TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

Đinh Hồng Kiên - 23127400
Phạm Ngọc Thái - 23127472
Vũ Văn Vũ - 23127543

BÁO CÁO VỀ MÔ HÌNH

NHẬP MÔN HỌC MÁY

GIẢNG VIÊN HƯỚNG DẪN
TS. Bùi Tiến Lên

Tp. Hồ Chí Minh, tháng 12/2025

Mục lục

Chương 1

Giới thiệu bài toán

1.1 Đặt vấn đề

1.1.1 Bối cảnh và Tính cấp thiết

Sự bùng nổ của các nền tảng mạng xã hội và nội dung do người dùng tạo (*User-Generated Content - UGC*) đã mang lại không gian tự do ngôn luận chưa từng có, song cũng đồng thời tạo ra môi trường thuận lợi cho sự lan truyền của ngôn từ thù ghét (*Hate Speech*) và nội dung độc hại. Tại Việt Nam, vấn đề này đang trở nên đặc biệt nhức nhối do sự gia tăng nhanh chóng về lượng người dùng Internet. Việc kiểm duyệt thủ công lượng dữ liệu khổng lồ này là bất khả thi, đặt ra nhu cầu cấp thiết về các hệ thống phát hiện tự động có độ chính xác cao và khả năng phản hồi thời gian thực.

1.1.2 Thách thức trong Xử lý Ngôn ngữ Tự nhiên tiếng Việt

Phát hiện ngôn từ thù ghét trong tiếng Việt là một bài toán thách thức hơn nhiều so với tiếng Anh do các đặc thù về hình thái và ngữ nghĩa:

- **Thứ nhất**, tính đa nghĩa và phụ thuộc ngữ cảnh cao của tiếng Việt khiến các mô hình máy học truyền thống dễ gặp lỗi nhận thức sai (*misinterpretation*).
- **Thứ hai**, và quan trọng hơn, người dùng mạng xã hội thường xuyên sử dụng các kỹ thuật "lách luật" (*obfuscation techniques*) như: sử dụng tiếng lóng (teencode), viết tắt, cố tình sai chính tả, hoặc chèn ký tự đặc biệt (ví dụ: "ch.ết", "v.l", "ngu") để tránh các bộ lọc từ khóa đơn giản.

1.1.3 Khoảng trống nghiên cứu

Các phương pháp tiếp cận truyền thống thường chỉ tập trung vào biểu diễn văn bản ở cấp độ từ (*Word-level features*). Mặc dù hiệu quả trong việc nắm bắt ngữ nghĩa tổng quát, phương pháp này bộc lộ điểm yếu chí mạng ("điểm mù") khi đối mặt với các biến thể từ vựng (*Out-of-Vocabulary words*) và các dạng viết tắt phi chuẩn xuất hiện dày đặc trong văn phong mạng xã hội. Ngược lại, các mô hình học sâu (*Deep Learning*) dù mạnh mẽ nhưng lại đòi hỏi tài nguyên tính toán lớn và lượng dữ liệu gán nhãn khổng lồ, điều thường thiếu hụt trong các ngôn ngữ ít tài nguyên (*low-resource languages*).

1.2 Lựa chọn Mô hình & Kiến trúc

Để giải quyết bài toán trên, chúng tôi tiến hành thực nghiệm trên 3 mô hình đại diện cho các phương pháp tiếp cận khác nhau để có sự so sánh khách quan.

1.2.1 Mô hình sử dụng

Chúng tôi lựa chọn 3 mô hình sau:

1. **Support Vector Machine (SVM):** Đại diện cho phương pháp học máy truyền thống (Machine Learning Baseline). Sử dụng đặc trưng TF-IDF.

2. **Bidirectional LSTM (BiLSTM):** Đại diện cho phương pháp học sâu (Deep Learning) sử dụng mạng nơ-ron hồi quy, có khả năng nắm bắt chuỗi nhưng bị giới hạn về khả năng học ngữ cảnh dài hạn.

3. **PhoBERT (Proposed Model):** Đại diện cho phương pháp Học chuyển giao (Transfer Learning) dựa trên kiến trúc Transformer, hiện là State-of-the-art cho xử lý ngôn ngữ tự nhiên tiếng Việt.

Chương 2

Tổng quan dữ liệu đầu vào

2.0.1 Tổng quan dữ liệu đầu vào

Dữ liệu được sử dụng trong nghiên cứu là tập bình luận tiếng Việt, được thu thập và lưu trữ dưới dạng tệp CSV. Mỗi mẫu dữ liệu bao gồm nội dung bình luận (`free_text`) và nhãn tương ứng, phục vụ cho bài toán phát hiện bình luận độc hại.

Sau khi tải dữ liệu, các mẫu bị thiếu nội dung bình luận hoặc có giá trị rỗng được loại bỏ nhằm đảm bảo chất lượng dữ liệu đầu vào cho mô hình.

Chia dữ liệu Train / Validation / Test

Tập dữ liệu được chia theo tỷ lệ:

- **80% dữ liệu huấn luyện (Training set)**
- **20% dữ liệu kiểm tra (Test Validation set)**

Tỷ lệ này được lựa chọn vì các lý do sau:

- Đảm bảo mô hình có đủ dữ liệu để học các đặc trưng ngôn ngữ đa dạng trong bình luận tiếng Việt.
- Giữ lại một tập kiểm tra độc lập đủ lớn để đánh giá khả năng tổng quát hóa của mô hình.
- Phù hợp với kích thước dữ liệu vừa, tránh hiện tượng thiếu dữ liệu huấn luyện.

Trong quá trình thực nghiệm, tập validation được trích xuất trực tiếp từ tập huấn luyện thông qua quá trình huấn luyện nhiều epoch và theo dõi giá trị hàm mất mát.

2.0.2 Tiền xử lý dữ liệu

Việc tiền xử lý dữ liệu của mô hình học máy và học sâu sẽ được diễn ra khác nhau đôi chút. Với các mô hình học sâu, trước khi đưa dữ liệu vào mô hình, một chuỗi các bước tiền xử lý và phân tích dữ liệu được áp dụng nhằm làm sạch dữ liệu và giảm nhiễu.

