

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN 1



BÁO CÁO

Môn: Thực tập cơ sở

Đề tài: Ứng dụng LLM trong tự động phân loại tin nhắn.

Nhóm: 17

Giảng viên hướng dẫn:

Ngô Tiến Đức

Sinh viên thực hiện:

Đào Ngọc Đức

Mã sinh viên:

B22DCCN221

Hà Nội - 2025

Lời mở đầu

Trong kỷ nguyên bùng nổ của dữ liệu số và trí tuệ nhân tạo, các mô hình ngôn ngữ lớn (Large Language Models – LLM) đã và đang thể hiện tiềm năng vượt trội trong việc xử lý và hiểu ngôn ngữ tự nhiên. Một trong những ứng dụng thiết thực và cấp thiết hiện nay là việc phân loại và lọc tin nhắn rác – một vấn nạn gây ảnh hưởng nghiêm trọng đến trải nghiệm người dùng, tính bảo mật và hiệu quả vận hành của các hệ thống thông tin.

Xuất phát từ nhu cầu thực tiễn đó, em đã chọn đề tài: "Ứng dụng mô hình ngôn ngữ lớn (LLM) trong phân loại tin nhắn rác" (Spam & Ham), nhằm tìm hiểu khả năng áp dụng các mô hình AI hiện đại vào việc tự động phát hiện và phân loại tin nhắn rác một cách hiệu quả, chính xác và thích ứng với ngữ cảnh tiếng Việt.

Trong quá trình thực hiện đề tài, em xin bày tỏ lòng biết ơn sâu sắc đến thầy **Ngô Tiến Đức**, người đã tận tình định hướng, góp ý chuyên môn và tạo điều kiện thuận lợi để em hoàn thành tốt báo cáo này.

Mặc dù đã nỗ lực tối đa, nhưng do thời gian và kiến thức còn hạn chế, báo cáo đề tài sẽ không tránh khỏi những thiếu sót. Em rất mong nhận được sự góp ý quý báu từ thầy và bạn đọc để hoàn thiện hơn trong những dự án nghiên cứu tiếp theo.

Trân trọng!

Hà Nội, ngày 29 tháng 05 năm 2025

Sinh viên thực hiện

Đào Ngọc Đức

Mục lục

Danh mục hình vẽ.....	1
Danh mục từ viết tắt.....	2
Giới thiệu	3
CHƯƠNG 1. TỔNG QUAN MÔ HÌNH.....	4
1.1 Bài toán đặt ra:	4
1.2 Giới hạn đề tài:	4
1.3 Các nghiên cứu và giải pháp liên quan:	5
1.4 Kiến trúc, thuật toán sử dụng trong mô hình:	6
1.4.1 Kiến trúc mô hình:.....	6
1.4.2 Thuật toán, thư viện sử dụng:	6
CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ MÔ HÌNH	9
2.1 Khái quát:	9
2.2 Thiết kế và cài đặt mô hình:	9
2.2.1 Xây dựng tập dữ liệu (Dataset):.....	9
2.2.2 Tiền xử lý (Preprocessing):	16
2.2.3 Huấn luyện mô hình:	20
CHƯƠNG 3: ĐÁNH GIÁ MÔ HÌNH.....	27
3.1 Tải mô hình cuối và đánh giá:	27
3.2 Kết quả đạt được:	27
KẾT LUẬN.....	29
HƯỚNG PHÁT TRIỂN	30
TÀI LIỆU THAM KHẢO	31

Danh mục hình vẽ

Hình 1. Kiến trúc mô hình	6
Hình 2. Hàm synonym replacement	10
Hình 3. Hàm random date sinh ngày ngẫu nhiên	10
Hình 4. Hàm thêm các lỗi chính tả.....	11
Hình 5. Hàm random casing	11
Hình 6. Hàm random suffix	12
Hình 7. Giá trị thay thế cho tin nhắn spam	12
Hình 8. Mẫu tin nhắn spam.....	12
Hình 9. Sinh tin nhắn spam.....	13
Hình 10. Mẫu tin nhắn ham	13
Hình 11. Thay thế giá trị cho tin nhắn ham	14
Hình 12. Hàm tạo tập dữ liệu	14
Hình 13. Dữ liệu gốc	15
Hình 14. Biểu đồ phân bố nhãn.....	16
Hình 15. Danh sách các stopwords	17
Hình 16. Hàm clean text	18
Hình 17. Đầu ra tiền xử lý văn bản	18
Hình 18. Biểu đồ đặc trưng.....	19
Hình 19. Các từ xuất hiện nhiều nhất	20
Hình 20. Chuyển đổi label	20
Hình 21. Bước thiết lập.....	21
Hình 22. Định nghĩa dataset.....	21
Hình 23. Xác suất EDA	22
Hình 24. Tải tokenizer và chia tập dữ liệu.....	22
Hình 25. Tạo các bộ nạp dữ liệu	23
Hình 26. Tải mô hình và thiết lập fine-tuning.....	24
Hình 27. Tạo hàm mất mát	24
Hình 28. Tối ưu quá trình huấn luyện	25
Hình 29. Huấn luyện mô hình.....	26
Hình 30. Bước đánh giá	27
Hình 31. Đánh giá mô hình.....	27

Danh mục từ viết tắt

Từ viết tắt	Thuật ngữ tiếng Anh/Giải thích	Thuật ngữ tiếng Việt/Giải thích
LLM	Large Language Model	Mô hình ngôn ngữ lớn
NLP	Natural Language Processing	Xử lý ngôn ngữ tự nhiên
Spam	Spam messages	Tin nhắn rác
Ham	Normal messages	Tin nhắn thông thường
AUC	Area Under Curve	Thước đo khả năng phân biệt các lớp
TP	True Positive	Dự đoán đúng nhãn spam
TN	True Negative	Dự đoán đúng nhãn ham
FP	False Positive	Dự đoán nhầm ham thành spam
FN	False Negative	Dự đoán nhầm spam thành ham
EDA	Easy Data Augmentation	Tăng cường dữ liệu đơn giản
BCE	Binary Cross Entropy	Hàm mất mát Entropy chéo nhị phân

Giới thiệu

Trong thời đại chuyên đổi số mạnh mẽ, việc trao đổi thông tin qua tin nhắn văn bản đã trở thành một phần không thể thiếu trong đời sống cá nhân và hoạt động của các tổ chức, doanh nghiệp. Tuy nhiên, sự phát triển nhanh chóng này cũng kéo theo sự gia tăng đáng kể của các loại tin nhắn không mong muốn, đặc biệt là tin nhắn rác – gây ảnh hưởng tiêu cực đến trải nghiệm người dùng, làm gián đoạn thông tin, thậm chí tiềm ẩn rủi ro về an ninh mạng và lừa đảo trực tuyến.

Trước thực trạng đó, việc xây dựng một hệ thống có khả năng tự động phát hiện và phân loại tin nhắn rác một cách chính xác và hiệu quả là hết sức cần thiết. Trong những năm gần đây, các mô hình ngôn ngữ lớn (Large Language Models – LLM) như GPT, BERT, RoBERTa, ... đã chứng minh khả năng vượt trội trong việc xử lý ngôn ngữ tự nhiên, đặc biệt trong các tác vụ phân loại văn bản. Việc ứng dụng LLM vào bài toán phân loại tin nhắn không chỉ mở ra hướng tiếp cận mới có độ chính xác cao, mà còn giúp tăng khả năng thích ứng với ngôn ngữ linh hoạt, đa dạng và thường xuyên biến đổi trong các loại tin nhắn hiện nay.

Với mục tiêu tìm hiểu và đánh giá khả năng của LLM trong việc phân loại tin nhắn rác, đề tài “Ứng dụng mô hình ngôn ngữ lớn (LLM) trong bài toán phân loại tin nhắn” được thực hiện nhằm xây dựng, thử nghiệm và phân tích hiệu quả của mô hình khi áp dụng vào dữ liệu thực tế. Kết quả nghiên cứu kỳ vọng sẽ góp phần cung cấp giải pháp hiệu quả cho các hệ thống lọc tin nhắn thông minh, tăng cường bảo mật và cải thiện trải nghiệm người dùng trong các ứng dụng nhắn tin.

Báo cáo đề tài gồm 3 chương với nội dung chính như sau:

- **Chương 1** nghiên cứu tổng quan về mô hình Large Language Model (LLM) cho bài toán phân loại tin nhắn, bao gồm các nội dung khái quát về bài toán đặt ra và các thuật toán, mô hình.
- **Chương 2** thực hiện việc phân tích, thiết kế mô hình từ việc xây dựng bộ dataset, tiền xử lý dữ liệu và huấn luyện mô hình.
- **Chương 3** thực hiện việc thử nghiệm, đánh giá mô hình.

CHƯƠNG 1. TỔNG QUAN MÔ HÌNH

1.1 Bài toán đặt ra:

Trong thời đại bùng nổ về thông tin truyền thông hiện nay, người dùng cá nhân và doanh nghiệp nhận một lượng lớn các tin nhắn điện tử hàng ngày, trong đó có không ít những tin nhắn rác, tin nhắn lừa đảo hoặc mang nội dung không mong muốn. Việc phân loại tin nhắn một cách thủ công không chỉ tốn rất nhiều thời gian mà còn tiềm ẩn nhiều rủi ro bảo mật nếu người dùng vô tình tương tác với các nội dung độc hại. Trước thực trạng đó, nhu cầu về một hệ thống phân loại tin nhắn tự động, chính xác và thông minh đang ngày càng trở nên cấp thiết.

