PROJECT 2B:
Phân loại chủ đề bài báo trên trang báo Dân Trí

1st Mai Thanh Phuc, 2nd Hoang Thi Yen Nhi, 3rd Tran Trong Thanh, and Le Nhat Tung
HUTECH University, Vietnam
{Mai Thanh Phuc, Hoang Thi Yen Nhi, Tran Trong Thanh}@hutech.edu.vn, and lenhattung@hutech.edu.vn

Tóm tắt nội dung

Trong bối cảnh lượng thông tin báo chí trực tuyến ngày càng gia tăng, việc tự động phân loại bài báo theo chủ đề trở thành một nhiệm vụ quan trọng trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP). Nghiên cứu này tập trung vào việc xây dựng hệ thống phân loại chủ đề bài báo tiếng Việt dựa trên hai hướng tiếp cận chính: phương pháp biểu diễn truyền thống bằng TF-IDF và phương pháp học sâu sử dụng mô hình ngôn ngữ PhoBERT. Sau khi thu thập và tiền xử lý dữ liệu, các văn bản được chuyển đổi thành dạng vector đặc trưng để huấn luyện nhiều mô hình học máy khác nhau như Logistic Regression, Random Forest, XGBoost, Naive Bayes và MLP Neural Network. Thông qua việc so sánh hai hướng biểu diễn đặc trưng, nghiên cứu nhằm đánh giá khả năng ứng dụng của các kỹ thuật học máy trong việc hiểu và phân loại văn bản tiếng Việt, đồng thời làm tiền đề cho các nghiên cứu mở rộng về Fine-tuning mô hình ngôn ngữ tiền huấn luyện trong tương lai.

Index Terms

Text Classification, Vietnamese News, TF-IDF, PhoBERT, Logistic Regression, Random Forest, XGBoost, Naive Bayes, MLP Neural Network, Feature Representation, Machine Learning, Deep Learning, Natural Language Processing (NLP).

I. Giới thiệu

Trong thời đại công nghệ thông tin phát triển mạnh mẽ, lượng dữ liệu văn bản được tạo ra và chia sẻ trên các nền tảng trực tuyến ngày càng nhiều, đặc biệt là trong lĩnh vực báo chí điện tử. Việc tự động phân loại các bài báo theo chủ đề không chỉ giúp người đọc tiếp cận thông tin nhanh hơn mà còn hỗ trợ các hệ thống quản lý nội dung, gợi ý tin tức và tổng hợp thông tin hoạt động hiệu quả hơn. Tuy nhiên, tiếng Việt là ngôn ngữ có cấu trúc phức tạp, giàu ngữ nghĩa và chịu ảnh hưởng mạnh từ ngữ cảnh, điều này khiến việc xử lý và phân loại văn bản gặp nhiều thách thức.

Trong nghiên cứu này, nhóm tác giả tập trung vào bài toán *phân loại chủ đề bài báo tiếng Việt*, cụ thể là từ tập dữ liệu được thu thập từ trang tin tức Dân Trí. Nghiên cứu áp dụng hai hướng tiếp cận chính để biểu diễn văn bản: (1) phương pháp truyền thống dựa trên thống kê tần suất từ bằng TF-IDF, và (2) phương pháp học sâu sử dụng mô hình ngôn ngữ tiền huấn luyện PhoBERT, được thiết kế chuyên biệt cho tiếng Việt.

Sau khi dữ liệu được tiền xử lý và biểu diễn thành các vector đặc trưng, nhiều mô hình học máy khác nhau được huấn luyện và đánh giá, bao gồm Logistic Regression, Random Forest, XGBoost, Naive Bayes và MLP Neural Network. Mục tiêu của nghiên cứu là so sánh hiệu quả giữa hai phương pháp biểu diễn đặc trưng, từ đó rút ra nhận định về khả năng ứng dụng của các mô hình học máy trong việc phân loại văn bản tiếng Việt.

Nghiên cứu này không chỉ góp phần làm rõ sự khác biệt giữa các phương pháp biểu diễn ngôn ngữ truyền thống và hiện đại, mà còn đặt nền tảng cho việc mở rộng sang hướng *Fine-tuning PhoBERT* hoặc các mô hình ngôn ngữ tiên tiến hơn trong tương lai. Từ đó, có thể cải thiện độ chính xác của các hệ thống phân loại tin tức, hỗ trợ xây dựng nền tảng xử lý ngôn ngữ tự nhiên tiếng Việt hiệu quả và hiện đại hơn.

II. Nghiên cứu liên quan

Bài toán phân loại văn bản đã được nghiên cứu rộng rãi trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP) và được xem là một trong những nhiệm vụ nền tảng của học máy. Các hướng tiếp cận truyền thống dựa trên đặc trưng thống kê và các phương pháp hiện đại dựa trên học sâu đều đã được áp dụng cho nhiều ngôn ngữ khác nhau, bao gồm cả tiếng Việt.

A. Phân loại văn bản truyền thống

Trong giai đoạn đầu, các mô hình thống kê như *Bag-of-Words (BoW)*, *TF-IDF (Term Frequency–Inverse Document Frequency)* và *n-grams* được sử dụng phổ biến để biểu diễn văn bản dưới dạng vector đặc trưng. Công trình của Joachims (1998) [1] là một trong những nghiên cứu tiên phong khi áp dụng SVM kết hợp TF-IDF cho bài toán phân loại văn bản tiếng Anh. Đối với tiếng Việt, phương pháp TF-IDF cũng được chứng minh mang lại hiệu quả cao trong các hệ thống phân loại tin tức [2], đặc biệt khi dữ liệu có độ nhiễu thấp và nội dung mang tính ngữ cảnh hạn chế.

B. Phân loại văn bản dựa trên học sâu

Với sự phát triển của học sâu, các mô hình ngôn ngữ tiền huấn luyện (*Pre-trained Language Models*) như BERT [3], RoBERTa [4], và đặc biệt là PhoBERT [5] đã mở ra hướng tiếp cận mới cho việc biểu diễn ngữ nghĩa của văn bản tiếng Việt. PhoBERT được phát triển bởi nhóm nghiên cứu của Nguyễn và cộng sự (2020) dựa trên kiến trúc RoBERTa, huấn luyện trên 20GB dữ liệu tiếng Việt thu thập từ Wikipedia và báo chí. Mô hình này đã đạt kết quả nổi bật trên nhiều tác vụ NLP như phân loại cảm xúc, gán nhãn thực thể (NER) và phân tích cú pháp phụ thuộc.

C. Ứng dụng học máy trong phân loại báo chí tiếng Việt

Trong những năm gần đây, nhiều nghiên cứu đã ứng dụng các mô hình học máy cho bài toán phân loại bài báo tiếng Việt. Phạm và Trần (2021) [6] sử dụng TF-IDF kết hợp với Logistic Regression và Random Forest để phân loại tin tức theo 10 chủ đề khác nhau. Trong khi đó, Nguyễn và cộng sự (2022) [7] thử nghiệm PhoBERT cho cùng bài toán và ghi nhận sự cải thiện đáng kể về độ chính xác khi so sánh với các mô hình truyền thống. Những kết quả này cho thấy sự kết hợp giữa kỹ thuật biểu diễn ngôn ngữ tiên tiến và mô hình học sâu mang lại tiềm năng lớn trong xử lý văn bản tiếng Việt.

III. Phương pháp nghiên cứu

Phần này trình bày quy trình nghiên cứu được áp dụng trong đề tài, bao gồm các bước chính: thu thập và cân bằng dữ liệu bài báo, tiền xử lý văn bản, biểu diễn đặc trưng bằng hai phương pháp (TF-IDF và PhoBERT), chia dữ liệu thành tập huấn luyện và kiểm thử, huấn luyện các mô hình học máy khác nhau, và cuối cùng là đánh giá hiệu suất phân loại dựa trên các chỉ số chuẩn. Mỗi bước được mô tả chi tiết nhằm đảm bảo tính minh bạch, khả năng tái lập và độ tin cậy của quy trình thực nghiệm.

A. Thu thập dữ liệu

Dữ liệu được thu thập từ **báo điện tử Dân Trí** — một trong những nguồn tin tức phổ biến và đa dạng chủ đề nhất tại Việt Nam. Mục tiêu là thu thập tập hợp các bài viết thuộc nhiều chuyên mục khác nhau (chính trị, kinh tế, giáo dục, sức khỏe, thể thao, giải trí, v.v.), nhằm đảm bảo dữ liệu có độ bao phủ nội dung rộng, phục vụ tốt cho quá trình phân cụm chủ đề.

1) Công cụ và ngôn ngữ lập trình: Việc thu thập dữ liệu được thực hiện bằng ngôn ngữ **Python**, sử dụng hai thư viện chính là:

- **Selenium**: mô phỏng thao tác người dùng trên trình duyệt, cho phép cuộn trang và tải động nội dung.
- **BeautifulSoup4**: dùng để trích xuất dữ liệu HTML (tiêu đề, ngày đăng, tác giả, nội dung) từ từng bài viết.

Ngoài ra, thư viện `pandas` được sử dụng để xử lý dữ liệu dạng bảng và lưu trữ kết quả cào dưới dạng **CSV**, giúp thuận tiện cho các bước tiền xử lý sau này.

2) Quy trình cào dữ liệu: Quá trình thu thập được chia làm hai giai đoạn:

a) (1) Cào danh sách bài viết theo chuyên mục.: Trình duyệt Chrome được chạy ở chế độ `headless` để tăng tốc độ và tiết kiệm tài nguyên. Mỗi chuyên mục trên trang `https://dantri.com.vn` được tự động duyệt qua tối đa 15 trang (cuộn 10 lần mỗi trang) để thu thập các liên kết bài viết, tiêu đề, chuyên mục và thời điểm cào. Hệ thống tự động loại bỏ trùng lặp bằng cách đối chiếu với dữ liệu đã có, sau đó lưu kết quả vào tệp **CSV** kèm timestamp.

`dantri_new_unique_2025-10-12_00-50-08.csv`

b) (2) Cào nội dung chi tiết từng bài báo.: Ở giai đoạn thứ hai, mỗi liên kết thu được từ bước trên được truy cập riêng lẻ bằng thư viện `requests`. Dữ liệu chi tiết được trích xuất gồm:

- **Tiêu đề bài viết (title)**
- **Ngày đăng (pub_date)**
- **Tác giả (author)**
- **Nội dung (content)** – gồm toàn bộ phần thân bài dưới thẻ HTML `<p>` trong vùng `singular-content`.