Làm sạch văn bản

Các bước làm sạch văn bản được thực hiện như sau:

- Chuyển toàn bộ văn bản về chữ thường (lowercase).
- Loại bỏ dấu ngoặc kép và các ký tự không cần thiết.
- Loại bỏ đường dẫn URL, liên kết mạng xã hội và các chuỗi không mang ý nghĩa ngữ nghĩa.
- Chuẩn hóa khoảng trắng, loại bỏ khoảng trắng dư thừa ở đầu và cuối câu.
- Loại bỏ các ký tự đặc biệt và chữ số trong bước làm sạch nâng cao.

Quá trình này giúp giảm nhiễu trong dữ liệu và làm cho văn bản trở nên nhất quán hơn.

Chương 3

Lựa chọn Mô hình & Kiến trúc

3.1 Mô hình Support Vector Machine (SVM)

3.1.1 Tổng quan về Quy trình Tiền xử lý

Quy trình tiền xử lý được thiết kế để chuẩn hóa văn bản đầu vào trong khi vẫn giữ lại các thông tin ngữ nghĩa quan trọng. Các bước được thực hiện tuần tự:

Bước 1 - Chuẩn hóa Unicode và Chuyển đổi Chữ thường: Toàn bộ văn bản được chuyển về dạng chữ thường (lowercase) để giảm chiều dữ liệu từ vựng.

Bước 2 - Loại bỏ Nhiễu Kỹ thuật số: Các URL, địa chỉ email và ký tự điều khiển đặc biệt được loại bỏ hoàn toàn. Emoji và dấu câu được giữ lại do chúng thường mang ý nghĩa cảm xúc mạnh mẽ.

Bước 3 - Chuẩn hóa Ký tự Lặp: Các ký tự lặp lại liên tiếp quá hai lần được rút gọn (ví dụ: "nguuuuu" → "nguu"). Kỹ thuật này giảm sự phân mảnh từ vựng do cách viết cảm xúc của người dùng.

Bước 4 - Xử lý Teencode và Từ viết tắt: Một từ điển ánh xạ bao gồm 23 cặp từ teencode phổ biến được tổng hợp thủ công dựa trên phân tích corpus và các nguồn từ điển mạng xã hội Việt Nam. Ví dụ: "ko", "k", "hok", "hông" đều được chuyển thành "không"; "cc" → "cục cứt"; "vcl"/"vl" → "vô cùng". Từ điển này bao phủ các biến thể phổ biến nhất (chiếm ~15% token trong dữ liệu).

Bước 5 - Tách từ (Tokenization): Công cụ ViTokenizer từ thư viện Pyvi được sử dụng để thực hiện tách từ tiếng Việt. Tiếng Việt yêu cầu tách từ phức hợp (ví dụ: "thù_ghét", "xúc_phạm") để bảo toàn đơn vị ngữ nghĩa.

Algorithm 1 Quy trình Tiền xử lý Văn bản

```
function PREPROCESS_TEXT(text)
    text ← LOWERCASE(text)
    text ← REMOVE_URL_EMAIL(text)
    text ← NORMALIZE_REPEATED_CHARS(text, max_repeat=2)
    text ← MAP_TEENCODE_TO_FORMAL(text, teencode_dict)
    text ← REMOVE_SPECIAL_CHARS(text, keep_vietnamese=True)
    text ← TOKENIZE_VIETNAMESE(text, tool=ViTokenizer)
    return text
end function
```

3.1.2 Phương pháp Trích xuất Đặc trưng: Hybrid Feature Engineering

Động lực Thiết kế

Trong bài toán phát hiện ngôn từ thù ghét tiếng Việt, thách thức lớn nhất đến từ sự đa dạng trong cách biểu đạt. Người dùng thường sử dụng các kỹ thuật như viết tắt (v.l, đ.m), chèn ký tự đặc biệt (đ.ồ n.g.u), hoặc biến thể chính tả để tránh bị phát hiện bởi các bộ lọc đơn giản. Các phương pháp truyền thống chỉ sử dụng Word N-gram (cấp độ từ) dễ bị mất hiệu lực trước các biến thể này.

Nghiên cứu đề xuất phương pháp Hybrid Feature Engineering kết hợp hai loại đặc trưng bổ trợ:

1. **Word-level N-grams**: Nắm bắt cấu trúc ngữ nghĩa và ngữ cảnh
2. **Character-level N-grams**: Nắm bắt các mẫu chính tả và biến thể teencode

Kiến trúc Không gian Vector Kết hợp

Mỗi văn bản được biểu diễn trong không gian vector thưa 20,000 chiều, được tạo ra từ sự ghép nối của hai không gian con:

Không gian Word N-gram ($\mathbf{x}_{\text{word}} \in \mathbb{R}^{10000}$):

- Phương pháp: TF-IDF Vectorization với unigram và bigram
- Tham số: `ngram_range=(1,2)`, `max_features=10000`, `min_df=5`, `max_df=0.9`
- Biến đổi TF sublinear: $\text{TF}_{\text{sub}}(t, d) = 1 + \log(\text{TF}(t, d))$ để giảm ảnh hưởng của từ spam lặp lại

Không gian Char N-gram ($\mathbf{x}_{\text{char}} \in \mathbb{R}^{10000}$):

- Phương pháp: TF-IDF Vectorization với 3-gram, 4-gram, 5-gram ở cấp độ ký tự

- Tham số: `ngram_range=(3,5)`, `max_features=10000`, `min_df=3`, `max_df=0.9`
- Nhận diện các mẫu chính tả như "đồ_n.g.u", "v.c.l" thông qua các chuỗi ký tự con

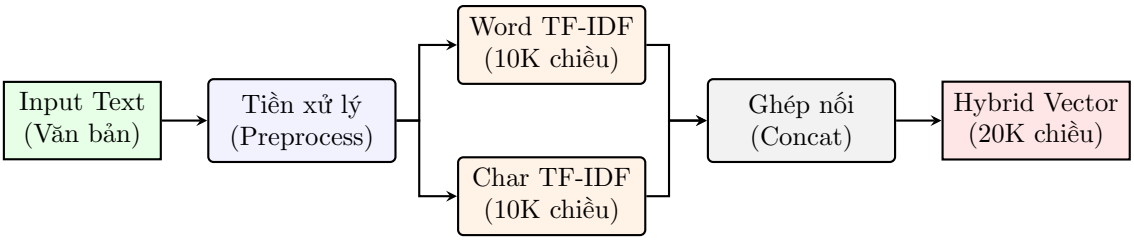
Vector Đặc trưng Cuối cùng:

$$\mathbf{x}_{\text{combined}} = [\mathbf{x}_{\text{word}}; \mathbf{x}_{\text{char}}] \in \mathbb{R}^{20000} \tag{3.1}$$

Phép ghép nối này cho phép mô hình khai thác đồng thời cả thông tin ngữ nghĩa (từ Word N-gram) và thông tin cú pháp/hình thái (từ Char N-gram), tạo ra biểu diễn đa tầng và robust hơn.