Việc xây dựng ứng dụng mô hình ngôn ngữ lớn trong hệ thống phân loại tin nhắn giúp cá nhân và các doanh nghiệp tiết kiệm thời gian và công sức cũng như là vận dụng sự phát triển mạnh mẽ của các mô hình xử lý ngôn ngữ tự nhiên (NLP). Nhờ khả năng học biểu diễn ngữ cảnh sâu sắc và thích nghi với cả nhiều dữ liệu ngôn ngữ (trong đó có tiếng Việt), LLM đã mở ra một hướng tiếp cận mới cho bài toán phân loại tin nhắn một cách chính xác, linh hoạt và đồng thời có khả năng mở rộng cho nhiều mục tiêu khác nhau.

1.2 Giới hạn đề tài:

Trong khuôn khổ đề tài này, bài toán được giới hạn ở các khía cạnh sau:

- Phạm vi dữ liệu: Do gặp khó khăn trong việc tìm tập dữ liệu tin nhắn tiếng Việt đáng tin cậy nên em đã sử dụng phương pháp sinh dữ liệu ngẫu nhiên dựa trên những mẫu tin nhắn phổ biến. Dữ liệu được thu thập là tiếng Việt, bao gồm các nội dung tin nhắn ngắn gọn có độ dài từ 5 đến 50 từ, được gán nhãn theo hai lớp chính là: spam (rác) và ham (bình thường).
- Mục tiêu bài toán: Tập trung vào phân loại nội dung tin nhắn đầu vào dựa trên ngữ nghĩa, không xét đến các yếu tố bổ sung như thời gian gửi, người gửi hay tác động từ các yếu tố bên ngoài.
- Đặc điểm mô hình: Sử dụng mô hình ngôn ngữ lớn (LLM) có khả năng xử lý tiếng Việt PhoBERT. Chủ yếu sử dụng fine-tuning trên mô hình đã được huấn luyện trước.
- Giới hạn về kỹ thuật: Đề tài không tập trung vào xây dựng hệ thống triển khai thực tế mà chủ yếu thử nghiệm mô hình trên tập dữ liệu với độ lớn vừa phải, được đánh giá bằng các chỉ số như Accuracy, F1-score, AUC và CM.

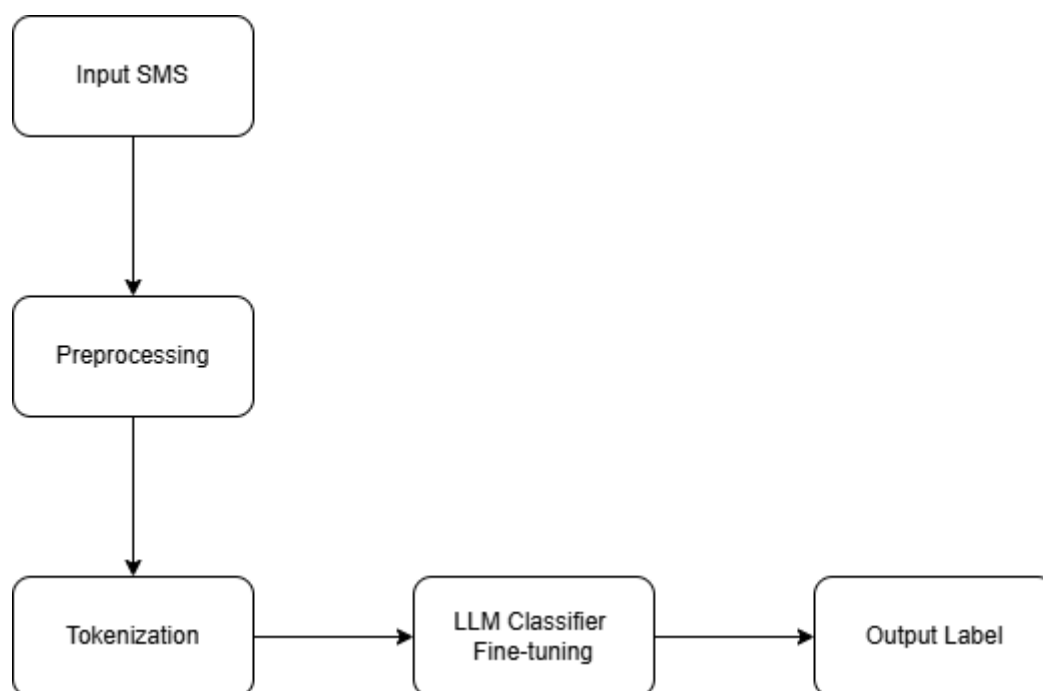
1.3 Các nghiên cứu và giải pháp liên quan:

Trong những năm gần đây, nhiều hướng nghiên cứu và giải pháp đã được triển khai nhằm giải quyết bài toán phân loại tin nhắn, sử dụng các kỹ thuật hiện đại trong xử lý ngôn ngữ tự nhiên:

1. Các phương pháp truyền thống:
 - Sử dụng kỹ thuật TF – IDF kết hợp với mô hình học máy như Naïve Bayes, SVM, Random Forest.
 - Ưu điểm là đơn giản, dễ huấn luyện nhưng khả năng hiểu ngữ nghĩa hạn chế, đặc biệt là với tiếng Việt không dấu hoặc nội dung ngắn, rút gọn.
2. Các mô hình học sâu (Deep Learning):
 - Mạng nơ-ron hồi tiếp (RNN, LSTM, GRU) được áp dụng với embedding đầu vào như Word2Vec hoặc FastText.
 - Ưu điểm là cho kết quả tốt hơn các phương pháp truyền thống, nhưng vẫn gặp khó khăn trong việc xử lý ngữ cảnh đa nghĩa.
3. Ứng dụng mô hình ngôn ngữ lớn (LLM):
 - BERT, RoBERTa, XLM-R và đặc biệt là PhoBERT – LLM được huấn luyện riêng cho tiếng Việt đã cho thấy kết quả vượt trội trong các tác vụ phân loại văn bản.
 - Đã có nhiều nghiên cứu tại Việt Nam (như VLSP – Vietnamese Language and Speech Processing) đã chứng minh rằng LLM giúp cải thiện F1-score từ 5 – 10% so với các mô hình trước đó.
 - Ngoài ra, LLM còn hỗ trợ fine-tuning với lượng dữ liệu nhỏ, giúp giảm chi phí huấn luyện và triển khai thực tế.
4. Các hệ thống thực tế:
 - Nhiều ứng dụng trong lĩnh vực tài chính, ngân hàng, chăm sóc khách hàng đã sử dụng LLM để lọc tin nhắn rác, hỗ trợ chatbot hoặc phân loại phản hồi khách hàng.
 - Một số nhà mạng tại Việt Nam cũng đã bước đầu tích hợp AI để nhận diện và chặn tin nhắn rác dựa trên phân tích nội dung.

1.4 Kiến trúc, thuật toán sử dụng trong mô hình:

1.4.1 Kiến trúc mô hình:



Hình 1. Kiến trúc mô hình

Các thành phần hệ thống:

- Tiếp nhận tin nhắn đầu vào từ bước sinh dữ liệu ngẫu nhiên (Input SMS)
- Tiền xử lý: Loại bỏ các ký tự đặc biệt nhưng giữ lại những từ có dấu, xử lý URL, xử lý các cách viết số điện thoại và tiền, xử lý khoảng trắng thừa và đưa về định dạng chuẩn unicode.
- Tách từ (tokenize): tách từ theo ngôn ngữ tiếng Việt sử dụng thư viện underthesea, loại bỏ stopwords.
- Embedding Extraction: sử dụng tokenizer và embedding của mô hình đã pre-trained để chuyển token thành vecto ngữ cảnh.
- LLM Classifier: Fine-tune một LLM trên bộ dữ liệu tin nhắn đã gán nhãn. Mô hình đưa vecto embedding vào, xác suất ra cho từng nhãn.
- Output: Trả về nhãn cho hệ thống, các tham số đánh giá mô hình.

1.4.2 Thuật toán, thư viện sử dụng:

Xử lý dữ liệu:

- Sinh dữ liệu sử dụng thư viện random và calendar cho xử lý ngày tháng.
- Thư viện Underthesea dùng để hỗ trợ xử lý ngôn ngữ tự nhiên (NLP) tiếng Việt. Trong đề tài này, thư viện được sử dụng để tách từ (tokenize) hỗ trợ loại bỏ các stopwords.

- Thư viện re (regular expression) dùng cho tiền xử lý dữ liệu.

Vẽ biểu đồ, thống kê:

- Thư viện matplotlib được sử dụng để vẽ biểu đồ trực quan hoá dữ liệu, vận dụng cho việc thống kê số từ trong tin nhắn spam và ham.
- Thư viện seaborn dùng để vẽ biểu đồ phân bố tin nhắn spam và ham.

Chia tập dữ liệu, train và đánh giá mô hình:

- Thư viện transformers để làm việc với các mô hình ngôn ngữ lớn, là thư viện chính để làm việc với phoBERT.
- Các thư viện dùng để tối ưu mô hình (optimize) như AdamW.
- Scikit-learn: thư viện dùng cho học máy, cung cấp công cụ cho bài toán phân loại (classification) và cung cấp các tham số để đánh giá mô hình.
 - Accuracy score: tỉ lệ mô hình dự đoán đúng trên tổng số mẫu.

$$Accuracy = \frac{\text{Số lượng dự đoán đúng}}{\text{Tổng số mẫu}}$$

- Precision score: độ chính xác của mô hình trong việc dự đoán đúng nhãn spam (TP).

$$Precision = \frac{TP}{(TP + FP)}$$

- Recall score: tỷ lệ phát hiện được trong các trường hợp nhãn là spam.

$$Recall = \frac{TP}{(TP + FN)}$$

- F1 score: trung bình điều hoà giữa precision và recall.

$$f1\ score = 2 * \frac{(precision * recall)}{precision + recall}$$

- auc (area under curve) score: thước đo cho khả năng phân biệt của các lớp.
 - 0.5: mô hình dự đoán ngẫu nhiên
 - 1.0: mô hình dự đoán hoàn hảo.
- Confusion matrix: ma trận thể hiện số lượng dự đoán đúng / sai từng nhãn.

