BỘ GIÁO DỤC VÀ ĐÀO TẠO

ĐẠI HỌC KINH TẾ THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG CÔNG NGHỆ VÀ THIẾT KẾ

BÁO CÁO ĐỒ ÁN XỬ LÝ NGÔN NGỮ TỰ NHIÊN

ĐỀ TÀI: PHÂN TÍCH QUAN ĐIỂM DỰA TRÊN PHẢN HỒI CỦA HỌC SINH ĐẠI HỌC

Giảng viên hướng dẫn: TS. Đặng Ngọc Hoàng Thành

Mã học phần: 24C1INF50907601

Danh sách nhóm:

1. **Nguyễn Trần Thế Anh - 31221026655**
2. **Phạm Bằng - 31221024364**
3. **Lâm Vĩ Kiệt - 31221020030**
4. **Trần Mỹ Thiên Tân - 31221026343**

Tp. Hồ Chí Minh, tháng 12 năm 2024

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG BIỂU

BẢNG PHÂN CÔNG NHIỆM VỤ

Thành viên	MSSV	Nhiệm vụ phân công
Nguyễn Trần Thế Anh	31221026655	- Nội dung chương 3, 4 - Thuật toán SVM, word tokenizer
Phạm Bằng	31221024364	- Nội dung chương 3, 4 - Thuật toán LSTM, vectorizer
Lâm Vĩ Kiệt	31221020030	- Chương 3, 5 - Thuật toán Naive Bayes, so sánh kết quả
Trần Mỹ Thiên Tân	31221026343	- Nội dung chương 1, 2 - Thuật toán XGBoost, khám phá dữ liệu

LINK GITHUB

CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI

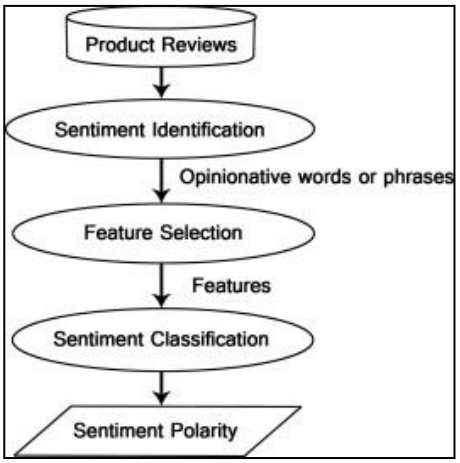
1.1 Giới thiệu về xử lý ngôn ngữ tự nhiên

Xử lý ngôn ngữ tự nhiên (NLP) là việc nghiên cứu về các mô hình toán học và tính toán của nhiều khía cạnh khác nhau của ngôn ngữ và cũng như sự phát triển của nhiều hệ thống. Chúng bao gồm các hệ thống ngôn ngữ nói tích hợp lời nói và ngôn ngữ tự nhiên; giao diện hợp tác với cơ sở dữ liệu và cơ sở tri thức mô hình hóa các khía cạnh tương tác giữa người với người; giao diện đa ngôn ngữ; dịch máy; và các hệ thống hiểu thông điệp, trong số những thứ khác. Nghiên cứu về NLP có tính liên ngành cao, bao gồm các khái niệm trong khoa học máy tính, ngôn ngữ học, logic và tâm lý học. NLP có vai trò đặc biệt trong khoa học máy tính vì nhiều khía cạnh của lĩnh vực này liên quan đến các tính năng ngôn ngữ của tính toán và NLP tìm cách mô hình hóa ngôn ngữ bằng tính toán.[1] Ngày nay, Xử lý ngôn ngữ tự nhiên (NLP) đã thu hút được nhiều sự chú ý vì khả năng biểu diễn và phân tích ngôn ngữ của con người bằng máy tính. Nó đã mở rộng ứng dụng của mình trong nhiều lĩnh vực như dịch máy, phát hiện thư rác email, trích xuất thông tin, tóm tắt, y tế và trả lời câu hỏi, v.v.

Tóm lại, NLP là một chuyên ngành của Trí tuệ nhân tạo và Ngôn ngữ học, dành riêng để giúp máy tính hiểu các câu lệnh hoặc từ được viết bằng ngôn ngữ của con người. Nó ra đời để giúp người dùng dễ dàng hơn trong công việc và thỏa mãn mong muốn giao tiếp với máy tính bằng ngôn ngữ tự nhiên, và có thể được phân loại thành hai phần, tức là Hiểu ngôn ngữ tự nhiên (Natural Language Understanding) hoặc Ngôn ngữ học và Tạo ngôn ngữ tự nhiên (Linguistics and Natural Language Generation), phát triển nhiệm vụ hiểu và tạo văn bản.[2]

1.2 Giới thiệu về phân tích cảm xúc

Phân tích cảm xúc (Sentiment Analysis - SA) là nghiên cứu tính toán về ý kiến, thái độ và cảm xúc của mọi người đối với một thực thể [(*)]. Mục tiêu của SA là tìm ra các ý kiến, xác định các tình cảm mà chúng thể hiện và sau đó phân loại cực tính của chúng, thể hiện ở hình sau[3]:



Hình 1.1: Quá trình phân tích tình cảm khi đánh giá sản phẩm.

(*) entity

Phân tích cảm xúc có thể được coi là một quá trình phân loại như minh họa trong Hình 1.1. Có ba cấp độ phân loại chính trong SA: cấp độ tài liệu (document-level), cấp độ câu (sentence-level) và cấp độ khía cạnh (aspect-level). [3]

SA cấp độ tài liệu (document-level SA) tập trung vào phân loại một tài liệu ý kiến xem liệu rằng nó thể hiện ý kiến/tình cảm tích cực hay tiêu cực. Nó coi toàn bộ tài liệu là một đơn vị thông tin cơ bản (nói về một chủ đề).

SA cấp độ câu (sentence-level SA) có mục đích phân loại tình cảm được thể hiện trong mỗi câu. Bước đầu tiên là xác định xem câu là chủ quan hay khách quan. Nếu câu là chủ quan, SA cấp độ câu sẽ xác định xem câu thể hiện ý kiến tích cực hay tiêu cực. Wilson và cộng sự [4] đã chỉ ra rằng các biểu hiện về cảm xúc không nhất thiết phải mang tính chủ quan. Tuy nhiên, không có sự khác biệt cơ bản nào giữa phân loại cấp độ tài liệu và cấp độ câu vì căn bản, câu chỉ là những tài liệu ngắn. Phân loại văn bản ở cấp độ tài liệu hoặc cấp độ câu không cung cấp chi tiết cần thiết cho các ý kiến về mọi khía cạnh của thực thể, điều này là cần thiết trong nhiều ứng dụng. Do đó, để có được những chi tiết này; chúng ta cần chuyển sang phân tích cảm xúc theo cấp độ khía cạnh.

SA cấp độ khía cạnh (aspect-level SA) có mục đích phân loại tình cảm liên quan đến các khía cạnh cụ thể của thực thể. Bước đầu tiên là xác định các thực thể và khía cạnh của chúng. Những người đưa ra ý kiến có thể đưa ra các ý kiến khác nhau cho các khía cạnh khác nhau của cùng một thực thể. Ví dụ như câu này: "Chất lượng giọng nói của điện thoại này không tốt, nhưng thời lượng pin dài".

1.3 Phát biểu bài toán

Phản hồi và đánh giá của sinh viên không chỉ là nguồn thông tin hữu ích để cải thiện chất lượng giáo dục mà nó còn đồng thời là nguồn thông tin quan trọng để nghiên cứu về phân tích cảm xúc và giáo dục. Việc tạo ra một ngữ liệu liên quan đến giáo dục không chỉ có lợi cho việc phát triển phân tích cảm xúc cho tiếng Việt mà còn thực sự hữu ích cho nghiên cứu về giáo dục, đặc biệt là ở Việt Nam.

Phản hồi và đánh giá của sinh viên là một trong những yếu tố ảnh hưởng mạnh mẽ nhất đến việc học và thành tích của họ, nhưng tác động này có thể là tích cực hoặc tiêu cực. Có hai loại phản hồi chính: (1) phản hồi từ giảng viên cho sinh viên giúp sinh viên nhận thức được điểm yếu và điểm mạnh của mình để cải thiện việc học và (2) phản hồi từ sinh viên cho giảng viên cho phép giảng viên phản ánh và cải thiện việc giảng dạy của mình. Đặc biệt, sinh viên đưa ra ý kiến về nhiều vấn đề khác nhau. Phản hồi được thu thập vào cuối mỗi học kỳ thông qua khảo sát khóa học.