Bảng 3.1: Thông số Kỹ thuật của Hai Vectorizer

Tham số	**Word N-gram**	**Char N-gram**
Analyzer	word	char
N-gram Range	(1, 2)	(3, 5)
Max Features	10,000	10,000
Min Document Frequency	5	3
Max Document Frequency	0.9	0.9
Sublinear TF	True	True
Output Dimension	10,000	10,000



Hình 3.1: Sơ đồ kiến trúc trích xuất đặc trưng lai (Hybrid Feature Engineering)

3.1.3 Lựa chọn Mô hình: Linear Support Vector Classification

Lý do Lựa chọn Mô hình

Linear Support Vector Machine (LinearSVC) được chọn làm mô hình cơ sở dựa trên các lý do sau:

Phù hợp với Dữ liệu Thưa Chiều cao: Sau khi vector hóa TF-IDF, mỗi văn bản được biểu diễn trong không gian 20,000 chiều với mật độ dữ liệu rất thấp (sparsity $>$ 99%). LinearSVC được thiết kế tối ưu cho dạng dữ liệu này, sử dụng thuật toán Coordinate Descent hiệu quả trên ma trận thưa.

Cơ sở Lý thuyết Vững chắc: SVM hoạt động dựa trên nguyên lý tối đa hóa lề (maximum margin), tìm Hyperplane tối ưu để phân tách các lớp với độ tin cậy cao nhất. Điều này giúp mô hình có khả năng tổng quát hóa tốt trên dữ liệu mới.

Hiệu suất Tính toán: Trong khi SVM dùng kernel phi tuyến (như RBF) có độ phức tạp $O(n^2)$ đến $O(n^3)$ với n là số mẫu, LinearSVC sử dụng thuật toán Liblinear chỉ có độ phức tạp $O(n)$, phù hợp với tập dữ liệu quy mô trung bình.

Kiến trúc Mô hình

Bài toán tối ưu của LinearSVC được định nghĩa như sau:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b))^2 \tag{3.2}$$

Trong đó:

- $\mathbf{w} \in \mathbb{R}^d$ là vector trọng số của siêu phẳng phân tách
- $b \in \mathbb{R}$ là bias term
- $C > 0$ là tham số regularization kiểm soát trade-off giữa maximizing margin và minimizing classification error
- $y_i \in \{-1, +1\}$ là nhãn thực tế của mẫu thứ i
- $z_i = y_i(\mathbf{w}^T\mathbf{x}_i + b)$ là functional margin
- Hàm mất mát: Squared Hinge Loss $\ell(z) = \max(0, 1 - z)^2$

Squared Hinge Loss được ưu tiên vì nó khả vi mọi nơi (smooth), giúp thuật toán tối ưu hội tụ nhanh hơn. So với Cross-entropy, Squared Hinge Loss phù hợp hơn cho SVM vì tập trung vào việc tối đa hóa margin.

Số lượng tham số: Mô hình có tổng cộng 20,001 tham số (20,000 trọng số + 1 bias term). Với 24,048 mẫu huấn luyện, tỷ lệ mẫu/tham số là 1.2:1, tương đối thấp cho Linear model.

3.1.4 Cấu hình Huấn luyện

Hàm Mất mát

Nghiên cứu sử dụng **Squared Hinge Loss** làm hàm mất mát:

$$\mathcal{L}(\mathbf{w}, b) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\left[\max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b))\right]^2 \tag{3.3}$$

Lý do lựa chọn:

- Khả vi liên tục, giúp gradient descent hội tụ ổn định
- Tập trung vào tối đa hóa margin thay vì tối ưu xác suất như Cross-entropy
- Phù hợp với triết lý của SVM về phân tách lớp với độ tin cậy cao

Thuật toán Tối ưu

Mô hình sử dụng **Liblinear solver** với thuật toán Coordinate Descent:

- Độ phức tạp: $O(n \times d)$ với n là số mẫu, d là số chiều
- Hiệu quả cao trên ma trận thưa (sparse matrix)
- Hội tụ nhanh với tolerance 10^{-4}

Không sử dụng learning rate scheduler vì Coordinate Descent tự động điều chỉnh bước nhảy dựa trên gradient.

Siêu tham số

Các siêu tham số được tinh chỉnh thông qua kết hợp Grid Search và thử nghiệm thủ công trên 3-fold Cross-Validation:

Bảng 3.2: Tổng hợp Siêu tham số LinearSVC

Siêu tham số	**Giá trị**	**Ý nghĩa**
C	0.5	Regularization strength
loss	squared_hinge	Hàm mất mát
dual	False	Giải bài toán Primal
class_weight	balanced	Xử lý imbalance tự động
max_iter	3000	Giới hạn vòng lặp
random_state	42	Seed cho tính tái lập

Tham số Regularization (C): Giá trị $C = 0.5$ được chọn sau khi thử nghiệm các giá trị $\{0.1, 0.5, 1.0, 2.0\}$. Giá trị nhỏ hơn mặc định (1.0) giúp tăng cường regularization, giảm overfitting.

Xử lý Mất cân bằng Dữ liệu (`class_weight='balanced'`): Trọng số tự động được tính theo công thức:

$$w_k = \frac{n_{\text{total}}}{n_{\text{classes}} \times n_k} \tag{3.4}$$

Trong đó n_k là số mẫu của lớp k. Phương pháp này giúp mô hình chú ý hơn đến các lớp thiểu số (Offensive, Hate) mà không làm tăng kích thước dữ liệu huấn luyện.