- Các thành phần:

- TP: True Positive – dự đoán đúng nhãn spam.

- TN: True Negative – dự đoán đúng nhãn ham.
- FP: False Positive – dự đoán nhầm nhãn ham thành spam.
- FN: False Negative – dự đoán nhầm nhãn spam thành ham.

Mô hình PhoBERT cho bài toán phân loại tin nhắn rác:

- Là mô hình ngôn ngữ tiếng Việt được huấn luyện theo kiến trúc RoBERTa – biến thể cải tiến của BERT, được coi là một trong những mô hình hiệu quả nhất cho tác vụ xử lý ngôn ngữ tự nhiên (NLP) tiếng Việt như phân loại tin nhắn
- Ưu điểm:
 - Được huấn luyện trên tập dữ liệu lớn: PhoBERT được huấn luyện trên hơn 20GB dữ liệu văn bản tiếng Việt.
 - Kiến trúc mạnh mẽ: Dựa trên RoBERTa, vốn cải thiện đáng kể so với BERT về tốc độ huấn luyện và độ chính xác.
 - Tối ưu cho tiếng Việt: Không cần xử lý đặc biệt như chuyển dấu hoặc thay đổi ngôn ngữ trung gian.
 - Dễ tích hợp với Hugging Face Transformers.

CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ MÔ HÌNH

2.1 Khái quát:

Mô hình phân loại tin nhắn thực hiện nạp đầu vào là file .csv gồm 2 cột là tin nhắn gốc được sinh ra từ hàm random và 1 cột gán nhãn cho dữ liệu (spam / ham). Sau đó tập dữ liệu sẽ được tiền xử lý nhằm loại bỏ các từ dư thừa không mang ngữ nghĩa (stopwords), xử lý định dạng các số điện thoại, loại bỏ các ký tự đặc biệt. Sau đó, dữ liệu sạch được tokenize và đưa vào mô hình phoBERT để fine-tuning và trả ra kết quả phân loại trong tập kiểm tra.

2.2 Thiết kế và cài đặt mô hình:

2.2.1 Xây dựng tập dữ liệu (Dataset):

- a. Mục tiêu: thực hiện sinh bộ dữ liệu (Dataset) khoảng 10000 mẫu tin nhắn với tỉ lệ spam là 30% và ham là 70%. Ngoài việc sinh dữ liệu theo các mẫu template có sẵn của tin nhắn, hàm đã được cải tiến để tạo sự đa dạng trong dữ liệu, tránh bộ dữ liệu quá đơn giản. Một số phương thức được thêm vào hàm sinh dữ liệu bao gồm:
 - Synonym replacement: thay các từ đồng nghĩa tạo các cách diễn đạt khác nhau cho tin nhắn.
 - Inject typo: tạo các lỗi chính tả đơn giản bằng cách xoá dấu câu và hoán vị hai ký tự khác nhau.
 - Random casing: ngẫu nhiên viết hoa và viết thường tin nhắn, tạo độ khó cho quá trình tiền xử lý và mô hình học.
 - Random suffix: ngẫu nhiên chèn thêm các ký tự, emoji, ở cuối tin nhắn.
- b. Các hàm chính:

Hàm synonym_replacement thay thế các từ đồng nghĩa gồm các thành phần chính sau:

- Khởi tạo danh sách các từ đồng nghĩa: danh sách các từ đồng nghĩa được tạo dựa trên các mẫu tin nhắn sử dụng các từ thường xuất hiện và từ đồng nghĩa của nó.
- Hàm thay thế: thay thế các từ đồng nghĩa được chọn ngẫu nhiên trong danh sách với xác suất prob.

```

# synonym replacement
SYNONYMS = {
    "giảm giá": ["sale", "hạ giá", "xả kho", "deal hot"],
    "khuyến mãi": ["promo", "ưu đãi", "ưu đãi sốc"],
    "nhận quà": ["nhận gift", "nhận quà tặng"],
    "liên hệ": ["LH", "call ngay", "liên lạc"],
    "đăng ký": ["đk", "ghi danh", "tham gia"],
    "quà": ["gift", "phần quà", "quà tặng"],
    "voucher": ["mã giảm", "coupon", "mã ưu đãi"],
    "freeship": ["miễn phí vận chuyển", "freeship toàn quốc"]
}

def synonym_replace(text, prob=0.2):
    for src, tgt_list in SYNONYMS.items():
        if random.random() < prob and src in text:
            text = text.replace(src, random.choice(tgt_list))
    return text

```

Hình 2. Hàm synonym replacement

Hàm `random_date` sinh ngẫu nhiên các ngày/tháng/năm trong khoảng từ năm 2004 đến năm 2025 theo định dạng dd/mm/yyyy.

```

# random date
def random_date(start_year=2004, end_year=2025) -> str:
    year = random.randint(start_year, end_year)
    month = random.randint(1, 12)
    _, max_day = calendar.monthrange(year, month)
    day = random.randint(1, max_day)
    return f"{day:02d}/{month:02d}/{year}"

```

Hình 3. Hàm random date sinh ngày ngẫu nhiên

Hàm `inject_typo` thực hiện loại bỏ dấu câu và hoán vị hai ký tự ngẫu nhiên với xác suất `prob`

```

# inject typo
def inject_typo(text, prob):
    # Loại bỏ dấu với xác suất prob
    if random.random() < prob:
        text = unicodedata.normalize('NFD', text)
        text = re.sub(r'[\u0300-\u036f]', '', text)
    # Hoán vị hai ký tự (typo) với xác suất prob
    if random.random() < prob and len(text) > 1:
        idx = random.randint(0, len(text)-2)
        lst = list(text)
        lst[idx], lst[idx+1] = lst[idx+1], lst[idx]
        text = ''.join(lst)
    return text

```

Hình 4. Hàm thêm các lỗi chính tả

Hàm random casing thay đổi tỉ lệ viết hoa viết thường theo xác suất:

- 0 – 20% là chữ viết thường.
- 20 – 40% là chữ viết hoa.
- 60% còn lại sẽ giữ nguyên văn bản ban đầu.

```

# Hàm random_casing
def random_casing(text):
    r = random.random()
    if r < 0.2:
        return text.upper()
    elif r < 0.4:
        return text.lower()
    else:
        return text

```

Hình 5. Hàm random casing

Hàm random suffix thực hiện chèn các biểu tượng hay ký tự kết thúc thường thấy trong tin nhắn văn bản SMS theo tỉ lệ nhỏ hơn 20%.

```
# Hàm random_suffix
def random_suffix(text):
    if random.random() < 0.2:
        suffix = random.choice(["!!!", "...", "?!", " 😊", " 🥰",
                                "~", "^^", "nha", "ạ", "hihi", "...",
                                ":)", "UwU", "T-T", "*OwO", ":D", "xD",
                                "~!", "@@", "lol", ":("])
        return text + suffix
    return text
```

Hình 6. Hàm random suffix

Template sinh dữ liệu tin nhắn spam:

```
# Các mẫu cho tin nhắn spam
URL_LIST = [
    "http://example.com", "https://chiase.mobifone.vn", "https://smarttravel.vn",
    "https://fpt-voice.net", "https://promo.now.vn/deal", "http://shop.example.com/giangia"
]
PHONE_LIST = [
    "0930451223", "093.045.1223", "093-045-1223", "0975551945", "097.555.1945", "097-555-1945"
]
MONEY_LIST = [
    "100k", "200k", "500k", "1.000.000đ", "50.000vnd", "100000000đ",
    "2.500.000vnd", "50000đ", "10K", "20k", "375.000vnd", "3 triệu", "5Tr"
]
EMOJI_LIST = ["🎁", "🔥", "💎", "✅", "!", "☀️", "📺", "👉"]
VOUCHER_LIST = ["10%", "20%", "30%", "50%", "100k", "200k"]
PRODUCT_LIST = [
    "điện thoại", "máy tính", "laptop", "thực phẩm chức năng",
    "kem trắng da", "serum trị nám", "kem trị mụn", "collagen",
    "thuốc đông y", "đồng hồ", "giày dép", "quần áo", "nước hoa"
]
REALESTATE_LIST = ["đất nền", "shophouse", "căn hộ", "khu nghỉ dưỡng", "chung cư", "khu đô thị"]
OFFER_LIST = ["ưu đãi", "khuyến mãi", "flash sale", "deal hot", "tri ân khách hàng"]
```

