BỘ GIÁO DỤC VÀ ĐÀO TẠO
ĐẠI HỌC KINH TẾ TP. HỒ CHÍ MINH
TRƯỜNG CÔNG NGHỆ VÀ THIẾT KẾ
KHOA CÔNG NGHỆ THÔNG TIN KINH DOANH

ĐỒ ÁN MÔN HỌC

XỬ LÝ NGÔN NGỮ TỰ NHIÊN

Đề tài:

PHÂN LOẠI CẢM XÚC VÀ CHỦ ĐỀ DỰA TRÊN ĐÁNH GIÁ CỦA SINH VIÊN VIỆT NAM

Thành viên:

1. Võ Ngọc Mỹ Kim
2. Văn Ngọc Như Quỳnh
3. Nguyễn Nhật Thảo Vy

Giảng viên: *Ts.* Đặng Ngọc Hoàng Thành

Thành phố Hồ Chí Minh, ngày 21 tháng 12 năm 2022

MỤC LỤC

CHƯƠNG I: TỔNG QUAN ĐỀ TÀI

1.1. Giới thiệu đề tài

Giáo dục là một lĩnh vực đóng vai trò quan trọng đối với sự phát triển của mỗi quốc gia, và việc đảm bảo chất lượng giáo dục luôn là một ưu tiên hàng đầu. Trong thời đại số hóa hiện nay, đặc biệt là ở Việt Nam, việc thu thập và phân tích phản hồi từ học sinh trở nên vô cùng quan trọng. Nó không chỉ giúp cải thiện chất lượng giáo dục mà còn đóng góp vào sự phát triển của ngành công nghệ thông tin, đặc biệt là lĩnh vực xử lý ngôn ngữ tự nhiên và phân tích cảm xúc. Vì thế, ***"Phân loại Cảm xúc và Chủ đề" (Sentiment and Topic Classification)*** đã trở thành mục tiêu nghiên cứu chính của nhiều nhà giáo dục và nhà nghiên cứu. Hiện nay, có rất nhiều mô hình giúp người nghiên cứu có thể có cái nhìn tổng quan nhất về vấn đề, thu thập được các nhận định, đánh giá từ học sinh để cải thiện chất lượng giáo dục theo nhu cầu của học sinh, nâng cao chất lượng giáo dục và tạo ra môi trường học tập tốt nhất.

Để hiểu rõ hơn về vấn đề này, nhóm chúng em đã cùng nhau tìm hiểu bài toán ***"Phân loại Cảm xúc và Chủ đề dựa trên Đánh giá của Sinh viên Việt Nam" (Sentiment and Topic Classification on Vietnamese Students' Feedback Corpus)*** nhằm thu thập phản hồi của học sinh Việt Nam về chất lượng giáo dục. Sự phát triển của các công cụ để cải thiện chất lượng giáo dục là hoàn toàn cần thiết và có lợi cho các trường học ngày nay.

1.2. Mục tiêu nghiên cứu

Đề tài này tập trung vào việc xây dựng mô hình phân tích, dựa trên phản hồi của sinh viên để đánh giá chất lượng giáo dục tại Việt Nam. Nhóm đã sử dụng các mô hình Random Forest, Logistic Regression và LSTM để phân loại cảm xúc và chủ đề từ phản hồi của sinh viên. Mục tiêu chính là hỗ trợ các cơ sở giáo dục nắm bắt được ý kiến của học sinh, từ đó đưa ra các phương án cải tiến chất lượng giáo dục một cách hiệu quả nhất.

1.3. Phương pháp nghiên cứu:

- Tiền xử lý dữ liệu: loại bỏ giá trị bị thiếu, loại bỏ dấu câu, các kí tự lạ, tách từ và embedding.
- Sử dụng các mô hình phân lớp để phân loại phản hồi của sinh viên và đánh giá chất lượng giáo dục.
- Đánh giá kết quả: Dùng các chỉ số đánh giá như Accuracy, Recall, F1 Score hay Precision để chọn ra thuật toán tối ưu nhất.

1.4. Tài nguyên sử dụng

- Ngôn ngữ sử dụng: Ngôn ngữ lập trình Python

- Bộ dữ liệu: Vietnamese Students' Feedback Corpus

CHƯƠNG II: CƠ SỞ LÝ THUYẾT

2.1. Các Chỉ Số Đánh Giá Mô Hình

Trong bài báo cáo này, nhóm đã sử dụng các chỉ số để đánh giá hiệu suất của mô hình như sau:

- **Accuracy**: Độ chính xác của mô hình, được tính bằng tỷ lệ giữa số mẫu được dự đoán đúng và tổng số mẫu. Accuracy càng cao, mô hình càng tốt.
- **Recall**: Độ phủ của mô hình, được tính bằng tỷ lệ giữa số mẫu thuộc một lớp được dự đoán đúng và số mẫu thực sự thuộc lớp đó. Recall càng cao, mô hình càng ít bỏ sót các mẫu thuộc lớp đó.
- **Precision**: Độ chính xác của mô hình, được tính bằng tỷ lệ giữa số mẫu thuộc một lớp được dự đoán đúng và số mẫu được dự đoán thuộc lớp đó. Precision càng cao, mô hình càng ít nhầm lẫn các mẫu của các lớp khác với lớp đó.
- **F1 score**: Điểm F1 của mô hình, được tính bằng trung bình điều hòa của precision và recall. F1 score càng cao, mô hình càng cân bằng giữa precision và recall.
- **Confusion matrix**: Ma trận nhầm lẫn của mô hình, là một bảng biểu thể hiện số lượng các mẫu thuộc mỗi lớp thực tế và mỗi lớp dự đoán. Confusion matrix giúp đánh giá hiệu suất của mô hình trên từng lớp và phát hiện các lỗi phân lớp thường gặp.
- **Biểu đồ ROC:** Biểu đồ ROC (Receiver Operating Characteristic) là một biểu đồ thể hiện khả năng phân loại chính xác của một mô hình dự đoán. Biểu đồ ROC giúp đánh giá và so sánh hiệu suất của các mô hình dự đoán. Biểu đồ ROC càng gần góc trên bên trái, mô hình dự đoán càng tốt.
- **Macro-average AUC**: Diện tích dưới đường cong ROC (AUC) trung bình của mô hình, được tính bằng cách lấy trung bình cộng của AUC của từng lớp. Macro-average AUC càng cao, mô hình càng có khả năng phân biệt các lớp khác nhau.
- **Micro-average AUC**: Diện tích dưới đường cong ROC (AUC) tổng hợp của mô hình, được tính bằng cách lấy tổng số mẫu thuộc mỗi lớp nhân với AUC của lớp đó, rồi chia cho tổng số mẫu. Micro-average AUC càng cao, mô hình càng có khả năng phân biệt các mẫu khác nhau.

2.2. Các Mô Hình Phân Tích Dữ Liệu

2.2.1. Mô Hình Random Forest

Random Forest là một thuật toán học máy được tạo ra lần đầu tiên bởi Tin Kam Ho (1995)[1] và sau đó được mở rộng bởi Leo Breiman (2001)[2]. Đây là phương pháp học có giám sát dựa trên kỹ thuật ensemble learning thường được sử dụng trong bài toán phân loại và hồi quy. Ensemble learning là một kỹ thuật trong học máy trong đó nhiều mô hình được huấn luyện để

giải quyết cùng một vấn đề và kết quả cuối cùng được quyết định dựa trên kết quả của tất cả các mô hình.

Random Forest tạo một tập hợp các cây quyết định từ tập hợp con được lấy mẫu ngẫu nhiên từ tập huấn luyện. Đối với bài toán phân lớp, đầu ra của mô hình chính là kết quả phổ biến nhất của hầu hết các cây quyết định. Đây chính là phương pháp bootstrap aggregating, hay còn gọi là bagging, giúp giảm hiện tượng overfitting (mô hình học quá khớp với dữ liệu huấn luyện và hoạt động kém trên dữ liệu mới) so với sử dụng một cây quyết định duy nhất.

