TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA TOÁN - TIN HỌC

XỬ LÝ NGÔN NGỮ TỰ NHIÊN
BÁO CÁO ĐỒ ÁN
Thành viên

Lê Quốc An	**22280001**
Phạm Lê Hồng Đức	**22280013**
Lê Đức Hòa	**22280027**

Ngày 16, tháng 6, năm 2025

BẢNG ĐÁNH GIÁ ĐÓNG GÓP CÁC THÀNH VIÊN TRONG NHÓM

No	Name	Id	Contribution
1	Lê Quốc An	22280001	100%
2	Phạm Lê Hồng Đức	22280013	100%
3	Lê Đức Hòa	22280027	100%

Link google drive: https://drive.google.com/drive/folders/1aZ7KTVO183ofl2Av1e8REg-r-fDa-SqM?usp=sharing

MỤC LỤC

1. Giới Thiệu

Trong thời đại kỹ thuật số, sự lan truyền thông tin trở nên nhanh chóng và rộng rãi thông qua mạng xã hội, báo chí trực tuyến và các nền tảng truyền thông. Bên cạnh những lợi ích to lớn, hiện tượng tin giả (fake news) đã và đang trở thành một vấn đề nhức nhối, gây ra nhiều hệ lụy nghiêm trọng như: làm sai lệch nhận thức cộng đồng, ảnh hưởng đến uy tín tổ chức và cá nhân, tác động đến chính trị, y tế, kinh tế và an ninh xã hội.

Việc phát hiện và xử lý tin giả là một thách thức lớn do tính chất tinh vi và đa dạng về ngôn ngữ, văn phong của các bài viết. Tuy nhiên, với sự phát triển nhanh chóng của các kỹ thuật Xử lý ngôn ngữ tự nhiên (Natural Language Processing - NLP) và Học máy (Machine Learning), việc phát hiện tin giả thông qua nội dung văn bản ngày càng khả thi và hiệu quả hơn.

Mục tiêu của đề tài này là xây dựng một mô hình học máy có khả năng phân loại văn bản thành tin thật hoặc tin giả, dựa trên nội dung bài viết. Qua đó, đề tài hướng đến việc hỗ trợ công tác sàng lọc thông tin, góp phần nâng cao khả năng phòng chống và nhận diện thông tin sai lệch trên Internet.

2. Mô tả bộ dữ liệu- Vấn đề và thử thách xung quanh bộ dữ liệu

Bộ dữ liệu sử dụng trong bài toán phân loại tin tức thật – giả ban đầu bao gồm hai tập tin:

- DataSet_Misinfo_FAKE.csv: Chứa các mẫu tin tức giả mạo (fake news) với tổng cộng 43,642 dòng.
- DataSet_Misinfo_TRUE.csv: Chứa các mẫu tin tức thật (true news) với tổng cộng 34,975 dòng.

Mỗi tập tin ban đầu đều gồm hai cột:

- Unnamed: 0: Chỉ số dòng (index), không mang ý nghĩa phân tích.
- text: Nội dung bài viết hoặc tiêu đề bài báo.

Sau quá trình tiền xử lý dữ liệu, các bước sau đã được thực hiện:

- Đã hợp nhất hai tập tin thành một DataFrame duy nhất.
- Đã xóa cột 'Unnamed: 0'.
- Đã loại bỏ 29 hàng có giá trị thiếu (NaN) trong cột 'text'.
- Đã loại bỏ 9,984 hàng trùng lặp.

Kết quả sau tiền xử lý, bộ dữ liệu cuối cùng bao gồm tổng cộng 68,600 mẫu văn bản, được phân chia thành hai nhãn như sau:

- Nhãn 0: Đại diện cho tin giả (FAKE), với 34,076 mẫu.
- Nhãn 1: Đại diện cho tin thật (TRUE), với 34,524 mẫu.

Với cấu trúc này, bộ dữ liệu phù hợp cho các bài toán phân loại văn bản nhị phân, đặc biệt là trong lĩnh vực phát hiện tin giả (fake news detection). Dữ liệu ở dạng văn bản thuần túy, cần áp dụng các kỹ thuật xử lý ngôn ngữ tự nhiên (NLP) như làm sạch sâu hơn, chuẩn hóa văn bản, trích xuất đặc trưng (vector hóa) và huấn luyện mô hình học máy hoặc học sâu.

Vấn đề / Thử thách	Mô tả	Giải pháp
Thiếu thông tin ngữ cảnh	Dữ liệu chỉ gồm văn bản thô, không có thông tin bổ sung về ngày tháng, tác giả, nguồn đăng,... Điều này gây khó khăn trong việc khai thác các khía cạnh liên quan đến thời gian hay độ tin cậy của nguồn.	Bổ sung thêm dữ liệu nếu có thể (ví dụ: thu thập từ các API hoặc cơ sở dữ liệu có sẵn thông tin metadata). Hoặc huấn luyện mô hình chỉ dựa trên nội dung văn bản, kết hợp trích xuất đặc trưng nâng cao như TF-IDF, Word2Vec, hoặc các lớp Embedding từ các mô hình học sâu.
Chất lượng văn bản không đồng đều	Một số tin rất ngắn, chỉ 1-2 câu, trong khi các tin khác lại dài nhiều đoạn. Ngoài ra, văn bản có thể chứa các ký tự đặc biệt không mong muốn như \r\n, dấu câu thừa, hoặc các lỗi chính tả cần xử lý.	Thực hiện tiền xử lý văn bản kỹ lưỡng: làm sạch ký tự đặc biệt, chuẩn hóa từ viết tắt, tách câu, lọc bỏ các từ dừng (stop words), chuẩn hóa chính tả (nếu cần). Có thể đặt ngưỡng độ dài tối thiểu cho văn bản để loại bỏ các mẫu quá ngắn không đủ thông tin.
Khó phân biệt nội dung thật/giả qua ngôn ngữ	Một số tin giả được viết rất giống tin thật, sử dụng văn phong nghiêm túc, khách quan. Ngược lại, một số tin thật có thể chứa tiêu đề giật tít hoặc nội dung gây hiểu nhầm, khiến việc phân biệt chỉ dựa vào đặc điểm ngôn ngữ trở nên phức tạp.	Áp dụng các mô hình học sâu nâng cao như BERT, RoBERTa hoặc XLNet để tận dụng khả năng hiểu ngữ cảnh sâu của câu. Kết hợp với kỹ thuật attention hoặc fine-tune các mô hình pre-trained để nắm bắt những sắc thái ngôn ngữ tinh tế.
Tính thời điểm và sự thay đổi ngôn ngữ theo thời gian	Hiện tại, không có thông tin về thời điểm dữ liệu được thu thập, trong khi ngôn ngữ và cách viết của tin giả/thật có thể thay đổi, phát triển theo thời gian. Điều này có thể ảnh hưởng đến khả năng khái quát hóa của mô hình đối với dữ liệu mới.	Nếu mở rộng nghiên cứu, nên thu thập thêm dữ liệu có gắn kèm thông tin thời gian rõ ràng. Trong phạm vi dữ liệu hiện có, có thể nghiên cứu các phương pháp huấn luyện mô hình theo dạng domain-invariant (bất biến với miền) để giảm thiểu ảnh hưởng của sự lệch thời điểm.

3. Tiền xử lý dữ liệu văn bản

Trước khi tiến hành huấn luyện mô hình, dữ liệu văn bản được xử lý để loại bỏ nhiễu và chuẩn hóa đầu vào, nhằm tối ưu hóa hiệu suất của mô hình. Tùy thuộc vào hướng tiếp cận (mô hình học máy truyền thống hay học sâu hiện đại) mà nhóm áp dụng mức độ tiền xử lý khác nhau.

3.1. Làm sạch mạnh (cho mô hình truyền thống)

Với các mô hình học máy cổ điển như Naïve Bayes, Logistic Regression, SVM,... thì đầu vào thường là các đặc trưng (features) rút trích từ văn bản như TF-IDF hoặc CountVectorizer. Do đó, văn bản cần được xử lý kỹ lưỡng để đảm bảo loại bỏ các thành phần gây nhiễu hoặc không có giá trị ngữ nghĩa.