Hình 7. Giá trị thay thế cho tin nhắn spam

```
# Hàm sinh tin nhắn spam đa dạng
def generate_spam_message():
    spam_templates = [
        "- Trong ngày {date}, ABC chính thức ra mắt sản phẩm. với nhiều tính năng nổi bật cùng mức giá {offer} hấp dẫn. Xem chi tiết tại {url} {emoji}",
        "Nhanh tay nhận quà tặng {emoji}, click ngay {url}",
        "Mừng lễ 30/4, giảm đồng loạt 10% cho quý khách sử dụng các sản phẩm {product}. Thời gian áp dụng kể từ {date}. Xem chi tiết tại {url}.",
        "HÌNH NHƯ BẠN ĐANG THIẾU GÌ ĐÓ? MỖI NGÀY {url} để nhận được {offer} GIẢM GIÁ KHÔNG GIỚI HẠN! Chi tiết truy cập {url}, Liên hệ {phone}",
        "Chào anh/chị, xin chia sẻ với chị thông tin về khu chung cư cao cấp {real_estate}. {real_estate} có vị trí nằm ngay trung tâm thành phố, đang được mở bán với lãi",
        "Nhân dịp sinh nhật bạn, Boshop tặng bạn voucher {voucher} khi mua hàng vào ngày sinh nhật. Chúc bạn có 1 ngày sinh nhật vui vẻ! Chi tiết LH: {phone}",
        "- XYZ cho ra đời sản phẩm {product} mới với nhiều tính năng vượt trội, độc lạ. Đăng ký ngay tại {url} để trở thành người đầu tiên nhận nhiều {offer} hấp dẫn!",
        "Giảm giá sốc {voucher}, ưu đãi cực lớn chỉ trong hôm nay tại {url}! {emoji}",
        "Khu biệt thự ven biển {real_estate} - resort mở bán vào ngày xx với lãi suất vay vốn chỉ 0%. Còn chần chờ gì nữa mà không nhanh tay sở hữu cho mình một căn hộ vip",
        "Thời trang XYZ đang có {offer} {voucher} tất cả các sản phẩm tại cửa hàng từ ngày {date}. Để biết thêm thông tin chi tiết LH {phone}",
        "- Lễ 30/4 - ĐỒNG GIÁ TẤT CẢ SP {product} chỉ với {money}. Trong ngày {date} freeship online. Xem chi tiết tại {url}.",
        "- Mừng lễ 2/9, eSMS giảm đồng loạt {voucher} cho mọi doanh nghiệp sử dụng các sản phẩm {product} từ ngày {date}. Xem chi tiết tại {url}.",
        "Bạn đã trúng thưởng {prize} phần quà, liên hệ {phone} để nhận! {emoji}",
        "Chỉ với {money} bạn có thể sở hữu {product}, đừng bỏ lỡ! {emoji}",
        "Khuyến mãi hot, mua 1 tặng 1, nhanh tay đặt hàng ngay! {emoji}",
        "Cơ hội có 1-0-2, đăng ký ngay để nhận {offer}! {emoji}",
        "Nhận ngay {money} khi giới thiệu bạn mới đăng ký! {emoji}",
        "Tặng bạn mã {voucher} khi đặt món {emoji}",
        "TRÁI NGHIỆM LIỀN TAY, NHẬN NGAY QUÀ KHỦNG, cơ hội trúng tới {money} tiền mặt và nhiều phần quà hấp dẫn khi tham gia chương trình tại {url}. {emoji}",
        "VÔ VẠN ƯU ĐÃI, chỉ với 1 phút đăng ký, nhận ngay hàng loạt voucher {voucher}. Số lượng có hạn, nhanh tay tải app {url} để đăng ký ngay! {emoji}",
    ]
```

Hình 8. Mẫu tin nhắn spam

Sau đó gán giá trị ngẫu nhiên trên cho các biến trong template và thực hiện thay các từ đồng nghĩa, thêm lỗi chính tả, viết hoa viết thường và thêm từ kết thúc làm đa dạng dữ liệu. (Hình

```

# Giá trị ngẫu nhiên cho biến
url          = random.choice(URL_LIST)
prize        = random.choice(["100", "200", "300", "500", "1.000", "2.000"])
phone        = random.choice(PHONE_LIST)
money        = random.choice(MONEY_LIST)
product      = random.choice(PRODUCT_LIST)
real_estate  = random.choice(REALESTATE_LIST)
offer        = random.choice(OFFER_LIST)
voucher      = random.choice(VOUCHER_LIST)
date         = random_date(2004, 2025)
emoji        = random.choice(EMOJI_LIST)

template = random.choice(spam_templates)
message = template.format(
    url=url, prize=prize, phone=phone, money=money,
    product=product, real_estate=real_estate,
    offer=offer, voucher=voucher, date=date, emoji=emoji
)
message = synonym_replace(message, prob=0.3)
message = inject_typo(message, prob=0.2)
message = random_casing(message)
message = random_suffix(message)

return message

```

Hình 9. Sinh tin nhắn spam

Template sinh dữ liệu tin nhắn ham:

```

# Hàm sinh tin nhắn ham
def generate_ham_message():
    ham_templates = [
        "Mai m thi rồi, t chúc m may mắn nhaaaa {emoji}",
        "Ôn thi hk? M chép đề cương giúp t với :(",
        "Tối muộn tí nha, kẹt xe vl!!!",
        "Hôm qua coi phim hài đã man. Vừa cười vừa khóc lun :)",
        "Tao đi trc nha. Mày nhớ mang bài tập theo đấy. Ko là toi cả đám á.",
        "Đi học hơm? Trời mưa z mà bắt đi học chán ghê ",
        "Lúc nào rảnh ghé qua ăn cơm nha, mẹ làm món con thích nè {emoji}",
        "Ba chuyển tiền học rồi đó nghen. Kiểm tra xem vô tk chưa nha con.",
        "Con ơi mai nhớ ghé chợ mua 1 bó rau muống với 2 quả cà chua nha. Mẹ dặn đấy nhớ đó.",
        "Mã OTP của bạn là {otp}. Có hiệu lực trong 5 phút. Tuyệt đối không chia sẻ mã này.",
        "Đơn hàng #{order_id} đã giao thành công. Cảm ơn bạn đã mua sắm cùng Shopee!",
        "Quý khách vừa thanh toán {money} tại Circle K lúc {time}. Số dư: {balance}.",
        "Các em lưu ý nộp bài giữa kỳ đúng hạn. Thầy ko nhận file sau 23h ngày 15/05.",
        "Cả lớp note lại nhé: Sáng mai học bù môn Triết, 7h30 tại D205 nha.",
        "Phòng đào tạo nhắc bạn kiểm tra lại lịch học tuần này trên cổng LMS. Có thay đổi môn CNPM.",
        "Hehe tối đi nhậu ko? Lâu ko gặp ae, alo t cái nha.",
        "Chiều nay có tiết KTLT đó, đừng skip như hôm trc nha.",
        "M nhớ nộp bài t.a cô Hồng hnay trc 23h59 nha. Tao ms nộp xog, deadline đi vcl.",
        "Trưa nay ăn gì zq? Tao chán cơm vp vl.",
        "Ê mai có đi cf vs nhóm k? T 4h xong việc, qua đón nhé.",
        "Hôm nay trời đẹp quá, mình đi dạo công viên nhé. {emoji}",
        "Bạn có muốn cùng mình ăn trưa không?",
        "Mình đã xem bộ phim hay vừa rồi, bạn nên đi xem nhé.",
        "Hẹn gặp lại bạn vào cuối tuần này.",
        "Mình vừa đọc một cuốn sách thú vị, chia sẻ với bạn.",
        "Cảm ơn bạn đã giúp đỡ.",
        "Sắp tới mình có kế hoạch du lịch, bạn có tham gia không?",
        "Mọi thứ đều ổn, cảm ơn bạn đã hỏi thăm.",
        "Mình sẽ liên hệ với bạn sau khi có thông tin thêm.",
        "Hãy giữ liên lạc và chúc bạn một ngày tốt lành! {emoji}",
    ]

```

Hình 10. Mẫu tin nhắn ham

Gán các giá trị ngẫu nhiên cho tin nhắn ham và tạo đa dạng dữ liệu bằng cách tương tự như các tin nhắn spam.


```

template = random.choice(ham_templates)

otp      = random.randint(100000, 999999)
order_id = random.randint(1000000, 9999999)
money    = random.choice(MONEY_LIST)
time     = f"{random.randint(0,23):02d}:{random.randint(0,59):02d}"
balance  = f"{random.randint(0,9)}.{random.randint(0,999):03d}.{random.randint(0,999):03d}đ"
emoji    = random.choice(EMOJI_LIST)

message = template.format(
    otp=otp, order_id=order_id, money=money,
    time=time, balance=balance, emoji=emoji
)
message = synonym_replace(message, prob=0.2)
message = inject_typo(message, prob=0.1)
message = random_casing(message)
message = random_suffix(message)

return message

```

Hình 11. Thay thế giá trị cho tin nhắn ham

Hàm sinh tập dữ liệu tổng (10000 tin nhắn) với tỉ lệ spam và ham là 30 – 70.

```

# Hàm tổng sinh dataset
def generate_dataset(num_messages=10000, spam_ratio=0.3):
    data = []
    num_spam = int(num_messages * spam_ratio)
    num_ham = num_messages - num_spam
    for _ in range(num_spam):
        data.append({"message": generate_spam_message(), "label": "spam"})
    for _ in range(num_ham):
        data.append({"message": generate_ham_message(), "label": "ham"})
    random.shuffle(data)
    return data

if __name__ == "__main__":
    import pandas as pd
    dataset = generate_dataset(10000, spam_ratio=0.3)
    df_out = pd.DataFrame(dataset)
    df_out.to_csv("vietnamese_sms_dataset.csv", index=False, encoding="utf-8-sig")
    print("Dataset đã được tạo và lưu vào file 'vietnamese_sms_dataset.csv'")
    print(df_out.head(20))

```

Hình 12. Hàm tạo tập dữ liệu

Các bước mô tả bước tạo dữ liệu:


- Khởi tạo một mảng data[] chứa các tin nhắn được tạo ngẫu nhiên.
- Khởi tạo số lượng tin nhắn spam và ham.
 - $num_{spam} = num_{messages} * spam_{ratio}$: tỉ lệ ratio được đặt là 0.3 với tổng số lượng tin nhắn là 10000 tin. Số lượng tin nhắn spam sẽ là 3000 tin.

- $num_{ham} = num_{messages} - num_{spam}$: số lượng tin nhắn bình thường là 7000 tin.
- Gọi hàm `generate_spam_message()` khởi tạo tin nhắn spam, đồng thời gán nhãn “spam” và thêm vào mảng.
- Gọi hàm `generate_ham_message()` khởi tạo tin nhắn ham, đồng thời gán nhãn “ham” và thêm vào mảng.
- Xáo trộn các tin nhắn: hàm `shuffle()` thực hiện xáo trộn thứ tự các tin nhắn trong mảng `data`.