Random Forest bắt đầu bằng cách áp dụng bagging, tức là tạo nhiều tập dữ liệu từ dữ liệu huấn luyện ban đầu bằng cách sử dụng lấy mẫu bootstrap (chọn ngẫu nhiên các điểm dữ liệu có thay thế). Sau đó tạo cây quyết định cho từng tập dữ liệu. Triển khai mỗi cây quyết định thu có được một kết quả phân lớp tương ứng. Cuối cùng, kết quả phân lớp của đa số các cây quyết định được chọn làm đầu ra của mô hình.

Kỹ thuật này giúp cho hiệu suất mô hình tốt hơn vì nó làm giảm phương sai (variance) của mô hình mà không làm tăng độ lệch (bias).

Ưu điểm:

- Giảm overfitting: nhờ việc kết hợp đầu ra của nhiều cây quyết định ngẫu nhiên (không tương quan)
- Ít nhạy cảm với nhiễu và outliers
- Có độ chính xác cao

Khuyết điểm:

- Thời gian thực thi chậm: tuy có thể xử lý các tập dữ liệu lớn nên chúng có thể đưa ra dự đoán chính xác hơn nhưng có thể xử lý dữ liệu chậm vì chúng tính toán trên từng cây quyết định riêng lẻ.
- Sử dụng nhiều tài nguyên: vì Random Forest cần không gian dữ liệu lớn hơn nên chúng sẽ cần nhiều tài nguyên hơn để lưu trữ dữ liệu đó.
- Phức tạp: dự đoán của một cây quyết định đơn lẻ sẽ dễ diễn giải hơn khi so sánh với một tập hợp nhiều cây quyết định.

2.2.2. Mô Hình Logistic Regression

Mô hình hồi quy logistic còn được gọi là mô hình logit, là một công cụ thống kê giúp phân loại và dự đoán xác suất của một sự kiện nhất định. Ví dụ, giả sử chúng ta muốn dự đoán xem khách truy cập trang web sẽ nhấp vào nút thanh toán trong giỏ hàng của họ hay không. Chúng ta có thể sử dụng hồi quy logistic để phân tích xem xét hành vi của khách hàng truy cập trước đây,

chẳng hạn như thời gian họ dành cho trang web và số lượng mặt hàng trong giỏ hàng của họ. Vì kết quả là một xác suất nên biến phụ thuộc có giới hạn từ 0 đến 1.

Trong mô hình hồi quy logistic, áp dụng một biến đổi logit dựa trên tỷ lệ cược. Tỷ lệ cược ở đây là tỷ lệ giữa xác suất thành công và xác suất thất bại. Điều này còn thường được biết đến là log odds hoặc logarit tự nhiên của tỷ lệ cược và hàm logistic này được biểu thị bằng các công thức sau[6]:

$$logit\ (pi)\ = \frac{1}{1\ +\ e - pi}$$

$$ln\ (\frac{pi}{1 - pi}) = \beta_0 + \beta_1 \times x_1 + \ldots + \beta_k \times x_k$$

Trong đó:

- logit là biến phụ thuộc
- x là biến độc lập
- β được ước lượng thông qua phương pháp ước lượng tối đa hóa hợp lý (MLE). MLE sẽ thử nghiệm các giá trị khác nhau của β qua nhiều lần lặp để tìm ra giá trị tối ưu nhất phù hợp với log odds.

Có 3 loại mô hình hồi quy logistic, được xác định dựa trên phản ứng phân loại.

- Hồi quy logistic nhị phân (Binary logistic regression): Đây là mô hình dự đoán kết quả có hai lựa chọn, ví dụ như dự đoán email có phải là spam (1) hay không (0), hoặc một khối u là ác tính (1) hay lành tính (0).
- Hồi quy logistic đa nhóm (Multinomial logistic regression): Mô hình này được sử dụng khi có nhiều hơn hai kết quả có thể xảy ra và không có thứ tự cụ thể. Ví dụ, các hãng phim muốn dự đoán thể loại phim mà người xem có thể thích xem như hành động, tình cảm, hài hước, kinh dị, v.v. để quảng cáo phim hiệu quả hơn.
- Hồi quy logistic thứ bậc (Ordinal logistic regression): Mô hình này cũng dự đoán kết quả có nhiều hơn hai lựa chọn, nhưng các lựa chọn này có thứ tự cụ thể. Ví dụ, dự đoán xếp hạng của một sản phẩm từ 1 đến 5 sao.

Ưu điểm:

- Hồi quy logistic dễ hiểu và dễ thực hiện hơn so với các phương pháp Machine Learning khác.
- Có thể xử lý dữ liệu lớn nhanh vì nó cần ít tài nguyên máy tính máy tính.
- Giúp xử lý dữ liệu và giải thích mối quan hệ giữa một biến nhị phân phụ thuộc và một hoặc nhiều biến độc lập.

Nhược điểm:

- Hồi quy logistic yêu cầu các biến độc lập phải độc lập tuyến tính với logit của biến phụ thuộc. Nếu mối quan hệ giữa biến phụ thuộc và biến độc lập không phải là tuyến tính, mô hình hồi quy logistic có thể không mô tả chính xác sự biến đổi trong dữ liệu, dẫn đến kết quả dự đoán không chính xác.
- Mô hình hồi quy logistic có thể bị ảnh hưởng bởi điểm ngoại lai trong dữ liệu, đặc biệt là khi có sự chênh lệch lớn giữa các quan sát.
- Nếu có ít quan sát hoặc biến động thấp trong dữ liệu, mô hình có thể không hoạt động hiệu quả và dự đoán có thể không chính xác.

2.2.3. Mô hình LSTM

LSTM (Long-Short Term Memory) được giới thiệu lần đầu vào năm 1997 bởi ***S. Hochreiter*** và ***J. Schmidhuber*** và được cải tiến bởi ***A. Graves*** vào năm 2013. LSTM là một loại mạng nơ-ron có khả năng học được các mối quan hệ giữa các phần tử trong một chuỗi dữ liệu, kể cả khi chúng cách nhau rất xa[7]. LSTM có thể nhớ được những thông tin quan trọng và bỏ qua những thông tin không cần thiết trong quá trình xử lý chuỗi. LSTM có thể giải quyết được vấn đề gradient biến mất, tức là sự mất dần của các thông tin ở những bước thời gian trước đó mà RNN thường gặp phải. LSTM được sử dụng rộng rãi trong các bài toán nhận dạng giọng nói, dịch máy, phân tích cảm xúc,...

Mỗi bước thời gian của LSTM có một đầu vào là dữ liệu hiện tại và một đầu ra là dữ liệu trước đó. Đầu vào và đầu ra này được xử lý bởi một đơn vị LSTM, có chức năng như một bộ nhớ có thể lưu trữ, quên và cập nhật thông tin. Đầu ra của đơn vị LSTM được gửi đến bước thời gian tiếp theo và cũng được sử dụng để phân loại dữ liệu.

LSTM gồm ba cổng: input gate, forget gate và output gate. Mỗi cổng có một chức năng cụ thể trong việc điều khiển luồng thông tin. Input gate quyết định cách cập nhật trạng thái nội bộ dựa trên đầu vào hiện tại và trạng thái nội bộ trước đó. Forget gate xác định mức độ quên đi của trạng thái nội bộ trước đó. Cuối cùng, output gate điều chỉnh ảnh hưởng của trạng thái nội bộ lên hệ thống.

Ưu điểm:

- Như đã đề cập thì LSTM có thể giải quyết được vấn đề gradient biến mất hoặc bùng nổ, tức là sự mất dần hoặc tăng quá nhanh của các thông tin ở những bước thời gian trước đó, mà RNN thường gặp phải.
- LSTM có thể nhớ được những thông tin quan trọng và bỏ qua những thông tin không cần thiết trong quá trình xử lý chuỗi, nhờ có các cổng để điều khiển luồng thông tin trong các đơn vị LSTM.
- LSTM có thể giải quyết được các bài toán khó, như xử lý ngôn ngữ tự nhiên, nhận dạng giọng nói, dịch máy, phân tích cảm xúc,.. mà RNN không thể.