1.4 Thách thức khi giải quyết bài toán

1.4.1. So sánh giữa tiếng Việt và tiếng Anh:

Đặc điểm của Tiếng Việt	Đặc điểm của Tiếng Anh
Là loại hình đơn lập (isolated), hay còn được gọi là loại hình phi hình thái, đơn tiết.	Là loại hình biến cách (flexion), hay còn được gọi là loại hình khuất chiết
Từ không biến đổi hình thái, ý nghĩa ngữ pháp nằm ở ngoài từ Ví dụ: "Tôi chơi đá banh với Nam" và "Nam chơi đá banh với tôi"	Từ có biến đổi hình thái, ý nghĩa ngữ pháp nằm trong từ Ví dụ: "I play football with him" và "He plays football with me"
Phương thức ngữ pháp chủ yếu là trật tự từ và hư từ Ví dụ: "học bài" và "bài học"; "đang học bài" và "học bài rồi";...	Phương thức ngữ pháp chủ yếu là phụ tố Ví dụ: "studying" và "studied"
Ranh giới giữa các từ không được tự động xác định bằng khoảng cách/khoảng trắng	Kết hợp giữa các hình vị là chặt chẽ, khó xác định và quan trọng hơn, là nó được nhận diện bằng khoảng trắng hoặc dấu câu.
Tồn tại loại từ đặc biệt "từ phân loại" (classifier) (hay còn gọi là phó danh từ chỉ loại) kèm theo với danh từ Ví dụ: cái bàn; con chó; con sông; bức thư;...	Hiện tượng cấu tạo bằng từ ghép thêm phụ tố (affix) vào gốc từ rất phổ biến Ví dụ: international (inter-nation-al)
Có hiện tượng từ láy và nói lái trong tiếng Việt Ví dụ: long lanh, lắp lánh, lao xao, rộn ràng;… hiện đại -> hại điện;…	

1.4.2. Nhận xét

Từ những đặc điểm nêu trên của tiếng Việt, ta có thể dễ dàng nhận thấy rằng về bản chất, ngôn ngữ này là một loại hình phi hình thái. Do đó, việc có thể phân biệt các loại từ (danh từ, động từ, tính từ,...) và ý nghĩa của chúng là rất khó. Điều này dẫn tới việc tiền xử lý văn bản (tách từ, tách đoạn, tách câu,...) sẽ có độ phức tạp khá cao, khi mà quá trình này sẽ phải bao gồm những khâu như xử lý các hư từ, phụ từ, từ láy,...

Ngoài ra, ngữ pháp tiếng Việt chủ yếu sử dụng phương thức là trật tự từ nên nếu áp dụng phương pháp tính xác suất từ xuất hiện có thể cho ra kết quả không đúng và thiếu chính xác. Ranh giới từ giữa 2 ngôn ngữ cũng có phần khác biệt khi đối với

tiếng Việt, ranh giới này không được xác định mặc nhiên bằng khoảng trắng/khoảng trống.

Tóm lại, những khác biệt giữa tiếng Anh và tiếng Việt là quá nhiều và cân nhắc vào độ phức tạp của tiếng Việt, chúng ta không thể áp dụng hoàn toàn các thuật toán sử dụng của ngôn ngữ này cho ngôn ngữ còn lại.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1 Giới thiệu về các mô hình học máy

2.1.1 XGBoost

XGBoost (eXtreme Gradient Boosting) là một thuật toán học máy mạnh mẽ được sử dụng rộng rãi trong các bài toán phân loại và hồi quy. Thuật toán này hoạt động dựa trên cây quyết định (decision trees) và sử dụng kỹ thuật gradient boosting để xây dựng mô hình. Các cây quyết định được tạo ở dạng tuần tự, trọng số đóng vai trò quan trọng trong XGBoost, mỗi cây mới sẽ cố gắng giảm thiểu sai số của cây trước.

Công thức tối ưu:

$$min_{c_n, w_n} L(y, W_{n-1} + c_n w_n)$$

Trong đó:

- y là giá trị dự đoán.
- W_{n-1} là mô hình được huấn luyện trước đó.
- c_n, w_n là các tham số cần cập nhật trong mỗi vòng học.
- $L(y, W_n)$ là hàm mất mát (loss function), đánh giá độ chính xác mô hình.

Điều làm XGBoost có hiệu suất ấn tượng và khả năng tính toán tốt là:

- Tích hợp các cơ chế để giảm nguy cơ bị overfitting như Regularization, Sub-sampling columns/ rows,..
- Tận dụng tốt tài nguyên hệ thống.
- Xử lý missing data và tiết kiệm thời gian huấn luyện

2.1.2 Multinomial Naive Bayes

Multinomial Naive Bayes là một biến thể của thuật toán Naive Bayes được sử dụng chủ yếu trong phân loại văn bản, nơi dữ liệu thường được biểu diễn bằng số lượng từ hoặc tần suất xuất hiện của từ trong tài liệu.

Thuật toán này giả định rằng các đặc trưng tuân theo phân phối đa thức (Multinomial Distribution) đối với mỗi lớp y. Và để tính xác suất $P(x_i \mid y)$, thuật toán sử dụng MLE (Maximum Likelihood Estimation) với kỹ thuật làm mịn để tránh xác suất bằng 0.

$$\mathrm{P}(x_i \mid y) = \frac{N_{yi} + \alpha}{N_y + \alpha n}$$

Trong đó:

- N_{yi}: số lần đặc trưng i xuất hiện trong các tài liệu thuộc lớp y
- $N_y = \sum_{i=1}^{n} N_{yi}$: Tổng số lần các đặc trưng xuất hiện trong lớp y.
- α: Tham số làm mịn

Sau đó, thực hiện tính xác suất hậu nghiệm cho từng lớp y theo định lý Bayes:

$$P(y|x) \propto P(y)\prod_{i=1}^{n} P(x_i \mid y)$$

Cuối cùng, để phân loại tài liệu ta chọn lớp y sao cho xác suất hậu nghiệm $P(y \mid x_i)$ lớn nhất:

$$\hat{y} = argmax_y(P(y)\prod_{i=1}^{n} P(x_i \mid y))$$

2.1.3 SVM

Phương pháp Support Vector Machine (SVM) là một thuật toán học máy giám sát được sử dụng cho cả hai bài toán phân loại và hồi quy. Nhưng hầu hết nó phổ biến hơn với các bài toán phân loại. Mục tiêu của SVM là tìm ra một siêu phẳng tốt nhất để phân tách các điểm dữ liệu thuộc các lớp khác nhau. Siêu phẳng tốt nhất là siêu phẳng có khoảng cách lớn nhất đến các điểm dữ liệu gần nhất của mỗi lớp, được gọi là các vector hỗ trợ.

Với một tập huấn luyện có N điểm dữ liệu, mỗi điểm được biểu diễn bởi một vectơ x_i và một nhãn lớp y_i :

$$(x_i, y_i) \text{ với i } = 1, 2, \ldots, N$$

SVM tìm kiếm một siêu phẳng phân tách dữ liệu sao cho khoảng cách từ siêu phẳng đến các điểm dữ liệu gần nhất là lớn nhất. Phương trình của siêu phẳng:

$$w \cdot x - b = 0$$

Trong đó w là vectơ trọng số và b là hằng số. Để đảm bảo rằng các điểm dữ liệu thuộc các lớp khác nhau nằm ở hai phía khác nhau của siêu phẳng, ta có điều kiện:

$$y_i(w \cdot x_i - b) \geq 1 \ \forall i$$

Để tìm siêu phẳng tốt nhất, ta cần tối ưu hóa bài toán sau

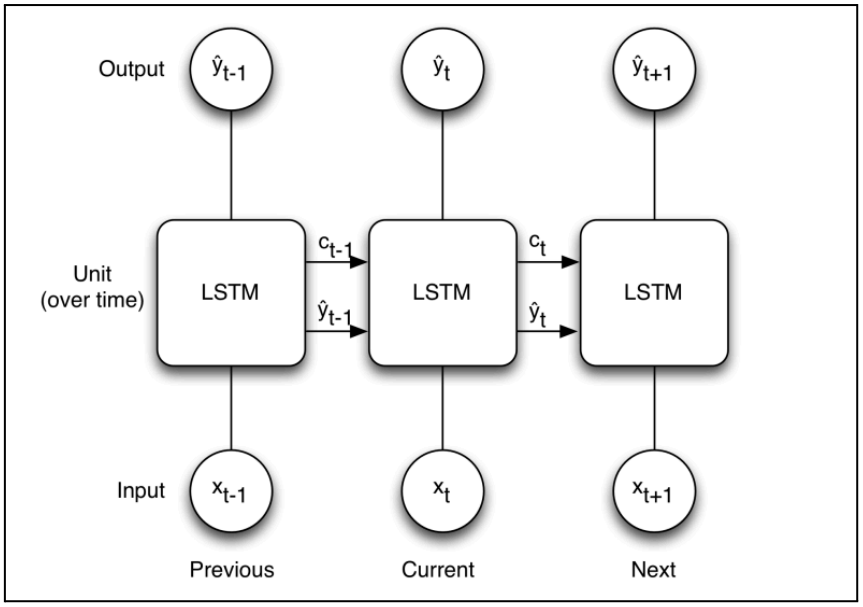
$$\min_{w,b} \frac{1}{2}||w||^2$$

với ràng buộc:

$$y_i(w \cdot x_i - b) \geq 1 \ \forall i$$

2.2 Giới thiệu mô hình học sâu

LSTM là một kiến trúc mạng thần kinh tuần hoàn, được thiết kế để giải quyết vấn đề biến mất gradient trong các mạng RNN truyền thống. LSTM sử dụng các cổng (gates) để điều khiển dòng chảy thông tin, bao gồm cổng quên, cổng vào và cổng xuất. Nhờ cơ chế này, LSTM có thể "nhớ" thông tin quan trọng và "quên" đi thông tin không cần thiết giúp mô hình phù hợp với các bài toán liên quan đến dữ liệu chuỗi như xử lý ngôn ngữ tự nhiên, dự báo chuỗi thời gian và nhận dạng giọng nói.