Hàm main chính:

- Gọi hàm khởi tạo `generate_dataset`.
- Chuyển dataset thành DataFrame sử dụng thư viện `pandas`. DataFrame là cấu trúc dữ liệu dạng bảng giống như Excel gồm 2 cột chính là “message” và “label”.
- Lưu DataFrame thành file csv là “`vietnamese_sms_dataset.csv`”.
 - `encoding = “utf-8-sig”`: đảm bảo tiếng Việt hiển thị đúng trong file Excel.
 - `index = False`: không ghi chỉ số dòng vào file.

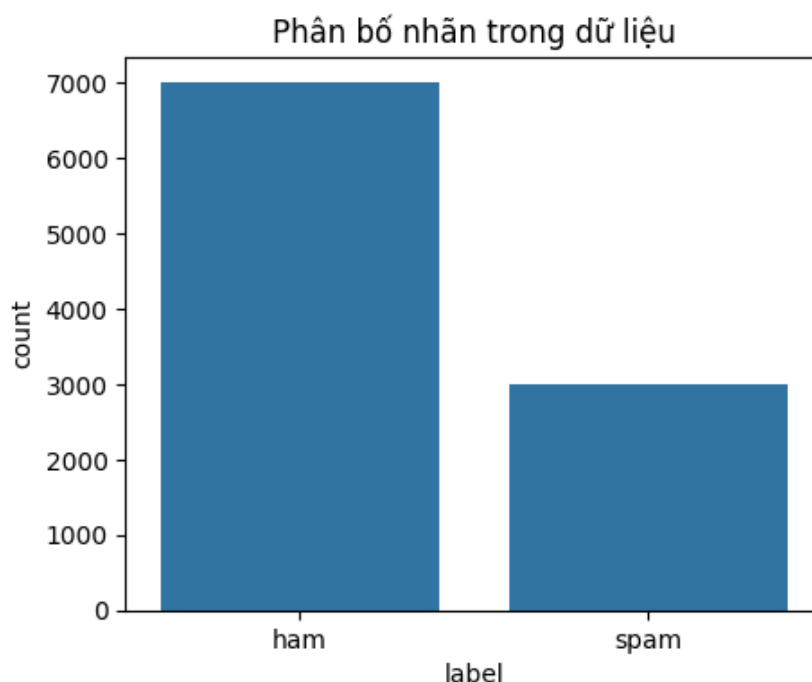
Dữ liệu gốc (raw text) sau khi tạo và lưu vào trong file csv dạng bảng sẽ có dạng như Hình 13.:

 Dataset đã được tạo và lưu vào file 'vietnamese_sms_dataset.csv'

	message	label
0	M nhớ nộp bài t.a cô Hồng hnay trc 23h5 9nha. ...	ham
1	Chào anh/chị, xin chia sẻ với chị thông tin về...	spam
2	giảm giá sốc 200k, ưu đãi cực lớn chỉ trong hô...	spam
3	Chào anh/chị, xin chia sẻ với chị thông tin về...	spam
4	NHANH TAY NHẬN QUÀ TẶNG 📺, CLICK NGAY HTTP://S...	spam
5	Con ơi mai nhớ ghé chợ mua 1 bó rau muống với ...	ham
6	TRƯA NAY ẮN GÌ ZƠ? TAO CHẤN CƠM VP VL.	ham
7	con ơi mai nhớ ghé chợ mua 1 bó rau muống với ...	ham
8	SAP TOI MINH CO KE HOACH DU LICH, BAN CO THAM ...	ham
9	Cả lớp note lại nhé: Sáng mai học bù môn Triết...	ham
10	sức khỏe của bạn: huyết áp 120/80, nhịp tim 75...	ham
11	Nhan ngay 10K khi giới thiệu bạn mới đang ky ! 📺	spam
12	Mình đã xem bộ phim hay vừa rồi, bạn nên đi xe...	ham
13	phòng đào tạo nhắc bạn kiểm tra lại lịch học t...	ham
14	Hehe tối đi nhậu ko? Lâu ko gặp ae, alo t cái ...	ham
15	Mẹ nhắc: hôm nay 5h có họp trực tuyến, nhớ tha...	ham
16	ĐƠN HÀNG #3013778 ĐÃ GIAO THÀNH CÔNG. CẢM ƠN B...	ham
17	trai nghiêm liên tay, nhan ngay qua khung, co ...	spam
18	Chiều nay có tiết KTLT đó, đừng skip như hôm r...	ham
19	Gia nha đang co xu huong gia tang nhanh chóng,...	ham

Hình 13. Dữ liệu gốc

Biểu đồ dưới đây thể hiện phân bố số lượng tin nhắn ham và spam trong quá trình sinh dữ liệu.



Hình 14. Biểu đồ phân bố nhãn

Với phân bố 70% spam và 30% ham trong tập dữ liệu phân loại tin nhắn là một lựa chọn cân bằng giữa thực tế và khả năng học của mô hình. Điều này giúp mô hình tránh học theo hướng đoán quá nghiêng về một phía, tránh mất cân bằng dữ liệu mà vẫn đảm bảo tính thực tế.

2.2.2 Tiền xử lý (Preprocessing):

- a. Mục đích: làm sạch dữ liệu, loại bỏ các từ ngữ dư thừa và định dạng lại các dữ liệu thường thấy trong tin nhắn. Một số phương pháp tiền xử lý được sử dụng trong đề tài này bao gồm:
 - Chuyển các tin nhắn về lowercase.
 - Thay thế URL, số điện thoại, date, tiền và các emoji về dạng <token>.
 - Loại bỏ các ký tự đặc biệt nhưng giữ lại các chữ, số và dấu tiếng Việt.
 - Xóa các khoảng trắng thừa.
 - Chuẩn hoá unicode về dạng NFC.
 - Tách từ và xóa các stopwords từ list stopwords có sẵn.
 - Áp dụng văn bản đã làm sạch lên DataFrame và lưu dưới tên `cleaned_message`
- b. Các hàm chính:

Việc loại bỏ stopwords (từ dừng) trong tin nhắn tiếng Việt là một bước tiền xử lý quan trọng trong bài toán phân loại văn bản.

- Giảm nhiễu dữ liệu: những từ dừng không mang nhiều thông tin về mặt ngữ nghĩa để phân biệt giữa spam và ham. Giữ lại sẽ gây loãng đặc trưng quan trọng của văn bản.
- Giảm số lượng từ giúp giảm số chiều của vector khi cho vào mô hình học, tăng tốc độ huấn luyện và tiết kiệm tài nguyên tính toán.
- Tăng độ tập trung vào các từ khoá có ý nghĩa phân loại.

Trước khi bắt đầu tiền xử lý, mảng `vietnamese_stopwords[]` được tạo lưu các từ dừng trong tiếng Việt.

```
# Stopwords
vietnamese_stopwords = set([
    'và', 'của', 'cho', 'là', 'để', 'trong', 'có', 'không', 'được', 'tại', 'những',
    'với', 'này', 'về', 'như', 'đó', 'các', 'một', 'từ', 'bạn', 'tôi', 'làm', 'việc',
    'theo', 'năm', 'khi', 'sẽ', 'đã', 'lên', 'đây', 'nhưng', 'do', 'vào', 'ra', 'nếu',
    'thì', 'nên', 'quá', 'lại', 'đến', 'đi', 'còn', 'chỉ', 'mà', 'vì', 'cũng', 'ngoài',
    'rồi', 'hay', 'nào', 'sau', 'trên', 'dưới', 'đang', 'mới', 'bởi', 'ở', 'phải', 'vẫn',
    'luôn', 'làm', 'hơn', 'thế', 'đều', 'vậy', 'rất', 'cần', 'lúc', 'bị', 'sao', 'ai',
    'mình', 'chúng', 'cùng', 'đôi', 'trước', 'nhiều', 'cứ', 'để', 'thật', 'chứ', 'suốt',
    'rằng', 'nhé', 'vừa', 'đây', 'kia', 'nhi', 'nhá', 'ạ', 'à', 'á', 'ấy', 'ày',
    'anh', 'chị', 'em', 'cô', 'chú', 'ông', 'bà', 'họ', 'ta', 'hôm',
    'nay', 'mai', 'gi', 'đó', 'ừ', 'ơ', 'ừm', 'hừm', 'huhu', 'híc', 'ừh',
    'k', 'ok', 'oke', 'đc', 'đc rồi', 'rồi', 'đâu', 'hà', 'hơ', 'khôg', 'khong',
    'úi', 'ôi', 'ồ', 'ơi', 'chứ', 'chỉ', 'chị', 'kệ', 'hay', 'thôi', 'nữa',
    'tiếp', 'tiếp theo', 'nhé', 'nè', 'nà', 'nha', 'nữa'
])
```

Hình 15. Danh sách các stopwords

Hàm `clean_text()` để tiền xử lý dữ liệu:

- `text.lower()`: chuyển tất cả văn bản về chữ thường.
- Chuẩn hoá URL: mọi đường link dạng `http/https/www` đều được thay bằng `<url>` để mô hình nhận biết sự hiện diện của đường link thay vì từng link cụ thể.
- Số điện thoại: Tất cả các số điện thoại thường gặp (bao gồm định dạng `xxx-xxxx-xxx` hay `xxx.xxxx.xxx`) đều được thay bằng `<phone>`.
- Các thông tin về ngày tháng năm được chuyển về `<date>`.
- Các thông tin liên quan đến tiền: Các biểu thức tiền tệ như `500.000đ`, `200k`, `3.500vnd` đều được thay bằng `<money>`.
- Các emoji cảm xúc trong tin nhắn thường thấy được thay bằng `<emoji>`
- Loại bỏ các ký tự đặc biệt nhưng giữ lại các chữ cái có dấu của tiếng Việt.
- Loại bỏ các khoảng trắng thừa, giữ văn bản sạch sẽ.
- Chuẩn hoá mã Unicode: chuẩn hoá ký tự văn bản tiếng Việt tránh lỗi phân rã với các chữ cái có dấu.