Các bước xử lý mạnh mẽ được áp dụng tuần tự trong hàm _single_text_preprocess như sau:

- **0. Chuẩn hóa Unicode**: Chuyển đổi văn bản sang dạng chuẩn NFKD để xử lý các ký tự có dấu và biến thể.
- **1. Bỏ các ký tự không thuộc bảng mã ASCII**: Mã hóa và giải mã lại văn bản để loại bỏ các ký tự không phải ASCII, giúp đảm bảo tính nhất quán của dữ liệu.
- **2. Chuyển tất cả thành chữ thường (lowercase)**: Để thống nhất các từ (ví dụ: "Trump" và "trump" được coi là một).
- **3. Xóa các URL, liên kết Twitter ảnh và đề cập (**mentions**)**: Loại bỏ các đường dẫn web (bắt đầu bằng http://, https://, www.), liên kết hình ảnh Twitter (pic.twitter.com), và các từ chứa ký tự @ (thường là đề cập người dùng trong mạng xã hội) vì chúng không mang thông tin phân loại hữu ích.
- **4. Xóa các từ lặp lại không cần thiết**: Loại bỏ các chuỗi ký tự hoặc từ lặp lại quá mức (ví dụ: "ha ha ha" thành "ha ha", "twitter twitter" thành "twitter", và các ký tự lặp lại quá 2 lần như "looooong" thành "loong").
- **5. Xóa dấu câu và ký tự đặc biệt**: Chỉ giữ lại các chữ cái tiếng Anh (a-zA-Z) và khoảng trắng, loại bỏ tất cả các ký tự khác (bao gồm dấu câu, ký tự đặc biệt, và cả số ở bước này nếu regex bao quát).
- **6. Xóa số**: Loại bỏ tất cả các chữ số còn lại trong văn bản.
- **7. Chuẩn hóa khoảng trắng**: Chuyển nhiều khoảng trắng thành một khoảng trắng duy nhất và xóa khoảng trắng ở đầu/cuối chuỗi.
- **8. Xóa stop words**: Loại bỏ các từ phổ biến nhưng không có ý nghĩa cụ thể trong việc phân loại (ví dụ: "the", "is", "of",...).

Ví dụ trước và sau khi xử lý mạnh:

Văn bản gốc: “The president is trying to make America great again!”
Sau khi xử lý: “president trying make america great again”

Việc loại bỏ các thành phần không mang tính phân biệt giúp mô hình tập trung vào các đặc trưng quan trọng hơn trong văn bản, nâng cao khả năng học của các mô hình truyền thống.

3.2. Không làm sạch (cho mô hình Transformer: BERT, XLNet)

Với các mô hình học sâu hiện đại như BERT và XLNet, **nhóm sẽ không thực hiện bất kỳ bước làm sạch nào** liên quan đến việc loại bỏ dấu câu, chữ hoa, chữ số, URL hay stopwords. Dữ liệu sẽ được giữ ở trạng thái "thô" nhất có thể sau khi đảm bảo tính hợp lệ của ký tự.

Lý do: Các mô hình này đã được huấn luyện trước (pretrained) trên tập dữ liệu lớn và có khả năng hiểu ngữ cảnh, ngữ pháp, và trọng số của từng từ, từng dấu câu thông qua cơ chế attention phức tạp. Việc loại bỏ các thành phần này có thể khiến mô hình mất đi thông tin ngữ nghĩa quan trọng mà chúng được thiết kế để tận dụng. Chúng có thể phân biệt được ý nghĩa của một từ trong các ngữ cảnh khác nhau, ngay cả khi có dấu câu hay các từ phổ biến.

3.3. Lọc theo độ dài văn bản

Ngoài các bước làm sạch trên, quá trình tiền xử lý còn bao gồm bước lọc các văn bản dựa trên độ dài để loại bỏ các mẫu ngoại lai (outliers) quá ngắn hoặc quá dài. Hàm filter_by_length_text thực hiện các bước sau:

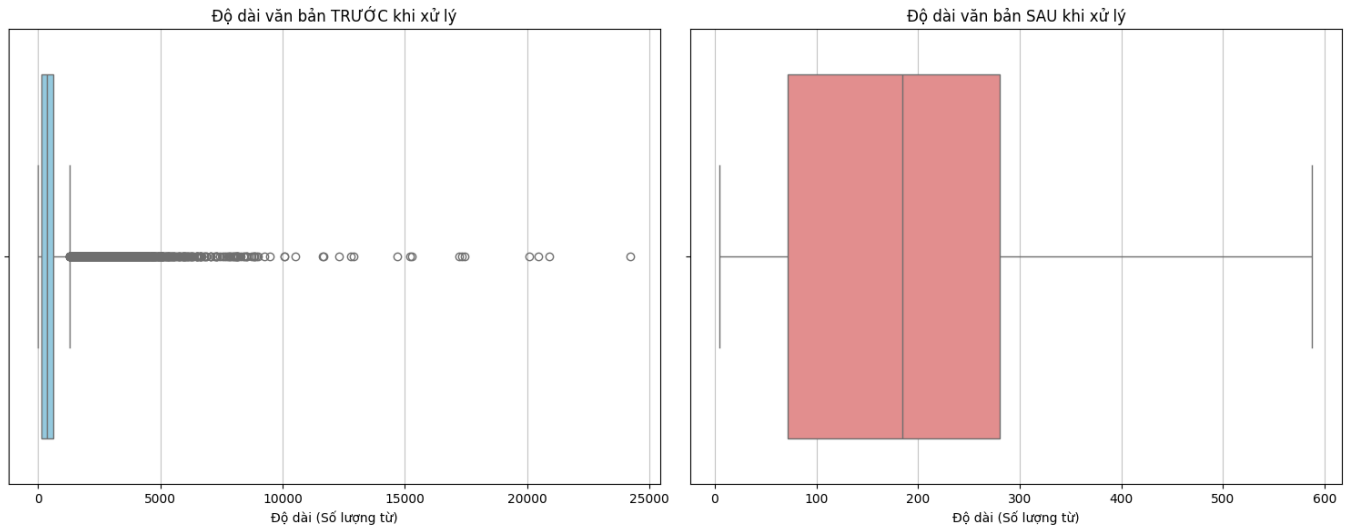
- Tính toán số lượng từ (word_count) cho mỗi văn bản trong cả df_ML và df_DL.
- Xác định các ngưỡng phân vị (lower_percentile và upper_percentile, mặc định là 1% và 90%) của số lượng từ.
- Giữ lại chỉ những văn bản có số lượng từ nằm trong khoảng giữa hai ngưỡng phân vị này. Điều này giúp loại bỏ các văn bản cực đoan có thể gây nhiễu cho quá trình huấn luyện mô hình.

Tổng kết: Sau khi hoàn tất các bước tiền xử lý này, nhóm sẽ tiến hành triển khai hai hướng tiếp cận chính để giải quyết bài toán: một dựa trên các mô hình học máy cổ điển với dữ liệu được làm sạch mạnh, và một dựa trên các mô hình học sâu hiện đại với dữ liệu được giữ nguyên.