Để đảm bảo tính ổn định, chương trình thực hiện:

- Tạm dừng ngẫu nhiên (`time.sleep(random.uniform(0.5, 1.2))`) giữa các yêu cầu để tránh bị chặn IP.
- Bỏ qua các bài không tải được hoặc thiếu nội dung chính.
- Gộp dữ liệu đã cào mới với danh sách link cũ bằng `pandas.merge()` theo khóa "link".

Sau khi hoàn tất, toàn bộ dữ liệu được lưu lại trong một tệp CSV tổng hợp, ví dụ:

`dantri_crawl_full_2025-10-12_01-40-30.csv`

c) (3) Quy mô dữ liệu.: Tổng cộng có **7.580 bài viết** được thu thập, trải rộng trên hơn 20 chuyên mục, bao gồm cả các chủ đề phổ biến như thời sự, thể thao, sức khỏe, giải trí và kinh doanh. Mỗi bản ghi trong dữ liệu gồm các trường:

`[category, title, link, pub_date, author, content, crawl_date]`

3) Đánh giá chất lượng dữ liệu: Sau khi cào xong, dữ liệu được kiểm tra và làm sạch sơ bộ nhằm loại bỏ:

- Các bài viết trùng lặp hoặc bị lỗi khi tải nội dung.
- Những bài có phần nội dung quá ngắn (dưới 30 ký tự).
- Các bài không có tiêu đề hoặc ngày đăng.

Quy trình này giúp đảm bảo dữ liệu đầu vào có chất lượng tốt, phục vụ hiệu quả cho các bước tiền xử lý ngôn ngữ và biểu diễn đặc trưng tiếp theo.

B. Tiền xử lý dữ liệu

Sau khi thu thập được dữ liệu thô từ báo điện tử Dân Trí, bước tiếp theo là tiến hành **tiền xử lý dữ liệu văn bản** nhằm loại bỏ nhiễu, chuẩn hóa ngôn ngữ và chuyển đổi nội dung thành dạng có thể xử lý bằng mô hình học máy. Toàn bộ quá trình được thực hiện bằng ngôn ngữ **Python**, sử dụng các thư viện chính như `pandas`, `re`, `underthesea`, `tqdm` và `matplotlib`.

1) Đọc và lựa chọn dữ liệu đầu vào: Từ tệp dữ liệu gốc `dantri_crawl_full.csv`, chỉ hai trường quan trọng được giữ lại là:

- **category**: chuyên mục của bài viết.
- **content**: phần nội dung chi tiết của bài báo.

2) Làm sạch văn bản: Mỗi bài viết được xử lý thông qua hàm `clean_text()` với các bước cụ thể như sau:

- **Chuyển toàn bộ văn bản về chữ thường** nhằm giảm sự phân biệt không cần thiết giữa chữ hoa và chữ thường.
- **Loại bỏ liên kết URL** bằng biểu thức chính quy để tránh nhiễu từ các đường dẫn trong bài viết.
- **Xóa ký tự đặc biệt, số, emoji và dấu câu**, chỉ giữ lại các ký tự chữ cái và khoảng trắng.
- **Chuẩn hóa khoảng trắng** giúp loại bỏ các dấu cách thừa.
- **Tách từ tiếng Việt (word tokenization)** bằng thư viện `Underthesea` ở định dạng văn bản (`format="text"`), giúp xác định rõ ràng ranh giới giữa các từ đơn và từ ghép.

Kết quả của bước này được lưu vào cột mới `clean_text` trong bảng dữ liệu.

3) Loại bỏ từ dừng (Stopwords): Danh sách **stopwords tiếng Việt** được tải từ tệp `vietnamese-stopwords.txt`, bao gồm các từ phổ biến nhưng không mang ý nghĩa phân biệt chủ đề (ví dụ: "là", "và", "những", "các", "được", "của", "một"). Hàm `remove_stopwords()` được áp dụng để loại bỏ toàn bộ các từ này ra khỏi mỗi câu, giúp giảm nhiễu và tăng tính tập trung cho mô hình biểu diễn từ.

4) Loại bỏ trùng lặp và câu ngắn: Sau khi làm sạch, dữ liệu được xử lý để loại bỏ:

- **Các văn bản trùng nội dung**: sử dụng hàm `drop_duplicates()` dựa trên cột `clean_text`.
- **Các văn bản quá ngắn**: loại bỏ những dòng có ít hơn 8 từ (`len(str(x).split()) < 8`) nhằm đảm bảo mỗi mẫu dữ liệu có đủ ngữ cảnh phục vụ cho việc trích xuất đặc trưng.

Sau khi lọc, tập dữ liệu còn lại **7.278 bài viết** hợp lệ.

5) Cân bằng dữ liệu bằng kỹ thuật Oversampling: Phân bố dữ liệu giữa các chuyên mục (*category*) ban đầu không đồng đều — một số chuyên mục như *Thời sự* và *Kinh doanh* chiếm tỷ lệ lớn, trong khi các chuyên mục khác như *Tình yêu – Giới tính* hay *Infographic* có rất ít mẫu. Để khắc phục hiện tượng mất cân bằng lớp, kỹ thuật **Random Oversampling** được áp dụng nhằm nhân bản ngẫu nhiên các mẫu thuộc lớp thiểu số cho đến khi tất cả các lớp có số lượng tương đương nhau. Quá trình này được thực hiện bằng thư viện `imbalanced-learn` (`imblearn.over_sampling.RandomOverSampler`).

Kết quả sau khi cân bằng cho thấy mỗi chuyên mục có số lượng ít được tăng thêm mẫu, tạo thành tập dữ liệu `dantri_balanced.csv`. Việc cân bằng dữ liệu giúp mô hình học máy không bị thiên lệch khi huấn luyện và cải thiện đáng kể khả năng phân loại các lớp nhỏ.

6) Lưu kết quả tiền xử lý: Toàn bộ dữ liệu sau khi làm sạch và cân bằng được lưu vào tệp:

`/content/drive/MyDrive/project2/project2b/dantri_balanced.csv`

Tệp này sẽ là đầu vào cho bước biểu diễn đặc trưng văn bản (embedding) ở các phương pháp TF-IDF và PhoBERT.

C. Biểu diễn đặc trưng văn bản (Embedding)

Sau khi dữ liệu được tiền xử lý và cân bằng, bước tiếp theo là **biểu diễn văn bản dưới dạng vector số (embedding)** nhằm giúp mô hình học máy hiểu và xử lý được ngữ nghĩa của ngôn ngữ tự nhiên. Trong nghiên cứu này, hai phương pháp biểu diễn phổ biến được sử dụng là **TF-IDF** (phương pháp truyền thống dựa trên thống kê tần suất từ) và **PhoBERT** (mô hình ngôn ngữ hiện đại được huấn luyện trước cho tiếng Việt).

1) Biểu diễn bằng TF-IDF: Phương pháp *Term Frequency–Inverse Document Frequency (TF-IDF)* [8] là một kỹ thuật thống kê nhằm đánh giá tầm quan trọng của một từ trong toàn bộ tập văn bản. Giá trị TF-IDF của một từ t trong văn bản d được tính theo công thức:

$$TF\text{-}IDF(t,d) = TF(t,d) \times \log\left(\frac{N}{DF(t)}\right)$$

trong đó:

- $TF(t,d)$: tần suất xuất hiện của từ t trong văn bản d.
- $DF(t)$: số lượng văn bản có chứa từ t.
- N: tổng số văn bản trong tập dữ liệu.

Giá trị TF-IDF càng cao thể hiện rằng từ đó mang tính phân biệt tốt giữa các văn bản khác nhau. Trong thực nghiệm, mô hình `TfidfVectorizer` từ thư viện `scikit-learn` [9] được sử dụng với các tham số:

- `max_features = 3000`: chỉ chọn 3.000 đặc trưng có giá trị thông tin cao nhất.
- `ngram_range = (1, 2)`: xét cả từ đơn và cụm từ hai từ (unigram và bigram).
- `sublinear_tf = True`: áp dụng log-scaling để giảm ảnh hưởng của các từ xuất hiện quá thường xuyên.

Kết quả là mỗi bài viết được mã hoá thành một vector chiều 3.000, lưu trong tệp `tfidf_embeddings.npy`. Phương pháp TF-IDF có ưu điểm là đơn giản, dễ huấn luyện và hiệu quả tốt khi dữ liệu được làm sạch kỹ lưỡng.

2) Biểu diễn bằng PhoBERT: Khác với TF-IDF chỉ dựa trên tần suất, **PhoBERT** là mô hình ngôn ngữ tiền huấn luyện (Pre-trained Language Model) dành riêng cho tiếng Việt, được xây dựng dựa trên kiến trúc *RoBERTa* [4] và được phát triển bởi Viện nghiên cứu Trí tuệ nhân tạo VinAI [5]. PhoBERT được huấn luyện trên hơn 20GB dữ liệu tiếng Việt, giúp mô hình học được các biểu diễn ngữ nghĩa sâu và phù hợp hơn cho bài toán xử lý ngôn ngữ tự nhiên tiếng Việt.

Trong nghiên cứu này, mô hình `vinai/phobert-base` được tải từ thư viện `transformers` [10] và sử dụng lớp đầu ra `pooler_output` để trích xuất vector biểu diễn cho toàn bộ câu. Quy trình tạo embedding như sau:

1) Mỗi văn bản được mã hoá bằng `AutoTokenizer` với độ dài tối đa `max_length = 256`.
2) Dữ liệu đầu vào được đưa qua mô hình `AutoModel` để thu được đầu ra ẩn.
3) Giá trị `pooler_output` (hoặc trung bình các token nếu không có) được sử dụng làm vector đặc trưng cho văn bản.

Kết quả của PhoBERT là các vector có chiều 768, lưu trong tệp `phobert_embeddings.npy`. Nhờ khả năng học ngữ nghĩa sâu, PhoBERT cung cấp đặc trưng phù hợp hơn cho các bài toán phân loại văn bản và nhận dạng chủ đề.

3) So sánh hai phương pháp biểu diễn:

- **TF-IDF**: dựa trên thống kê tần suất, phù hợp với dữ liệu có ngữ cảnh ngắn, ít tốn tài nguyên tính toán và huấn luyện nhanh.
- **PhoBERT**: dựa trên ngữ nghĩa toàn văn, học được quan hệ giữa các từ trong câu, cho kết quả tốt hơn trong các bài toán hiểu ngữ cảnh, nhưng yêu cầu tài nguyên cao hơn.

Việc sử dụng song song hai phương pháp này giúp đánh giá toàn diện hiệu quả giữa hướng tiếp cận truyền thống và hướng hiện đại dựa trên mô hình ngôn ngữ tiền huấn luyện.