Nhược điểm:

- LSTM có độ phức tạp tính toán cao hơn so với RNN, do có nhiều cổng và trạng thái nội bộ cần được cập nhật.
- LSTM có thể bị ảnh hưởng bởi các điểm ngoại lai hoặc nhiễu trong dữ liệu, làm giảm độ chính xác của mô hình.
- LSTM có thể không phù hợp với các dữ liệu có cấu trúc phức tạp hơn so với định dạng tuần tự, ví dụ như dữ liệu có dạng cây hoặc đồ thị.

CHƯƠNG III: CÁC KẾT QUẢ THỰC NGHIỆM

3.1. Bộ Dữ Liệu

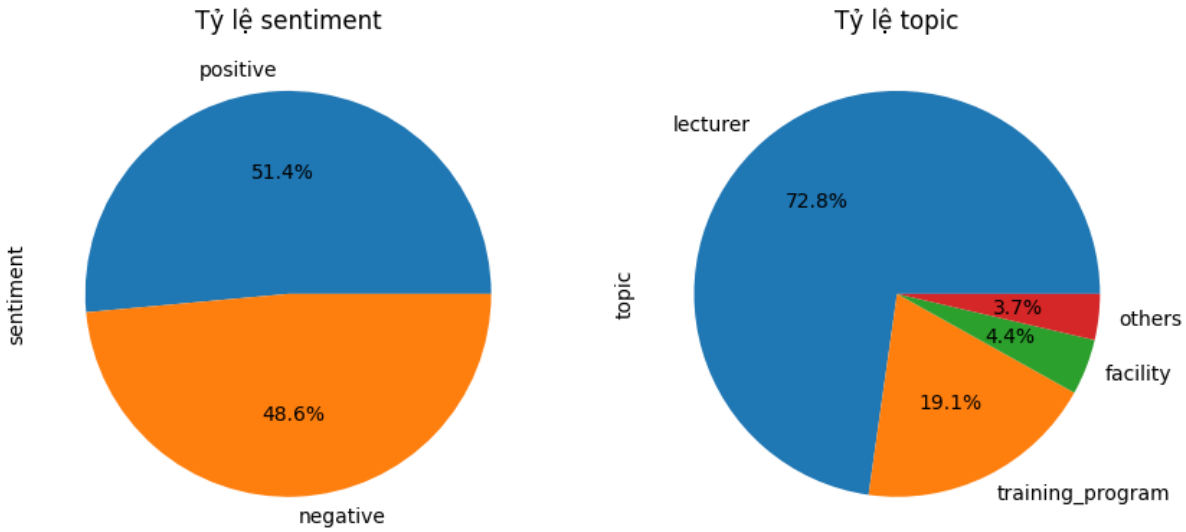
Bộ dữ liệu *Vietnamese Students' Feedback Corpus* ghi lại những phản hồi từ sinh viên Việt Nam về chất lượng giáo dục. Bộ dữ liệu này bao gồm 3 thuộc tính và tổng cộng có 10,968 dòng dữ liệu. Đáng chú ý, dữ liệu đã được phân chia thành ba tập: tập train, tập test và tập validation.

Mô tả thuộc tính:

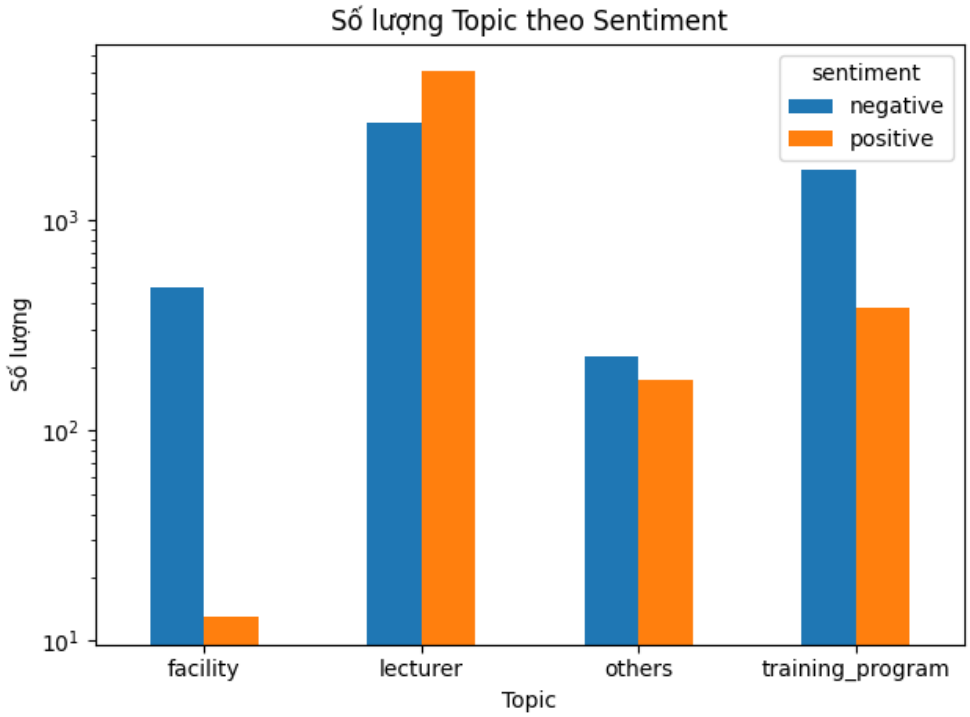
STT	Tên thuộc tính	Mô tả	Ghi chú
1	sentence	Các đánh giá của sinh viên	
2	sentiment	Cảm xúc	2 giá trị: 0 = negative (tiêu cực), 1 = positive (tích cực)
3	topic	Chủ đề	4 giá trị: 0 = lecturer, 1 = training_program, 2 = facility, 3 = others

Bộ dữ liệu chứa 51,4% phản hồi tích cực (sentiment = 1), và phần còn lại 48,6% là phản hồi tiêu cực (sentiment = 0).

Đồng thời, về chủ đề giảng viên (lecturer) chiếm ưu thế với 72,8%, nhiều hơn một nửa tổng số. Tiếp đến là chủ đề chương trình đào tạo (training_program) chiếm tỷ lệ 19,1%, cơ sở vật chất (facility) chiếm 4,4% và cuối cùng là các chủ đề khác (others) với 3,7%.



Trong số các chủ đề được đánh giá, giảng viên (lecturer) đứng đầu cả về số lượng phản hồi tích cực lẫn tiêu cực. Ngược lại, cơ sở vật chất (facility) chỉ nhận được một lượng phản hồi tích cực rất ít, thấp hơn nhiều so với lượng phản hồi tích cực. Nhìn chung, sinh viên có xu hướng đưa ra phản hồi tiêu cực nhiều hơn đối với các chủ đề như cơ sở vật chất (facility), chương trình đào tạo (training_program) và các chủ đề khác (others), trong khi giảng viên (lecturer) là chủ đề duy nhất nhận được nhiều phản hồi tích cực hơn.



3.2. Tiền xử lý dữ liệu

3.2.1. Xử lý dữ liệu bị thiếu

Đầu tiên tiến hành kiểm tra dữ liệu bị thiếu.

```
train.isna().sum()
```

Output:

```
sentence     0
sentiment    0
topic        0
dtype: int64
```

Nhận xét: bộ dữ liệu không có dòng nào bị thiếu.