- Tách từ và loại bỏ stopwords: Dùng hàm `word_tokenize()` trong thư viện `underthesea` để tách từ tiếng Việt có dấu sau đó loại bỏ những từ nằm trong danh sách stopwords (từ dùng) nhưng vẫn giữ lại các thẻ đặc biệt ở trên.
- Ghép lại các token và trả về chuỗi văn bản sạch.

[illegible]

Hình 16. Hàm clean text

Kết quả tiền xử lý được thể hiện trong Hình 17.

Văn bản chưa xử lý: Các em lưu ý nộp bài giữa kỳ đúng hạn. Thầy ko nhận file sau 23h ngày 15/05.
Tokens: ['càc', 'lưu', 'ý', 'nộp', 'bài', 'giữa', 'kỳ', 'đúng', 'hạn', 'thầy', 'ko', 'nhận', 'file', '23', 'h', 'ngày', '15', '05']

Văn bản đã xử lý: các lưu ý nộp bài giữa kỳ đúng hạn thầy ko nhận file 23 h ngày 15 05

Văn bản chưa xử lý: Mình vừa đọc một cuốn sách thú vị, chia sẻ với bạn.
Tokens: ['đọc', 'cuốn', 'sách', 'thú', 'vị', 'chia', 'sẻ']

Văn bản đã xử lý: đọc cuốn sách thú vị chia sẻ

Văn bản chưa xử lý: Chiều nay có tiết KTLT đó, đừng skip như hôm trc nha.
Tokens: ['chiều', 'tiết', 'ktlt', 'đừng', 'skip', 'trc']

Văn bản đã xử lý: chiều tiết ktlt đừng skip trc

Văn bản chưa xử lý: Con ơi mai nhớ ghé chợ mua 1 bó rau muống với 2 quả cà chua nha. Mẹ dặn đấy nhớ đó~
Tokens: ['con', 'nhớ', 'ghé', 'chợ', 'mua', '1', 'bó', 'rau', 'muống', '2', 'quả', 'cà', 'chua', 'mẹ', 'dặn', 'đấy', 'nhớ']

Văn bản đã xử lý: con nhớ ghé chợ mua 1 bó rau muống 2 quả cà chua mẹ dặn đấy nhớ

Văn bản chưa xử lý: PHÒNG ĐÀO TẠO NHẮC BẠN KIỂM TRA LẠI LỊCH HỌC TUẦN NÀY TRÊN CỐNG LMS. CÓ THAY ĐỔI MÔN CNPM.
Tokens: ['phòng', 'đào', 'tạo', 'nhắc', 'kiểm', 'tra', 'lịch', 'học', 'tuần', 'trên', 'cống', 'lms', 'thay', 'đổi', 'môn', 'cnpm']

Văn bản đã xử lý: phòng đào tạo nhắc kiểm tra lịch học tuần cống lms thay đổi môn cnpn

Văn bản chưa xử lý: LÚC NÀO RÀNH GHÉ QUÀ ĂN CƠM NHA, MẸ LÀM MÓN CON THÍCH NÈ 🍱!!
Tokens: ['lúc', 'nào', 'rảnh', 'ghé', 'qua', 'ăn', 'cơm', 'mẹ', 'món', 'con', 'thích', '< emoji']

Văn bản đã xử lý: lúc nào rảnh ghé qua ăn cơm mẹ món con thích < emoji

Văn bản chưa xử lý: Các em lưu ý nộp bài giữa kỳ đúng hạn. Thầy ko nhận file sau 23h ngày 15/05.
Tokens: ['lưu', 'ý', 'nộp', 'bài', 'giữa', 'kỳ', 'đúng', 'hạn', 'thầy', 'ko', 'nhận', 'file', '23', 'h', 'ngày', '15', '05']

Văn bản đã xử lý: lưu ý nộp bài giữa kỳ đúng hạn thầy ko nhận file 23 h ngày 15 05

Hình 17. Đầu ra tiền xử lý văn bản

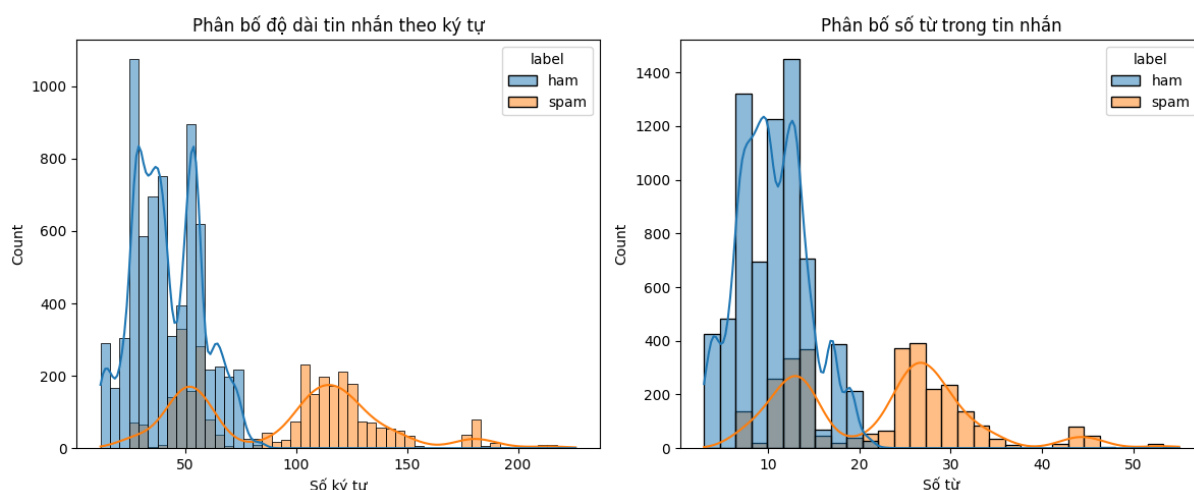
Sau khi tiền xử lý văn bản, ta nhận thấy được một vài đặc trưng của tin nhắn spam và ham như sau (Hình 18):

a. Phân bố độ dài tin nhắn theo ký tự:

- Tin nhắn ham (màu xanh dương): độ dài tập trung chủ yếu từ 30 đến 70 ký tự, với nhiều nhất là khoảng từ 40 đến 50 ký tự. Điều này cho thấy hầu hết tin nhắn bình thường ngắn gọn, nhỏ hơn 70 ký tự và súc tích.
- Tin nhắn spam (màu cam): phân bố rộng hơn, có hai “đỉnh” chính:
 - Một nhóm nhỏ từ 30 đến 40 ký tự (thường là các mẫu spam ngắn, giới thiệu nhanh).
 - Nhóm lớn hơn từ 80 đến 200 ký tự là các tin nhắn dài, lồng nhiều thông tin khuyến mãi, đường link, số điện thoại, sale, ...
- Mặc dù có sự trùng lặp giữa spam và ham trong khoảng từ 30 đến 40 ký tự, nhưng trung bình thì tin nhắn spam sẽ dài hơn ham → chiều dài có thể giúp mô hình phân biệt 2 lớp.

b. Phân bố số từ trong tin nhắn:

- Tin nhắn ham: số từ nhiều nhất trong khoảng từ 8 đến 12 từ, có thể lên đến 20 từ.
- Tin nhắn spam:
 - Nhóm nhỏ có từ 10 đến 15 từ nhưng đa phần nằm trong khoảng từ 20 đến 40 từ và có thể lên đến 50 từ.
- Từ đó cho thấy các tin nhắn spam không chỉ dài về ký tự mà thường chứa nhiều từ hơn → số từ vượt quá 20 từ thì khả năng cao là spam.



Hình 18. Biểu đồ đặc trưng

c. Thống kê các token (từ) xuất hiện nhiều nhất (Hình 15)

- Trong đây có một vài từ đặc biệt xuất hiện nhiều như là:
 - Emoji: xuất hiện 1675 lần.

- Money: xuất hiện 867 lần.
- Phone: xuất hiện 841 lần.
- URL: xuất hiện 1666 lần.
- Nhận thấy các đặc trưng spam như “phone” và “url” hay “money” xuất hiện nhiều lần cho thấy các tin nhắn spam thường xuất hiện số điện thoại / đường link khuyến mãi.

('ngay', 2831), ('chỉ', 1436), ('nhận', 1044), ('tiết', 962), ('phone', 931), ('url', 923), ('quà', 693), ('tặng', 671), ('san', 666), ('pham', 666), ('gia', 665), ('tai', 665), ('nha', 1919), ('t', 1277), ('học', 1176), ('m', 904), ('nay', 870), ('mai', 845), ('nhớ', 818), ('ko', 803), ('cả', 697), ('ngày', 671), ('nộp', 656), ('tao', 656), ('qua', 656)

Hình 19. Các từ xuất hiện nhiều nhất

2.2.3 Huấn luyện mô hình:

a. Chuẩn bị:

Mục đích chính: chuyển đổi chuỗi spam/ham sang dạng nhị phân 1/0 chuẩn bị cho mô hình học máy giúp việc tính toán hàm loss cũng như đo lường các metric.

```

▶ label_mapping = {"spam":1, "ham":0}
df['label'] = df['label'].map(label_mapping)
print(df['label'].head(20))

```

0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	1
13	1
14	0
15	0
16	1
17	0
18	0
19	0

Name: label, dtype: int64

Hình 20. Chuyển đổi label

b. Thiết lập:

- Gán hạt ngẫu nhiên (seed) cho các module random, numpy, torch để kết quả huấn luyện lặp lại được.