D. Chia tập huấn luyện và kiểm thử

Sau khi dữ liệu được biểu diễn dưới dạng vector số (embedding), bước tiếp theo là **phân chia dữ liệu thành hai tập: huấn luyện (train) và kiểm thử (test)** nhằm đánh giá khách quan hiệu quả của mô hình học máy. Quá trình này được thực hiện bằng hàm `train_test_split()` trong thư viện `scikit-learn` [9].

Tỷ lệ chia được lựa chọn là **80% cho huấn luyện** và **20% cho kiểm thử**. Cụ thể:

- **Tập huấn luyện (Train set)**: dùng để huấn luyện các mô hình học máy trên dữ liệu đã biết.
- **Tập kiểm thử (Test set)**: dùng để đánh giá độ tổng quát (generalization) của mô hình đối với dữ liệu chưa từng thấy trước đó.

Để đảm bảo các chuyên mục (nhãn) trong tập dữ liệu được phân bố đồng đều ở cả hai tập, tham số `stratify=y` được sử dụng trong hàm chia dữ liệu. Điều này giúp tránh tình trạng mất cân bằng lớp trong tập kiểm thử, vốn có thể làm sai lệch kết quả đánh giá mô hình.

Quá trình được áp dụng cho cả hai dạng biểu diễn:

1) **PhoBERT embedding:**
   `train_test_split(emb_phobert, y_pho, test_size=0.2, stratify=y_pho, random_state=42)`
2) **TF-IDF embedding:**
   `train_test_split(emb_tfidf, y_tfidf, test_size=0.2, stratify=y_tfidf, random_state=42)`

Kết quả thu được:

- **PhoBERT:** khoảng 6.142 mẫu huấn luyện và 1.536 mẫu kiểm thử, mỗi vector có chiều 768.
- **TF-IDF:** tương tự về số lượng, mỗi vector có chiều 3.000.

Việc sử dụng `random_state=42` đảm bảo tính tái lập của thí nghiệm, tức là cùng một bộ dữ liệu và tham số sẽ luôn tạo ra cùng kết quả chia.

E. Chuẩn hóa dữ liệu

Trước khi đưa dữ liệu vào huấn luyện, các vector đặc trưng cần được **chuẩn hóa (scaling)** nhằm đưa các giá trị về cùng thang đo và giảm độ chênh lệch giữa các chiều đặc trưng. Việc chuẩn hóa giúp quá trình huấn luyện mô hình học máy trở nên ổn định hơn, tránh tình trạng một số đặc trưng chi phối quá trình tối ưu do có biên độ giá trị lớn hơn [11].

1) Chuẩn hóa dữ liệu PhoBERT: Các vector embedding sinh ra từ PhoBERT có phân phối giá trị không đồng đều giữa các chiều, vì vậy cần được chuẩn hóa bằng **StandardScaler** trong thư viện `scikit-learn` [9]. Phương pháp này áp dụng chuẩn hóa z-score theo công thức:

$$z = \frac{x - \mu}{\sigma}$$

với μ là giá trị trung bình và σ là độ lệch chuẩn của từng đặc trưng trong tập huấn luyện.

Trong quá trình thực hiện:

- Bộ `StandardScaler()` được **fit** trên tập huấn luyện để tính toán μ và σ.
- Sau đó áp dụng **transform** cho cả tập huấn luyện và kiểm thử để đảm bảo không xảy ra rò rỉ dữ liệu (data leakage).

Cụ thể trong mã nguồn:

```
scaler_pho = StandardScaler()
X_train_pho_scaled = scaler_pho.fit_transform(X_train_pho)
X_test_pho_scaled  = scaler_pho.transform(X_test_pho)
```

Điều này đảm bảo dữ liệu đầu vào của PhoBERT phân bố đồng đều quanh gốc tọa độ, giúp các mô hình như Logistic Regression, MLP hay XGBoost hội tụ nhanh và ổn định hơn.

2) Giữ nguyên dữ liệu TF-IDF: Ngược lại, đối với TF-IDF, các vector đã được chuẩn hóa theo chuẩn L2 (`norm='l2'`) ngay trong quá trình tính toán. Do đó, bước scale không cần thiết nữa — chỉ cần sao chép dữ liệu gốc để đưa vào mô hình:

```
X_train_tfidf_scaled = X_train_tfidf.copy()
X_test_tfidf_scaled  = X_test_tfidf.copy()
```

Việc giữ nguyên TF-IDF giúp bảo toàn mối quan hệ giữa các từ trong không gian vector, đồng thời tránh làm thay đổi ý nghĩa của trọng số TF-IDF ban đầu.

3) Tổng kết:

- PhoBERT: được chuẩn hóa bằng `StandardScaler` để đưa các giá trị về phân phối chuẩn.
- TF-IDF: giữ nguyên vì đã được chuẩn hóa L2 trong quá trình vector hóa.

Bước chuẩn hóa dữ liệu là giai đoạn quan trọng, đảm bảo tính đồng nhất giữa các đặc trưng đầu vào và giúp cải thiện hiệu suất, độ ổn định của các mô hình học máy.

F. Huấn luyện mô hình

Sau khi dữ liệu đã được biểu diễn dưới dạng vector đặc trưng và chuẩn hóa, bước tiếp theo là **huấn luyện các mô hình học máy** nhằm phân loại chủ đề bài báo dựa trên nội dung văn bản. Mục tiêu của giai đoạn này là xây dựng các mô hình có khả năng học được mối quan hệ giữa đặc trưng văn bản và nhãn chuyên mục, từ đó dự đoán chính xác chủ đề của các bài viết mới.

Toàn bộ quá trình huấn luyện được thực hiện bằng thư viện `scikit-learn` [9] và `xgboost` [12], với năm thuật toán được lựa chọn nhằm so sánh hiệu suất giữa các nhóm mô hình tuyến tính, phi tuyến và mạng nơ-ron.

1) Logistic Regression: **Logistic Regression** [13] là một mô hình tuyến tính được sử dụng phổ biến cho bài toán phân loại. Mô hình học một hàm sigmoid để ước lượng xác suất một mẫu x thuộc về lớp y:

$$P(y = 1|x) = \frac{1}{1 + e^{-(w^T x + b)}}$$

Trong bài toán đa lớp, phương pháp **One-vs-Rest (OvR)** được sử dụng. Mô hình này có ưu điểm là đơn giản, dễ huấn luyện và hoạt động tốt với dữ liệu có tính tuyến tính. Trong thí nghiệm, Logistic Regression cho kết quả nổi bật khi kết hợp với TF-IDF.

2) Random Forest: **Random Forest** [14] là thuật toán học tập tổ hợp (ensemble learning), sử dụng nhiều cây quyết định (Decision Trees) huấn luyện trên các mẫu ngẫu nhiên và trích chọn đặc trưng ngẫu nhiên. Dự đoán cuối cùng được quyết định dựa trên bỏ phiếu đa số giữa các cây:

$$\hat{y} = \text{mode}\{h_1(x), h_2(x), ..., h_n(x)\}$$

Thuật toán giúp giảm phương sai (variance), tránh hiện tượng quá khớp (overfitting) và thích hợp cho dữ liệu phi tuyến.

3) XGBoost: **XGBoost** (Extreme Gradient Boosting) [12] là phiên bản nâng cao của Gradient Boosting, tối ưu hiệu suất tính toán và khả năng tổng quát hóa. Mỗi cây mới được huấn luyện để sửa lỗi còn lại của cây trước đó:

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + \eta f_t(x_i)$$

Trong đó η là learning rate và f_t là cây quyết định tại bước lặp t. XGBoost được sử dụng với tham số `max_depth=8`, `learning_rate=0.05`, `n_estimators=300`, và `eval_metric="mlogloss"`.

4) Naive Bayes: **Naive Bayes** [15] là thuật toán phân loại dựa trên Định lý Bayes, giả định các đặc trưng độc lập có điều kiện:

$$P(y|x_1, x_2, ..., x_n) \propto P(y) \prod_{i=1}^{n} P(x_i|y)$$

Mặc dù đơn giản, Naive Bayes thường đạt hiệu quả cao với dữ liệu văn bản có số chiều lớn và thưa (như TF-IDF). Trong nghiên cứu này, mô hình `GaussianNB()` được áp dụng.

5) Multilayer Perceptron (MLP): **MLP (Multilayer Perceptron)** [16] là mạng nơ-ron nhân tạo nhiều lớp, có khả năng mô hình hóa các quan hệ phi tuyến giữa đặc trưng và nhãn. Mô hình bao gồm:

$$h = \sigma(W_1 x + b_1), \quad \hat{y} = \text{softmax}(W_2 h + b_2)$$

với σ là hàm kích hoạt ReLU. Trong thí nghiệm, MLP sử dụng hai tầng ẩn với 512 và 256 nơ-ron, hàm kích hoạt ReLU, và số vòng lặp tối đa `max_iter=300`.

6) Quy trình huấn luyện: Mỗi mô hình được huấn luyện độc lập trên hai dạng biểu diễn đặc trưng:

- **PhoBERT scaled:** đầu vào là embedding có kích thước 768, được chuẩn hóa bằng StandardScaler.
- **TF-IDF:** đầu vào là vector có 3.000 đặc trưng, giữ nguyên giá trị gốc.

Các tham số ngẫu nhiên được cố định với `random_state=42` để đảm bảo tính tái lập của kết quả.

G. Chỉ số đánh giá

Để đánh giá hiệu quả của các mô hình phân loại chủ đề bài báo, nhóm sử dụng bốn chỉ số chính: **Accuracy**, **Precision**, **Recall**, **F1-score** và ma trận nhầm lẫn (**Confusion Matrix**). Những chỉ số này giúp phản ánh toàn diện khả năng dự đoán đúng của mô hình trên từng lớp, đặc biệt khi dữ liệu không cân bằng giữa các chuyên mục [17].

1) Độ chính xác (Accuracy): **Accuracy** đo lường tỷ lệ mẫu được dự đoán đúng trên tổng số mẫu:

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

trong đó:

- TP: số mẫu dương được dự đoán đúng (True Positive),
- TN: số mẫu âm được dự đoán đúng (True Negative),
- FP: số mẫu âm bị dự đoán nhầm thành dương (False Positive),
- FN: số mẫu dương bị dự đoán nhầm thành âm (False Negative).

Đây là chỉ số tổng quát, tuy nhiên có thể gây sai lệch khi dữ liệu mất cân bằng.

2) Độ chính xác dự đoán (Precision): **Precision** phản ánh mức độ "tin cậy" của mô hình khi dự đoán một mẫu thuộc về một lớp cụ thể:

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

Giá trị Precision cao thể hiện rằng mô hình ít đưa ra các dự đoán sai dương tính (False Positive), phù hợp trong bài toán mà sai nhầm về chủ đề gây ảnh hưởng nghiêm trọng (ví dụ: nhầm "Pháp luật" với "Giải trí").