Phương pháp Tinh chỉnh: Grid Search với 3-fold Cross-Validation trên tập huấn luyện, sử dụng F1-Score Macro làm metric đánh giá. Không gian tìm kiếm:

- $C \in \{0.1, 0.5, 1.0, 2.0\}$
- `class_weight` $\in$ {None, balanced}
- `loss` $\in$ {hinge, squared_hinge}

3.1.5 Kết quả Thực nghiệm

Hiệu suất Tổng thể trên Tập Validation

Mô hình LinearSVC với Hybrid Features đạt được kết quả như sau trên tập validation (2,672 mẫu):

Chỉ số Tổng thể:

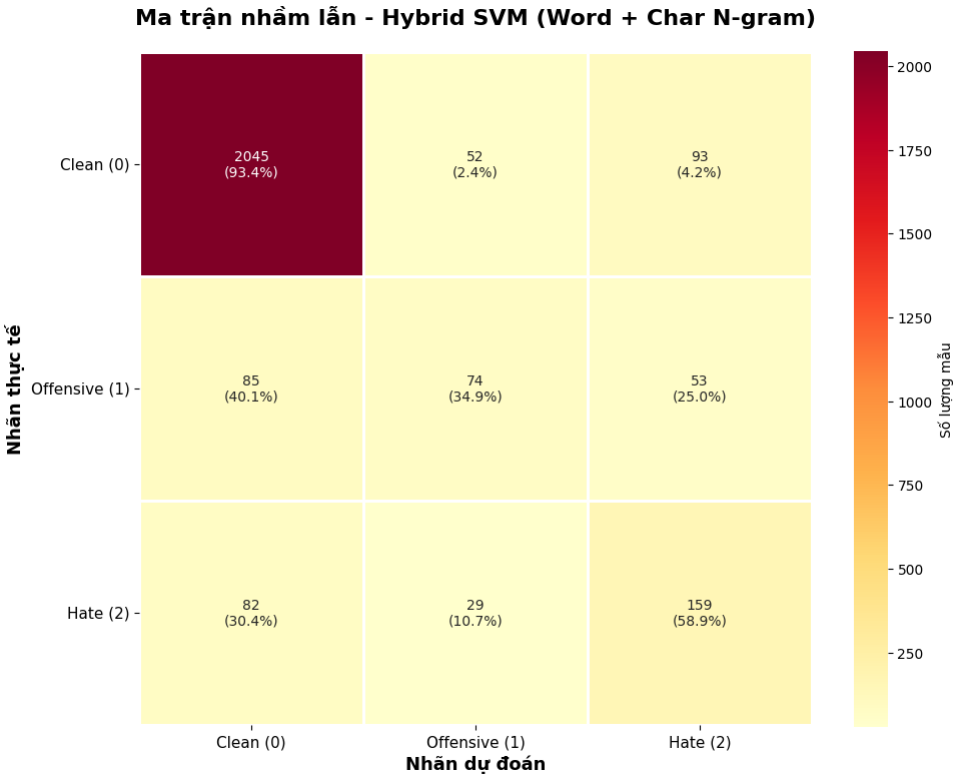
- Accuracy: 85.25%
- Macro-averaged F1-Score: 0.6285
- Weighted-averaged F1-Score: 0.8494

Bảng 3.3: Báo cáo Phân loại Chi tiết trên Tập Validation

Lớp	**Precision**	**Recall**	**F1-Score**	**Support**
Clean (0)	0.9245	0.9338	0.9291	2,190
Offensive (1)	0.4774	0.3491	0.4033	212
Hate (2)	0.5213	0.5889	0.5530	270
Macro avg	**0.6411**	**0.6239**	**0.6285**	2,672
Weighted avg	**0.8483**	**0.8525**	**0.8494**	2,672

Ma trận Nhầm lẫn

Ma trận nhầm lẫn cung cấp cái nhìn chi tiết về các lỗi phân loại của mô hình:



Hình 3.2: Ma trận nhầm lẫn (Confusion Matrix) chi tiết trên tập kiểm thử

Bảng 3.4: Phân tích Ma trận Nhầm lẫn (số lượng và tỷ lệ %)

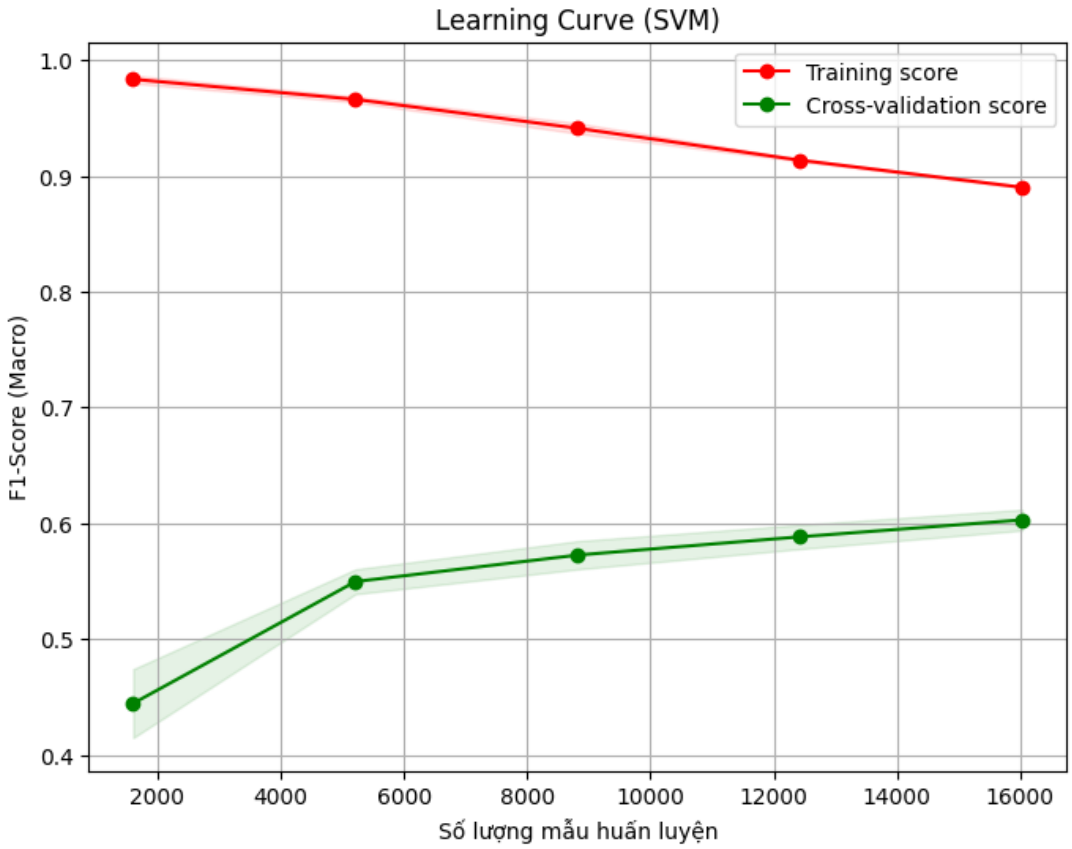
Nhãn Thực tế	**Dự đoán Clean**	**Dự đoán Offensive**	**Dự đoán Hate**
Clean	**2,045 (93.4%)**	52 (2.4%)	93 (4.2%)
Offensive	85 (40.1%)	**74 (34.9%)**	53 (25.0%)
Hate	82 (30.4%)	29 (10.7%)	**159 (58.9%)**