3.4. Trực quan hóa tác động của tiền xử lý mạnh và lọc theo độ dài văn bản (áp dụng cho dữ liệu mô hình truyền thống)

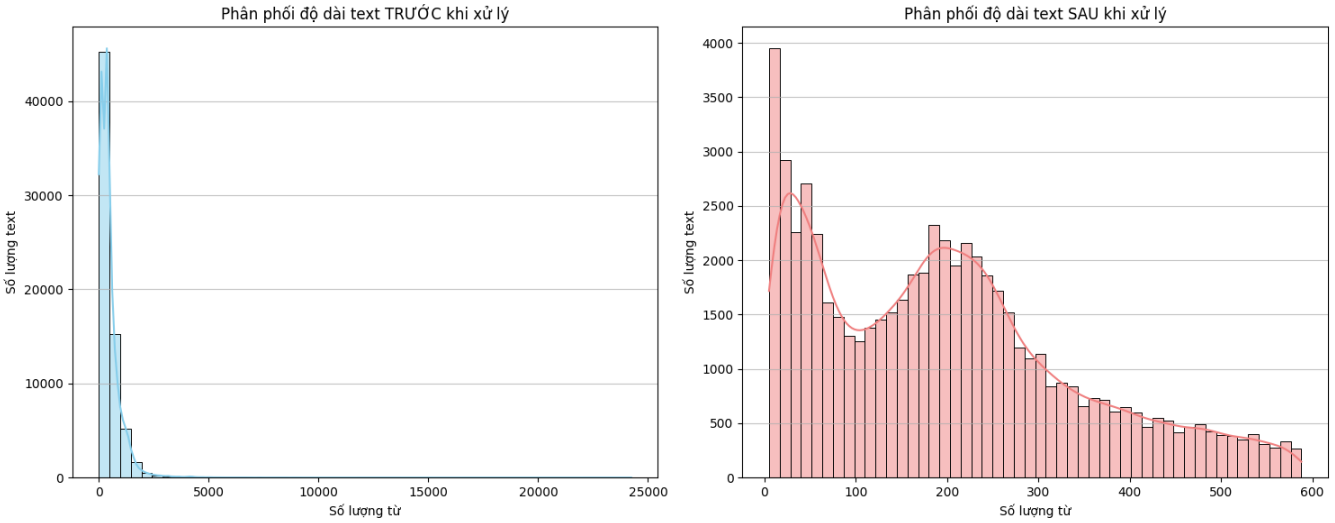
Sau khi áp dụng hai phương pháp làm sạch ở trên, nhóm tiến hành trực quan hóa để đánh giá ảnh hưởng của các bước xử lý đến dữ liệu đầu vào.

- Boxplot giúp so sánh độ dài văn bản (số lượng từ) trước và sau xử lý với cả hai phương pháp.
- Histogram giúp phân tích sự phân bố độ dài câu, từ đó hỗ trợ cho việc chọn padding hoặc cắt ngắn khi đưa vào mô hình học sâu.
- Pie charts giúp trực quan hóa tỷ lệ giữa các lớp trong tập dữ liệu, cụ thể là “tin thật” và “tin giả”. Hình này cho phép kiểm tra xem dữ liệu có bị mất cân bằng (imbalanced) hay không – một yếu tố rất quan trọng ảnh hưởng đến hiệu quả của mô hình học máy. Nếu một lớp chiếm đa số tuyệt đối, mô hình có thể bị thiên lệch khi huấn luyện.



Hình 3.4.1. Boxplot độ dài văn bản trước/sau xử lý

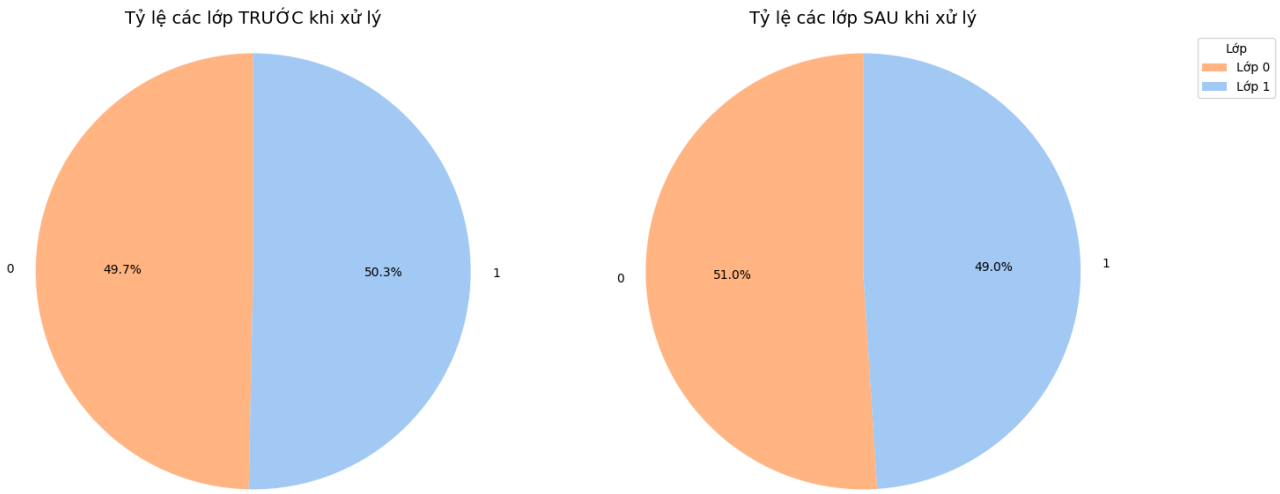
Quá trình tiền xử lý đã **hiệu quả** trong việc thu hẹp đáng kể phạm vi độ dài văn bản và loại bỏ các giá trị ngoại lệ, giúp dữ liệu trở nên **đồng nhất và tập trung hơn**, rất có lợi cho việc xây dựng mô hình.



Hình 3.4.2. Histogram phân phối số lượng từ

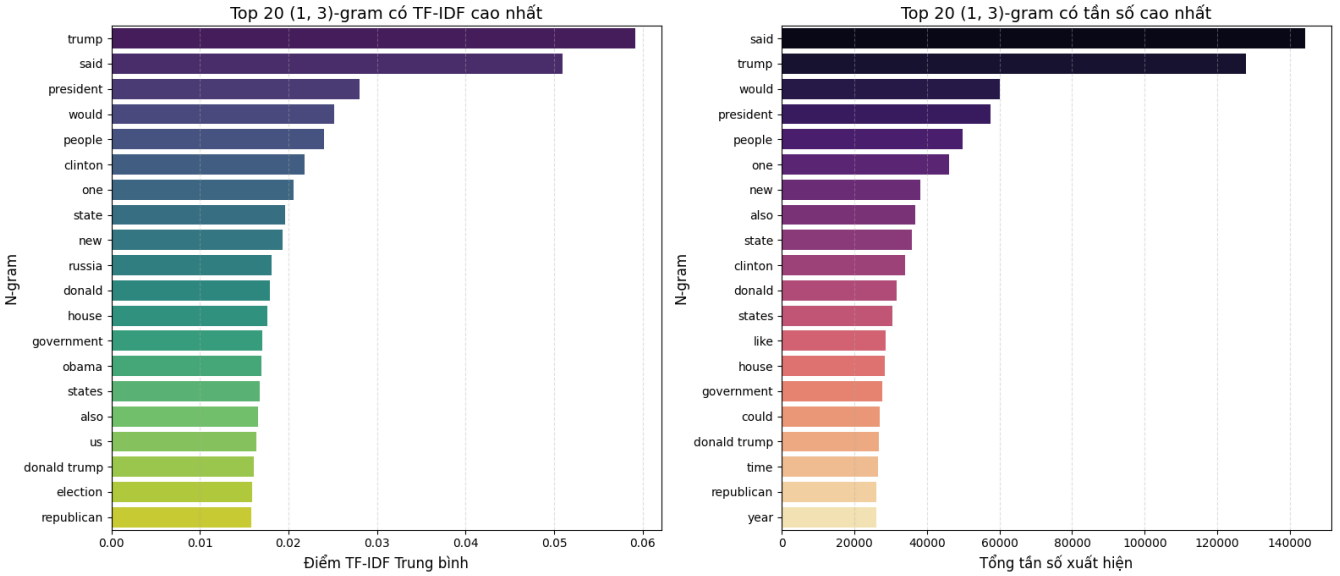
Biểu đồ trước xử lý: Phân phối độ dài văn bản rất lệch phải, tập trung cao ở các văn bản ngắn và có nhiều văn bản cực dài.

Biểu đồ sau xử lý: Phân phối trở nên đồng đều và tập trung hơn nhiều, với độ dài được giới hạn trong một phạm vi hợp lý, cho thấy quá trình tiền xử lý đã hiệu quả trong việc chuẩn hóa dữ liệu.