3) Độ bao phủ (Recall): **Recall** đo lường khả năng phát hiện toàn bộ các mẫu thực sự thuộc về một lớp:

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

Recall cao cho thấy mô hình ít bỏ sót các mẫu dương tính, tức là nhận diện được hầu hết các bài viết đúng chủ đề.

4) Chỉ số F1-score: **F1-score** là trung bình điều hòa giữa Precision và Recall, thể hiện sự cân bằng giữa hai yếu tố:

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Khi Precision hoặc Recall thấp, F1-score cũng giảm tương ứng, giúp mô hình không thiên lệch về một trong hai chỉ số.

5) Ma trận nhầm lẫn (Confusion Matrix): **Confusion Matrix** là công cụ trực quan hóa kết quả phân loại, hiển thị số lượng dự đoán đúng và sai giữa các lớp. Ma trận có kích thước $n \times n$ với n là số lượng chuyên mục:

$$\text{CM}_{i,j} = \Big\{\text{số mẫu thuộc lớp } i \text{ nhưng được dự đoán là } j$$

Đường chéo chính ($i = j$) biểu thị số lượng dự đoán đúng cho từng lớp, trong khi các ô ngoài đường chéo thể hiện lỗi phân loại. Trong nghiên cứu này, ma trận nhầm lẫn được trực quan hóa bằng `seaborn.heatmap` để dễ quan sát tỷ lệ sai lệch giữa các chủ đề.

IV. Thiết lập thí nghiệm

A. Môi trường thực nghiệm

Toàn bộ quá trình thực nghiệm được thực hiện trên máy tính cá nhân với cấu hình như sau:

Bảng I
Cấu hình môi trường thực nghiệm

Thành phần	Thông tin
Hệ điều hành	Windows 11 64-bit
Bộ xử lý (CPU)	Intel Core i7-12700H (14 nhân, 20 luồng)
RAM	16 GB DDR5
Ngôn ngữ lập trình	Python 3.13.2
IDE	PyCharm Community Edition 2022.2.3

Thư viện chính sử dụng: Các mô hình được phát triển và huấn luyện bằng ngôn ngữ lập trình **Python**, sử dụng các thư viện phổ biến trong lĩnh vực xử lý ngôn ngữ tự nhiên và học máy:

- **pandas, numpy**: xử lý và thao tác dữ liệu dạng bảng.
- **scikit-learn** [9]: triển khai các thuật toán học máy truyền thống như Logistic Regression, Random Forest, Naive Bayes và MLP.
- **xgboost** [12]: huấn luyện mô hình tăng cường cây quyết định (Gradient Boosted Trees).
- **transformers** (HuggingFace): tải và sử dụng mô hình ngôn ngữ tiền huấn luyện PhoBERT.
- **matplotlib, seaborn**: trực quan hóa dữ liệu và kết quả mô hình.
- **underthesea**: tách từ tiếng Việt và tiền xử lý văn bản.

B. Dữ liệu đầu vào

Nguồn dữ liệu được sử dụng trong nghiên cứu là tập các bài viết tiếng Việt thu thập từ trang báo điện tử **Dân Trí** (https://dantri.com.vn). Dữ liệu phản ánh đa dạng các chủ đề thuộc nhiều lĩnh vực khác nhau trong đời sống xã hội Việt Nam, từ đó phù hợp cho bài toán phân loại văn bản đa lớp.

1) Thu thập và lựa chọn dữ liệu: Dữ liệu được thu thập tự động thông qua kỹ thuật **Web Scraping** bằng thư viện `Selenium` và `BeautifulSoup`, lưu vào tệp `dantri_crawl_full.csv`. Mỗi bản ghi chứa các thông tin chính sau:

- **category** – chuyên mục của bài viết (ví dụ: "Thời sự", "Giáo dục", "Thể thao", "Giải trí", "Kinh doanh"...).
- **title** – tiêu đề của bài viết.
- **content** – nội dung chi tiết của bài báo.
- **link** – đường dẫn đến bài viết gốc.

Trong quá trình xử lý, chỉ hai cột **category** và **content** được giữ lại để phục vụ cho bài toán phân loại. Các bản ghi bị thiếu dữ liệu, trùng lặp hoặc có nội dung ngắn dưới 8 từ bị loại bỏ để đảm bảo chất lượng đầu vào. Sau bước lọc, tập dữ liệu được tăng mẫu nhằm ổn định cho việc huấn luyện với số bài là **7.678 bài viết** hợp lệ.

2) Cấu trúc và phân bố dữ liệu: Mỗi bài viết thuộc về một trong 18 chuyên mục chính, phản ánh các lĩnh vực báo chí phổ biến của Việt Nam. Bảng II trình bày phân bố số lượng bài viết theo từng chuyên mục trước khi cân bằng dữ liệu.

Bảng II
PHÂN BỐ DỮ LIỆU GỐC THEO TỪNG CHUYÊN MỤC TRÊN BÁO DÂN TRÍ

Chuyên mục	**Số lượng bài viết**
Thời sự	661
Giáo dục	641
Giải trí	576
Sức khỏe	570
Thể thao	527
Kinh doanh	502
Du lịch	480
Việc làm	480
Ô tô - Xe máy	439
Bạn đọc	426
Pháp luật	413
Tình yêu - Giới tính	412
Thế giới	395
Infographic	298
Tâm điểm	253
Xã hội	206
Văn hóa	201
Nhịp sống trẻ	198
Tổng cộng	**7.678**

3) Cân bằng dữ liệu bằng Oversampling

Quan sát bảng phân bố cho thấy dữ liệu bị mất cân bằng, trong đó một số chuyên mục như "Văn hóa", "Xã hội" và "Nhịp sống trẻ" có số lượng mẫu thấp hơn đáng kể so với các nhóm khác. Để khắc phục hiện tượng này, nghiên cứu áp dụng kỹ thuật **Oversampling** bằng hàm `resample()` trong thư viện `scikit-learn`, giúp nhân bản ngẫu nhiên các mẫu thiểu số đến khi đạt được kích thước mong muốn. Việc này giúp mô hình học máy có khả năng học đều trên các lớp và tránh hiện tượng thiên lệch về các lớp chiếm đa số.

Sau khi tăng cường dữ liệu, tập dữ liệu cân bằng được lưu thành tệp:

`/content/drive/MyDrive/project2/project2b/dantri_balanced.csv`

Tập dữ liệu này được sử dụng làm đầu vào cho bước biểu diễn đặc trưng văn bản (embedding) ở phần tiếp theo.

C. Thuật toán và quy trình thực hiện

Phần này trình bày tổng quan các thuật toán và quy trình thực nghiệm được sử dụng trong đề tài. Hệ thống được thiết kế theo chuỗi xử lý dữ liệu gồm 6 giai đoạn chính: (1) thu thập dữ liệu,
(2) tiền xử lý văn bản,
(3) biểu diễn đặc trưng (embedding),
(4) chia dữ liệu train/test,
(5) huấn luyện mô hình,
(6) đánh giá kết quả.

1. Mô tả quy trình thực hiện

Quy trình được thể hiện trong Hình 1, mô tả toàn bộ luồng xử lý từ dữ liệu thô đến kết quả đánh giá mô hình.

Hình 1. Sơ đồ quy trình thực hiện đề tài

2. Các thuật toán được sử dụng

Trong nghiên cứu, năm mô hình học máy được lựa chọn nhằm so sánh hiệu quả phân loại giữa hai phương pháp biểu diễn đặc trưng TF-IDF và PhoBERT:

- **Logistic Regression (LR)** – mô hình tuyến tính dựa trên hàm sigmoid, thích hợp cho bài toán phân loại đa lớp [13].
- **Random Forest (RF)** – mô hình tổ hợp (ensemble) dựa trên nhiều cây quyết định nhằm giảm overfitting và tăng độ chính xác [14].
- **XGBoost** – thuật toán tăng cường gradient hiệu quả, tối ưu hóa tốc độ huấn luyện và khả năng tổng quát hóa [12].
- **Naive Bayes (NB)** – thuật toán xác suất dựa trên định lý Bayes, phù hợp cho dữ liệu văn bản có đặc trưng độc lập [15].
- **MLP Neural Network** – mạng nơ-ron nhiều lớp (Multilayer Perceptron), có khả năng học các mối quan hệ phi tuyến giữa đặc trưng và nhãn [**?**].

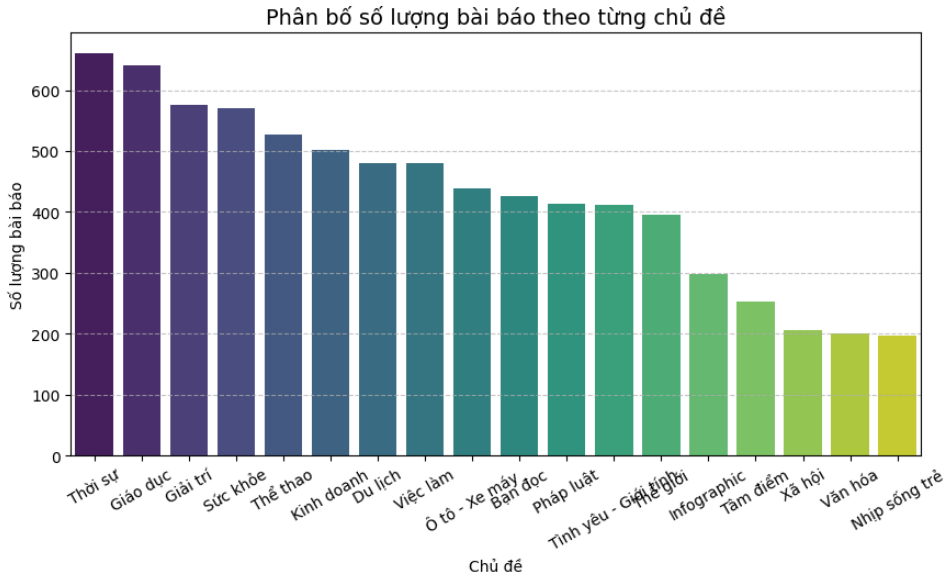
Mỗi mô hình được huấn luyện độc lập trên hai dạng embedding (TF-IDF và PhoBERT) nhằm đánh giá ảnh hưởng của phương pháp biểu diễn đặc trưng đến hiệu suất phân loại.

V. Kết quả và thảo luận

A. Tổng quan dữ liệu

Tập dữ liệu được sử dụng trong nghiên cứu bao gồm các bài viết thu thập từ báo điện tử **Dân Trí**, sau khi làm sạch và xử lý còn lại **7.678 bài viết** thuộc **18 chuyên mục khác nhau**. Mỗi bài viết được gán nhãn chuyên mục duy nhất, phản ánh các

lĩnh vực phổ biến trong đời sống xã hội Việt Nam như chính trị, giáo dục, thể thao, giải trí, kinh tế, sức khỏe, v.v.