3.2.2. Chuẩn hóa văn bản

Xây dựng hàm normalize_text(s) để chuẩn hóa dữ liệu văn bản. Hàm có chức năng lần lượt như sau:

- Chuyển tất cả các ký tự trong chuỗi s thành chữ thường bằng cách sử dụng phương thức lower(),
- Loại bỏ tất cả các dấu câu trong chuỗi s. Điều này được thực hiện bằng cách tạo một chuỗi mới chỉ chứa các ký tự không phải dấu câu. Trong đó string.punctuation là hàm hỗ trợ bởi python trả về một chuỗi chứa tất cả các dấu câu.
- Thay tên mã hóa (các từ có dạng wzjwz theo sau là số) bằng khoảng trắng bằng cách sử dụng biểu regular expression [r'\b((\w+|)wzjwz\d+)\b].
- Loại bỏ tất cả các số trong chuỗi s bằng cách sử dụng regular expression [r'\d']
- Thay thế các khoảng trắng liên tiếp bằng một khoảng trắng duy nhất bằng cách sử dụng regular expression [r'\s+']
- Xóa khoảng trắng đầu và cuối bằng phương thức strip().

Trong quá trình khảo sát dữ liệu, nhóm nhận thấy tên của giảng viên được mã hóa dưới dạng wzjwz, nên nhóm chọn xóa chúng đi vì đây là dữ liệu không có giá trị đối với bài toán phân lớp.

```
punctuations=list(string.punctuation)
def normalize_text(s):
    # Uncased
    s = s.lower()

    # Remove punctuations
    s = ''.join(ch for ch in s if ch not in string.punctuation)
```

```
    # Remove entities
    s = re.sub(r'\b((\w+|)wzjwz\d+)\b', " ", s)

    # Remove numbers
    s = re.sub(r'\d', ' ', s)

     # Fix whitespaces
    s = re.sub(r'\s+', ' ', s)

    #Remove leading and trailing spaces
    s = s.strip()

    return s
```

3.2.3. Tách từ (Tokenize)

Tiếp theo nhóm xây dựng hàm tokenizer (text) để tách các từ có trong văn bản thành các token. Đầu tiên hàm sẽ khởi tạo một danh sách rỗng tokens để lưu trữ các từ sau khi đã được tách. Sau đó áp dụng hàm sent_tokenize để tách một đoạn văn bản thành các câu riêng biệt. Chạy vòng lặp duyệt qua từng câu để tách mỗi câu thành các từ riêng biệt bằng cách gọi hàm word_tokenize. Cuối cùng là thêm các từ vừa tách được vào danh sách tokens. Cả 2 hàm sent_tokenize và word_tokenize đều được hỗ trợ bởi thư viện underthesea, một thư viện xử lý Tiếng Việt phổ biến.

```
def tokenizer(text):
    tokens = []
    for sent in sent_tokenize(text):
        words = word_tokenize(sent)
        tokens.extend(words)
    return tokens
```

3.2.4. Text Mining

Sau khi xây dựng hàm normalize_text(s) và tokenizer(text), nhóm tiến hành gọi hàm để khai thác văn bản (text mining).
Đầu tiên chuyển cột ‘sentence’ trong DataFrame ‘train’ (tập huấn luyện) thành một danh sách và lưu vào biến train_x.
Sau đó sử dụng hàm normalize_text để chuẩn hóa mỗi câu trong train_x. Kết quả là một danh sách các câu đã được chuẩn hóa và lưu lại vào biến train_x.

Tiếp theo sử dụng hàm tokenizer để tách mỗi câu trong train_x thành các token. Kết quả là `all_tokens_train`, một danh sách các danh sách token, với mỗi danh sách con tương ứng với một câu.
Thực hiện tương tự với các tập điều chỉnh huấn luyện (valid) và tập kiểm thử (test).
Để đếm tần suất xuất hiện các từ, nhóm thực hiện như sau:
Sử dụng hàm chain.from_iterable từ module itertools để nối tất cả các danh sách con trong all_tokens_train thành một danh sách duy nhất (`all_tokens`). Sau đó sử dụng lớp Counter từ module collections để đếm số lần xuất hiện của mỗi token trong all_tokens.
Các công đoạn đếm từ này giúp chúng ta dễ truy vấn các từ xuất hiện nhiều hoặc ít nhất trong văn bản nếu cần.

```
train_x=train['sentence'].tolist()
train_x=[normalize_text(sentence)for sentence in train_x]
all_tokens_train = [tokenizer(sentence) for sentence in train_x]
#Đếm từ
all_tokens=list(chain.from_iterable(all_tokens_train))
counter=Counter(all_tokens)
```

3.2.5. WordCloud

Để xem trực quan các token xuất hiện nhiều nhất trong bộ dữ liệu, nhóm dùng biểu đồ WordCloud, trong đó kích thước của mỗi từ thể hiện tần suất xuất hiện của nó trong dữ liệu văn bản. Các từ xuất hiện nhiều lần sẽ lớn hơn so với các từ xuất hiện ít hơn.

Có thể thấy, các từ nổi bật bao gồm “sinh viên”, “nhiệt tình”, “thực hành”, “giảng viên”, “bài tập”, và “bài giảng”, “dễ hiểu”, “tận tâm”. Điều này có thể cho thấy rằng đánh giá trong bộ dữ liệu tập trung xoay quanh về sinh viên việc học thực hành, và các bài giảng của giảng viên đồng thời có khả năng các đánh giá này có xu hướng tích cực.

3.2.6. Word2Vec

Word2Vec là một mô hình học máy không giám sát được sử dụng để tạo ra các vector đặc trưng cho từ ngữ dựa trên ngữ cảnh của chúng trong đoạn văn.

Trước khi xây dựng mô hình LSTM, việc sử dụng Word2Vec giúp giảm kích thước dữ liệu đầu vào. Thay vì sử dụng các vector one-hot có kích thước lớn, chúng ta có thể sử dụng các vector chiều thấp hơn được học bằng Word2Vec. Tiếp theo mô hình này giúp trích xuất đặc trưng trước, giảm lượng huấn luyện mà LSTM cần để đạt được hiệu quả tương tự. Cuối cùng, các vector đặc trưng được tạo ra bởi Word2Vec không phụ thuộc vào dữ liệu và ngữ cảnh, chúng giúp tăng cường khả năng tổng quát hóa của mô hình.

Ở bài báo cáo này, nhóm sử dụng Word2Vec để huấn luyện cho ba tập đó là tập train, valid và test đã tách từ rồi.

Đầu tiên ta cần xác định các tham số nhất định cho mô hình, bao gồm:

- min_count: số lượng tối thiểu của một từ cần có trong tập dữ liệu để được xem xét
- window: số lượng từ xung quanh một từ mục tiêu để xem xét trong quá trình huấn luyện
- vector_size: kích thước của vector đặc trưng
- alpha: tốc độ học ban đầu
- min_alpha: tốc độ học tối thiểu
- negative: số lượng từ nhiễu được sử dụng trong mô hình skip-gram
- sg: chọn một trong hai mô hình: mô hình CBOW (0) và skip-gram (1)

Tiếp theo ta xây dựng từ điển cho tập dữ liệu train và huấn luyện mô hình trên tập dữ liệu này. Quá trình huấn luyện này được lặp lại 100 lần. Tiếp đó là huấn luyện trên tập valid và test, các kết quả được cập nhật tiếp nối vào mô hình trước đó đã huấn luyện. Cuối cùng là lưu mô hình và sử dụng.

3.2. Huấn Luyện Dữ Liệu với các mô hình

Đầu tiên, xây dựng hàm get_sentence_vector để tạo vector cho mỗi câu. Hàm này sẽ duyệt qua từng từ trong câu, kiểm tra xem từ đó có trong từ điển của mô hình Word2Vec hay không. Nếu có, vector của từ đó sẽ được cộng vào tổng vector của câu. Cuối cùng, vector tổng sẽ được chia cho số từ trong câu để tạo ra vector trung bình.

3.2.1. Random Forest

Lần lượt chuyển đổi cột ‘topic’ của DataFrame train, test thành một danh sách và gán cho train_y, test_y. Train_y, test_y chứa nhãn của dữ liệu huấn luyện, tập kiểm thử. Sau đó chuyển đổi mỗi câu trong word_sents_train thành một vector sử dụng hàm sentence_to_vec và mô hình w2v_model được xây dựng ở phần trên, tiếp tục chuyển đổi danh sách các vector này thành một mảng numpy và gán cho X_train_w2v. Tiến hành tương tự cho các câu của tập test và gán cho X_test_w2v.