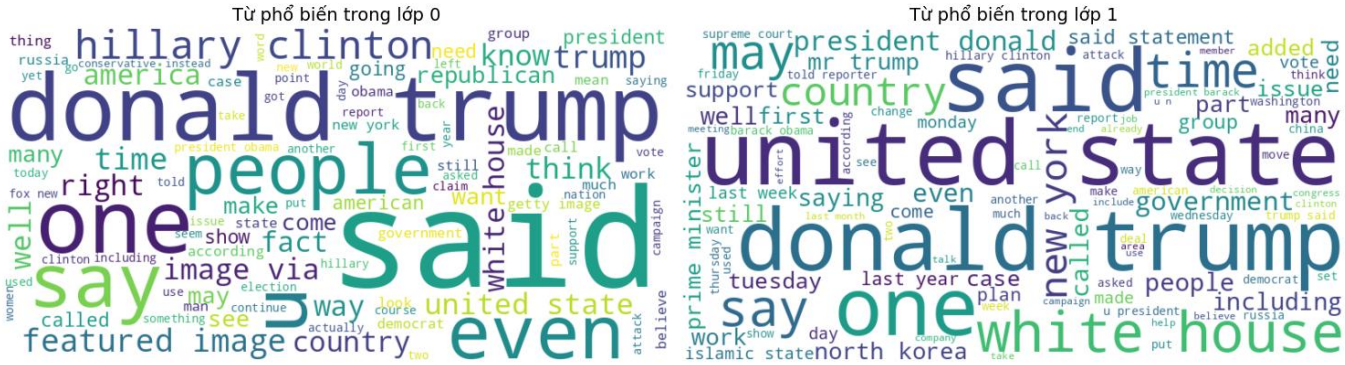


Hình 3.4.3. Biểu đồ tròn tỉ lệ các lớp (tin thật / tin giả) trước xử lý

Các biểu đồ tròn cho thấy tỷ lệ phân bố giữa Lớp 0 và Lớp 1 trong bộ dữ liệu **gần như cân bằng** cả trước và sau quá trình xử lý. Cụ thể, trước xử lý, tỷ lệ là 49.7% (Lớp 0) và 50.3% (Lớp 1); sau xử lý, tỷ lệ thay đổi rất ít thành 51.0% (Lớp 0) và 49.0% (Lớp 1). Điều này chứng tỏ quá trình tiền xử lý dữ liệu **không làm thay đổi đáng kể sự cân bằng** của các lớp, đảm bảo tính đại diện của dữ liệu cho các bước phân tích và huấn luyện mô hình sau này.

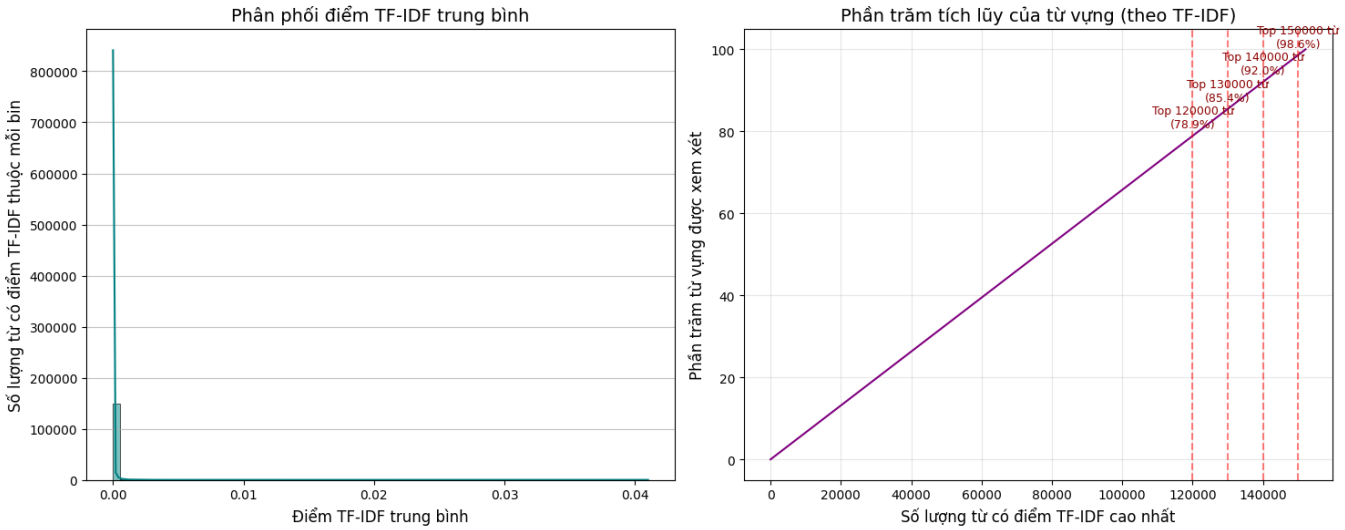


Hình 3.4.4. Biểu đồ Top 20 unigram có TF-IDF/ Tần số cao nhất



Hình 3.4.5. Biểu đồ đám mây từ khóa phổ biến trong dữ liệu (sau khi tiền xử lý)

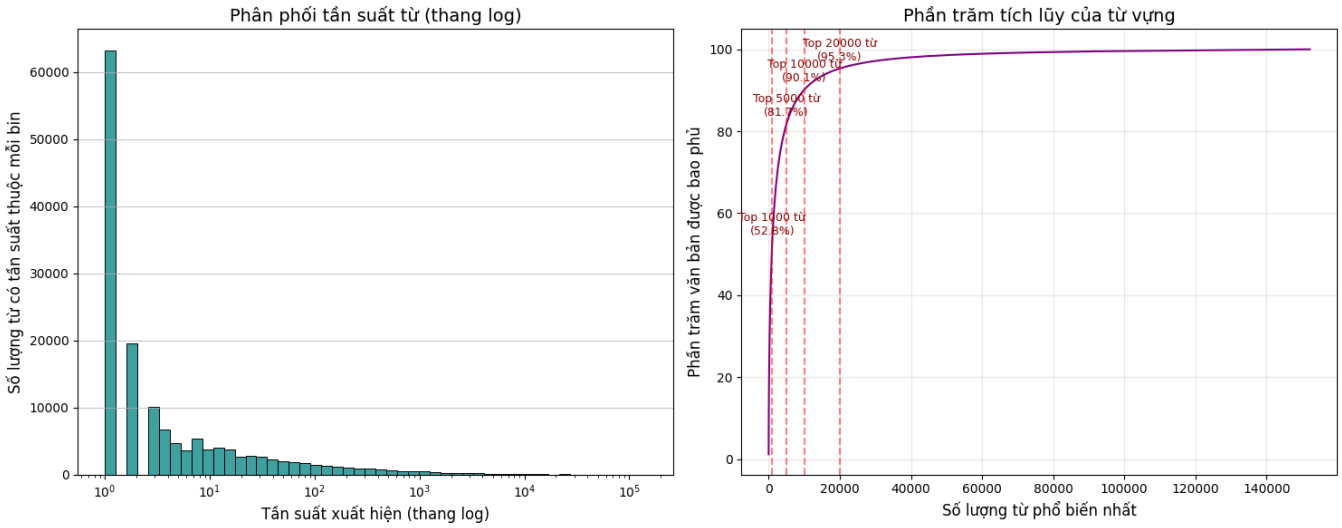
Dựa trên biểu đồ, các từ/cụm từ phổ biến nhất liên quan đến **chính trị Mỹ** ("trump", "president", "clinton", "russia", "election"), cho thấy dữ liệu có khả năng là tin tức chính trị. TF-IDF làm nổi bật các từ khóa phân biệt, trong khi tần số cao bao gồm cả các từ chung.



Hình 3.4.6. Biểu đồ phân phối điểm TF-IDF trung bình của từ vựng và Đồ thị tích lũy từ vựng theo điểm TF-IDF

Biểu đồ phân bố TF-IDF: Cho thấy phần lớn các từ có điểm TF-IDF rất thấp (gần 0), chỉ một số ít từ có điểm TF-IDF cao hơn đáng kể. Điều này phản ánh rằng đa số các từ là phổ biến và ít mang tính phân biệt, trong khi một số từ khóa đặc trưng mới có giá trị TF-IDF cao.