- Device tự động chọn GPU (nếu có CUDA) hoặc CPU, đảm bảo sẽ không crash trong quá trình.

```
# Setup
SEED = 42
random.seed(SEED)
np.random.seed(SEED)
torch.manual_seed(SEED)
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
```

Hình 21. Bước thiết lập

c. Định nghĩa dataset:

```
# Simple Dataset
class SMSDataset(Dataset):
    def __init__(self, texts, labels, tokenizer, max_length=128):
        self.texts = texts
        self.labels = labels
        self.tokenizer = tokenizer
        self.max_length = max_length

    def __len__(self):
        return len(self.texts)

    def __getitem__(self, idx):
        text = self.texts[idx]
        enc = self.tokenizer(
            text,
            padding='max_length',
            truncation=True,
            max_length=self.max_length,
            return_tensors='pt'
        )
        return {
            'input_ids': enc.input_ids.squeeze(),
            'attention_mask': enc.attention_mask.squeeze(),
            'labels': torch.tensor(self.labels[idx], dtype=torch.long)
        }
```

Hình 22. Định nghĩa dataset

- Hàm `__len__`: trả về số lượng mẫu,
- `__getitem__`: với mỗi chỉ số `idx`:
 - Lấy văn bản đã clean.
 - Tokenize với PhoBERT (padding, truncation về độ dài cố định).
 - Trả về:
 - `input_ids`: chỉ số token.
 - `attention_mask`: mask cho phần padding.
 - Labels: nhãn 0/1 dưới dạng LongTensor.

- Phương pháp tăng cường dữ liệu đơn giản (EDA – Easy Data Augmentation):
 - Mục đích: Tạo ra các tin nhắn mới từ dữ liệu ban đầu nhưng có thêm sự thay đổi nhỏ nhưng vẫn giữ nguyên ý nghĩa chính, không bị sai lệch nhãn. Từ đó, phương pháp này giúp tăng số lượng và sự đa dạng của dữ liệu huấn luyện giúp mô hình học được nhiều đặc điểm hơn.
 - Với các tập dữ liệu lớn (trên 5000) dataset thì xác suất phù hợp cho phương pháp EDA là 10% (theo Jason Wei, Kai Zou – *EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks*).

N_{train}	α	n_{aug}
500	0.05	16
2,000	0.05	8
5,000	0.1	4
More	0.1	4

Table 3: Recommended usage parameters.

Hình 23. Xác suất EDA

- Các phương pháp sử dụng:
 - Random deletion: Xóa từ ngẫu nhiên với xác suất 10%.
 - Random swap: Đổi vị trí ngẫu nhiên giữa hai từ trong câu với xác suất 10%.

d. Tải tokenizer và chia tập dữ liệu:

```
#Tải tokenizer của PhoBERT và chia tập dữ liệu.
tokenizer = AutoTokenizer.from_pretrained("vinai/phobert-base", use_fast=False)
train_df = df.sample(frac=0.9, random_state=SEED)
test_df = df.drop(train_df.index)
train_texts, train_labels = train_df["cleaned_message"].tolist(), train_df["label"].tolist()
test_texts, test_labels = test_df["cleaned_message"].tolist(), test_df["label"].tolist()
```

Hình 24. Tải tokenizer và chia tập dữ liệu

- Tải tokenizer cho PhoBERT.
- Chia DataFrame thành:
 - 90% làm tập train
 - 10% làm tập kiểm tra (test)
- Chuyển cột ‘cleaned_message’ và ‘label’ thành hai list.

- Tạo SMSDataset và DataLoader với batch size phù hợp, shuffle = True cho tập train để huấn luyện đa dạng.

e. Tạo các bộ nạp dữ liệu:

Mục đích: tạo các bộ nạp dữ liệu để phục vụ cho quá trình huấn luyện và đánh giá mô hình phân loại tin nhắn.

- SMSDataset: chuẩn bị dữ liệu đầu vào cho mô hình.
- augment = true: áp dụng kỹ thuật tăng cường dữ liệu (EDA) cho tập huấn luyện nhằm cải thiện khả năng tổng quát của mô hình.
- augment = false: không áp dụng kỹ thuật tăng cường cho tập test để đánh giá chính xác hiệu suất của mô hình.
- DataLoader: dùng để duyệt qua dữ liệu theo batch với tự động xáo trộn (shuffle):
 - batch_size:
 - train_loader: đặt batch_size = 16 sẽ giúp mô hình học kỹ hơn.
 - test_loader: đặt batch_size = 32 giúp mô hình đánh giá nhanh hơn.
 - shuffle = true: chỉ áp dụng cho tập huấn luyện, tránh tình trạng mô hình học theo thứ tự cố định.
 - num_workers = 2: sử dụng 2 tiến trình để load dữ liệu song song. Do các bước xử lý cũng như số lượng dataset có nên nếu để mặc định (num_workers = 0) thì việc tải dữ liệu sẽ diễn ra tuần tự, rất chậm)
 - pin_memory = True: tăng tốc độ truyền dữ liệu.

```
#DataLoaders
train_ds = SMSDataset(train_texts, train_labels, tokenizer, augment=True)
test_ds = SMSDataset(test_texts, test_labels, tokenizer, augment=False)
train_loader = DataLoader(train_ds, batch_size=16, shuffle=True, num_workers=2, pin_memory=True)
test_loader = DataLoader(test_ds, batch_size=32, shuffle=False, num_workers=2, pin_memory=True)
```

Hình 25. Tạo các bộ nạp dữ liệu

f. Tải mô hình và thiết lập fine-tuning:

Mục đích: Tải mô hình PhoBERT và thiết lập chiến lược fine-tuning có chọn lọc để huấn luyện mô hình phân loại văn bản theo cách tối ưu tài nguyên và hiệu suất.

- Tạo cấu hình mô hình (cfg):
 - Tải cấu hình gốc của vinai/phobert-base.
 - Num_labels = 1: dùng cho bài toán phân loại.
 - Hidden_dropout_prob = 0.3 và attention_probs_dropout_prob = 0.3: tăng dropout để giảm overfitting khi fine-tune.

- Tải mô hình vinai/phobert-base với cấu hình đã chỉnh.
- Đóng băng một phần mô hình (partially fine-tuning):
 - Mục tiêu: chỉ fine-tune các layer cuối (từ 8 trở đi), các layer trước đó sẽ đóng băng giúp ổn định mô hình khi dữ liệu không quá lớn và tránh vấn đề “quá hoàn hảo” của mô hình, tiết kiệm thời gian train.
- Chuyển toàn bộ mô hình sang device.

```
#Tải mô hình và thiết lập fine-tuning
cfg = AutoConfig.from_pretrained(
    "vinai/phobert-base",
    num_labels=1,
    hidden_dropout_prob=0.3,
    attention_probs_dropout_prob=0.3
)
model = AutoModelForSequenceClassification.from_pretrained("vinai/phobert-base", config=cfg)
for name, p in model.named_parameters():
    if name.startswith("roberta.encoder.layer"):
        layer_id = int(name.split(".")[3])
        p.requires_grad = layer_id >= 8
    else:
        p.requires_grad = True
model.to(device)
```

Hình 26. Tải mô hình và thiết lập fine-tuning

g. Cấu hình hàm mất mát:

Mục đích: Xây dựng hàm mất mát BCEWithLogitsLoss có trọng số lớp để cân bằng tác động giữa mẫu “spam” và “ham” trong mô hình phân loại nhị phân giúp mô hình không thiên lệch về lớp chiếm đa số. Do dữ liệu thường bị mất cân bằng (70% ham và 30% spam) nên nếu không xử lý thì mô hình sẽ học cách đoán ham để đạt được accuracy cao.

- pos_weight: là trọng số áp dụng cho lớp positive (spam=1) trong hàm BCEWithLogitsLoss, dùng để bù đắp khi số mẫu spam quá ít → nếu không đặt mô hình thì sẽ có xu hướng thiên lệch về ham.

```
#Class-weighted loss để cân bằng
spam_count = sum(train_labels)
ham_count = len(train_labels) - spam_count
pos_weight = torch.tensor(ham_count/spam_count, device=device)
loss_fn = BCEWithLogitsLoss(pos_weight=pos_weight)
```

Hình 27. Tạo hàm mất mát

h. Tối ưu quá trình huấn luyện:

Mục đích: thiết lập quá trình tối ưu hoá cho huấn luyện mô hình bằng cách chọn:

- Tạo optimizer bằng thư viện AdamW và tối ưu các layer đang được mở.
 - Đặt `learning_rate = 2e-5`.
 - `weight-decay = 0.01` để chống overfitting.
 - `eps = 1e-8`: tránh chia cho 0 trong tính toán Adam.
- Tính tổng số bước huấn luyện:

$$Total_{steps} = num_{batches} * 3$$

- Tạo scheduler để điều chỉnh learning rate.

```
#6. Thiết lập tối ưu huấn luyện.
optimizer = AdamW(
    filter(lambda p: p.requires_grad, model.parameters()),
    lr=2e-5,
    weight_decay=0.01,
    eps=1e-8
)
total_steps = len(train_loader) * 3
scheduler = get_linear_schedule_with_warmup(optimizer, num_warmup_steps=0, num_training_steps=total_steps)
```

Hình 28. Tối ưu quá trình huấn luyện

i. Vòng huấn luyện (training loop):

Mục đích: huấn luyện mô hình phân loại, tính toán `train_loss` và `val_loss` và dừng sớm (early stopping) nếu `val_loss` không cải thiện.