Hình 2.1: Mô hình LSTM

Cấu Trúc Cơ Bản của LSTM: Một LSTM có thể được biểu diễn bằng các đơn vị (units) hoặc ô (cell) được kết nối với các cổng quan trọng sau:

• Cổng Quên (Forget Gate): Quyết định thông tin nào sẽ bị xóa từ bộ nhớ trước đó.

• Cổng Đầu Vào (Input Gate): Quyết định thông tin nào sẽ được cập nhật vào bộ nhớ.

• Cổng Đầu Ra (Output Gate): Quyết định thông tin nào sẽ được đưa ra từ bộ nhớ.

Công thức cập nhật trạng thái bộ nhớ ct tại thời điểm t được mô tả như sau:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

Trong đó:

• c_t là trạng thái bộ nhớ tại thời điểm t.

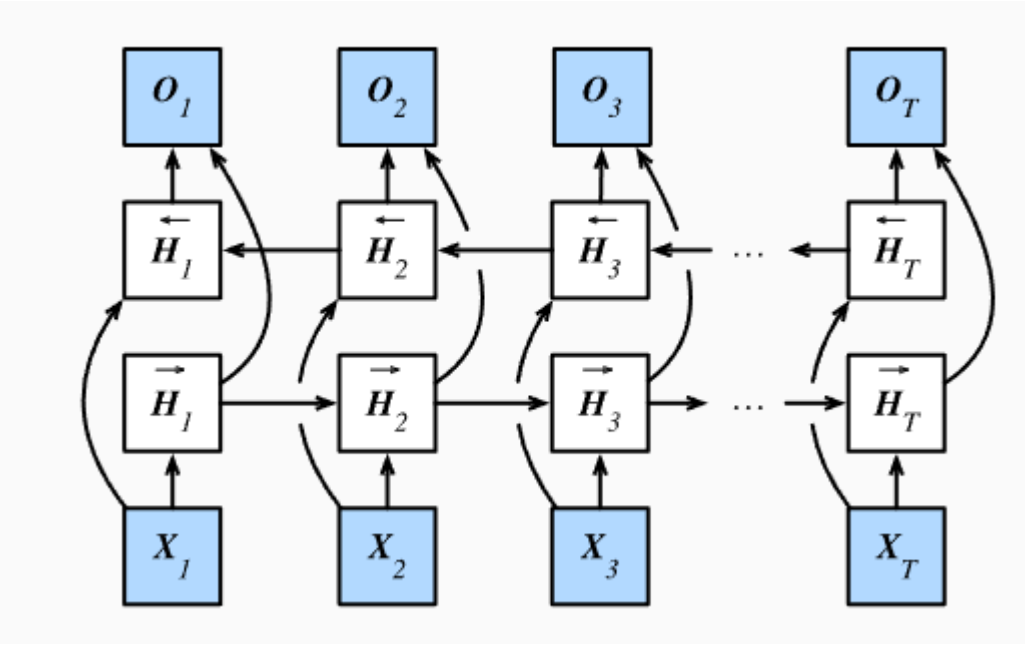
• f_t là giá trị của cổng quên.

• i_t là giá trị của cổng đầu vào.

• $\tilde{c}_t$ là giá trị ứng với thông tin mới được đề xuất.

• $\odot$ là phép nhân element-wise.

BiLSTM (LSTM hai chiều) là một mô hình xử lý chuỗi sử dụng hai mạng LSTM: một mạng xử lý dữ liệu theo hướng thuận và mạng còn lại theo hướng ngược. Cách tiếp cận này giúp mạng tiếp cận được nhiều thông tin hơn, đồng thời cải thiện khả năng nắm bắt ngữ cảnh. Nhờ đó, BiLSTM có thể hiểu rõ hơn ý nghĩa của một từ trong câu thông qua cả các từ đứng trước lẫn các từ đứng sau.



Hình 2.2: Quy trình xử lý dựa trên mô hình BiLSTM

CHƯƠNG 3: KHÁM PHÁ VÀ PHÂN TÍCH DỮ LIỆU

3.1 Tổng quan bộ dữ liệu

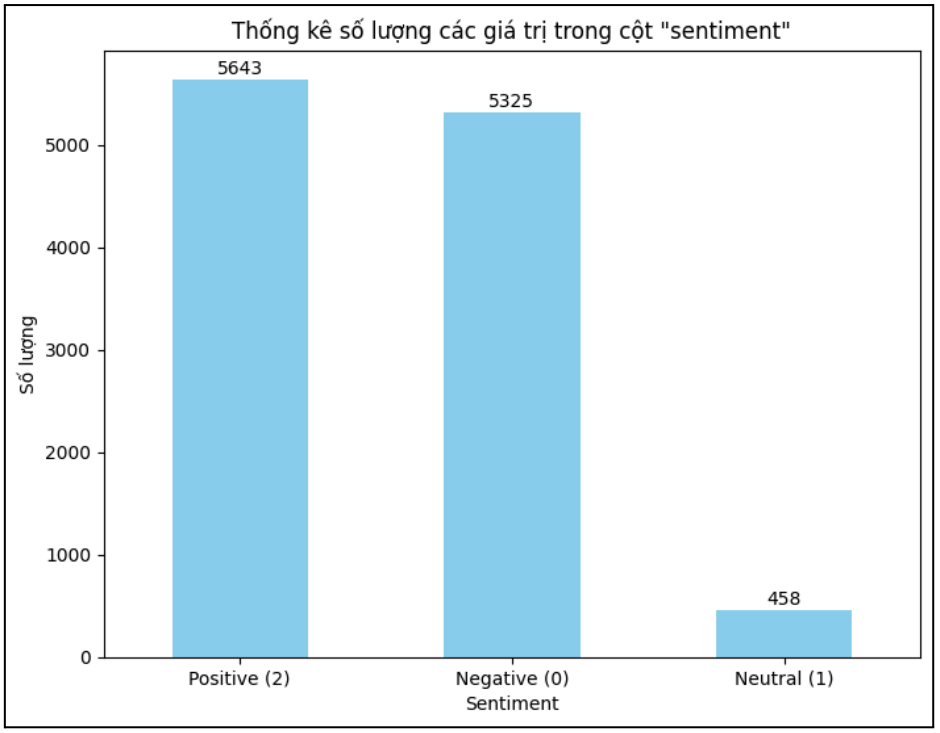
Bộ dữ liệu này được thu thập từ Huggingface và có nguồn gốc từ bài báo **"UIT-VSFC: Vietnamese Students' Feedback Corpus for Sentiment Analysis"**. Dữ liệu phản hồi của sinh viên được ghi nhận từ một trường đại học tại Việt Nam thông qua các khảo sát vào cuối mỗi học kỳ trong khoảng thời gian từ năm 2013 đến 2016, với tổng cộng 16.175 câu phản hồi. Bộ dữ liệu này cung cấp cái nhìn sâu sắc về ý kiến phản hồi của sinh viên và là nguồn tài nguyên quan trọng cho nghiên cứu phân tích cảm xúc.

Mô tả bộ dữ liệu như sau:

Thuộc tính	Mô tả	Giá trị
Sentence	Văn bản mô tả phản hồi của sinh viên.	Chuỗi văn bản(string)
Sentiment	Phân loại cảm xúc phản hồi của sinh viên.	0 (tiêu cực), 1 (trung lập), 2 (tích cực).
Topic	Phân loại chủ đề phản hồi của sinh viên.	0 (giảng viên), 1 (chương trình đào tạo), 2 (cơ sở vật chất), 3 (khác).

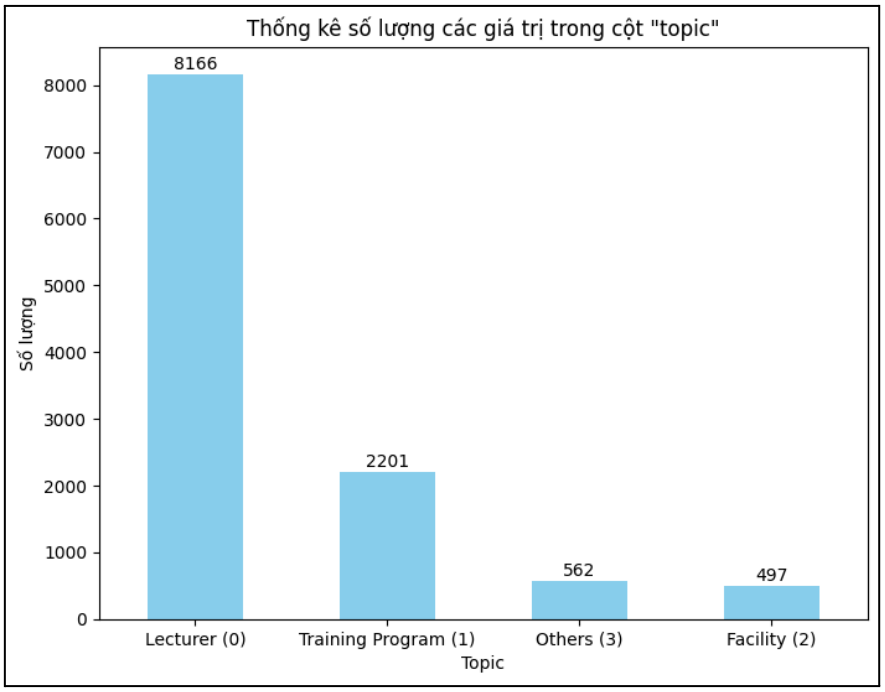
3.2. Phân tích khám phá dữ liệu (Exploratory Data Analysis - EDA)