Đồ thị tích lũy từ vựng theo điểm TF-IDF: Biểu đồ này chỉ ra rằng để đạt được một phần trăm đáng kể của tổng vốn từ vựng được xem xét (ví dụ: 98.6% ở top 150000 từ), cần phải giữ lại một lượng lớn từ vựng. Điều này gợi ý rằng ngay cả các từ có điểm TF-IDF thấp cũng đóng góp vào sự đa dạng của ngôn ngữ và có thể cần được xem xét trong các tác vụ NLP phức tạp



Hình 3.4.7. Biểu đồ phân phối tần suất từ vựng (theo thang log) và Biểu đồ tích lũy từ vựng theo số lượng từ phổ biến nhất

Biểu đồ phân phối tần suất từ vựng: Biểu đồ này hiển thị sự phân bố tần suất của các từ, cho thấy phần lớn các từ xuất hiện rất ít (tần suất thấp), trong khi một số lượng nhỏ các từ lại có tần suất xuất hiện rất cao. Đây là một đặc điểm phổ biến của ngôn ngữ tự nhiên, tuân theo luật Zipf, nơi một vài từ phổ biến chiếm phần lớn văn bản.

Biểu đồ tích lũy từ vựng: Biểu đồ này minh họa rằng chỉ một số lượng tương đối nhỏ các từ phổ biến (ví dụ: Top 20000 từ) đã chiếm một phần rất lớn (gần 99%) của tổng văn bản. Điều này rất quan trọng trong việc xây dựng bộ từ vựng (vocabulary) cho các mô hình xử lý ngôn ngữ tự nhiên, cho phép tập trung vào các từ quan trọng nhất để giảm kích thước mô hình và cải thiện hiệu quả.

4. Hướng tiếp cận và phát triển mô hình

Để đánh giá hiệu quả phát hiện tin giả, nhóm đã xây dựng mô hình theo hai hướng chính: Học máy cổ điển (dựa trên đặc trưng thủ công như TF-IDF, CountVectorizer) và Học sâu hiện đại (sử dụng kiến trúc Transformer như BERT, XLNet).

4.1. Hướng truyền thống: TF-IDF / CountVectorizer kết hợp mô hình học máy

4.1.1. Biểu diễn đặc trưng

Sử dụng hai kỹ thuật chính để chuyển văn bản thành vector:

- CountVectorizer: Đếm tần suất từ xuất hiện. Đơn giản nhưng dễ quá khớp.
- TF-IDF: Giảm trọng số từ phổ biến, tăng trọng số từ đặc trưng, giúp phân biệt tốt hơn.

4.1.2. Các mô hình học máy áp dụng

Các mô hình được sử dụng bao gồm:

- **Logistic Regression**: Là mô hình tuyến tính đơn giản, dựa vào hàm sigmoid để dự đoán xác suất văn bản thuộc lớp "thật" hay "giả":

$$\sigma(z) = \frac{1}{1 + e^{-z}}, \quad z = w^T x + b$$

- **Naive Bayes (Multinomial) :** Sử dụng định lí Bayes với giả định các đặc trưng độc lập có diều kiện. Dù giả định đơn giản, nhưng hiệu quả cao với dữ liệu rời rạc như văn bản:

$$P(C|x) = \frac{P(x|C)P(C)}{P(x)}$$

- **Decision Tree**: Phân chia dữ liệu tại mỗi nút theo độ lợi thông tin hoặc chỉ số Gini. Mô hình dễ hiểu nhưng dễ bị quá khớp (overfitting).
- **Random Forest:** Tập hợp nhiều cây quyết định, học trên các tập con ngẫu nhiên. Giúp cải thiện độ chính xác và giảm overfitting.
- **Support Vector Machine** (SVM): Tìm siêu phẳng tối ưu phân chia hai lớp đặc biệt hiệu quả khi kết hợp với đặc trưng TF-IDF và kernel tuyến tính.

4.1.3. Tối ưu mô hình

Sử dụng RandomizedSearchCV để tìm siêu tham số tối ưu (số lượng từ, n-gram, trọng số...).

4.2. Hướng hiện đại: Mô hình Transformer (BERT, XLNet)

4.2.1. Tổng quan kiến trúc Transformer

Transformer là kiến trúc học sâu sử dụng cơ chế self – attention, giúp mô hình hóa mối quan hệ giữa các từ mà không cần mạng tuần tự như RNN hay LSTM.
Công thức attention:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Trong đó:

Q, K, V: ma trận truy vấn, khóa và giá trị.
d_k: số chiều của vector khóa.

4.2.2. Mô hình BERT

BERT (bert-base-cased) học ngữ cảnh hai chiều qua **Masked Language Modeling (MLM)**. Nó được fine-tune để phân loại tin giả, giúp mô hình hiểu sâu ngữ nghĩa và cấu trúc ngôn ngữ.
BERT: 108,311,810 tham số.

4.2.3. Mô hình XLNet

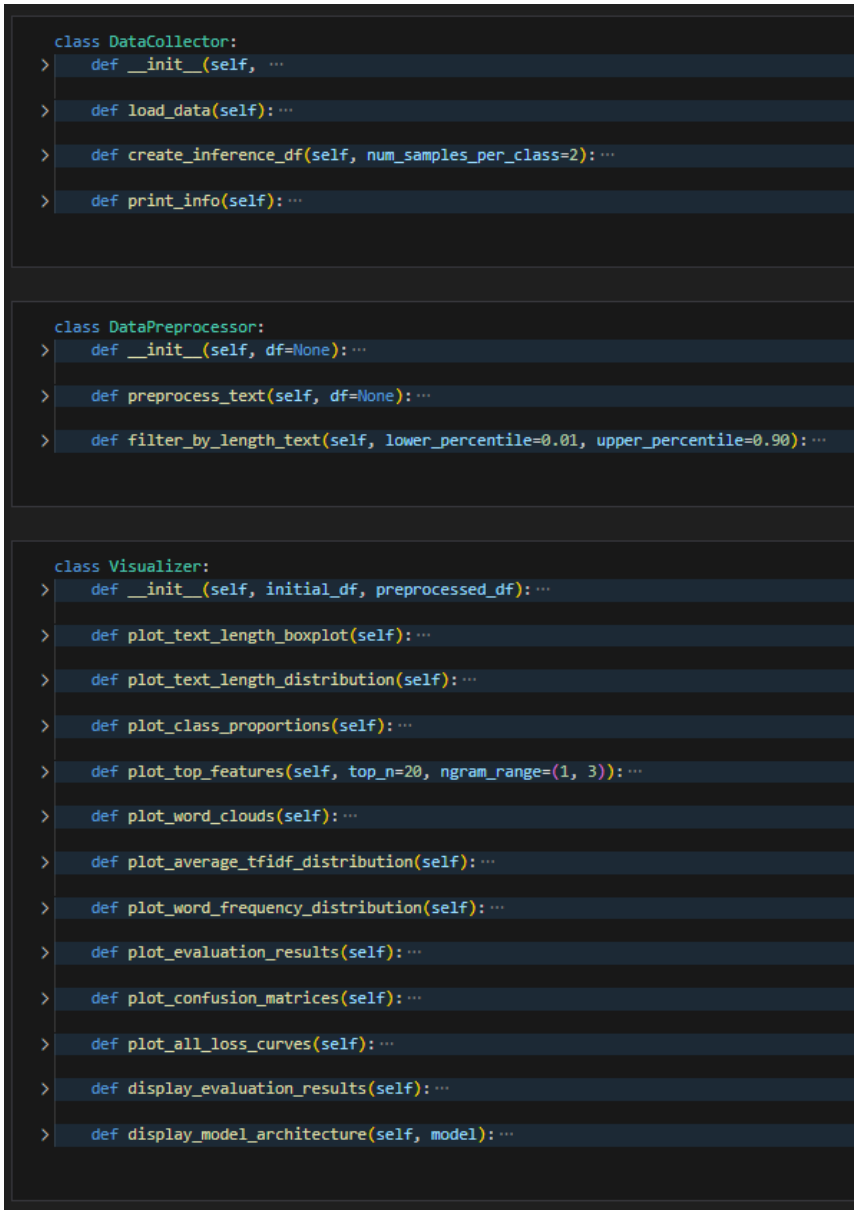
XLNet (xlnet-base-cased) sử dụng **Permutation Language Modeling**, khắc phục điểm yếu của BERT bằng cách giữ thông tin thứ tự từ. Điều này giúp XLNet hiểu rõ hơn về ngữ pháp và ngữ nghĩa, mang lại kết quả chính xác cao hơn.
XLNet: 117,310,466 tham số.
XLNet có số lượng tham số lớn hơn **BERT**, cho thấy mô hình này có kiến trúc phức tạp hơn một chút, dù cả hai đều là các mô hình Transformer quy mô lớn.