- Khởi tạo trước vòng lặp:
 - `Best_val_loss`: dùng để lưu loss.
- `Model.train()`: bật chế độ training với `train_loss` khởi tạo ban đầu = 0.
- Với mỗi batch:
 - `zero_grad()`: reset gradient từ bước trước.
 - `outs = model()`: chuyển toàn bộ input tokens qua mô hình, lấy embedding CLS và qua lớp classification layer để ra logits. Tính loss so với nhãn đã truyền.
 - logits: định dạng output.
 - loss: tính loss
 - `backward()`: lan truyền ngược gradient.
 - `clip_grad_norm`: cắt gradient nếu quá lớn, chống exploding gradient.
 - `optimizer.step()`: cập nhật trọng số.
 - `scheduler.step()`: cập nhật learning rate.
 - Tính train loss trung bình.
- Validation:
 - Không cập nhật trọng số.

- Tắt dropout để mô hình ổn định hơn.
- Tính val loss trung bình.
- Lưu lại train_losses và val_losses để xem tiến trình hội tụ.

```
#Training loop
best_val_loss = math.inf
for epoch in range(1, 6):
    model.train()
    train_loss = 0
    for batch in train_loader:
        optimizer.zero_grad()
        outs = model(
            input_ids=batch["input_ids"].to(device),
            attention_mask=batch["attention_mask"].to(device),
        )
        logits = outs.logits.view(-1)
        labels = batch["labels"].to(device)
        loss = loss_fn(logits, labels)
        loss.backward()
        torch.nn.utils.clip_grad_norm_(model.parameters(), 1.0)
        optimizer.step()
        scheduler.step()
        train_loss += loss.item()
    train_loss /= len(train_loader)
    # Validation
    model.eval()
    val_loss = 0
    with torch.no_grad():
        for batch in test_loader:
            outs = model(
                input_ids=batch["input_ids"].to(device),
                attention_mask=batch["attention_mask"].to(device),
            )
            logits = outs.logits.view(-1)
            labels = batch["labels"].to(device)
            val_loss += loss_fn(logits, labels).item()
    val_loss /= len(test_loader)

    print(f"Epoch {epoch}: train_loss={train_loss:.4f}, val_loss={val_loss:.4f}")
    if val_loss < best_val_loss :
        best_val_loss = val_loss
        torch.save(model.state_dict(), "best_model.pt")
```

Hình 29. Huấn luyện mô hình

CHƯƠNG 3: ĐÁNH GIÁ MÔ HÌNH

3.1 Tải mô hình cuối và đánh giá:

- Tải lại trọng số mô hình tốt nhất đã lưu khi val_loss min.
- Khởi tạo list để lưu kết quả dự đoán (preds), nhãn thật (trues), xác suất (probs).
- Tính toán và lưu các kết quả (preds, trues, probs).
- Tính toán các metric cuối cùng:
 - Accuracy: tỉ lệ dự đoán đúng trên toàn bộ test set.
 - Precision: tỷ lệ dự đoán spam đúng
 - Recall: tỷ lệ dự đoán spam đúng trên các mẫu là spam thật.
 - F1: trung bình điều hoà giữa precision và recall.
 - AUC: đo độ phân biệt của mô hình.
 - Confusion matrix: ma trận phân loại.

```
#8 Đánh giá mô hình
model.load_state_dict(torch.load("best_model.pt"))
model.eval()
preds, trues, probs = [], [], []
with torch.no_grad():
    for batch in test_loader:
        outs = model(
            input_ids=batch["input_ids"].to(device),
            attention_mask=batch["attention_mask"].to(device),
        )
        logit = outs.logits.view(-1)
        prob = torch.sigmoid(logit)
        preds += (prob > 0.5).cpu().int().tolist()
        probs += prob.cpu().tolist()
        trues += batch["labels"].int().tolist()

acc = accuracy_score(trues, preds)
prec, rec, f1, _ = precision_recall_fscore_support(trues, preds, average="binary", zero_division=0)
auc = roc_auc_score(trues, probs)
cm = confusion_matrix(trues, preds)

print(f"Final Test → Acc {acc:.4f} Prec {prec:.4f} Rec {rec:.4f} F1 {f1:.4f} AUC {auc:.4f}")
print("CM:", cm)
```

Hình 30. Bước đánh giá

3.2 Kết quả đạt được:

Kết quả được lấy trong epoch cuối, đánh giá trên tập kiểm tra (Hình 22)

```
Epoch 1: train_loss=0.4752, val_loss=0.3793
Epoch 2: train_loss=0.2013, val_loss=0.3537
Epoch 3: train_loss=0.1475, val_loss=0.1455
Epoch 4: train_loss=0.1537, val_loss=0.1455
Epoch 5: train_loss=0.1580, val_loss=0.1455
Final Test → Acc 0.9770 Prec 0.9270 Rec 1.0000 F1 0.9621 AUC 0.9996
CM: [[685  23]
      [  0 292]]
```

Hình 31. Đánh giá mô hình

- Loss ban đầu rất cao ($\text{train_loss} = 0.4752$ và $\text{val_loss} = 0.3793$): mô hình bắt đầu học và từ epoch 2 đến 3, loss giảm đáng kể, mô hình học tốt. Trên 2 epoch cuối thì loss đã giữ nguyên, chứng tỏ mô hình đã hội tụ và không cải thiện thêm ($\text{val_loss} = 0.1455$)
- Không có dấu hiệu overfitting vì val_loss không tăng khi train_loss giảm.
- Các chỉ số đánh giá:
 - Accuracy: 97%. Mô hình đoán khá chính xác với tập dữ liệu.
 - Precision: 92%. Đa phần các dự đoán “spam” là đúng
 - Recall: 100%. Mô hình không bỏ sót bất kỳ mẫu spam nào.
 - F1-score: 96%. Cân bằng giữa precision và recall là khá tốt.
 - AUC: ~99.9%. Mô hình phân biệt spam/ham cực kỳ tốt.
- Confusion Matrix (CM) ma trận nhầm lẫn: Mô hình dự đoán sai 23 mẫu spam nhưng không hề bỏ sót spam nào.

KẾT LUẬN

Thông qua quá trình tìm hiểu, việc xây dựng mô hình phân loại tin nhắn spam/ham trên nền tảng PhoBERT đã trải qua các bước chủ chốt: tiền xử lý và làm sạch dữ liệu, tokenization với PhoBERT, chia tập train/test, fine-tune mô hình cùng tối ưu hyper-parameters đơn giản (batch size, learning rate), theo dõi train/val loss và đánh giá bằng các chỉ số Precision, Recall, F1, AUC.

Kết quả trên tập thử đã đạt 97% accuracy và mô hình đã hội tụ sau 3 vòng lặp. Đây là một kết quả cho thấy mô hình đã phân loại tin nhắn spam rất tốt với tỉ lệ bỏ sót tin nhắn spam là 0%. Bên cạnh đó, mô hình vẫn còn mắc sai lầm khi dự đoán sai 23 mẫu ham thành spam. Điều này cho thấy vẫn có những ngưỡng mô hình chưa thể hiểu và cách cấu hình thiên về bắt spam khiến cho mô hình nghĩ đó là tin nhắn spam.

Thông qua đề tài lần này, em đã có cơ hội tìm hiểu và mở rộng kiến thức thêm trong lĩnh vực học máy như kỹ thuật fine-tuning, tăng cường dữ liệu đơn giản (EDA) xử lý mất cân bằng dữ liệu sử dụng class-weighted loss, tối ưu quá trình huấn luyện và đánh giá mô hình qua các chỉ số như AUC, Accuracy, F1-score, precision, recall. Đây là nền tảng vững chắc cho em để tiếp cận các ứng dụng thực tế và triển khai mô hình về sau.

HƯỚNG PHÁT TRIỂN

Để bổ sung thêm cho đề tài của em, dưới đây là một số hướng phát triển có thể mở rộng và nâng cao cho đề tài phân loại tin nhắn:

1. Tìm kiếm các bộ dữ liệu thực tế:
 - Sử dụng các bộ dữ liệu thực tế từ đời sống, các tin nhắn quảng cáo, spam thực tế sẽ có độ uy tín và hiệu suất đánh giá tốt hơn.
2. Phân loại nhãn chi tiết:
 - Tách spam thành các nhóm con nhỏ hơn: quảng cáo, phishing, scam tài chính hay tin hệ thống.
3. Tận dụng thêm các ngữ cảnh và metadata:
 - Kết hợp thêm các đặc trưng như thời gian gửi, người gửi, số lần gửi lại giúp phân biệt spam điển hình.

TÀI LIỆU THAM KHẢO

1. Dat Quoc Nguyen, Anh Tuan Nguyen (2020), *PhoBERT: Pre-trained language models for Vietnamese*, (arXiv:2003.00744). Available: <https://arxiv.org/pdf/2003.00744>
2. Van-Duyet Le, *Vietnamese-stopwords*. Available: <https://github.com/stopwords/vietnamese-stopwords>
3. *Danh sách stopwords*. Available: https://xltiengviet.fandom.com/wiki/Danh_sách_stop_word
4. Hugging Face, *PhoBERT documentation*. Available: <https://huggingface.co/vinai/phobert-base>
5. Vu Minh Tuan, Nguyen Xuan Thang, Tran Quang Anh (2022), *Vietnamese SMS Spam Detection with Deep Learning and Pre-trained Language Model*. Available: <https://jstic.ptit.edu.vn/jstic-ptit/index.php/jstic/article/view/484>
6. Jason Wei and Kai Zou. 2019. EDA: *Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks*. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 6382–6388, Hong Kong, China. Association for Computational Linguistics.
7. Pham Dinh Khanh, *Kiến trúc mô hình bài toán classification – Thực hành ứng dụng BERT*. Available: https://phamdinhkhanh.github.io/2020/06/04/PhoBERT_Fairseq.html#81-kiến-trúc-mô-hình
8. Documetation, *BCEWithLogitsLoss*. Available: <https://docs.pytorch.org/docs/stable/generated/torch.nn.BCEWithLogitsLoss>
9. Documentation, *Binary Cross Entropy for Binary Classification*. Available: <https://www.geeksforgeeks.org/binary-cross-entropy-log-loss-for-binary-classification/>