Đầu tiên, nhóm tiến hành khám phá xem sự phân bố của những nhãn trong cột sentiment theo hình dưới đây



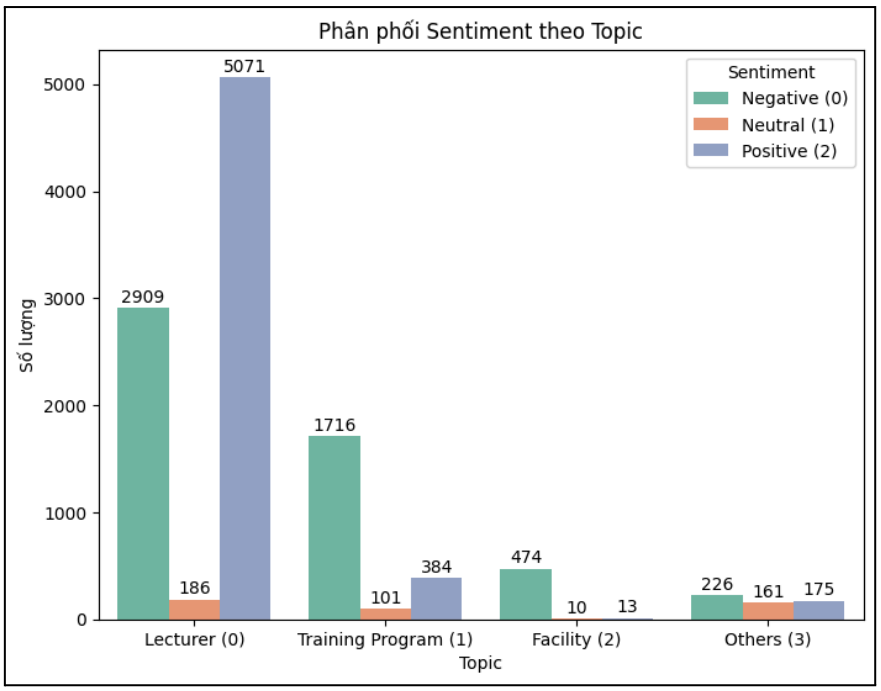
Biểu đồ 3.1: Thống kê số lượng các giá trị trong cột "sentiment"

Nhận thấy được rằng dữ liệu có sự mất cân bằng rõ rệt với số lượng nhãn Neutral (1) là 458 thấp hơn rất nhiều so với hai nhãn còn lại là Positive (2) với 5643 dòng và Negative (0) với 5325 dòng. Do đó nhóm quyết định sẽ khắc phục sự mất cân bằng dữ liệu bằng cách sử dụng phương pháp RandomOverSamplercuar thư viện imblearn, điều này sẽ được đề cập cụ thể hơn ở phần sau. Tiếp đến, nhóm thực hiện thống kê số lượng các giá trị trong cột "topic"



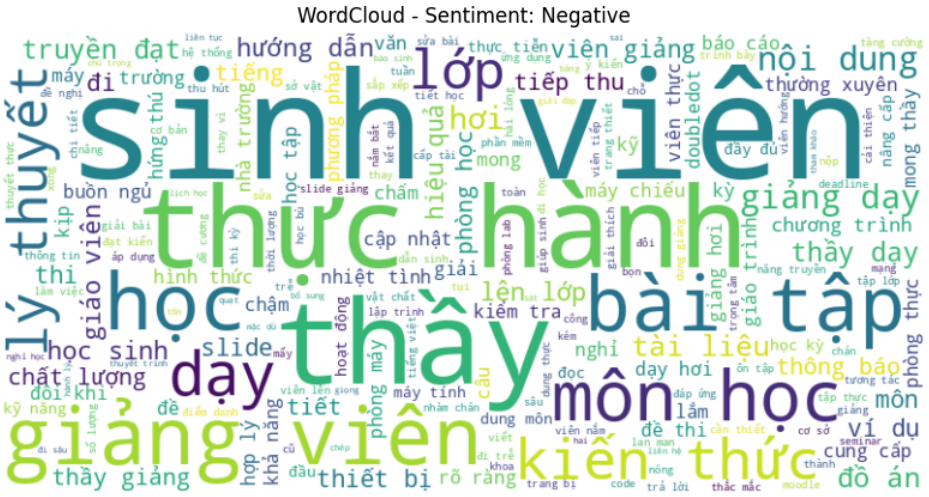
Biểu đồ 3.2: Thống kê số lượng các giá trị trong cột "topic"

Kết quả thu được cho rằng hầu hết những bình luận đánh giá là về chủ đề giảng viên (ví dụ như những bình luận mang tính tích cực về việc giảng viên dạy học hay, quan tâm sinh viên hoặc những bình luận mang tính tiêu cực đề cập đến việc giảng viên khó tính, giao nhiều bài). Những bình luận còn lại phần lớn thuộc về chủ đề trương trình đào tạo, còn chủ đề cơ sở vật chất và những chủ đề khác ít được quan tâm hơn.

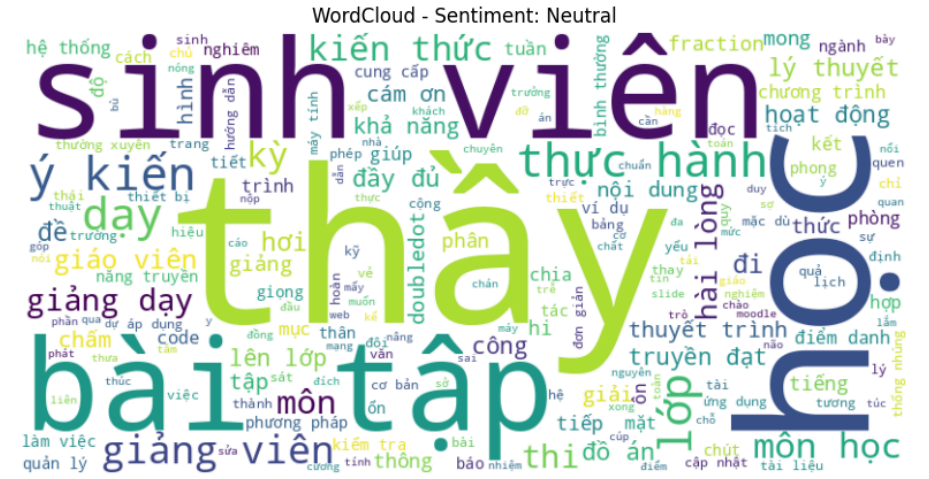


Biểu đồ 3.3: Phân phối Sentiment theo Topic

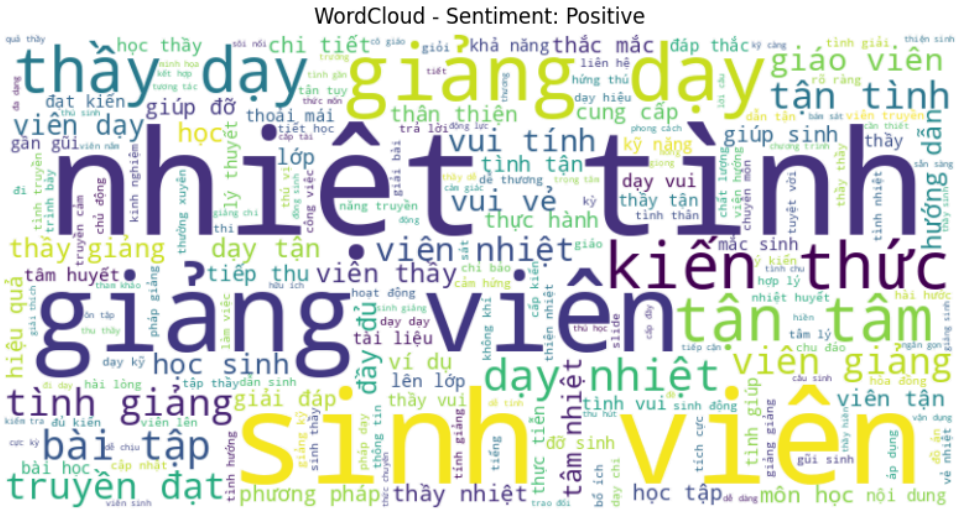
Biểu đồ phân tích phản hồi của sinh viên theo ba loại cảm xúc: tiêu cực, trung lập và tích cực trên bốn chủ đề: “Giảng viên”, “Chương trình đào tạo”, “Cơ sở vật chất” và “Khác”. Chủ đề “Giảng viên” nhận được lượng phản hồi tích cực cao nhất (5071), vượt xa phản hồi trung lập (2909) và tiêu cực (186), cho thấy đa số sinh viên đánh giá cao chất lượng giảng dạy và thái độ của giảng viên. Về “Chương trình đào tạo”, với 1716 phản hồi tích cực, so với 384 trung lập và 101 tiêu cực, chất lượng chương trình được đánh giá cao, phù hợp với phần lớn sinh viên tham gia khảo sát. Chủ đề “Cơ sở vật chất” nhận được số lượng phản hồi ít hơn nhưng tỷ lệ phản hồi tích cực (474) vượt trội so với trung lập (13) và tiêu cực (10), phản ánh mức độ hài lòng đáng kể. Riêng mục “Khác” có sự phân bố cảm xúc cân bằng hơn, với 175 phản hồi tích cực, 161 trung lập và 226 tiêu cực, cho thấy vẫn tồn tại những vấn đề cần được chú ý và cải thiện. Từ phân tích này, có thể thấy việc nâng cao chất lượng giáo dục cần tập trung duy trì và cải thiện các khía cạnh đã được đánh giá cao, đồng thời nghiên cứu sâu hơn các ý kiến tiêu cực, đặc biệt trong mục "Khác", để đưa ra các giải pháp khắc phục hiệu quả.