5. Mô tả code

5.1. Cấu trúc Code

Cấu trúc code được thiết kế theo hướng **module hóa và dễ bảo trì**, với các lớp (class) riêng biệt đảm nhiệm từng chức năng cụ thể:

- **DataCollector**: Xử lý việc nạp và tạo thông tin từ dữ liệu gốc, bao gồm đọc dữ liệu, tạo tập dữ liệu suy diễn và hiển thị thông tin tổng quan.
- **DataPreprocessor**: Thực hiện các bước tiền xử lý văn bản như chuẩn hóa và lọc theo độ dài.
- **Visualizer**: Tập trung vào việc trực quan hóa dữ liệu và kết quả huấn luyện, cung cấp các hàm vẽ biểu đồ đa dạng (boxplot, TF-IDF, tần suất từ, wordcloud, confusion matrix) và hiển thị kiến trúc mô hình.



```
class DataCollector:
    def __init__(self, ···
    def load_data(self): ···
    def create_inference_df(self, num_samples_per_class=2): ···
    def print_info(self): ···

class DataPreprocessor:
    def __init__(self, df=None): ···
    def preprocess_text(self, df=None): ···
    def filter_by_length_text(self, lower_percentile=0.01, upper_percentile=0.90): ···

class Visualizer:
    def __init__(self, initial_df, preprocessed_df): ···
    def plot_text_length_boxplot(self): ···
    def plot_text_length_distribution(self): ···
    def plot_class_proportions(self): ···
    def plot_top_features(self, top_n=20, ngram_range=(1, 3)): ···
    def plot_word_clouds(self): ···
    def plot_average_tfidf_distribution(self): ···
    def plot_word_frequency_distribution(self): ···
    def plot_evaluation_results(self): ···
    def plot_confusion_matrices(self): ···
    def plot_all_loss_curves(self): ···
    def display_evaluation_results(self): ···
    def display_model_architecture(self, model): ···
```

5.2. Tổ chức Huấn luyện (Workflow) và Suy diễn

Quy trình huấn luyện được chuẩn hóa thông qua hai lớp **WorkflowML** (cho mô hình Machine Learning truyền thống) và **WorkflowDL** (cho mô hình Deep Learning). Cả hai lớp này đều chia quá trình thành các bước rõ ràng: tách dữ liệu, tìm siêu tham số tối ưu, xây dựng pipeline, huấn luyện, đánh giá và lưu mô hình.

Điểm đặc biệt là việc sử dụng lớp **Hyperparameters** riêng biệt. Được định nghĩa với @dataclass, lớp này gom toàn bộ các siêu tham số tại một nơi duy nhất, giúp **dễ dàng tinh chỉnh, theo dõi và tái sử dụng** giữa các giai đoạn huấn luyện và suy diễn, đồng thời hỗ trợ mặc định cho các giá trị như "cuda" hoặc "cpu".

Sau khi huấn luyện, lớp **ModelInferencer** cho phép nạp lại mô hình đã lưu và thực hiện suy diễn. Điều này đảm bảo tính **tái sử dụng cao** của pipeline từ các lớp Workflow nếu mô hình được lưu trữ đúng định dạng.

```
class WorkflowML:
    def __init__(self, dataframe: pd.DataFrame): ...

    def split_data(self, test_ratio: float = 0.2, random_seed: int = 42): ...

    def find_best_hyperparameters(self, random_seed: int = 42, cv_splits: int = 5, models_to_optimize: list = 

    def set_model_pipline(self, model_pipline_name: str, model_pipeline: Pipeline): ...

    def train_model(self): ...

    def save_model(self): ...

    def evaluate_model(self): ...
```

```
@dataclass
class Hyperparameters: ...

class WorkflowDL:
    def __init__(self, dataframe: pd.DataFrame, hypers: Hyperparameters): ...

    def split_data(self, test_ratio: float = 0.15, val_ratio: float = 0.15, random_seed: int = 42): ...

    def tokenize_data(self): ...

    def create_data_loaders(self): ...

    def build_load_model(self): ...

    def _setup_training_components(self): ...

    def _process_batch(self, batch): ...

    def _train_epoch(self, dataloader, optimizer, scheduler, criterion): ...

    def _evaluate_epoch(self, dataloader, criterion): ...

    def train_model(self): ...

    def save_pipeline(self): ...

    def evaluate_model(self): ...
```

```
class ModelInferencer:
    def __init__(self, model_path: str, df: pd.DataFrame): ...

    def _print_results(self, texts, predictions, actual_labels, probabilities): ...

    def infer(self) -> list: ...
```

5.3. Ưu điểm Nổi bật

Cách tổ chức này mang lại nhiều lợi ích quan trọng:

- **Tính mở rộng và dễ kiểm thử**: Các class riêng biệt theo vai trò giúp dễ dàng mở rộng và kiểm thử độc lập, tuân thủ nguyên tắc SOLID.
- **Tái sử dụng cao**: Quy trình workflow rõ ràng và việc tách biệt các siêu tham số giúp giảm lặp code và tăng tính tái sử dụng giữa các loại mô hình.
- **Quản lý hiệu quả**: Tập trung các siêu tham số vào một nơi duy nhất đơn giản hóa việc tinh chỉnh và quản lý.
- **Phân tích chuyên sâu**: Lớp Visualizer chuyên biệt hỗ trợ phân tích dữ liệu và mô hình trực quan hơn.
- **Ứng dụng độc lập**: Phần suy diễn độc lập cho phép triển khai mô hình đã huấn luyện một cách linh hoạt.

6. Các phương pháp tối ưu và huấn luyện mô hình

Nhóm đã áp dụng các phương pháp tiên tiến để tối ưu hóa và nâng cao hiệu suất mô hình:

- Tối Ưu Hóa Siêu Tham Số: Để tìm kiếm các siêu tham số tối ưu, nhóm đã ưu tiên sử dụng **Random Search** thay vì Grid Search. Phương pháp này đặc biệt hiệu quả trong việc khám phá không gian siêu tham số rộng lớn, mang lại khả năng tìm thấy các tổ hợp tham số tốt hơn một cách hiệu quả về mặt tính toán so với việc duyệt tuần tự của Grid Search.

- Xử Lý Dữ Liệu Chuỗi: Trong các tác vụ liên quan đến dữ liệu chuỗi, kỹ thuật **Slide Window** được áp dụng để cho phép mô hình xem xét các đoạn dữ liệu cố định. Điều này giúp mô hình nắm bắt ngữ cảnh cục bộ và các mối quan hệ tuần tự quan trọng trong dữ liệu.

- Hàm Loss: Đối với bài toán phân loại, nhóm sử dụng hàm **CrossEntropyLoss** từ thư viện PyTorch. Hàm loss này rất phù hợp cho các tác vụ phân loại đa lớp và được điều chỉnh để xử lý hiệu quả các trường hợp dữ liệu mất cân bằng thông qua việc điều chỉnh trọng số (nếu cần) hoặc các kỹ thuật khác trong quá trình tiền xử lý dữ liệu.

- Ngừng Huấn Luyện Sớm: Nhằm ngăn chặn hiện tượng quá khớp (overfitting) và tối ưu hóa tài nguyên, **Early Stopping** đã được triển khai. Quá trình huấn luyện sẽ tự động dừng lại khi hiệu suất của mô hình trên tập kiểm định không cải thiện trong một số lượng epoch nhất định.