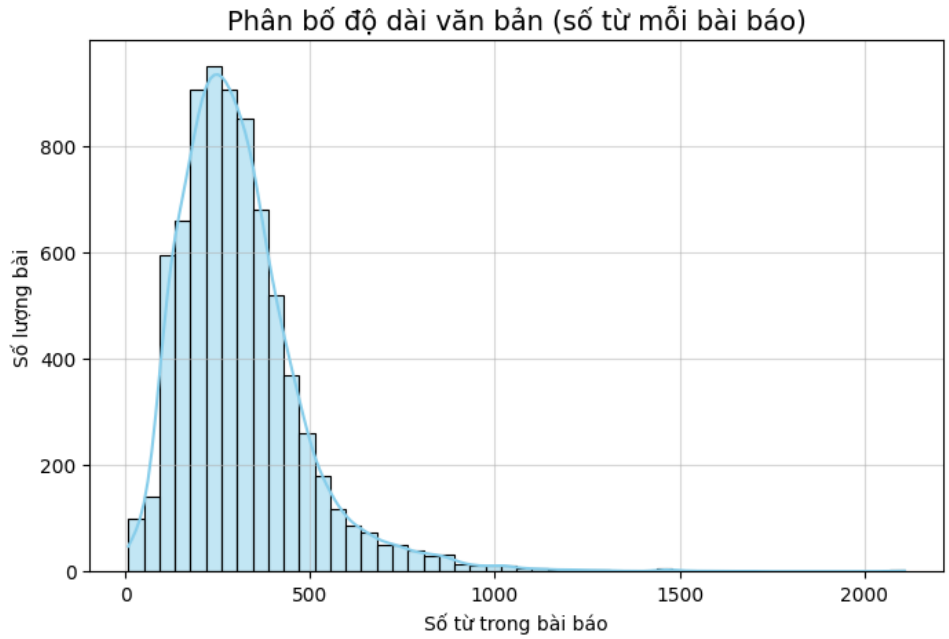
Hình 2. Phân bố số lượng bài báo theo từng chủ đề

Kết quả thống kê ở Bảng III cho thấy sự chênh lệch đáng kể giữa các chuyên mục. Những nhóm chủ đề "Thời sự" (8.6%) và "Giáo dục" (8.3%) chiếm tỷ trọng lớn nhất, trong khi "Văn hóa" và "Nhịp sống trẻ" chỉ chiếm khoảng 2.6%. Điều này thể hiện xu hướng nội dung của báo Dân Trí tập trung nhiều vào các vấn đề thời sự - chính trị và các lĩnh vực giáo dục, y tế, thể thao, trong khi các mảng văn hóa – đời sống chiếm tỉ trọng nhỏ hơn.

Bảng III
TỶ LỆ PHẦN TRĂM SỐ LƯỢNG BÀI VIẾT THEO CHUYÊN MỤC

Chuyên mục	**Tỷ lệ (%)**
Thời sự	8.6
Giáo dục	8.3
Giải trí	7.5
Sức khỏe	7.4
Thể thao	6.9
Kinh doanh	6.5
Du lịch	6.3
Việc làm	6.3
Ô tô - Xe máy	5.7
Bạn đọc	5.5
Pháp luật	5.4
Tình yêu - Giới tính	5.4
Thế giới	5.1
Infographic	3.9
Tâm điểm	3.3
Xã hội	2.7
Văn hóa	2.6
Nhịp sống trẻ	2.6

Về độ dài văn bản, Hình 3 cho thấy phân bố độ dài bài viết có dạng **phân phối lệch phải** (right-skewed distribution). Phần lớn các bài viết có độ dài trung bình từ **100 đến 400 từ**, phù hợp cho các phương pháp biểu diễn đặc trưng dựa trên tần suất như TF-IDF cũng như các mô hình ngữ nghĩa sâu như PhoBERT.



Hình 3. Phân bố độ dài văn bản (số từ trong mỗi bài báo)

Kết hợp giữa phân bố chủ đề và độ dài văn bản, có thể nhận thấy tập dữ liệu mang tính đa dạng cao về nội dung nhưng vẫn tồn tại sự chênh lệch nhẹ giữa các nhóm tin tức. Do đó, bước **cân bằng dữ liệu (oversampling)** ở giai đoạn tiền xử lý là cần thiết nhằm tránh hiện tượng *class imbalance* ảnh hưởng đến hiệu suất huấn luyện mô hình sau này.

B. Biểu diễn đặc trưng văn bản (Embedding)

Sau khi hoàn tất tiền xử lý, dữ liệu văn bản được chuyển thành vector số bằng hai phương pháp chính: **TF-IDF** và **PhoBERT**. Kết quả cho thấy embedding TF-IDF có kích thước $(7678, 3000)$ với giá trị trung bình 0.0034, trong khi PhoBERT có kích thước $(7678, 768)$ và giá trị trung bình 0.0004. Cả hai đều không chứa giá trị `NaN` hay `Infinity`, đảm bảo chất lượng dữ liệu đầu vào cho quá trình huấn luyện mô hình.

Bảng IV
SO SÁNH NHANH GIỮA HAI PHƯƠNG PHÁP EMBEDDING

Đặc trưng	**TF-IDF**	**PhoBERT**
Kích thước	(7678, 3000)	(7678, 768)
Giá trị trung bình	0.0034	0.0004
Khoảng giá trị	[0.000, 0.756]	[-0.793, 0.792]

Nhìn chung, TF-IDF thể hiện tốt ở mức biểu diễn tần suất, trong khi PhoBERT nắm bắt được thông tin ngữ nghĩa và ngữ cảnh sâu hơn. Cả hai embedding được lưu ở định dạng `.npy` để sử dụng trong các mô hình huấn luyện tiếp theo.

C. Chia dữ liệu train/test

Tập dữ liệu sau khi biểu diễn được chia thành hai phần theo tỉ lệ **80% train** và **20% test**, đảm bảo phân phối nhãn giữa các chuyên mục được giữ nguyên bằng phương pháp `stratified split`. Cả hai loại embedding đều có tổng cộng **7.678 mẫu** và được chia thành:

- **PhoBERT:** Train $(6142, 768)$, Test $(1536, 768)$.
- **TF-IDF:** Train $(6142, 3000)$, Test $(1536, 3000)$.

Kết quả cho thấy phân phối các lớp giữa tập huấn luyện và kiểm thử là tương đồng, giúp tránh hiện tượng mất cân bằng dữ liệu trong quá trình học. Ví dụ, chuyên mục *"Thời sự"* chiếm khoảng 8.6% tổng số bài, *"Giáo dục"* khoảng 8.3%, trong khi các chuyên mục nhỏ hơn như *"Văn hóa"* hay *"Nhịp sống trẻ"* chỉ chiếm gần 2.6%. Điều này phản ánh rõ tính đa dạng nhưng đồng thời có sự chênh lệch nhất định giữa các chủ đề báo chí.

Bảng V
TỔNG HỢP KÍCH THƯỚC VÀ SỐ LƯỢNG MẪU SAU KHI CHIA DỮ LIỆU

Embedding	Tổng mẫu	Train	Test
TF-IDF	7678	6142 (80%)	1536 (20%)
PhoBERT	7678	6142 (80%)	1536 (20%)

Đối với việc huấn luyện bằng **MLP Neural Network**, thay vì chỉ chia thành hai phần train/test, tập dữ liệu được tách thành ba phần theo tỉ lệ **70% train**, **10% validation** và **20% test**. Trong đó, *train set* được sử dụng để mô hình học tham số, *validation set* để theo dõi hiệu suất, tinh chỉnh siêu tham số và kiểm soát hiện tượng overfitting, còn *test set* chỉ được dùng một lần ở cuối cùng để đánh giá khả năng tổng quát hóa của mô hình.

Việc chia ba tập giúp đảm bảo quy trình huấn luyện tuân theo chuẩn thực nghiệm trong học sâu, đồng thời cung cấp một điểm tham chiếu công bằng giữa hai loại biểu diễn (TF-IDF và PhoBERT) khi áp dụng cùng một mô hình MLP.

Bảng VI
KÍCH THƯỚC DỮ LIỆU SAU KHI CHIA CHO HUẤN LUYỆN MLP

Embedding	Tổng mẫu	Train (70%)	Validation (10%)	Test (20%)
TF-IDF	7678	5375	768	1535
PhoBERT	7678	5375	768	1535

Nhìn chung, kết quả chia dữ liệu đảm bảo tính đại diện cho toàn bộ 18 chuyên mục báo chí, tạo nền tảng ổn định cho quá trình huấn luyện và đánh giá các mô hình học máy ở các phần tiếp theo.

D. Chuẩn hóa dữ liệu

Để đảm bảo các đặc trưng của mô hình PhoBERT có cùng thang đo, dữ liệu được chuẩn hóa bằng phương pháp **StandardScaler**, trong khi TF-IDF được giữ nguyên do đặc tính thưa (sparse) và đã được chuẩn hóa sẵn trong quá trình tính trọng số.

Kết quả cho thấy tập huấn luyện PhoBERT sau chuẩn hóa có giá trị trung bình bằng 0 và độ lệch chuẩn xấp xỉ 1, trong khi tập kiểm thử có giá trị trung bình gần 0 và độ lệch chuẩn ổn định. Điều này chứng tỏ quá trình chuẩn hóa được thực hiện đúng cách, giúp mô hình hội tụ nhanh hơn và ổn định hơn trong huấn luyện.

Kích thước dữ liệu sau chuẩn hóa vẫn được giữ nguyên: PhoBERT có $(6142, 768)$ cho tập train và $(1536, 768)$ cho tập test, trong khi TF-IDF có $(6142, 3000)$ và $(1536, 3000)$ tương ứng. Kết quả này đảm bảo sự nhất quán cho các mô hình huấn luyện ở bước tiếp theo.