Hình 3.1: WordCloud cho các phản hồi Negative

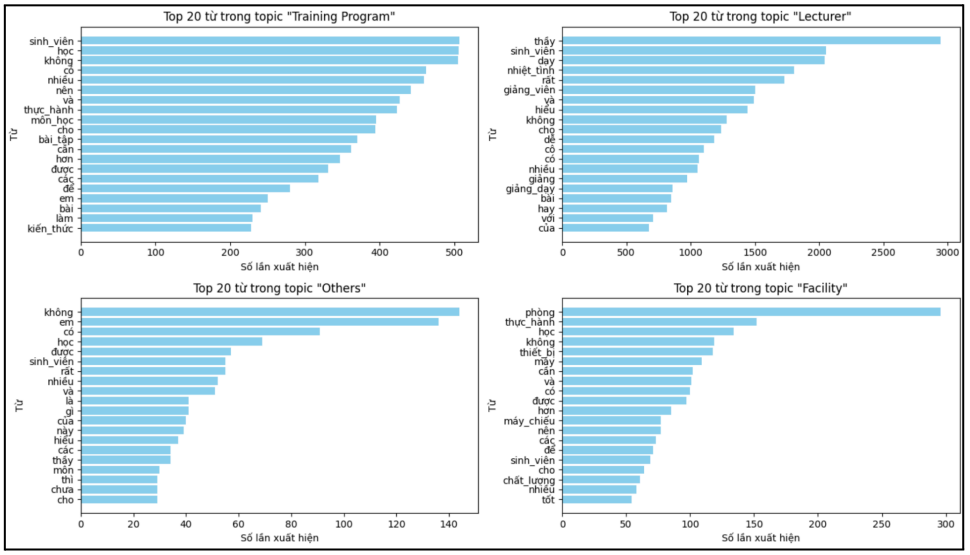


Hình 3.2: WordCloud cho các phản hồi Neutral



Hình 3.3: Wordcloud cho các phản hồi Positive

Có thể thấy qua các wordcloud của từng nhãn thì các từ phổ biến sẽ xoay quanh sinh viên, thầy, giảng viên. Đối với nhãn negative thì sẽ liên quan đến kiến thức, lý thuyết hay truyền đạt trong giảng dạy. Còn positive thì sẽ liên quan tới các cụm từ như nhiệt tình, tận tâm, vui tính ,... Qua các wordcloud có thể thấy những từ phổ biến sẽ xuất hiện trong từng nhãn.



Biểu đồ 3.4: Biểu đồ Top 20 từ trong các topic

Qua biểu đồ trên, ta có thể biết được các cụm từ xuất hiện nhiều nhất trong từng nhãn, từ đó hình dung sơ bộ về những đối tượng sẽ liên quan trong nhãn này.

3.3 Tiền xử lý dữ liệu

3.3.1 Word tokenizer

Word tokenizer là quá trình tách đoạn văn hoặc câu thành các từ riêng biệt, đây là bước quan trọng trong việc chuẩn hóa dữ liệu cho tác vụ phân tích quan điểm . Do đặc thù ngôn ngữ nên việc tách các từ trong tiếng Việt không dùng khoảng trắng để phân tách từ một cách rõ ràng, nó sẽ thách thức hơn so với tiếng Anh. Nhóm tiến hành sử dụng thư viện pyvi trong Python để hỗ trợ tách từ trong tiếng Việt. Kết quả cho thấy thư viện pyvi làm khá tốt trong việc tách các từ trong tiếng Việt.

	sentence	sentiment	topic	processed_text
0	slide giáo trình đầy đủ .	2	1	slide giáo_trình đầy_đủ .
1	nhiệt tình giảng dạy , gần gũi với sinh viên .	2	0	nhiệt_tình giảng_dạy , gần_gũi với sinh_viên .
2	đi học đầy đủ full điểm chuyên cần .	0	1	đi học đầy_đủ full điểm chuyên_cần .
3	chưa áp dụng công nghệ thông tin và các thiết ...	0	0	chưa áp_dụng công_nghệ thông_tin và các thiết_...
4	thầy giảng bài hay , có nhiều bài tập ví dụ ng...	2	0	thầy giảng bài hay , có nhiều bài_tập ví_dụ ng...

Hinh 3.4: Kết quả sau khi tách từ bằng thư viện pyvi

3.3.2 Vectorization

Vectorization là quá trình chuyển đổi dữ liệu văn bản thành các vector số thực mà các mô hình học máy có thể phân tích và xử lý dữ liệu. Nhóm sử dụng 2 phương pháp Vectorization là TF-IDF (Term Frequency-Inverse Document Frequency) và module Tokenizer do thư viện Keras cung cấp để so sánh kết quả.

➢ ***TF-IDF*** **:**

Đây là một phương pháp đánh trọng số cho các từ, giúp xác định mức độ quan trọng của các từ trong một tài liệu. Các thành phần của TF-IDF là TF (Term Frequency) đo lường tần suất một từ xuất hiện trong văn bản so với tổng số từ trong văn bản đó, IDF (Inverse Document Frequency) đo lường mức độ quan trọng của một từ trong toàn bộ văn bản. IDF sẽ giúp giảm trọng số của các từ xuất hiện thường xuyên trong văn bản nhưng không mang lại nhiều thông tin. Và cuối cùng là giá trị TF-IDF cho một từ *t* trong tài liệu *d* tính theo công thức:

$$\text{TF-IDF}(t,d) = \text{TF}(t,d) * \text{IDF}(t)$$

➢ ***Tokenizer:***

Keras tokenizer là một công cụ hữu ích để xử lý văn bản trong học học, nó hỗ trợ chuyển văn bản thành các token, lập từ điển và gán số cho mỗi từ trong văn bản, chuyển đổi văn bản thành chuỗi số nguyên, padding chuỗi để chúng có độ dài cố định. Ngoài ra, ta cũng có thể kiểm soát kích thước từ vựng qua tham số num_words , xử lý các từ lạ (OOV - Out of vocabulary) và truy cập thông tin từ điển nếu cần thiết.

3.3.3 Xử lý dữ liệu mất cân bằng

```
X = df_train.drop('sentiment', axis=1)
y = df_train['sentiment']
ros = RandomOverSampler(random_state=42)
X_resampled, y_resampled = ros.fit_resample(X, y)
df_train_balanced = pd.DataFrame(X_resampled, columns=X.columns)
df_train_balanced['sentiment'] = y_resampled
```

Nhóm tiến hành xử lý dữ liệu mất cân bằng theo phương pháp RandomOverSampler, giúp tăng cường số lượng mẫu của các lớp thiểu số bằng cách tạo thêm bản sao của các mẫu trong lớp đó. Kết quả sau khi áp dụng là số lượng nhãn trong mỗi lớp đã cân bằng.

sentiment	count
2	5643
0	5643
1	5643

dtype: int64

Hình 3.5: Số lượng nhãn trong mỗi lớp sau khi xử lý mất cân bằng

CHƯƠNG 4: HUẤN LUYỆN CÁC MÔ HÌNH

4.1 Xây dựng mô hình

4.1.1 Thực hiện trích xuất đặc trưng TF-IDF và Tokenizer của keras

```
def vectorize(max_features=5000):
      vectorizer = TfidfVectorizer(max_features=max_features)
      X_train_vec = vectorizer.fit_transform(X_train)
      X_val_vec = vectorizer.transform(X_val)
      X_test_vec = vectorizer.transform(X_test)

      return X_train_vec, X_val_vec, X_test_vec, vectorizer

X_train, y_train = df_train_balanced['processed_text'],
df_train_balanced['sentiment']
X_val, y_val = df_val['processed_text'], df_val['sentiment']
X_test, y_test = df_test['processed_text'], df_test['sentiment']
X_train_vec_tf, X_val_vec_tf, X_test_vec_tf, vectorizer_tf =
vectorize()
```

```
tokenizer_data1 = Tokenizer(oov_token = '<oov>', filters = '', split
= ' ')
tokenizer_data1.fit_on_texts(X_train)

tokenized_data_train1 = tokenizer_data1.texts_to_sequences(X_train)
tokenized_data_val1 = tokenizer_data1.texts_to_sequences(X_val)
tokenized_data_test1 = tokenizer_data1.texts_to_sequences(X_test)


vec_data_train1 = pad_sequences(tokenized_data_train1, maxlen = 50,
padding = 'post', truncating = 'post')
vec_data_val1 = pad_sequences(tokenized_data_val1, maxlen = 50,
padding = 'post', truncating = 'post')
vec_data_test1 = pad_sequences(tokenized_data_test1, maxlen = 50,
padding = 'post', truncating = 'post')

tokenizer_vocab1 = tokenizer_data1.word_index
tokenizer_vocab1 = {word: index - 1 for word, index in
tokenizer_vocab1.items()}

vectorizer1 = CountVectorizer(vocabulary=tokenizer_vocab1,
max_features=5000)
```

```
X_train_vec1 = vectorizer1.fit_transform(X_train).toarray()
X_val_vec1 = vectorizer1.transform(X_val).toarray()
X_test_vec1 = vectorizer1.transform(X_test).toarray()
```

Nhóm tiến hành sử dụng TF-IDF và Tokenizer của keras để trích xuất đặc trưng cho 3 tập train ,validation và test trước khi tìm kiếm siêu tham số và huấn luyện các mô hình.