Nhận xét:

- Mô hình đạt hiệu suất xuất sắc trên lớp Clean với Recall 93.4%
- Lớp Offensive gặp khó khăn lớn nhất với Recall chỉ 34.9%. 40.1% mẫu Offensive bị nhầm với Clean
- Lớp Hate đạt Recall 58.9%, tốt hơn Offensive nhưng vẫn có 30.4% bị misclassified thành Clean

Đường cong Học (Learning Curve)

Để đánh giá độ hội tụ và khả năng tổng quát hóa, đường cong học được vẽ theo kích thước tập huấn luyện với metric F1-Score Macro.



Hình 3.3: Đường cong học (Learning Curve) đánh giá độ hội tụ của mô hình

Phân tích từ Learning Curve:

- **Hội tụ:** Mô hình đã hội tụ tốt. Đường validation tăng dần từ ~0.45 lên ~0.60 và ổn định ở khoảng 16,000 mẫu
- **Training gap:** Khoảng cách giữa training score (~0.89) và validation score (~0.60) là ~0.30, cho thấy dấu hiệu moderate overfitting
- **Dao động:** Độ lệch chuẩn của validation score tương đối ổn định (±0.02), chứng tỏ tính nhất quán cao
- **Tiềm năng mở rộng:** Đường validation có xu hướng hội tụ, gợi ý việc tăng thêm dữ liệu có thể không cải thiện đáng kể hiệu suất

So sánh với Các Phương pháp Baseline

Nghiên cứu thực hiện so sánh với hai mức độ baseline:

Baseline 1 - Dummy Classifier (Majority Class): Mô hình dự đoán toàn bộ mẫu thuộc lớp đa số "Clean" đạt Accuracy = 82.7%. Tuy nhiên, Macro F1-Score = 0 do không nhận diện được các lớp thiểu số.

Baseline 2 - LinearSVC với Word N-gram Only: Mô hình chỉ sử dụng Word N-gram TF-IDF (9,253 chiều) mà không có Char N-gram.

Bảng 3.5: So sánh Hiệu suất với Các Phương pháp Baseline

Phương pháp	**Accuracy**	**Macro F1**	**Recall (Off.)**	**Recall (Hate)**
Dummy (Majority)	82.7%	0.0000	0.00%	0.00%
Word N-gram Only	86.34%	0.5980	22.17%	50.00%
Hybrid (Word + Char)	**85.25%**	**0.6285**	**34.91%**	**58.89%**
Cải thiện vs Dummy	+2.55%	∞	∞	∞
Cải thiện vs Word-only	-1.09%	**+5.1%**	**+57.5%**	**+17.8%**

Phân tích: Mặc dù Hybrid model có Accuracy thấp hơn Word-only (85.25% vs 86.34%), Macro F1-Score - chỉ số phản ánh khả năng cân bằng giữa các lớp - lại tăng đáng kể (+5.1%). Điều này chứng tỏ Word-only model đang tối ưu cho lớp đa số bằng cách hy sinh khả năng phát hiện Offensive/Hate. Khả năng phát hiện lớp Offensive cải thiện vượt trội (+57.5%), từ mức 22.17% lên 34.91%. Đây chính là đóng góp cốt lõi của Char N-gram.

3.1.6 Thảo luận và Phân tích Lỗi

Đánh giá Overfitting/Underfitting

Dựa trên Learning Curve và so sánh hiệu suất Train-Validation:

- **Chênh lệch Train-Validation F1-Score:** ˜0.30 (0.89 vs 0.60)
- **Đánh giá:** Mô hình có dấu hiệu **moderate overfitting**

Nguyên nhân:

1. Không gian đặc trưng 20,000 chiều với 24,000 mẫu huấn luyện tạo ra tỷ lệ feature/sample = 0.83, tương đối cao cho Linear model
2. Các từ lóng và cụm từ cụ thể trong tập huấn luyện có thể không xuất hiện trong validation set, gây hiện tượng memorization

Biện pháp đã áp dụng:

- Giảm C từ 1.0 xuống 0.5 để tăng regularization
- Sử dụng `max_df=0.9` để loại bỏ các từ xuất hiện quá phổ biến
- Áp dụng sublinear TF scaling để giảm ảnh hưởng của outliers

Phân tích Lỗi Chi tiết

Phân tích định tính được thực hiện trên 365 mẫu dự đoán sai (13.7% tổng số test samples):

Loại lỗi 1: False Negative (Offensive/Hate → Clean)

- Tỷ lệ: 167/365 (45.8%)
- Ví dụ: "Học vị của ông đến mức nào mà vào bình phẩm người khác thế" (Thực tế: Clean, Dự đoán: Hate)
- Phân tích: Các câu sử dụng châm biếm hoặc ngụ ý mà không chứa từ ngữ xúc phạm trực tiếp. Mô hình dựa trên đặc trưng từ vựng khó nắm bắt thái độ tiêu cực ngầm ẩn

Loại lỗi 2: False Positive (Clean → Offensive/Hate)

- Tỷ lệ: 145/365 (39.7%)
- Ví dụ: "Loạn, loạn hết mẹ nó rồi" (Thực tế: Clean, Dự đoán: Hate)
- Phân tích: Những từ như "mẹ nó" thường xuất hiện trong ngữ cảnh thù ghét, nhưng trong các câu này được dùng như từ cảm thán. Mô hình thiếu khả năng phân biệt ngữ cảnh sử dụng