Dùng thư viện sklearn.ensemble một thư viện cung cấp các thuật toán “Ensemble” trong học máy, bao gồm Random Forest để xây dựng mô hình. Nhóm khởi tạo một mô hình Random Forest với random_state được đặt là 42 để đảm bảo rằng kết quả có thể tái tạo (kết quả của các lần chạy là như nhau). Sau khi xây dưng mô hình, tiến hành lưu mô hình đã huấn luyện vào tệp ‘model_random_forest_topic.joblib’. Để sử dụng mô hình đã huấn luyện để dự đoán nhãn cho dữ liệu kiểm tra X_test_w2v và gán kết quả cho predicted_y, dùng câu lệnh predicted_y = classifier_topic.predict(X_test_w2v).

Tiến hành tương tự với sentiment, có được mô hình phân lớp cho sentiment đặt tên là 'model_random_forest_sentiment.joblib'.

3.2.2. Logistic Regression

Đầu tiên, nhóm xây dựng hàm get_sentence_vector để tạo vector cho mỗi câu. Hàm này sẽ duyệt qua từng từ trong câu, kiểm tra xem từ đó có trong từ điển của mô hình Word2Vec hay không. Nếu có, vector của từ đó sẽ được cộng vào tổng vector của câu. Cuối cùng, vector tổng sẽ được chia cho số từ trong câu để tạo ra vector trung bình.

Sau đó, nhóm sử dụng hàm get_sentence_vector để chuyển đổi các câu trong tập train và tập test thành vector.

```
X_train_w2v = [get_sentence_vector(sentence, w2v_model) for sentence
in all_tokens_train]

X_test_w2v = [get_sentence_vector(sentence, w2v_model) for sentence in
all_tokens_test]
```

Tiếp theo, nhóm tiến hành chia dữ liệu thành features (X_train_w2v, X_test_w2v) và labels (y_train_topic, y_test_topic) đối với label topic. Đối với label sentiment thì nhóm chia thành features (X_train_w2v, X_test_w2v) và labels (y_train_sentiment, y_test_sentiment).

Sau đó, tiến hành huấn luyện mô hình Logistic Regression với các tham số bao gồm:

- multi_class = 'multinomial': cho phép mô hình xử lý bài toán đa lớp, tức là có nhiều hơn hai lớp cần phân loại. Điều này có nghĩa là mô hình sẽ cố gắng dự đoán xác suất của mỗi lớp thay vì chỉ dự đoán lớp có xác suất cao nhất.
- solver = 'lbfgs': 'lbfgs' là thuật toán tối ưu hóa được sử dụng để tìm ra các trọng số tối ưu cho mô hình Logistic Regression.
- max_iter = 100: cài đặt số lần lặp tối đa mà thuật toán huấn luyện sẽ chạy. Nó xác định số lần mà thuật toán tối ưu hóa điều chỉnh tham số mô hình dựa trên dữ liệu huấn luyện để cải thiện hiệu suất. Trong trường hợp này, nếu thuật toán đã đạt đến trạng thái ổn định sau 100 lần lặp, quá trình huấn luyện sẽ dừng lại.
- random_state = 42: để đảm bảo rằng mỗi lần chạy mới sẽ ra kết quả như nhau.

Cuối cùng, sử dụng mô hình đã được huấn luyện để dự đoán chủ đề và cảm xúc cho các câu trong tập test.

3.2.3. LSTM

Sau hai mô hình trên, nhóm chọn mô hình học sâu (Deep Learning) thông dụng là LSTM (Long Short-Term Memory) để huấn luyện mô hình với bộ dữ liệu này.

Đầu tiên ta thực hiện huấn luyện trước với cột topic, ta xác định kích thước của nhóm vector sử dụng đầu vào từ Word2Vec. Tiếp theo ta xây dựng mô hình LSTM với nhiều lớp, cụ thể ở đây là 5. Bởi vì các lớp LSTM đa tầng có khả năng học được ở mức độ phức tạp cao hơn so với các mô hình đơn tầng. Thêm vào đó, việc sử dụng Dropout giữa các lớp mô hình có thể giúp làm giảm hiện tượng overfitting, đặc biệt là khi mô hình có kích thước lớn.

Sau đó, ta in các thông số để có cái nhìn tổng quan về kiến trúc của mô hình thông qua hàm `model_LSTM_topic.summary()`.

Kết quả:

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_8 (LSTM)               (None, 1, 100)            160400

 dropout_8 (Dropout)         (None, 1, 100)            0

 lstm_9 (LSTM)               (None, 1, 50)             30200

 dropout_9 (Dropout)         (None, 1, 50)             0

 lstm_10 (LSTM)              (None, 1, 50)             20200

 dropout_10 (Dropout)        (None, 1, 50)             0

 lstm_11 (LSTM)              (None, 50)                20200

 dropout_11 (Dropout)        (None, 50)                0

 dense_2 (Dense)             (None, 4)                 204

=================================================================
Total params: 231204 (903.14 KB)
Trainable params: 231204 (903.14 KB)
Non-trainable params: 0 (0.00 Byte)
```

Nhận xét:

- Layer (type): là tên của từng lớp trong mô hình.
- Output Shape: mô tả hình dạng của đầu ra mỗi lớp. Ở đây Layer lstm_8 có Output Shape là (None, 1, 100), có nghĩa đây là một tensor (cấu trúc dữ liệu đa chiều) 3D, "None" đại diện cho số lượng mẫu trong mỗi batch, mỗi mẫu là một chuỗi với "1" vector, và vector này có kích thước là 100. Tương tự, các Layer còn lại cũng có ý nghĩa gần giống như vậy.
- Param # (Number of Parameters): là số lượng tham số trong mỗi lớp, bao gồm cả trọng số và bias. Param được dùng để xây dựng mô hình. Tổng số params có ở mô hình này là 231204. Trainable params là số lượng tham số có thể được đào tạo trong mô hình. Trong trường hợp này, tất cả các tham số đều có thể được đào tạo.

Tiếp theo, để huấn luyện mô hình đầu tiên ta sẽ xác định các tập như X_train, y_train, X_valid. Đặc biệt ở đây dùng hàm 'reshape' để thay đổi dạng dữ liệu cho X_train và X_valid cho phù hợp với đầu vào của mô hình LSTM.

Xác định các thông số cho mô hình là epochs = 100, batch_size = 32, tập valid[‘topic’] dùng hàm to_categorical để one-hot encoding. Cải tiến thêm một chút, ta dùng hàm EarlyStopping để theo dõi sự cải thiện của mô hình qua các epoch và dừng lại nếu không cải thiện đủ lớn.

Kết quả:

```
Epoch 18/20
343/343 [==============================] - 4s 13ms/step - loss: 0.4158 - accuracy: 0.8477 - val_loss: 0.3970 - val_accuracy: 0.8576
Epoch 19/20
343/343 [==============================] - 5s 15ms/step - loss: 0.4156 - accuracy: 0.8501 - val_loss: 0.3983 - val_accuracy: 0.8530
Epoch 20/20
343/343 [==============================] - 5s 16ms/step - loss: 0.4121 - accuracy: 0.8506 - val_loss: 0.4171 - val_accuracy: 0.8510
<keras.src.callbacks.History at 0x7b150ead2290>
```

Nhận xét: sau khi chạy EarlyStopping nhiều lần và kết quả cho thấy mô hình có thể chạy tốt nhất từ 15 đến 20 epochs, nên ở đây nhóm chọn cố định epochs = 20 cho kết quả ổn định.

Cuối cùng ta thực hiện lưu mô hình vừa huấn luyện để có thể dùng cho sau này.

Với tập sentiment ta cũng thực hiện tương tự như trên, nhưng thông số Dense ở lớp cuối giảm còn 2.

3.3. Các Kết Quả Thực Nghiệm

3.3.1. Random Forest

a) Phân lớp Topic

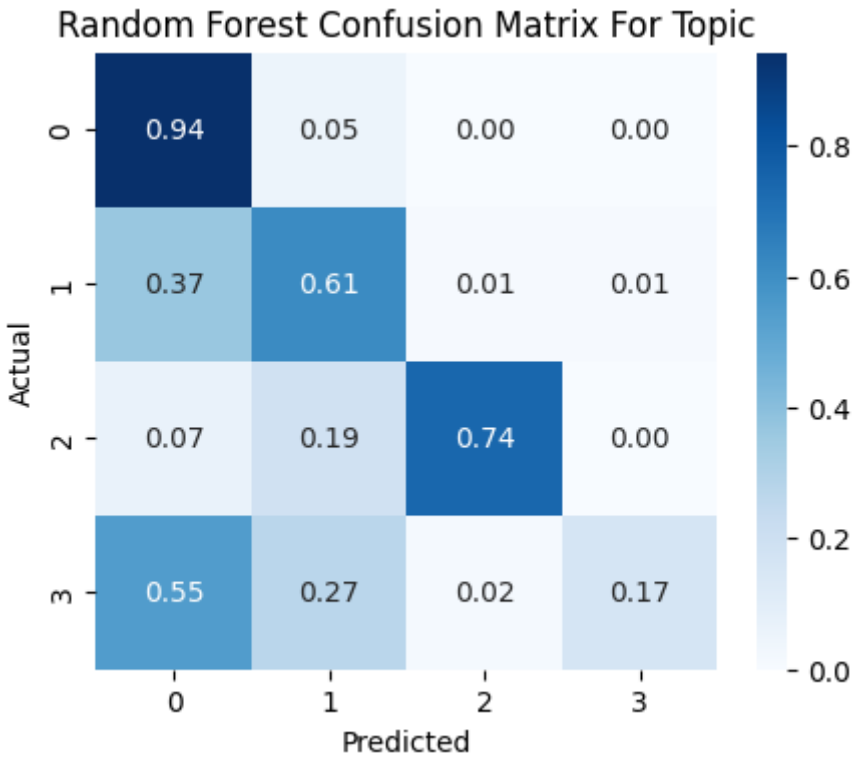
Sau khi huấn luyện mô hình trên tập huấn luyện, và chạy mô hình trên tập kiểm thử, nhóm thu được các chỉ số đánh giá như sau:

- Accuracy: 0.8449483161053685
- Recall: 0.8449483161053685
- Precision: 0.8344108974046336
- F1 Score: 0.834199309809798

Nhận xét: hiệu suất phân lớp của mô hình Random Forest có tương đối tốt. Trong đó:

- Độ chính xác (Accuracy) đạt 0.845, cho thấy mô hình dự đoán chính xác gần 85% các mẫu trong tập kiểm tra.
- Độ phủ (Recall) bằng 0.845, cho thấy mô hình đã phát hiện chính xác khoảng 85% các mẫu thuộc mỗi lớp trong tập kiểm tra.
- Độ chính xác (Precision) xấp xỉ 0.834, cho thấy khoảng 83% các mẫu được dự đoán thuộc một lớp thực sự thuộc lớp đó.
- Điểm F1 (F1 Score) là 0.834, là trung bình điều hòa của Precision và Recall, cho thấy giữa Precision và Recall của mô hình tương đối cân bằng.

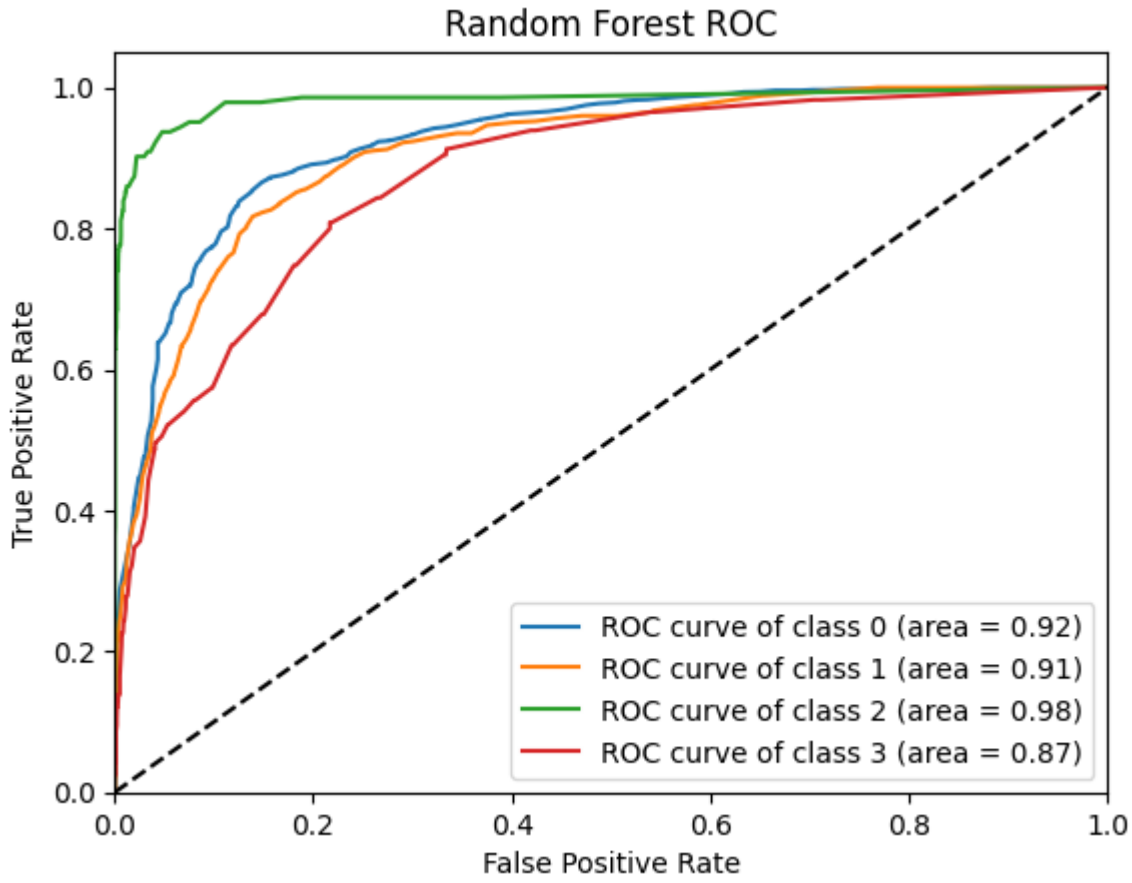
Để có cái nhìn chi tiết hơn về kết quả dự đoán phân lớp của mô hình, nhóm xem xét ma trận nhầm lẫn (Confusion Matrix) của mô hình.



Các tỷ số trên đường chéo chính của ma trận cho thấy kết quả dự đoán phân lớp chính xác của mô hình, có thể thấy mô hình dự đoán tương đối tốt đối với các lớp 0, 1, 2 vì giá trị của chúng đều khá lớn. Tuy nhiên ở lớp 3, mô hình phân lớp kém hiệu quả hơn khi nó chỉ dự đoán chính xác khoảng 17% các mẫu thuộc phân lớp này.

Bên cạnh đó mô hình cũng phân lớp sai các lớp khác thành lớp 0 khá nhiều, đặt biệt là 55% các mẫu ở lớp 3 bị phân thành lớp 1. Nguyên nhân cho việc này có thể là bởi vì bộ dữ liệu không cân bằng (imbalanced) với hơn 70% các đánh giá thuộc về lớp 0 và lớp 3 chỉ chiếm hơn 3% tổng thể. Đặc trưng này khiến cho mô hình có thể thiên vị (bias) về chủ đề 0 và dẫn đến hiệu suất kém khi phân loại các chủ đề còn lại. Thực tế, mô hình dự đoán rất chính xác chủ đề 0 với gần 95% các mẫu được phân lớp chính xác vì có nhiều dữ liệu hơn để học. Ngược lại, mô hình gặp khó khăn khi dự đoán các mẫu thuộc chủ đề 3 vì có ít dữ liệu được cung cấp, dẫn đến kết quả gần 83% các đánh giá thuộc chủ đề này bị phân lớp sai.

Tiếp theo nhóm xem qua biểu đồ ROC (Receiver Operating Characteristic) của mô hình



Macro-average AUC: 0.922203881786887

Micro-average AUC: 0.9703318171004576

Nhận xét: nhìn chung mô hình có hiệu suất tương đối tốt đối với tất cả các chủ đề. Trong đó:

- Chủ đề 0: Đường cong ROC cho chủ đề 0 có diện tích dưới đường cong (AUC) là 0.92, cho thấy mô hình hoạt động tốt với chủ đề này. Đây cũng là chủ đề được phân lớp tốt nhất.
- Chủ đề 1: Đường cong ROC cho chủ đề 1 có AUC (diện tích dưới đường cong) là 0.91, cho thấy mô hình cũng hoạt động tốt với chủ đề này.
- Chủ đề 2: Đường cong ROC cho chủ đề 2 có AUC là 0.98, cho thấy mô hình hoạt động xuất sắc với chủ đề này.
- Chủ đề 3: Đường cong ROC cho chủ đề 3 có AUC là 0.87, cho thấy mô hình hoạt động khá tốt với chủ đề này, nhưng có thể cần cải thiện. Đây là chủ đề có hiệu suất phân lớp thấp nhất của mô hình.

Bên cạnh đó, vì đây là bài phân lớp đa lớp nên biểu đồ ROC chỉ thể hiện hiệu suất phân lớp trên từng lớp, không đại diện cho tổng thể, nhóm sẽ xem xét thêm 2 chỉ số AUC trung bình (macro-average AUC) và AUC tổng (micro-average AUC). Có thể thấy từ kết quả trên, cả 2 chỉ số này của mô hình đều tương đối cao, trong đó:

- AUC trung bình đạt 0.922 cho thấy mô hình hoạt động tốt với từng lớp một cách đồng đều.
- AUC tổng đạt 0.970 cho thấy mô hình hoạt động rất tốt khi xem xét tổng thể tất cả các lớp.

b) Phân lớp Sentiment

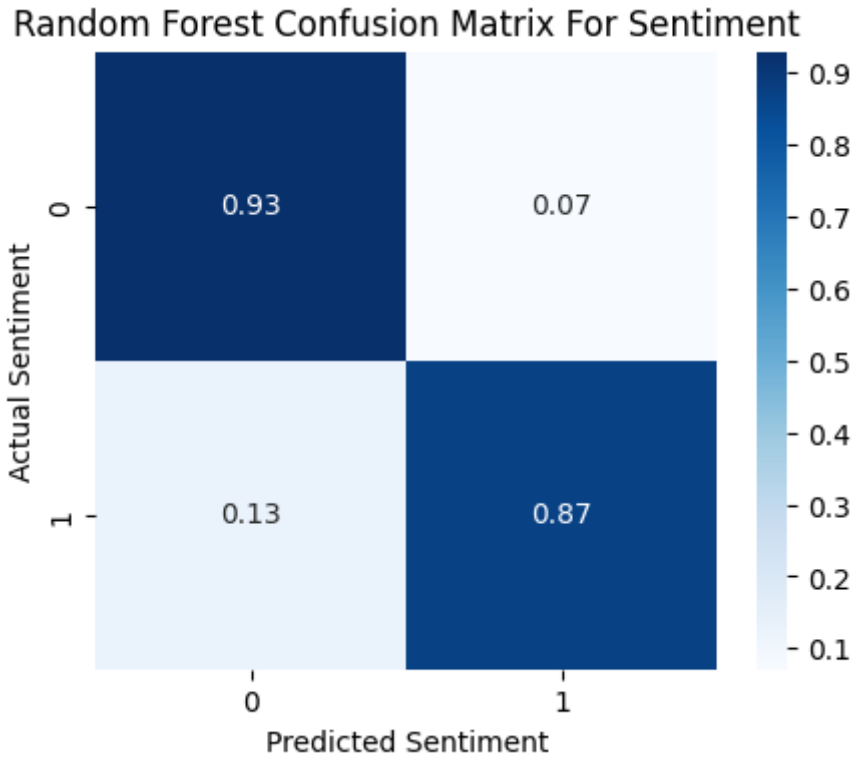
Sau khi huấn luyện mô hình trên tập huấn luyện với cột ‘sentiment’ và chạy trên tập test, nhóm cũng thu được các chỉ số đánh giá như sau:

- Accuracy: 0.8979659886628877
- Recall: 0.8979659886628877
- Precision: 0.9003902448073606
- F1 Score: 0.8980562352647355

Nhận xét: Nhìn chung, các chỉ số đánh giá phân lớp sentiment đều cao hơn so với phân lớp topic. Điều này cho thấy mô hình đang có hiệu suất phân lớp sentiment tốt hơn. Cụ thể:

- Accuracy = 0.898, cho thấy mô hình dự đoán chính xác lên đến 89% các mẫu trong tập kiểm tra.
- Recall = 0.898, cho thấy mô hình đã phát hiện chính xác hơn 89% các mẫu thuộc mỗi lớp trong tập kiểm tra.
- Precision = 0.90, cho thấy có đến 90% các mẫu được dự đoán thuộc một lớp thực sự thuộc lớp đó.
- F1 = 0.898, cho Precision và Recall của mô hình tương đối cân bằng.

Các chỉ số trên cho thấy hiệu suất phân lớp trên sentiment của mô hình tương đối ấn tượng. Để đánh giá chi tiết hơn, nhóm sẽ xem xét Confusion Matrix của mô hình khi phân lớp sentiment:



Từ ma trận trên, có thể thấy mô hình phân lớp rất tốt. Các cả 2 lớp 0 và 1 của sentiment đều được phân lớp với tỷ lệ chính xác cao. Trong đó, có 93% các mẫu thuộc lớp 0 được dự đoán phân lớp đúng, tỷ lệ nhầm lẫn của lớp này chỉ có khoảng 7%. Bên cạnh đó mô hình phân lớp chính xác 87% các đánh giá thuộc lớp 1, đồng thời có khoảng 13% các mẫu thuộc lớp 1 bị phân lớp nhầm thành lớp 0.

So với topic, mô hình phân lớp sentiment tốt hơn và có độ chính xác đồng đều trên tất cả cá lớp. Nguyên nhân có thể là do dữ liệu của sentiment cân bằng (balanced) hơn so với topic, tỷ lệ lớp 0 và l của sentiment tương đối bằng nhau chứ không chênh lệch rõ rệt như topic.

3.3.2. Logistic Regression

Để đánh giá mô hình, nhóm sẽ thực hiện tính toán các chỉ số đánh giá bao gồm: accuracy, precision, recall và F1 score.

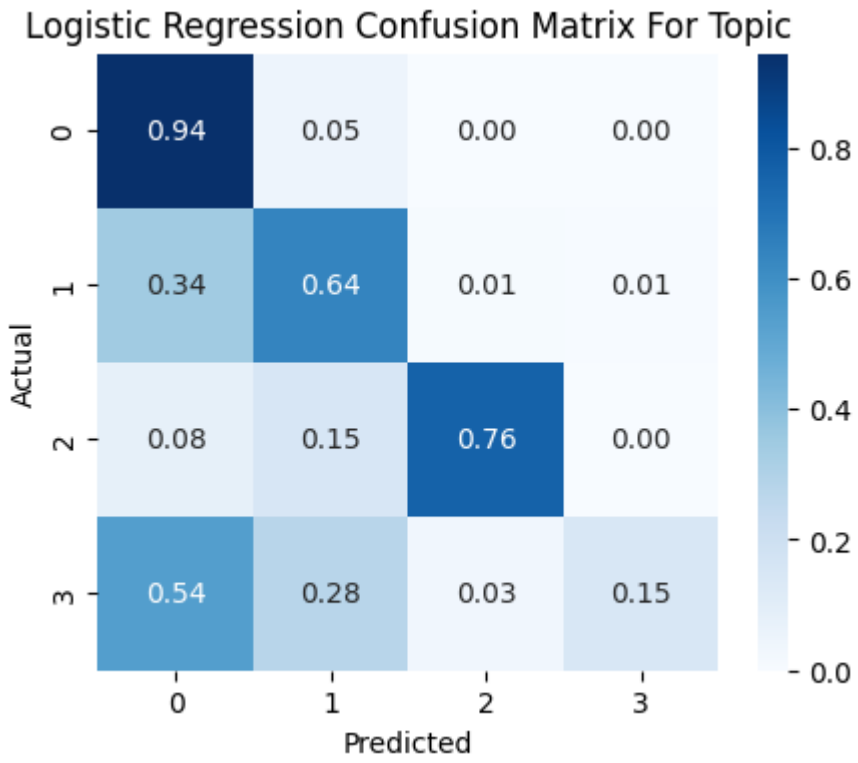
a) Phân lớp Topic:

- Accuracy: 0.8512837612537513
- Precision: 0.8414171335397487
- Recall: 0.8512837612537513
- F1 Score: 0.8401418294923361