4.1.2 Xây dựng mô hình XGBoost

```
param_grid = {
    'n_estimators': [50, 100, 200],
    'learning_rate': [0.01, 0.1, 0.2],
    'max_depth': [3, 5, 7],
    'subsample': [0.8, 0.9, 1.0],
    'colsample_bytree': [0.8, 0.9, 1.0],
}
xgb_model = xgb.XGBClassifier(objective='multi:softmax', num_class=3,
eval_metric='merror',
                              use_label_encoder=False, random_state =
42)
random_search_tf = RandomizedSearchCV(estimator=xgb_model,
param_distributions=param_grid,
                                      n_iter=5, scoring='accuracy',
cv=3, verbose=2, random_state=42, n_jobs=-1)

random_search_tf.fit(X_train_vec_tf, y_train)

print("Best parameters:", random_search_tf.best_params_)
print("Best score:", random_search_tf.best_score_)

best_xgb_model_tf = random_search_tf.best_estimator_

y_pred_val_tf = best_xgb_model_tf.predict(X_val_vec_tf)
print(classification_report(y_val, y_pred_val_tf));
```

Đối với mô hình XGBoost, nhóm sử dụng RamdomizedSearch để tìm kiếm các siêu tham số cho mô hình. Đầu tiên là khởi tạo mô hình cho bài toán phân loại đa lớp, sử dụng tỷ lệ lỗi phân loại để làm thước đo đánh giá. Các siêu tham số bao gồm:

- n_estimators: số lượng cây trong mô hình

- learning_rate: tốc độ học, ảnh hưởng đến việc cập nhật các trọng số trong quá trình huấn luyện
- max_depth: độ sâu tối đa của mỗi cây quyết định
- subsample: tỷ lệ mẫu được lấy ngẫu nhiên trong bộ dữ liệu để xây dựng mỗi cây quyết định
- colsample_bytree: tỷ lệ số lượng đặc trưng được chọn ngẫu nhiên để xây dựng mỗi cây (giảm overfitting)

TF-IDF : Đối với phương pháp vectorizer TF-IDF

```
  warnings.warn(smsg, UserWarning)
Best parameters: {'subsample': 0.9, 'n_estimators': 200, 'max_depth': 5, 'learning_rate': 0.2, 'colsample_bytree': 1.0}
Best score: 0.9234449760765551
              precision    recall  f1-score   support

           0       0.92      0.89      0.90       705
           1       0.34      0.64      0.44        73
           2       0.94      0.89      0.91       805

    accuracy                           0.88      1583
   macro avg       0.73      0.81      0.75      1583
weighted avg       0.90      0.88      0.89      1583
```

Hình 4.1: Kết quả tìm kiếm siêu tham số của XGBoost - TF-IDF

Kết quả cho thấy mô hình tốt nhất với các tham số subsample là 0.9, n_estimaters là 200, max_depth là 5, learning_rate là 0.2, colsample_bytree là 1. Và độ đo accuracy là 0.88, f1 macro là 0.75 cho thấy kết quả khá tốt, kết quả phân loại nhãn 0 và 2 rất tốt, tuy nhiên hiệu suất của nhãn 1 thấp dẫn đến f1 macro cũng thấp theo.

Tokenizer của keras : Phương pháp tokenizer của thư viện keras

```
  warnings.warn(smsg, UserWarning)
Best parameters: {'subsample': 0.9, 'n_estimators': 200, 'max_depth': 5, 'learning_rate': 0.2, 'colsample_bytree': 1.0}
Best score: 0.9104495244846121
              precision    recall  f1-score   support

           0       0.94      0.88      0.91       705
           1       0.29      0.73      0.41        73
           2       0.95      0.87      0.91       805

    accuracy                           0.87      1583
   macro avg       0.73      0.83      0.74      1583
weighted avg       0.92      0.87      0.89      1583
```

Hình 4.2: Kết quả tìm kiếm siêu tham số của XGBoost - Tokenizer

Kết quả cho thấy phương pháp TF-IDF có chỉ số accuracy và F1 macro cao hơn tokenizer của keras tí 1 nhưng không đáng kể, có thể nói không có sự khác biệt nhiều giữa 2 cách vectorizer.

4.1.3 Xây dựng mô hình Naive Bayes

```
print("\nTuning Multinomial Naive Bayes...")
mnb = MultinomialNB()
mnb_params = {
    'alpha': [0.1, 0.5, 1, 2, 5, 10],
    'fit_prior': [True, False]
}
```

```
mnb_grid_tf = GridSearchCV(mnb, mnb_params, cv=5, scoring='accuracy')
mnb_grid_tf.fit(X_train_vec_tf, y_train)


print("Best parameters for MultinomialNB:", mnb_grid_tf.best_params_)
y_pred_mnb_tf = mnb_grid_tf.best_estimator_.predict(X_val_vec_tf)
print(classification_report(y_val, y_pred_mnb_tf))
```

Nhóm tiến hành tìm kiếm các siêu tham số bằng GridSearcgCV theo 2 tham số:

- alpha: Tham số smoothing điều chỉnh độ trơn, giá trị nhỏ làm giảm khả năng bỏ qua các đặc trưng hiếm.
- fit_prior: Quyết định việc sử dụng xác suất tiên nghiệm (prior probabilities).

TF-IDF : Đối với phương pháp vectorizer TF-IDF

```
Tuning Multinomial Naive Bayes...
Best parameters for MultinomialNB: {'alpha': 0.1, 'fit_prior': True}
              precision    recall  f1-score   support

           0       0.84      0.86      0.85       705
           1       0.27      0.49      0.35        73
           2       0.93      0.85      0.89       805

    accuracy                           0.84      1583
   macro avg       0.68      0.73      0.70      1583
weighted avg       0.86      0.84      0.85      1583
```

Hình 4.3: Kết quả tìm kiếm siêu tham số của Multinomial Naive bayes - TF-IDF

Tokenizer của keras : Phương pháp tokenizer của thư viện keras

```
Tuning Multinomial Naive Bayes...
Best parameters for MultinomialNB: {'alpha': 0.1, 'fit_prior': True}
              precision    recall  f1-score   support

           0       0.88      0.86      0.87       705
           1       0.30      0.48      0.37        73
           2       0.92      0.89      0.91       805

    accuracy                           0.86      1583
   macro avg       0.70      0.74      0.71      1583
weighted avg       0.87      0.86      0.86      1583
```

Hình 4.4: Kết quả tìm kiếm siêu tham số của Multinomial Naive bayes - Tokenizer

4.1.4 Xây dựng mô hình SVM

```
print("\nTuning SVM...")
svm = SVC()
svm_params = {
    'C': [1, 10],
    'kernel': ['linear', 'rbf'],
    'gamma': [1e-2, 0.1],
}
svm_grid_tf = GridSearchCV(svm, svm_params, cv=3, scoring='accuracy')
svm_grid_tf.fit(X_train_vec_tf, y_train)

print("Best parameters for SVM:", svm_grid_tf.best_params_)
y_pred_svm_tf = svm_grid_tf.best_estimator_.predict(X_val_vec_tf)
print(classification_report(y_val, y_pred_svm_tf))
```

TF-IDF : Đối với phương pháp vectorizer TF-IDF

```
Tuning SVM...
Best parameters for SVM: {'C': 10, 'gamma': 0.1, 'kernel': 'rbf'}
              precision    recall  f1-score   support

           0       0.90      0.90      0.90       705
           1       0.31      0.52      0.39        73
           2       0.95      0.90      0.92       805

    accuracy                           0.88      1583
   macro avg       0.72      0.77      0.74      1583
weighted avg       0.90      0.88      0.89      1583
```

Hình 4.5: Kết quả tìm kiếm siêu tham số của SVM - TF-IDF

Tokenizer của keras : Phương pháp tokenizer của thư viện keras

```
Tuning SVM...
Best parameters for SVM: {'C': 10, 'gamma': 0.1, 'kernel': 'rbf'}
              precision    recall  f1-score   support

           0       0.90      0.94      0.92       705
           1       0.49      0.49      0.49        73
           2       0.95      0.91      0.93       805

    accuracy                           0.90      1583
   macro avg       0.78      0.78      0.78      1583
weighted avg       0.90      0.90      0.90      1583
```

Hình 4.6: Kết quả tìm kiếm siêu tham số của SVM - Tokenizer

4.1.5 Xây dựng mô hình học sâu BiLSTM từ TF-IDF

Đầu tiên, dữ liệu văn bản đã được vector hóa sử dụng TF-IDF chứa các đặc trưng của dữ liệu huấn luyện và kiểm thử. Tiếp theo, để đảm bảo tính đồng nhất cho mô hình học sâu, dữ liệu được chuyển đổi từ dạng sparse matrix sang dạng ma trận đặc trưng dày (dense matrix) và tiếp tục chuyển thành Tensor dưới dạng float32. Để đảm bảo tương thích với đầu vào của LSTM, dữ liệu được mở rộng thêm một chiều time-step, tạo thành kích thước (batch_size, num_features, 1).