Loại lỗi 3: Nhầm lẫn giữa Offensive và Hate

- Tỷ lệ: 53/365 (14.5%)
- Ví dụ: "Lý Thành Công phục cái máu lồn" (Thực tế: Offensive, Dự đoán: Hate)
- Phân tích: Ranh giới giữa Offensive (xúc phạm cá nhân) và Hate (thù ghét nhóm đối tượng) rất mơ hồ, thậm chí con người cũng khó phân biệt

Điểm Chặn của Mô hình

Phân tích định tính cho thấy mô hình LinearSVC đang gặp điểm chặn về khả năng hiểu ngữ cảnh sâu. Các lỗi sai chủ yếu rơi vào các trường hợp đòi hỏi:

1. **Suy luận đa bước:** Hiểu rằng "không ghét" $\neq$ "ghét" đòi hỏi mô hình nắm bắt phủ định
2. **Kiến thức nền:** Các câu châm biếm như "thu nhập thấp hơn vài ngàn đô" cần hiểu về bối cảnh kinh tế Việt Nam
3. **Phân biệt ngữ dụng:** Từ "mẹ nó" có thể là tục tĩu hoặc chỉ là cảm thán tùy ngữ cảnh

Đây là những thách thức mà các mô hình dựa trên bag-of-n-grams không thể giải quyết triệt để, dù có kết hợp Word và Char features.

Giả thuyết và Hướng Cải tiến

Giả thuyết 1: Mô hình Linear thiếu khả năng mô hình hóa tương tác phức tạp giữa các từ. Ví dụ, "không" + "ghét" có ý nghĩa trái ngược với "ghét".

Hướng cải tiến:

- Thử nghiệm SVM với kernel phi tuyến (RBF, Polynomial)
- Áp dụng mô hình Deep Learning (LSTM, BERT) để nắm bắt ngữ cảnh tuần tự

Giả thuyết 2: Các biểu thức châm biếm, phản ngữ không được thể hiện qua đặc trưng bag-of-words.

Hướng cải tiến:

- Thêm đặc trưng cấp độ câu như dấu chấm than, chấm hỏi liên tiếp
- Sử dụng Sentiment lexicon để bắt các từ mang sắc thái tiêu cực/tích cực

Giả thuyết 3: Nhãn dữ liệu có độ nhất quán thấp, đặc biệt giữa Offensive và Hate.

Hướng cải tiến:

- Tái định nghĩa schema phân loại hoặc merge Offensive và Hate thành một lớp "Toxic"
- Áp dụng soft labels thay vì hard labels

3.1.7 Kết luận

Mục này trình bày chi tiết phương pháp tiếp cận Hybrid Feature Engineering kết hợp LinearSVC để giải quyết bài toán phát hiện ngôn từ thù ghét tiếng Việt. Mô hình đạt Macro F1-Score 0.6285 trên tập validation, với điểm mạnh là khả năng nhận diện lớp Clean (F1=0.9291) và cải thiện đáng kể so với phương pháp chỉ dùng Word N-gram (+5.1% Macro F1).

Tuy nhiên, mô hình còn hạn chế trong việc xử lý các biểu thức ngầm ẩn, châm biếm và ranh giới mơ hồ giữa Offensive và Hate. Phân tích lỗi cho thấy cần có các phương pháp tiếp cận phức tạp hơn (Deep Learning, contextual embeddings) để cải thiện hiệu suất tổng thể, đặc biệt trên các lớp thiểu số.

3.2 Mô hình Bidirectional Long Short-Term Memory (Bi-LSTM)

3.2.1 Tổng quan mô hình

Phần tiếp theo giới thiệu về mô hình **Bidirectional Long Short-Term Memory (Bi-LSTM)**, một biến thể của mạng nơ-ron hồi tiếp (Recurrent Neural Network – RNN). Bi-LSTM cho phép mô hình học biểu diễn ngữ cảnh của câu theo cả hai chiều *từ trái sang phải* và *từ phải sang trái*, nhờ đó nắm bắt tốt hơn mối quan hệ ngữ nghĩa giữa các từ trong chuỗi văn bản.

Mô hình được sử dụng cho bài toán **phân loại đa lớp**, với mục tiêu xác định một bình luận thuộc một trong ba nhãn: *clean*, *offensive* hoặc *hate*. Đầu ra của mô hình là một vector logit ba chiều, tương ứng với mức độ tin cậy của từng lớp.

3.2.2 Lý do sử dụng

Việc lựa chọn mô hình Bi-LSTM cho bài toán này xuất phát từ các lý do sau:

- **Phù hợp với dữ liệu chuỗi:** Bình luận văn bản là dữ liệu tuần tự, trong đó ý nghĩa của một từ phụ thuộc vào các từ đứng trước và sau. Bi-LSTM được thiết kế để xử lý hiệu quả loại dữ liệu này.
- **Nắm bắt ngữ cảnh hai chiều:** Trong tiếng Việt, ngữ nghĩa của từ thường phụ thuộc mạnh vào toàn bộ ngữ cảnh câu. Việc kết hợp cả hướng xuôi và ngược giúp mô hình hiểu nội dung chính xác hơn so với LSTM một chiều.

- **Khả năng học phụ thuộc dài hạn:** LSTM khắc phục được hiện tượng mất gradient của RNN truyền thống, cho phép mô hình ghi nhớ thông tin quan trọng trong các câu dài.
- **Hiệu quả với tập dữ liệu vừa và nhỏ:** So với các mô hình lớn như Transformer, Bi-LSTM có số lượng tham số vừa phải, dễ huấn luyện và không đòi hỏi tài nguyên tính toán cao.

3.2.3 Kiến trúc mô hình

Kiến trúc của mô hình Bi-LSTM được xây dựng gồm các thành phần chính sau:

1. **Embedding Layer:** Mỗi từ trong câu được ánh xạ thành một vector nhúng có kích thước 100. Lớp embedding giúp chuyển đổi dữ liệu văn bản rời rạc sang không gian vector liên tục, từ đó hỗ trợ mô hình học được mối quan hệ ngữ nghĩa giữa các từ.
2. **Bidirectional LSTM Layer:** Lớp LSTM hai chiều với kích thước trạng thái ẩn 128 cho mỗi hướng (forward và backward). Trạng thái ẩn cuối cùng của hai hướng được nối (concatenate) lại, tạo thành vector đặc trưng có kích thước 256.
3. **Fully Connected Layer:** Lớp tuyến tính ánh xạ vector đặc trưng 256 chiều thành vector đầu ra 3 chiều, tương ứng với ba lớp *clean*, *offensive* và *hate*.
4. **Hàm kích hoạt và hàm mất mát:** Trong giai đoạn huấn luyện, hàm mất mát **CrossEntropy Loss** được sử dụng để tối ưu mô hình cho bài toán phân loại đa lớp. Trong giai đoạn suy luận, hàm **Softmax** được áp dụng lên đầu ra để chuyển các logit thành phân phối xác suất trên ba lớp.