E. Huấn luyện mô hình

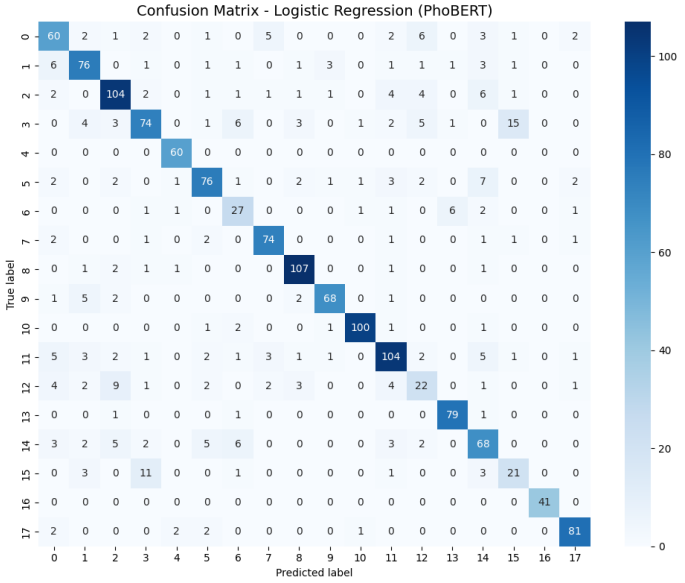
1. Logistic Regression

Mô hình **Logistic Regression** được sử dụng như một đường cơ sở (baseline model) để đánh giá hiệu quả của hai phương pháp biểu diễn văn bản: TF-IDF và PhoBERT. Kết quả huấn luyện cho thấy cả hai mô hình đều đạt độ chính xác cao, tuy nhiên TF-IDF thể hiện tốt hơn ở mọi chỉ số đánh giá.

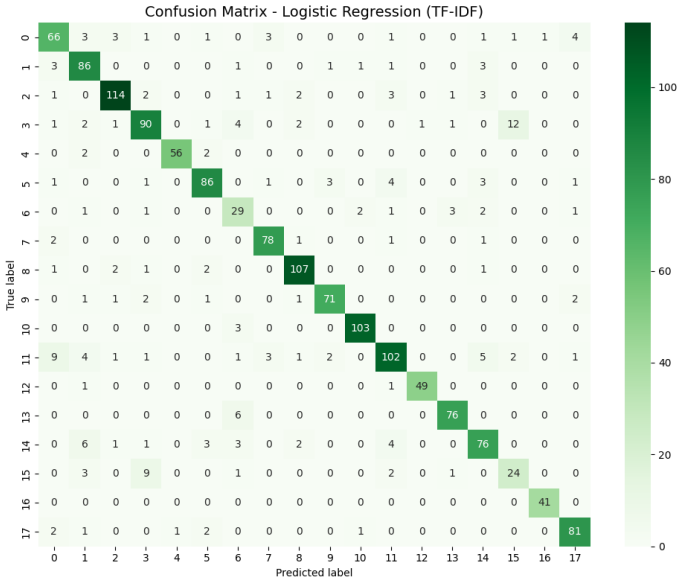
Bảng VII
HIỆU SUẤT MÔ HÌNH LOGISTIC REGRESSION TRÊN HAI EMBEDDING

Embedding	Accuracy	Precision	Recall	F1-score
PhoBERT	0.8086	0.7923	0.7977	0.7941
TF-IDF	**0.8691**	**0.8620**	**0.8658**	**0.8631**

Hình 4 và 5 trình bày ma trận nhầm lẫn (Confusion Matrix) của Logistic Regression cho hai loại embedding. Quan sát cho thấy TF-IDF phân loại chính xác hơn ở hầu hết các chủ đề, đặc biệt là các chuyên mục có số lượng bài viết lớn như *Thời sự*, *Giáo dục* và *Giải trí*. PhoBERT mặc dù nắm bắt được ngữ nghĩa, nhưng hiệu suất bị ảnh hưởng nhẹ do số lượng mẫu huấn luyện chưa đủ lớn để khai thác toàn bộ tiềm năng ngữ cảnh của mô hình.



Hình 4. Confusion Matrix - Logistic Regression (PhoBERT)



Hình 5. Confusion Matrix - Logistic Regression (TF-IDF)

Nhìn chung, Logistic Regression với TF-IDF embedding đạt hiệu quả cao và ổn định hơn, cho thấy mô hình tuyến tính này vẫn có khả năng phân loại mạnh khi đặc trưng đầu vào được biểu diễn tốt.

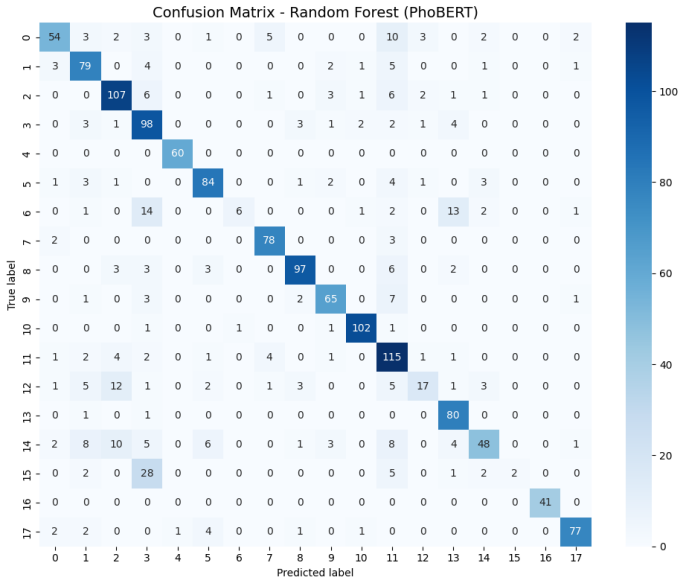
2. Huấn luyện mô hình Random Forest

Tiếp theo, mô hình **Random Forest** được huấn luyện trên hai tập đặc trưng PhoBERT và TF-IDF nhằm khai thác sức mạnh của các cây quyết định trong việc xử lý dữ liệu phi tuyến. Kết quả cho thấy mô hình TF-IDF tiếp tục cho hiệu suất cao hơn, chứng tỏ khả năng biểu diễn ổn định của phương pháp dựa trên tần suất từ, trong khi PhoBERT có xu hướng suy giảm nhẹ do chiều đặc trưng thấp hơn và số lượng mẫu hạn chế.

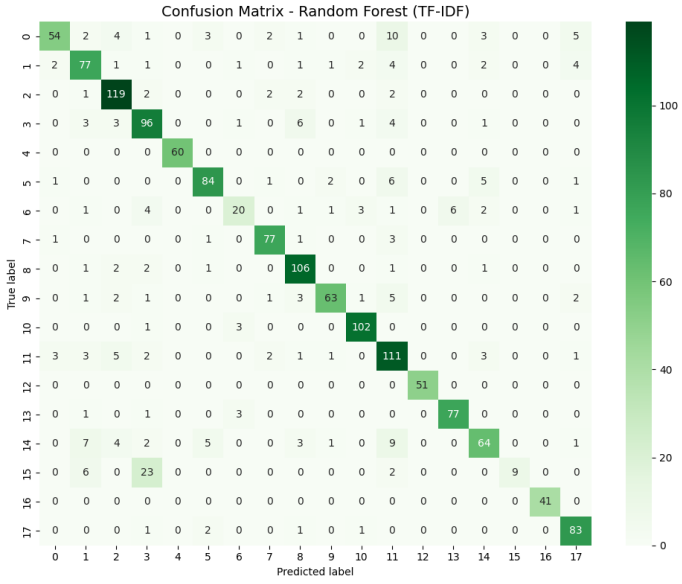
Bảng VIII
HIỆU SUẤT MÔ HÌNH RANDOM FOREST TRÊN HAI EMBEDDING

Embedding	Accuracy	Precision	Recall	F1-score
PhoBERT	0.7878	0.8265	0.7398	0.7363
TF-IDF	**0.8424**	**0.8682**	**0.8208**	**0.8270**

Các ma trận nhầm lẫn trong Hình 6 và 7 cho thấy TF-IDF giúp mô hình phân biệt tốt hơn giữa các chuyên mục lớn như *Giải trí*, *Thể thao*, *Thời sự*, đồng thời giảm lỗi chéo giữa các chủ đề tương tự. PhoBERT dù vẫn giữ được khả năng nhận diện ngữ nghĩa, nhưng chưa đạt hiệu quả tối ưu khi dữ liệu chưa đủ phong phú để tận dụng toàn bộ năng lực biểu diễn ngữ cảnh của mô hình transformer.



Hình 6. Confusion Matrix - Random Forest (PhoBERT)



Hình 7. Confusion Matrix - Random Forest (TF-IDF)

Nhìn chung, Random Forest cho thấy khả năng tổng quát hóa tốt, đặc biệt khi áp dụng trên TF-IDF embedding, nhờ cấu trúc cây ngẫu nhiên giúp giảm hiện tượng quá khớp và tăng độ ổn định của mô hình.

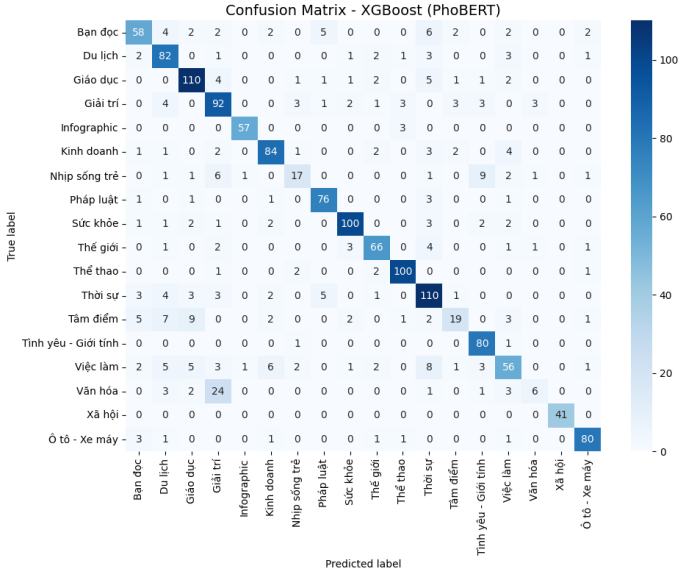
3. XGBoost

Mô hình **XGBoost** được lựa chọn nhằm tận dụng khả năng học phi tuyến mạnh mẽ của thuật toán boosting, giúp tối ưu hóa sai số huấn luyện thông qua cơ chế kết hợp nhiều cây yếu (weak learners). Kết quả cho thấy XGBoost đạt hiệu suất cao và ổn định hơn so với Logistic Regression và Random Forest, đặc biệt khi sử dụng TF-IDF embedding.