- Điều Chỉnh Tốc Độ Học: Để điều chỉnh tốc độ học (learning rate) trong suốt quá trình huấn luyện, nhóm sử dụng **get_linear_schedule_with_warmup** từ thư viện transformers. Scheduler này sẽ tăng tốc độ học từ từ trong giai đoạn "warmup" ban đầu, sau đó giảm tuyến tính theo thời gian, giúp mô hình hội tụ ổn định và hiệu quả hơn.

- Thuật Toán Tối Ưu Hóa: AdamW được lựa chọn làm thuật toán tối ưu hóa để cập nhật trọng số của mô hình. Đây là một biến thể của Adam, cải tiến việc xử lý thuật ngữ suy giảm trọng số (weight decay), giúp mô hình tổng quát hóa tốt hơn và tránh bị quá khớp.

- Giới Hạn Độ Lớn Gradient: Kỹ thuật **clip_grad_norm_** được áp dụng để giới hạn độ lớn của gradient, đặc biệt quan trọng trong các mạng nơ-ron sâu hoặc mạng tái phát. Điều này giúp ngăn chặn hiện tượng bùng nổ gradient, đảm bảo quá trình huấn luyện diễn ra ổn định.

- Lưu Trữ Thông Số Mô Hình để Tái Sử Dụng: Việc **lưu lại thông số của mô hình** (model checkpoints) là một bước thiết yếu. Điều này cho phép nhóm dễ dàng tải lại mô hình đã huấn luyện để thực hiện suy diễn, tiếp tục quá trình huấn luyện từ điểm dừng trước đó, hoặc so sánh và lựa chọn phiên bản mô hình tối ưu nhất.

7. Đánh giá mô hình

Sau khi huấn luyện các mô hình ở hai hướng tiếp cận (truyền thống và hiện đại), nhóm tiến hành đánh giá hiệu năng dựa trên các tiêu chí phổ biến trong bài toán phân loại nhị phân: Accuracy, Precision, Recall, F1 – score và ma trận nhầm lẫn (Confusion Matrix).

7.1. Tiêu chí đánh giá

- **Accuracy** (Độ chính xác): Tỉ lệ dự đoán đúng trên tổng số mẫu.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Precison** (Độ chính xác dương tính): Trong số những mẫu được dự đoán là "tin thật", có bao nhiêu là đúng.

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Recall** (Khả năng tìm đúng tin thật): Trong số những "tin thật" thực sự, có bao nhiêu được mô hình dự đoán đúng.

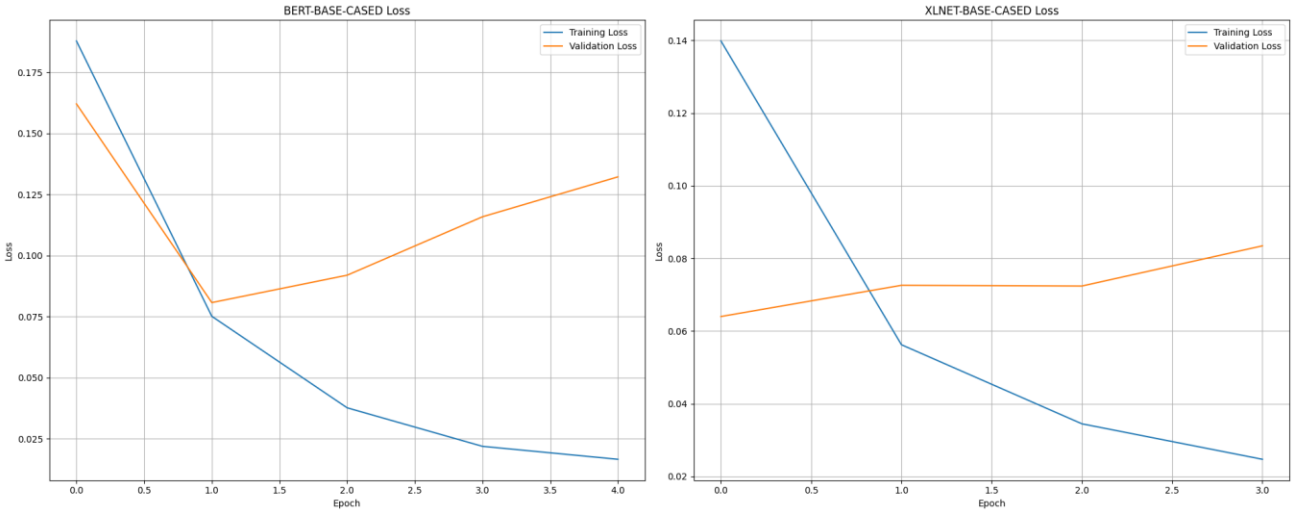
$$\text{Recall} = \frac{TP}{TP + FN}$$

- **F1 – score**: Trung bình điều hòa giữa Precision và Recall.

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

- **Confusion Matrix**: Cho biết số lượng True Positive, False Positive, True Negative, False Negative, giúp hiểu mô hình nhầm lẫn ở đâu.

7.2. Đánh giá hiệu suất mô hình dựa trên biểu đồ train/validation loss



Do cả BERT và XLNet đều được huấn luyện với cùng một bộ siêu tham số (hyperparameters), điều này đảm bảo tính công bằng và khách quan trong việc đánh giá, so sánh hiệu suất giữa hai kiến trúc mô hình. Dựa trên các biểu đồ loss, chúng ta có thể rút ra một số kết luận chi tiết hơn:

BERT-BASE-CASED:

- **Training Loss:** Giảm đều và liên tục qua các epoch, cho thấy mô hình học tốt dữ liệu huấn luyện và dần khớp với các mẫu trong tập huấn luyện.
- **Validation Loss:** Ban đầu giảm cùng với training loss (đến khoảng epoch 1). Tuy nhiên, sau **epoch 1**, validation loss bắt đầu tăng lên trong khi training loss vẫn tiếp tục giảm. Điều này rõ ràng cho thấy hiện tượng **overfitting** (mô hình học quá kỹ các chi tiết của dữ liệu huấn luyện, mất khả năng khái quát hóa trên dữ liệu chưa từng thấy).
- **Lưu model:** Model được lưu ở **epoch 2**. Đây là thời điểm khá hợp lý để dừng lại và lưu mô hình, trước khi hiện tượng overfitting trở nên quá nghiêm trọng và hiệu suất trên tập kiểm định bị suy giảm đáng kể. Quyết định này có thể dựa trên một ngưỡng kiên nhẫn nhất định của cơ chế Early Stopping.