Mô hình được xây dựng với kiến trúc chính là Bidirectional LSTM nhằm tận dụng khả năng học thông tin từ cả hai chiều của dữ liệu. Lớp LSTM đầu tiên có 30 đơn vị, tích hợp cơ chế loại bỏ ngẫu nhiên (dropout) để tăng cường khả năng tổng quát của mô hình. Lớp LSTM thứ hai tiếp tục xử lý sâu hơn các đặc trưng từ lớp trước đó, đồng thời giữ lại thông tin theo chuỗi để phục vụ các bước xử lý tiếp theo. Sau đó, lớp trung bình hóa toàn cục (Global Average Pooling) được sử dụng để giảm chiều dữ liệu, giúp tăng hiệu quả xử lý.

Tiếp theo, các lớp kết nối đầy đủ (Dense) được thêm vào nhằm học các mối quan hệ phức tạp hơn trong dữ liệu, đồng thời áp dụng kỹ thuật Dropout giữa các lớp để hạn chế hiện tượng quá khớp. Bài toán được xác định là phân lớp với 3 nhãn chính là “positive”, “negative” và “Neutral” vậy nên ở layer cuối cùng, nhóm sử dụng hàm kích hoạt (activation function) là hàm “softmax”.

Model: "sequential"

Layer (type)	Output Shape	Param #
bidirectional (Bidirectional)	(None, 3664, 120)	29,760
bidirectional_1 (Bidirectional)	(None, 3664, 120)	86,880
global_average_pooling1d (GlobalAveragePooling1D)	(None, 120)	0
dense (Dense)	(None, 40)	4,840
dropout (Dropout)	(None, 40)	0
dense_1 (Dense)	(None, 30)	1,230
dropout_1 (Dropout)	(None, 30)	0
dense_2 (Dense)	(None, 20)	620
dropout_2 (Dropout)	(None, 20)	0
dense_3 (Dense)	(None, 8)	168
dense_4 (Dense)	(None, 3)	27

Total params: 123,525 (482.52 KB)
Trainable params: 123,525 (482.52 KB)
Non-trainable params: 0 (0.00 B)

Hình 4.7: Tóm tắt mô hình học sâu BiLSTM từ TF-IDF

```
Epoch 1/10
265/265 ━━━━━━━━━━━━━━━━━━━━ 127s 453ms/step - accuracy: 0.3333 - loss: 1.0989 - val_accuracy: 0.5085 - val_loss: 1.0968
Epoch 2/10
265/265 ━━━━━━━━━━━━━━━━━━━━ 137s 447ms/step - accuracy: 0.3356 - loss: 1.0988 - val_accuracy: 0.5085 - val_loss: 1.0993
Epoch 3/10
265/265 ━━━━━━━━━━━━━━━━━━━━ 148s 469ms/step - accuracy: 0.3309 - loss: 1.0990 - val_accuracy: 0.0461 - val_loss: 1.1008
Epoch 4/10
265/265 ━━━━━━━━━━━━━━━━━━━━ 137s 449ms/step - accuracy: 0.3310 - loss: 1.0987 - val_accuracy: 0.0461 - val_loss: 1.1031
Epoch 5/10
265/265 ━━━━━━━━━━━━━━━━━━━━ 141s 445ms/step - accuracy: 0.3365 - loss: 1.0986 - val_accuracy: 0.5085 - val_loss: 1.0976
Epoch 6/10
265/265 ━━━━━━━━━━━━━━━━━━━━ 141s 441ms/step - accuracy: 0.3284 - loss: 1.0987 - val_accuracy: 0.0461 - val_loss: 1.1010
Epoch 7/10
265/265 ━━━━━━━━━━━━━━━━━━━━ 142s 441ms/step - accuracy: 0.3302 - loss: 1.0987 - val_accuracy: 0.4454 - val_loss: 1.0985
Epoch 8/10
265/265 ━━━━━━━━━━━━━━━━━━━━ 142s 442ms/step - accuracy: 0.3331 - loss: 1.0987 - val_accuracy: 0.4454 - val_loss: 1.0978
Epoch 9/10
265/265 ━━━━━━━━━━━━━━━━━━━━ 142s 441ms/step - accuracy: 0.3281 - loss: 1.0986 - val_accuracy: 0.5085 - val_loss: 1.0967
Epoch 10/10
265/265 ━━━━━━━━━━━━━━━━━━━━ 117s 442ms/step - accuracy: 0.3267 - loss: 1.0987 - val_accuracy: 0.5085 - val_loss: 1.0975
```

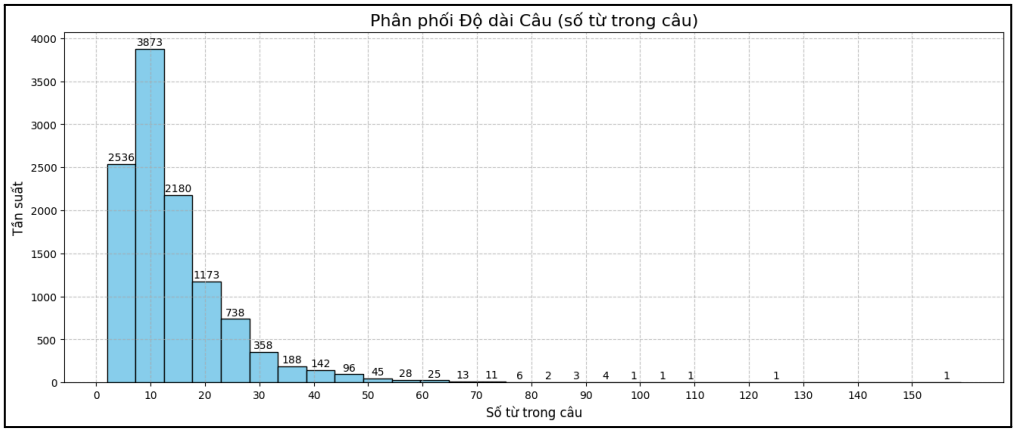
Hình 4.8: Kết quả huấn luyện của BiLSTM - tokenizer

```
Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       705
           1       0.00      0.00      0.00        73
           2       0.51      1.00      0.67       805

    accuracy                           0.51      1583
   macro avg       0.17      0.33      0.22      1583
weighted avg       0.26      0.51      0.34      1583
```

Hình 4.9: Kết quả dự đoán của BiLSTM- TF-IDF

4.1.6 Xây dựng mô hình học sâu BiLSTM từ Keras tokenizer

Nhóm thực hiện vẽ biểu đồ để quan sát độ dài của câu (số từ trong câu)



Biểu đồ 4.1: Phân phối Độ dài Câu (số từ trong câu)

Qua biểu đồ nhóm nhận thấy rằng hầu hết các câu có độ dài lớn hơn hoặc bằng 50 từ do đó nhóm quyết định sẽ sử dụng vecto với độ dài 50 và chấp nhận mất mát thông tin với những câu có độ dài lớn hơn 50.

Mô hình được xây dựng giống với mô hình học sâu BiLSTM từ TF-IDF chỉ khác lúc này cần thêm lớp Embedding để ánh xạ mỗi từ thành một vector có 60 chiều, giúp biểu diễn từ ngữ trong không gian liên tục và giảm chiều dữ liệu.

Model: "sequential_1"

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 50, 60)	222,360
bidirectional_2 (Bidirectional)	(None, 50, 120)	58,080
bidirectional_3 (Bidirectional)	(None, 50, 120)	86,880
global_average_pooling1d_1 (GlobalAveragePooling1D)	(None, 120)	0
dense_5 (Dense)	(None, 40)	4,840
dropout_3 (Dropout)	(None, 40)	0
dense_6 (Dense)	(None, 30)	1,230
dropout_4 (Dropout)	(None, 30)	0
dense_7 (Dense)	(None, 20)	620
dropout_5 (Dropout)	(None, 20)	0
dense_8 (Dense)	(None, 8)	168
dense_9 (Dense)	(None, 3)	27

Total params: 374,205 (1.43 MB)
Trainable params: 374,205 (1.43 MB)
Non-trainable params: 0 (0.00 B)