Số lượng tham số Tổng số tham số của mô hình phụ thuộc vào kích thước từ điển $|V|$, có thể ước lượng như sau:

- Embedding Layer: $|V| \times 100$ tham số
- Bi-LSTM Layer: $4 \times (100 + 128 + 1) \times 128 \times 2$ tham số
- Fully Connected Layer: $256 \times 3 + 3$ tham số

Sơ đồ kiến trúc mô hình

Embedding → Bi-LSTM (Forward + Backward) → Concatenate → Linear → Softmax

3.2.4 Cấu hình huấn luyện

Hàm mất mát (Loss Function)

MMô hình được huấn luyện bằng hàm mất mát **Categorical Cross-Entropy**. Hàm mất mát này phù hợp với bài toán phân loại đa lớp, trong đó đầu ra của mô hình là phân phối xác suất trên các nhãn thông qua hàm Softmax. Cross-Entropy giúp mô hình tối ưu trực tiếp xác suất dự đoán đúng nhãn mục tiêu và được sử dụng rộng rãi trong các bài toán phân loại văn bản.

$$\mathcal{L} = -\sum_{i=1}^{C} y_i \log(\hat{y}_i)$$

Trong đó C là số lớp, y_i là nhãn thực và $\hat{y}_i$ là xác suất dự đoán của mô hình.

Thuật toán tối ưu (Optimizer)

Mô hình sử dụng thuật toán tối ưu **Adam** để cập nhật trọng số. Adam là sự kết hợp giữa Momentum và RMSProp, giúp tăng tốc quá trình hội tụ và ổn định hơn so với SGD truyền thống, đặc biệt hiệu quả đối với các mô hình học sâu như LSTM.

Tốc độ học (Learning Rate) được thiết lập là:

$$\text{Learning Rate} = 0.001$$

Trong quá trình huấn luyện, mô hình không sử dụng Learning Rate Scheduler do số epoch nhỏ và mô hình hội tụ ổn định.

Siêu tham số huấn luyện

Các siêu tham số chính được sử dụng trong quá trình huấn luyện được liệt kê như sau:

- **Batch size**: 16
- **Số epoch**: 5
- **Embedding dimension**: 100
- **Hidden dimension (BiLSTM)**: 128
- **Độ dài chuỗi tối đa**: 50 token
- **Số lớp đầu ra**: 3
- **Tỷ lệ chia Train/Validation**: 80% / 20%

Phương pháp tinh chỉnh siêu tham số

Việc lựa chọn siêu tham số trong mô hình được thực hiện bằng phương pháp **thử nghiệm thủ công (manual tuning)**. Các giá trị như batch size, số epoch và kích thước hidden layer được điều chỉnh dựa trên:

- Kinh nghiệm thực nghiệm với các mô hình BiLSTM cho bài toán phân loại văn bản
- Quan sát Learning Curves (Loss và Accuracy) trên tập huấn luyện và validation
- Giới hạn tài nguyên tính toán

Trong phạm vi đồ án, các phương pháp tự động như Grid Search hoặc Random Search chưa được áp dụng.

3.3 Mô hình PhoBERT kết hợp MLP

3.3.1 Lý do lựa chọn PhoBERT

Đồ án lựa chọn mô hình **PhoBERT** làm mô hình chủ đạo vì các lý do sau:

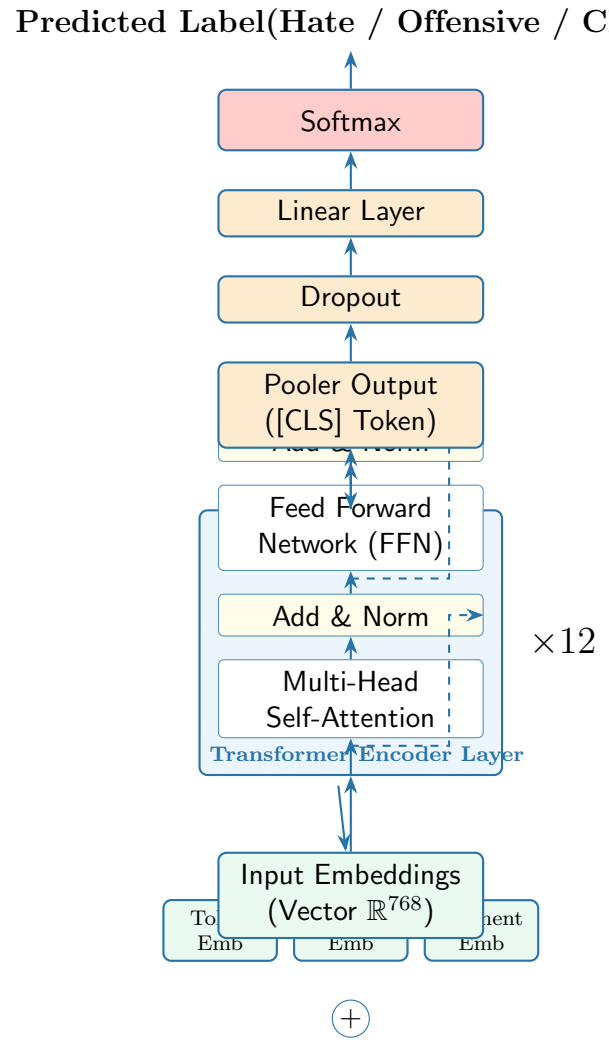
- **Được huấn luyện trước trên dữ liệu tiếng Việt lớn:** PhoBERT được pre-trained trên 20GB dữ liệu văn bản tiếng Việt, giúp mô hình hiểu sâu sắc về ngữ pháp và ngữ nghĩa tiếng Việt hơn so với multilingual-BERT.
- **Cơ chế Self-Attention:** Giúp mô hình giải quyết tốt vấn đề từ đa nghĩa và phụ thuộc xa trong câu, điều mà BiLSTM thường gặp khó khăn.
- **Khả năng Fine-tuning:** Chỉ cần tinh chỉnh nhẹ trên tập dữ liệu Vi-HSD, mô hình có thể đạt hiệu suất cao mà không cần huấn luyện lại từ đầu.