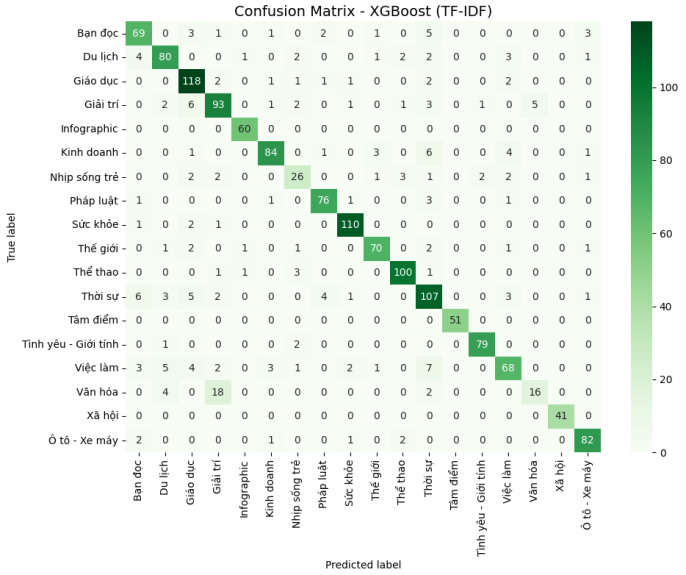
Bảng IX
HIỆU SUẤT MÔ HÌNH XGBOOST TRÊN HAI EMBEDDING

Embedding	Accuracy	Precision	Recall	F1-score
PhoBERT	0.8034	0.7921	0.7670	0.7683
TF-IDF	**0.8659**	**0.8699**	**0.8550**	**0.8587**

Từ các ma trận nhầm lẫn trong Hình 8 và 9, có thể thấy rằng mô hình với TF-IDF thể hiện khả năng phân biệt chủ đề tốt hơn, đặc biệt ở các nhóm có đặc trưng ngôn ngữ rõ ràng như *Thể thao*, *Giải trí* và *Thời sự*. Trong khi đó, PhoBERT tuy nắm bắt được ngữ nghĩa ngữ cảnh, nhưng chưa thể hiện vượt trội do kích thước dữ liệu huấn luyện còn hạn chế so với độ phức tạp của mô hình.



Hình 8. Confusion Matrix - XGBoost (PhoBERT)



Hình 9. Confusion Matrix - XGBoost (TF-IDF)

Nhìn chung, XGBoost với TF-IDF embedding đạt hiệu năng tốt nhất trong nhóm mô hình thử nghiệm, thể hiện sự cân bằng giữa độ chính xác và khả năng tổng quát hóa trên tập dữ liệu văn bản báo chí tiếng Việt.

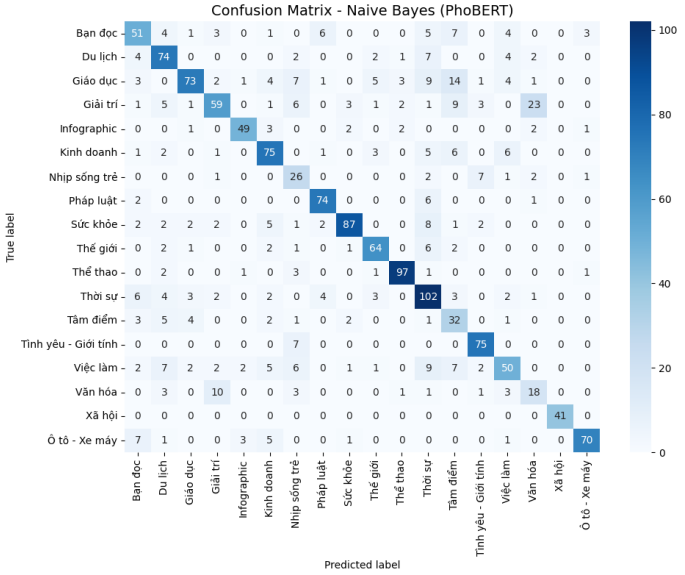
4. Naive Bayes

Mô hình **Naive Bayes** được áp dụng như một phương pháp truyền thống, có ưu điểm đơn giản và tốc độ huấn luyện nhanh, phù hợp với các bài toán phân loại văn bản có số lượng đặc trưng lớn. Kết quả cho thấy hiệu suất tổng thể của mô hình thấp hơn so với các phương pháp trước, đặc biệt trên tập TF-IDF, do giả định độc lập giữa các đặc trưng không hoàn toàn đúng trong ngữ cảnh ngôn ngữ tự nhiên.

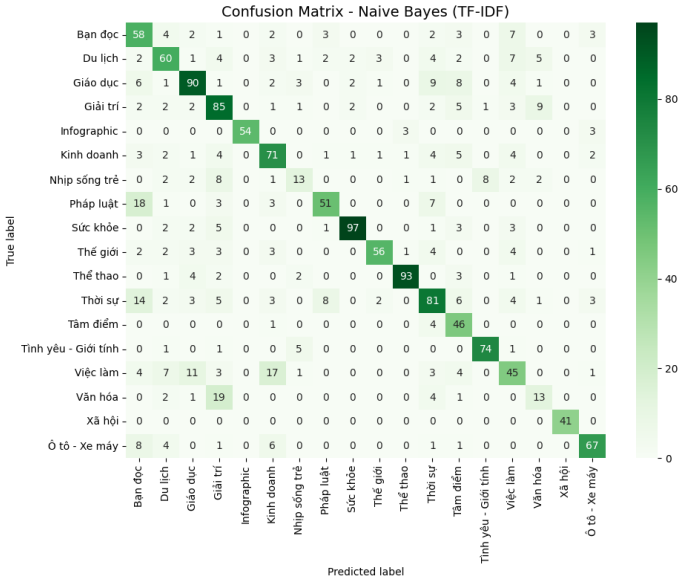
Bảng X
HIỆU SUẤT MÔ HÌNH NAIVE BAYES TRÊN HAI EMBEDDING

Embedding	Accuracy	Precision	Recall	F1-score
PhoBERT	**0.7272**	**0.7265**	**0.7296**	**0.7206**
TF-IDF	0.7129	0.7206	0.7061	0.7062

Hình 10 và 11 trình bày ma trận nhầm lẫn cho hai embedding. Kết quả cho thấy mô hình PhoBERT hoạt động ổn định hơn, đặc biệt ở các chuyên mục như *Thể thao*, *Thời sự* và *Sức khỏe*. Trong khi đó, TF-IDF có xu hướng dự đoán nhầm ở các nhóm chủ đề có ngữ nghĩa gần nhau như *Kinh doanh* và *Pháp luật*, thể hiện hạn chế của giả định độc lập trong mô hình Naive Bayes.



Hình 10. Confusion Matrix - Naive Bayes (PhoBERT)



Hình 11. Confusion Matrix - Naive Bayes (TF-IDF)

Nhìn chung, Naive Bayes cho thấy hiệu quả chấp nhận được trong bài toán phân loại văn bản, song không đạt độ chính xác cao bằng các mô hình hiện đại như Random Forest hoặc XGBoost do không tận dụng được mối quan hệ phụ thuộc giữa các đặc trưng ngôn ngữ.

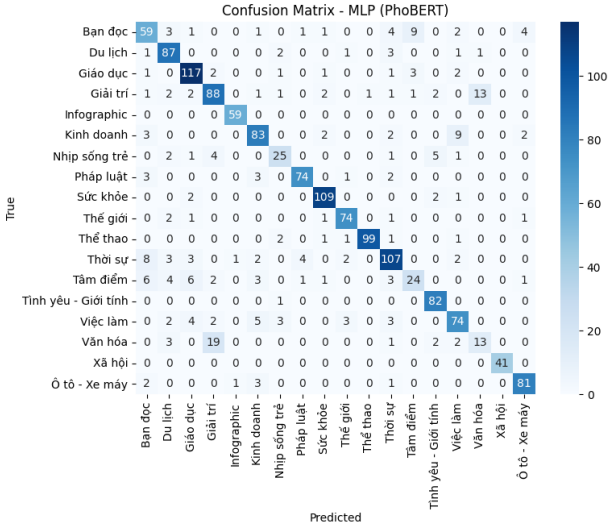
5. MLP Neural Network

Mô hình **MLP Neural Network (Multi-Layer Perceptron)** được triển khai nhằm tận dụng khả năng học phi tuyến mạnh mẽ của mạng nơ-ron nhiều tầng. Khác với các mô hình truyền thống như Logistic Regression hay Naive Bayes vốn chỉ mô tả được quan hệ tuyến tính, MLP có thể nắm bắt được các mối quan hệ phức tạp giữa các đặc trưng, đặc biệt hữu ích khi kết hợp với embedding có chiều cao như PhoBERT.

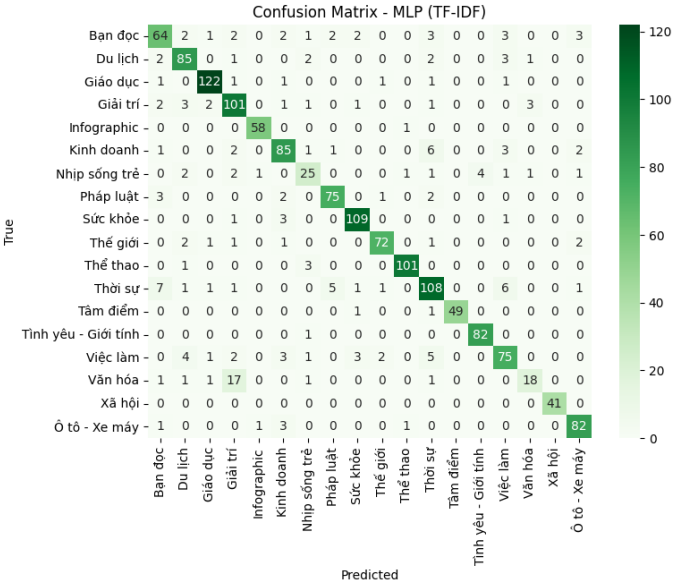
Bảng XI
HIỆU SUẤT MÔ HÌNH MLP NEURAL NETWORK TRÊN HAI EMBEDDING (VALIDATION VÀ TEST)

Embedding	Tập	Accuracy	Precision	Recall	F1-score
PhoBERT	Validation	0.8529	0.8316	0.8290	0.8294
PhoBERT	Test	0.8438	0.8265	0.8191	0.8205
TF-IDF	Validation	0.8802	0.8804	0.8731	0.8760
TF-IDF	Test	**0.8802**	**0.8803**	**0.8659**	**0.8702**

Kết quả cho thấy, **MLP kết hợp với TF-IDF** đạt hiệu suất cao nhất với **Accuracy 88.0%** và **F1-score 87.0%** trên tập kiểm thử, vượt trội so với PhoBERT embedding. Tuy nhiên, mô hình **PhoBERT + MLP** cũng mang lại kết quả khá tốt (F1-score khoảng 82%), chứng minh khả năng của embedding ngữ nghĩa khi huấn luyện bằng mạng nơ-ron sâu.