Hình 4.10: Tóm tắt mô hình học sâu BiLSTM từ Keras tokenizer

```
Epoch 1/15
265/265 ━━━━━━━━━━━━━━━━━━━━ 12s 25ms/step - accuracy: 0.3362 - loss: 1.0982 - val_accuracy: 0.1207 - val_loss: 1.1128
Epoch 2/15
265/265 ━━━━━━━━━━━━━━━━━━━━ 7s 25ms/step - accuracy: 0.4384 - loss: 1.0536 - val_accuracy: 0.4296 - val_loss: 1.1217
Epoch 3/15
265/265 ━━━━━━━━━━━━━━━━━━━━ 13s 37ms/step - accuracy: 0.5515 - loss: 0.9256 - val_accuracy: 0.4416 - val_loss: 1.0562
Epoch 4/15
265/265 ━━━━━━━━━━━━━━━━━━━━ 9s 34ms/step - accuracy: 0.5725 - loss: 0.8550 - val_accuracy: 0.4574 - val_loss: 0.9987
Epoch 5/15
265/265 ━━━━━━━━━━━━━━━━━━━━ 6s 17ms/step - accuracy: 0.6161 - loss: 0.7863 - val_accuracy: 0.7056 - val_loss: 0.7777
Epoch 6/15
265/265 ━━━━━━━━━━━━━━━━━━━━ 5s 16ms/step - accuracy: 0.7717 - loss: 0.6258 - val_accuracy: 0.8130 - val_loss: 0.5346
Epoch 7/15
265/265 ━━━━━━━━━━━━━━━━━━━━ 6s 21ms/step - accuracy: 0.8384 - loss: 0.5009 - val_accuracy: 0.8610 - val_loss: 0.4326
Epoch 8/15
265/265 ━━━━━━━━━━━━━━━━━━━━ 4s 15ms/step - accuracy: 0.8674 - loss: 0.4252 - val_accuracy: 0.8541 - val_loss: 0.4190
Epoch 9/15
265/265 ━━━━━━━━━━━━━━━━━━━━ 4s 15ms/step - accuracy: 0.8794 - loss: 0.3868 - val_accuracy: 0.8522 - val_loss: 0.4554
Epoch 10/15
265/265 ━━━━━━━━━━━━━━━━━━━━ 5s 20ms/step - accuracy: 0.8885 - loss: 0.3631 - val_accuracy: 0.8446 - val_loss: 0.4456
Epoch 11/15
265/265 ━━━━━━━━━━━━━━━━━━━━ 9s 16ms/step - accuracy: 0.9014 - loss: 0.3273 - val_accuracy: 0.8699 - val_loss: 0.3987
Epoch 12/15
265/265 ━━━━━━━━━━━━━━━━━━━━ 6s 21ms/step - accuracy: 0.9061 - loss: 0.3165 - val_accuracy: 0.8484 - val_loss: 0.4436
Epoch 13/15
265/265 ━━━━━━━━━━━━━━━━━━━━ 4s 15ms/step - accuracy: 0.9066 - loss: 0.3015 - val_accuracy: 0.8629 - val_loss: 0.4197
Epoch 14/15
265/265 ━━━━━━━━━━━━━━━━━━━━ 5s 16ms/step - accuracy: 0.9161 - loss: 0.2826 - val_accuracy: 0.8711 - val_loss: 0.4104
Epoch 15/15
265/265 ━━━━━━━━━━━━━━━━━━━━ 6s 19ms/step - accuracy: 0.9205 - loss: 0.2667 - val_accuracy: 0.8800 - val_loss: 0.3793
```

Hình 4.11: Kết quả huấn luyện của BiLSTM - tokenizer

```
Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.90      0.89       705
           1       0.35      0.55      0.43        73
           2       0.95      0.89      0.92       805

    accuracy                           0.88      1583
   macro avg       0.73      0.78      0.75      1583
weighted avg       0.89      0.88      0.88      1583
```

Hình 4.12: Kết quả dự đoán của BiLSTM - tokenizer

CHƯƠNG 5: KẾT LUẬN VÀ ĐÁNH GIÁ

5.1 Kết quả đạt được

	Model	Accuracy	F1 Micro	F1 Macro	Recall Micro	Recall Macro
0	XGBoost (TF-IDF)	0.865130	0.865130	0.737370	0.865130	0.774117
1	Naive Bayes (TF-IDF)	0.820910	0.820910	0.669123	0.820910	0.688285
2	SVM (TF-IDF)	0.872710	0.872710	0.727759	0.872710	0.739058
3	LSTM	0.869236	0.869236	0.736566	0.869236	0.757515

Hình 5.1: Kết quả tốt nhất

Kết quả cho thấy mô hình SVM với phương pháp vectorization là TF-IDF cho accuracy, F1 micro cao nhất và kế đến là mô hình LSTM với cách tokenizer của thư viện Keras. Tuy nhiên thì mô hình LSTM có chỉ số F1 macro tốt hơn cho thấy mô hình có hiệu suất giữa các nhãn tốt hơn SVM, hoạt động ổn định. XGBoost cũng là một mô hình mạnh với Recall Macro cao nhất, thể hiện khả năng dự đoán nhạy các nhãn. Cuối cùng là Naive Bayes có kết quả thất nhất, mặc dù accuracy khá cao nhưng F1 macro lại thấp nhất, cho thấy hiệu suất kém khi thực hiện tác vụ phân tích quan điểm này.

5.2 Hạn chế

- Trong phạm vi đề tài, nhóm hiện đang sử dụng hai phương pháp trích xuất vector đặc trưng văn bản là Tokenizer từ thư viện Keras và TF-IDF. Hai phương pháp này có nhược điểm là không đảm bảo giữ nguyên ngữ nghĩa của văn bản nên dẫn đến mất mát thông tin khi trích xuất các đặc trưng.
- Bộ dữ liệu nhóm hiện sử dụng để thực hiện nghiên cứu hiện gặp tình trạng mất cân bằng giữa các nhãn khá nặng (nhãn Neutral). Nhóm hiện đang chỉ sử dụng phương pháp Oversampling nhằm cân bằng lại các nhãn dẫn đến chưa so sánh được hiệu suất các mô hình khi sử dụng nhiều kĩ thuật cân bằng nhãn khác nhau.
- Các mô hình học máy hiện tại chưa được tối ưu hóa đầy đủ thông qua việc tìm kiếm trên phạm vi rộng các siêu tham số, dẫn đến hiệu suất chưa đạt mức cao nhất. Đồng thời, mô hình học sâu BiLSTM mà nhóm đang sử dụng vẫn chưa được tinh chỉnh chi tiết từng lớp, khiến cho việc khai thác tiềm năng tối đa của mô hình chưa được thực hiện hiệu quả.

5.3 Các phương pháp cải tiến

- Sử dụng các mô hình trích xuất đặc trưng nhưng vẫn giữ được ngữ nghĩa của câu. Có thể kể đến các mô hình nhúng từ (word embeddings) như Word2Vec, GloVe,... để biểu diễn từ ngữ dưới dạng vector trong không gian số. Các vector này không chỉ mã hóa từ ngữ mà còn thể hiện mối quan hệ ngữ nghĩa giữa chúng, giúp trích xuất đặc trưng hiệu quả hơn cho các mô hình học máy và học sâu.

- Tăng cường dữ liệu (Data Augmentation): Tìm kiếm thêm dữ liệu mới hoặc tạo dữ liệu tổng hợp để mở rộng tập huấn luyện, từ đó giúp mô hình học tốt hơn. Đặc biệt, tập trung vào cân bằng dữ liệu giữa các nhãn (ví dụ như nhãn Neutral bị mất cân bằng nặng) để tránh tình trạng lệch nhãn, đảm bảo mô hình hoạt động tốt hơn trên các nhãn ít xuất hiện.
- Sử dụng các mô hình hiện đại nhằm cải thiện hiệu suất
  Ví dụ như các mô hình dựa trên kiến trúc Transformer, như PhoBERT, mBERT,... để xử lý văn bản. Các mô hình này học được các mối quan hệ dài hạn giữa các từ và hiểu được ngữ cảnh mạnh mẽ, giúp cải thiện hiệu suất trong các tác vụ NLP như phân loại, sinh văn bản, và trả lời câu hỏi.
- Fine-tuning từng lớp cho mô hình học sâu hoặc mô hình đã được huấn luyện trước: Sử dụng các mô hình đã được huấn luyện trước (pretrained models) hoặc xây dựng các mô hình học sâu, sau đó thực hiện fine-tuning trên các lớp phù hợp. Ví dụ, có thể cố định các lớp đầu tiên và chỉ huấn luyện các lớp cuối để mô hình tập trung học các đặc trưng đặc thù của bài toán mà không làm mất đi các đặc trưng cơ bản đã được học từ tập dữ liệu lớn.

TÀI LIỆU THAM KHẢO

[1] Joshi, A. K. (1991). Natural Language Processing. Science, 253(5025), 1242–1249.
[2] Khurana, D., Koli, A., Khatter, K., & Singh, S. (2022). Natural language processing: state of the art, current trends and challenges. Multimedia Tools and Applications, 82(3), 3713–3744.
[3] Medhat, W., Hassan, A., Korashy, H., Ain Shams University, & Canadian International College. (2014). Sentiment analysis algorithms and applications: A survey. In Ain Shams Engineering Journal (Vols. 5–1093, pp. 1093–1113) [Journal-article]. Elsevier.
[4] Wilson T, Wiebe J, Hoffman P. (2005) Recognizing contextual polarity in phrase-level sentiment analysis. In: Proceedings of HLT/EMNLP.
[5] Tung, B. T. (2024, December 15). Gradient Boosting - Tất tần tật về thuật toán mạnh mẽ nhất trong Machine Learning. Viblo.
[6] Phạm, D. (2024, December 15). *TF-IDF (term frequency – inverse document frequency)*. Viblo.
[7] Van Huy, T. (n.d.). *TF-IDF*.
[8] Ong H. (2019, April 8). *XGBoost: thuật toán giành chiến thắng tại nhiều cuộc thi Kaggle*. Ông Xuân Hồng.
[9] Jha, A. (2023, August 11). *Vectorization Techniques in NLP [Guide]*. neptune.ai.