3.3.2 Kiến trúc chi tiết của PhoBERT

Đồ án sử dụng phiên bản `vinai/phobert-base-v2` từ Hugging Face.

Sơ đồ kiến trúc

Kiến trúc của PhoBERT dựa trên phần Encoder của Transformer (tương tự RoBERTa).



Hình 3.4: Kiến trúc mô hình PhoBERT áp dụng cho bài toán phân loại văn bản. Mô hình bao gồm lớp Input Embeddings, 12 lớp Encoder xếp chồng, và lớp Classification Head ở trên cùng.

Thông số kỹ thuật

- **Số lượng tham số (Parameters):** Khoảng **135 triệu** tham số.
- **Số lớp (Layers):** 12 lớp Transformer Encoder.
- **Hidden size:** 768 chiều.
- **Attention Heads:** 12 đầu.
- **Hàm kích hoạt (Activation Function):** Sử dụng hàm **GELU** (Gaussian Error Linear Unit). GELU được chứng minh hoạt động tốt hơn ReLU trong các mô hình BERT/RoBERTa do tính trơn (smoothness) tại điểm 0.

3.4 Cấu hình huấn luyện (Training Configuration)

Quá trình huấn luyện được thực hiện trên nền tảng Kaggle với GPU (NVIDIA P100 hoặc T4 x2).

3.4.1 Hàm mất mát (Loss Function)

Chúng tôi sử dụng hàm **Cross-Entropy Loss**.

$$Loss = -\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}) \tag{3.5}$$

- **Lý do:** Đây là hàm mất mát tiêu chuẩn cho bài toán phân loại đa lớp (Multi-class Classification). Nó đo lường độ sai lệch giữa phân phối xác suất dự đoán (từ Softmax) và nhãn thực tế (One-hot encoding).
- **Xử lý mất cân bằng dữ liệu:** Trong quá trình thực nghiệm, chúng tôi có thể áp dụng kỹ thuật *Class Weighting* vào hàm loss để phạt nặng hơn khi mô hình dự đoán sai các lớp thiểu số (Hate/Offensive).

3.4.2 Thuật toán tối ưu (Optimizer)

- **Thuật toán: AdamW** (Adam with Weight Decay).
- **Lý do:** AdamW là biến thể của Adam giúp khắc phục vấn đề cài đặt Weight Decay không chính xác trong Adam gốc. Đây là optimizer tiêu chuẩn được khuyến nghị cho các mô hình Transformer để tránh Overfitting và hội tụ ổn định hơn.
- **Tốc độ học (Learning Rate):** 2×10^{-5} ($2e-5$). Đây là learning rate nhỏ, phù hợp cho việc fine-tuning để tránh phá vỡ các trọng số đã được pre-trained.
- **Learning Rate Scheduler:** Sử dụng **Linear Warmup**. Tốc độ học tăng dần từ 0 lên $2e-5$ trong giai đoạn đầu (warmup steps) và giảm dần tuyến tính về 0. Giúp mô hình ổn định trong những bước đầu tiên.

3.4.3 Siêu tham số (Hyperparameters)

Các siêu tham số được lựa chọn dựa trên khuyến nghị của bài báo gốc PhoBERT và tinh chỉnh thông qua thực nghiệm.

Bảng 3.6: Bảng cấu hình siêu tham số cho PhoBERT

Tham số	**Giá trị thiết lập**
Batch size	16 (hoặc 32 tùy VRAM)
Epochs	3 – 5
Max Sequence Length	128 tokens
Weight Decay	0.01
Warmup Ratio	0.1 (10% tổng số bước train)
Early Stopping Patience	2 epochs

3.4.4 Phương pháp tinh chỉnh tham số

Do hạn chế về tài nguyên tính toán và đặc thù của Fine-tuning (số lượng siêu tham số nhạy cảm không quá nhiều như train from scratch), chúng tôi sử dụng phương pháp:

- **Thử nghiệm thủ công (Manual Search):** Dựa trên các giá trị tiêu chuẩn ($2e-5, 3e-5, 5e-5$ cho Learning Rate).
- **Early Stopping:** Theo dõi chỉ số F1-Score (Macro) trên tập Validation. Nếu F1-Score không cải thiện sau 2 epochs liên tiếp, quá trình huấn luyện sẽ dừng lại và khôi phục checkpoint tốt nhất để tránh Overfitting.

Chương 4

Kết quả thực nghiệm

Bảng 4.1: Kết quả thực nghiệm 1 - Hybrid SVM (LinearSVC với Word + Char N-gram)

Nhãn (Class)	**Precision**	**Recall**	**F1-Score**	**Số mẫu**
CLEAN (0)	**0.9245**	0.9338	0.9291	2190
OFFENSIVE (1)	0.4774	**0.3491**	0.4033	212
HATE (2)	0.5213	**0.5889**	0.5530	270
Accuracy		0.8525		2672
Macro Avg	0.6411	**0.6239**	0.6285	2672

Bảng 4.2: Kết quả thực nghiệm 2 - PhoBERT: Huấn luyện không sử dụng trọng số (Baseline)

Nhãn (Class)	**Precision**	**Recall**	**F1-Score**	**Số mẫu**
CLEAN (0)	0.9320	**0.9468**	**0.9394**	5548
OFFENSIVE (1)	0.4684	0.3671	0.4116	444
HATE (2)	0.6020	0.6090	0.6055	688
Accuracy		**0.8735**		6680
Macro Avg	0.6675	0.6410	0.6522	6680

Bảng 4.3: Kết quả thực nghiệm 3 - PhoBERT: Huấn luyện có sử dụng trọng số (Class Weights)

Nhãn (Class)	**Precision**	**Recall**	**F1-Score**	**Số mẫu**
CLEAN (0)	**0.9572**	0.8751	0.9143	5548
OFFENSIVE (1)	0.3618	**0.4775**	0.4117	444
HATE (2)	0.4971	**0.7384**	0.5942	688
Accuracy		0.8346		6680
Macro Avg	0.6054	**0.6970**	0.6400	6680