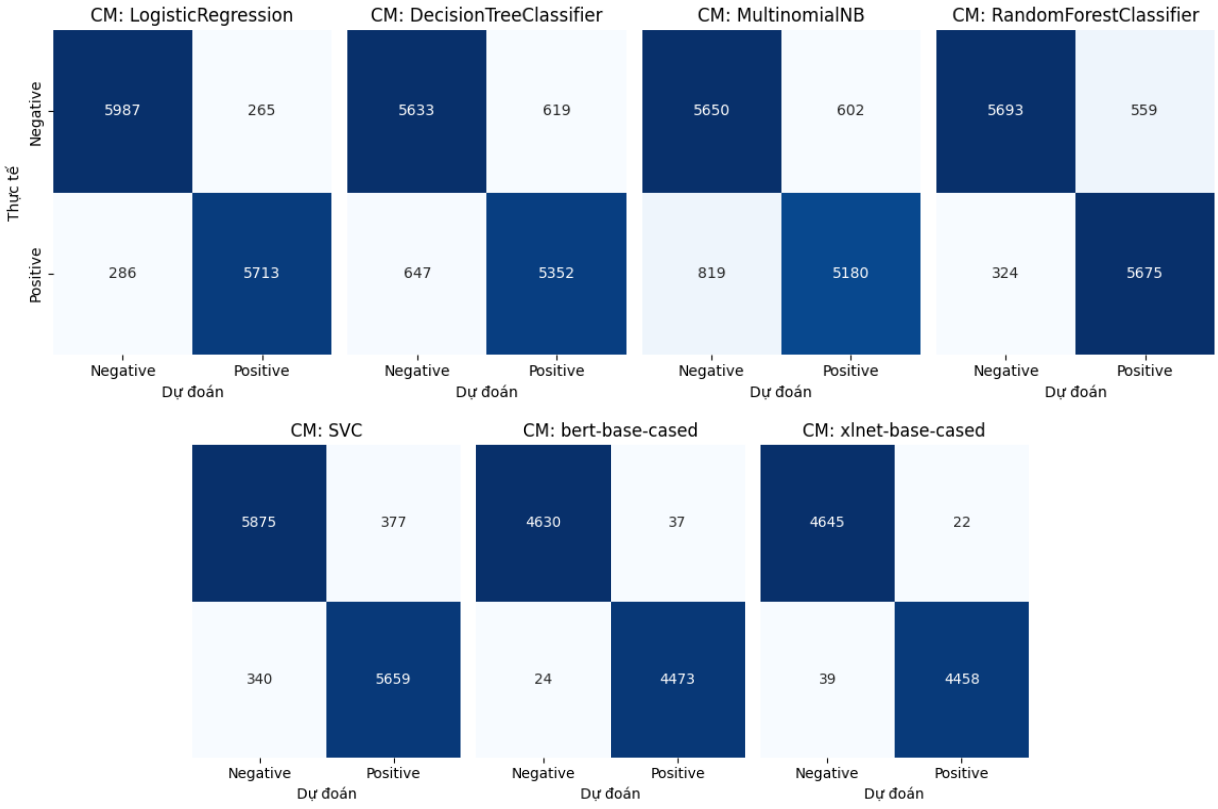
XLNET-BASE-CASED:

- **Training Loss:** Giảm đều, tương tự như BERT, cho thấy mô hình cũng đang học và thích nghi tốt với dữ liệu huấn luyện.
- **Validation Loss:** Giảm và duy trì tương đối ổn định hơn BERT trong các epoch đầu (đặc biệt từ epoch 1 đến epoch 2). Sau **epoch 2**, validation loss mới bắt đầu tăng nhẹ, cho thấy **overfitting** nhưng ít nghiêm trọng hơn và diễn ra chậm hơn so với BERT ở giai đoạn đầu của quá trình huấn luyện.
- **Lưu model:** Model được lưu ngay ở **epoch 1**. Đây là điểm tối ưu vì validation loss đạt mức thấp nhất và bắt đầu có dấu hiệu chững lại/tăng nhẹ sau đó, cho thấy mô hình đã đạt được khả năng tổng quát hóa tốt nhất tại thời điểm này.

Kết luận: Cả hai mô hình đều bị overfitting, nhưng XLNet có vẻ kiểm soát overfitting tốt hơn một chút ở các epoch đầu và đạt hiệu suất tốt nhất trên tập validation sớm hơn BERT. Việc lưu model ở các epoch tối ưu (epoch 1 cho XLNet, epoch 2 cho BERT) cho thấy chiến lược **"Ngừng huấn luyện sớm" (Early Stopping)** đã được áp dụng hiệu quả, giúp lựa chọn phiên bản mô hình tốt nhất để tránh hiện tượng học vẹt và tối ưu hóa tài nguyên huấn luyện. XLNet thể hiện tiềm năng vượt trội hơn trong việc đạt được khả năng khái quát hóa hiệu quả với số lượng epoch ít hơn.

7.3. Đánh giá hiệu suất mô hình dựa trên tập test

Mô hình	Accuracy	Precision	Recall	F1-score
Logistic Regression (TF-IDF Vectorizer)	0.9550	0.9550	0.9550	0.9550
Decision Tree Classifier (Count Vectorizer)	0.8967	0.8967	0.8967	0.8967
Multinomial Vectorizer (TF-IDF Vectorizer)	0.8840	0.8844	0.8840	0.8839
RandomForest Classifier (TF-IDF Vectorizer)	**0.9279**	**0.9286**	**0.9279**	**0.9279**
SVC (TF-IDF Vectorizer)	**0.9415**	**0.9415**	**0.9415**	**0.9415**
BERT (bert-base-cased)	**0.9933**	**0.9933**	**0.9933**	**0.9933**
XLNet (xlnet-base-cased)	**0.9933**	**0.9933**	**0.9933**	**0.9933**



Nhận xét:

Các mô hình học máy truyền thống kết hợp với kỹ thuật biểu diễn văn bản như TF-IDF hoặc CountVectorizer đều cho kết quả khá tốt:

- **Logistic Regression (TF–IDF)** đạt độ chính xác cao (Accurcacy = 95.5%) , là một mô hình đơn giản nhưng hiệu quả và ổn đinh.
- **Decision Tree (CountVectorizer)** là mô hình yếu trong số các mô hình thử nghiệm (Accuracy= 89.67%), có thể do cây quyết định dễ bị quá khớp (overfitting) và không xử lý tốt dữ liệu văn bản có chiều cao.
- **Multinomial Naive Bayes (TF–IDF)** có độ chính xác 88.50%, phù hợp với văn bản ngắn, tuy nhiên độ chính xác vẫn thấp hơn các mô hình khác do giả đinh độc lập mạnh mẽ giữa các từ.
- **Random Forest (TF–IDF)** cải thiện hơn so với Decision Tree, cho độ chính xác 92.79%, nhờ việc tổng hợp nhiều cây giảm thiểu overfitting.
- **SVC (TF–IDF)** là mô hình truyền thống có hiệu suất cao nhất với 94.15%, nhờ khả năng phân tách tốt các không gian chiều cao.

Các mô hình học sâu dựa trên Tranformers cho hiệu suất vượt trội:

- **BERT và XLNet** đều đạt Accuracy= 99.33%, gần như tuyệt đối, cho thấy khả năng hiểu ngữ cảnh và học sâu mạnh mẽ.
- Sự tương đồng về kết quả giữa BERT và XLNet cho thấy rằng ở tập dữ liệu này, cả hai đều khai thác tốt các đặc trưng ngữ nghĩa và cấu trúc của văn bản

7.4. Tổng kết:

Nếu yêu cầu mô hình đơn giản, dễ triển khai và hiệu quả cao, Logistic Regression (TF–IDF) và SVC (TF–IDF) là lựa chọn rất tốt.

Nếu không bị giới hạn về tài nguyên và cần độ chính xác cao nhất, BERT hoặc XLNet là lựa chọn tối ưu, đặc biệt phù hợp cho các hệ thống sản phẩn thực tế như chatbot, hệ thống phát hiện tin giả tự động.

8. Tài liệu tham khảo

[1] A. Ahmed, M. Traore, "Detecting Fake News Using Machine Learning and Natural Language Processing," in *Procedia Computer Science*, vol. 189, pp. 43–52, 2021.

[2] A. Kaliyar, A. Goswami, P. Narang, "FakeBERT: Fake news detection in social media with a BERT-based deep learning approach," *Multimedia Tools and Applications*, vol. 80, no. 8, pp. 11765–11788, 2021.

[3] Y. Liu et al., "RoBERTa: A Robustly Optimized BERT Pretraining Approach," *arXiv preprint*, arXiv:1907.11692, 2019.

[4] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," *arXiv preprint*, arXiv:1301.3781, 2013.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proc. NAACL-HLT*, pp. 4171–4186, 2019.

[6] H. Allcott and M. Gentzkow, "Social Media and Fake News in the 2016 Election," *Journal of Economic Perspectives*, vol. 31, no. 2, pp. 211–236, 2017.

[7] Kaggle, “Fake and real news dataset,” [Online]. Available: https://www.kaggle.com/clmentbisaillon/fake-and-real-news-dataset

[8] Scikit-learn, “TfidfVectorizer — scikit-learn documentation,” [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

[9] HuggingFace, “Transformers Documentation,” [Online]. Available: https://huggingface.co/docs/transformers/

[10] J. Zhou and R. Zafarani, "Fake News: A Survey of Research, Detection Methods, and Opportunities," *ACM Computing Surveys*, vol. 53, no. 5, pp. 1–40, 2021.