Hình 12. Confusion Matrix - MLP Neural Network (PhoBERT)



Hình 13. Confusion Matrix - MLP Neural Network (TF-IDF)

Nhìn chung, MLP cho thấy hiệu quả vượt trội, đặc biệt khi kết hợp với TF-IDF. Điều này chứng minh rằng việc khai thác các đặc trưng thống kê truyền thống kết hợp với mạng nơ-ron sâu vẫn mang lại kết quả tốt trong bài toán phân loại chủ đề báo chí tiếng Việt, thậm chí có thể vượt trội hơn embedding ngữ nghĩa như PhoBERT trong một số kịch bản.

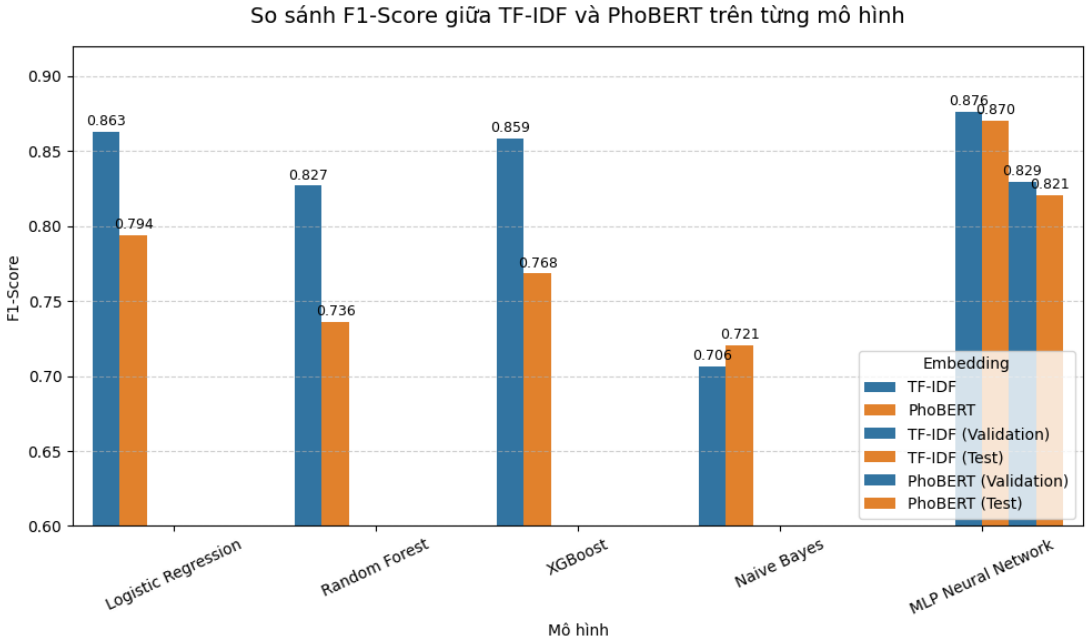
VI. Kết luận

A. Bảng tổng hợp kết quả

Bảng XII trình bày toàn bộ kết quả của năm mô hình được huấn luyện trên hai loại biểu diễn đặc trưng văn bản: TF-IDF và PhoBERT. Các chỉ số đánh giá bao gồm **Accuracy**, **Precision**, **Recall**, và **F1-score**.

Bảng XII
TỔNG HỢP HIỆU SUẤT CÁC MÔ HÌNH TRÊN HAI EMBEDDING

Model	Embedding	Accuracy	Precision	Recall	F1-score
Logistic Regression	TF-IDF	0.8691	0.8620	0.8658	0.8631
Logistic Regression	PhoBERT	0.8086	0.7923	0.7977	0.7941
Random Forest	TF-IDF	0.8424	0.8682	0.8208	0.8270
Random Forest	PhoBERT	0.7878	0.8265	0.7398	0.7363
XGBoost	TF-IDF	0.8659	0.8699	0.8550	0.8587
XGBoost	PhoBERT	0.8034	0.7921	0.7670	0.7683
Naive Bayes	TF-IDF	0.7129	0.7206	0.7061	0.7062
Naive Bayes	PhoBERT	0.7272	0.7265	0.7296	0.7206
MLP Neural Network	TF-IDF (Validation)	0.8802	0.8804	0.8731	0.8760
MLP Neural Network	TF-IDF (Test)	**0.8802**	**0.8803**	**0.8659**	**0.8702**
MLP Neural Network	PhoBERT (Validation)	0.8529	0.8316	0.8290	0.8294
MLP Neural Network	PhoBERT (Test)	0.8438	0.8265	0.8191	0.8205



Hình 14. So sánh F1-Score giữa TF-IDF và PhoBERT trên từng mô hình

B. Phân tích và đánh giá kết quả

1. Logistic Regression:

- Mô hình tuyến tính này thể hiện hiệu suất rất ổn định với TF-IDF, đạt **Accuracy = 0.8691** và **F1 = 0.8631**. - Trong khi đó, PhoBERT chỉ đạt **F1 = 0.7941**, thấp hơn khoảng 0.07 điểm, cho thấy TF-IDF phù hợp hơn cho các phương pháp tuyến tính. - Precision và Recall khá cân bằng, chứng tỏ Logistic Regression hoạt động ổn định, không quá thiên lệch giữa dự đoán dương và âm.

2. Random Forest:

- Với TF-IDF, mô hình đạt **F1 = 0.8270**, nhờ khả năng tổng hợp nhiều cây quyết định để giảm overfitting. - Tuy nhiên, trên PhoBERT, F1 giảm xuống **0.7363**, cho thấy các đặc trưng trừu tượng từ PhoBERT không tối ưu cho cây ngẫu nhiên do thiếu tính tuyến tính rõ ràng. - Mặc dù Precision cao (**0.8682**), Recall lại thấp hơn (**0.8208**), cho thấy mô hình thiên về dự đoán chắc chắn hơn là bao quát toàn bộ nhãn.

3. XGBoost:

- Đây là mô hình boosting mạnh mẽ, đạt **Accuracy = 0.8659** và **F1 = 0.8587** trên TF-IDF — gần tương đương Logistic Regression nhưng ổn định hơn với dữ liệu mất cân bằng. - Khi kết hợp với PhoBERT, F1 giảm còn **0.7683**. Dù vậy, XGBoost vẫn cho Precision cao (**0.8699**), thể hiện khả năng chọn lọc mẫu đúng mạnh mẽ nhưng vẫn bỏ sót một số nhãn hiếm (Recall thấp hơn). - So với Random Forest, XGBoost hoạt động tốt hơn trên cả hai embedding, chứng tỏ khả năng khử bias tốt hơn.

4. Naive Bayes:

- Đây là mô hình yếu nhất, với F1 chỉ đạt **0.7062** (TF-IDF) và **0.7206** (PhoBERT). - Mặc dù đơn giản, Naive Bayes không thể khai thác mối quan hệ phức tạp giữa các đặc trưng, dẫn đến Accuracy thấp nhất toàn bộ thí nghiệm. - Precision và Recall khá gần nhau, chứng tỏ độ ổn định chấp nhận được nhưng không đủ mạnh cho dữ liệu đa chủ đề như báo chí.

5. MLP Neural Network:

- Đây là mô hình mạnh nhất trong toàn bộ hệ thống, đặc biệt khi sử dụng TF-IDF. - Trên TF-IDF, mô hình đạt **F1 = 0.8760 (Validation)** và **0.8702 (Test)**, với Precision và Recall rất cân bằng (**0.8803** và **0.8659**). - Trên PhoBERT, kết quả thấp hơn nhưng vẫn khả quan, với **F1 = 0.8294 (Validation)** và **0.8205 (Test)**. - Nhờ khả năng học phi tuyến và tối ưu hóa qua nhiều lớp ẩn, MLP khai thác tốt hơn mối quan hệ phức tạp giữa đặc trưng và nhãn, giúp TF-IDF + MLP trở thành lựa chọn tốt nhất.

C. So sánh giữa hai biểu diễn Embedding

- **TF-IDF** vượt trội hơn PhoBERT trong hầu hết các mô hình, đặc biệt là các phương pháp học truyền thống và cả MLP. Điều này là do TF-IDF phản ánh trực tiếp tần suất từ vựng — đặc biệt hữu ích trong dữ liệu ngắn, ít ngữ cảnh như tiêu đề và mô tả tin tức.

- **PhoBERT**, mặc dù là embedding ngữ nghĩa tiên tiến, nhưng chưa phát huy hết tiềm năng cho bài toán phân loại. Các vector ẩn từ PhoBERT mang tính ngữ cảnh sâu, nhưng lại không phù hợp bằng TF-IDF trong tác vụ phân loại theo chủ đề ngắn gọn.

D. Kết luận tổng quát

Từ toàn bộ kết quả trên, có thể rút ra một số kết luận quan trọng:

- **MLP Neural Network với TF-IDF** là lựa chọn tốt nhất, đạt **F1 = 0.8702** trên tập test, vượt trội hơn tất cả các mô hình khác.
- Các mô hình tuyến tính như **Logistic Regression** vẫn hoạt động tốt, dễ huấn luyện và cho kết quả ổn định.
- **PhoBERT** cho kết quả khá tốt nhưng chưa vượt qua TF-IDF do chưa được fine-tune chuyên biệt.
- **Naive Bayes** chỉ đóng vai trò baseline, không phù hợp cho dữ liệu đa lớp và phức tạp.

TÀI LIỆU

[1] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *European conference on machine learning*. Springer, 1998, pp. 137–142.

[2] H. H. Phạm, "Vietnamese news classification using tf-idf and machine learning algorithms," *Vietnam Journal of Computer Science*, vol. 7, no. 4, pp. 241–254, 2020.

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2019.

[4] Y. Liu *et al.*, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[5] D. Q. Nguyen and A. T. Nguyen, "Phobert: Pre-trained language models for vietnamese," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020, pp. 1037–1042.

[6] M. T. Phạm and H. L. Trần, "Automated classification of vietnamese online news articles using tf-idf and machine learning," *Journal of Science and Technology*, 2021.

[7] Q. A. Nguyễn *et al.*, "Vietnamese news classification using phobert and transfer learning," *VNU Journal of Computer Science*, 2022.

[8] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information processing & management*, vol. 24, no. 5, pp. 513–523, 1988.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[10] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz *et al.*, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2020, pp. 38–45.

[11] A. K. Jain, R. P. Duin, and J. Mao, *Statistical pattern recognition: A review*. IEEE, 2000, vol. 22, no. 1.

[12] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 785–794.
[13] D. W. Hosmer, S. Lemeshow, and R. X. Sturdivant, *Applied Logistic Regression*. John Wiley & Sons, 2013.
[14] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
[15] I. Rish, "An empirical study of the naive bayes classifier," *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, no. 22, pp. 41–46, 2001.
[16] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.
[17] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing & Management*, vol. 45, no. 4, pp. 427–437, 2009.