Kết quả đánh giá cho thấy hiệu suất phân lớp của mô hình Logistic Regression tương đối tốt.

- Độ chính xác (Accuracy) đạt 0.851, cho thấy mô hình dự đoán chính xác khoảng 85% các mẫu trong tập kiểm tra.
- Độ chính xác (Precision) xấp xỉ 0.841, cho thấy khoảng 84% các mẫu được dự đoán thuộc một lớp thực sự thuộc lớp đó.
- Độ phủ (Recall) bằng 0.851, cho thấy mô hình đã phát hiện chính xác khoảng 85% các mẫu thuộc mỗi lớp trong tập kiểm tra.
- Điểm F1 (F1 Score) là 0.840, là trung bình điều hòa của Precision và Recall, cho thấy giữa Precision và Recall của mô hình tương đối cân bằng.

Nhóm sử dụng ma trận nhầm lẫn (Confusion Matrix) để có cái nhìn chi tiết hơn về kết quả dự đoán phân lớp của mô hình.



Mô hình phân lớp có thể dự đoán chính xác các lớp 0, 1, 2 với các tỷ số cao trên đường chéo chính của ma trận. Tuy nhiên ở lớp 3, mô hình phân lớp kém hiệu quả hơn khi nó chỉ dự đoán chính xác khoảng 15% các mẫu thuộc phân lớp này.

Ngoài ra, mô hình cũng phân lớp sai các lớp khác thành lớp 0 khá nhiều, đặt biệt là 54% các mẫu ở lớp 3 bị phân thành lớp 0. Điều này có thể là bởi vì bộ dữ liệu bị mất cân bằng (imbalanced) với hơn 70% các đánh giá thuộc về lớp 0 và lớp 3 chỉ chiếm hơn 3% tổng thể. Đặc trưng này khiến cho mô hình có thể thiên vị (bias) về chủ đề 0 và dẫn đến hiệu suất kém khi phân loại các chủ đề còn lại. Thực tế, mô hình dự đoán rất chính xác chủ đề 0 với gần 95% các mẫu được phân lớp chính xác vì có nhiều dữ liệu hơn để học. Ngược lại, mô hình gặp khó khăn khi dự đoán các mẫu thuộc chủ đề 3 vì có ít dữ liệu được cung cấp, dẫn đến kết quả có 85% các đánh giá thuộc chủ đề này bị phân lớp sai.

b) Phân lớp Sentiment:

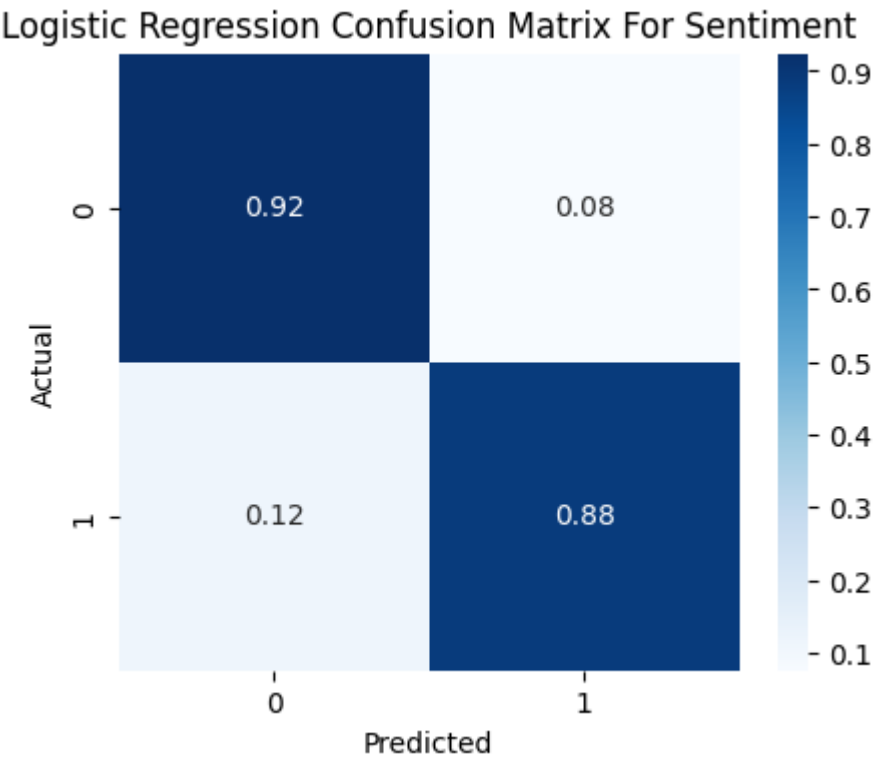
- Accuracy: 0.9033011003667889
- Precision: 0.9046330291604036
- Recall: 0.9033011003667889
- F1 Score: 0.9033870025465424

Sau khi huấn luyện mô hình trên tập huấn luyện với cột ‘sentiment’ và chạy trên tập test, nhóm cũng thu được các chỉ số đánh giá như sau:

- Độ chính xác (Accuracy) đạt 0.903, cho thấy mô hình dự đoán chính xác khoảng 90% các mẫu trong tập kiểm tra.

- Độ chính xác (Precision) xấp xỉ 0.905, cho thấy gần 91% các mẫu được dự đoán thuộc một lớp thực sự thuộc lớp đó.
- Độ phủ (Recall) bằng 0.903, cho thấy mô hình đã phát hiện chính xác khoảng 90% các mẫu thuộc mỗi lớp trong tập kiểm tra.
- Điểm F1 (F1 Score) là 0.903, là trung bình điều hòa của Precision và Recall, cho thấy giữa Precision và Recall của mô hình tương đối cân bằng.

Và nhóm cũng tiến hành vẽ ma trận nhầm lẫn (Confusion Matrix) để xem xét chi tiết hơn.



Từ ma trận trên, có thể thấy mô hình phân lớp rất tốt. Các cả 2 lớp 0 và 1 của sentiment đều được phân lớp với tỷ lệ chính xác cao. Trong đó, có 92% các mẫu thuộc lớp 0 được dự đoán phân lớp đúng, tỷ lệ nhầm lẫn của lớp này chỉ có khoảng 8%. Bên cạnh đó mô hình phân lớp chính xác 88% các đánh giá thuộc lớp 1, đồng thời có khoảng 12% các mẫu thuộc lớp 1 bị phân lớp nhầm thành lớp 0.

So với topic, mô hình phân lớp sentiment tốt hơn và có độ chính xác đồng đều trên tất cả cá lớp. Điều này có thể là do dữ liệu của sentiment cân bằng (balanced) hơn so với topic, tỷ lệ lớp 0 và l của sentiment tương đối bằng nhau chứ không chênh lệch rõ rệt như topic.

3.3.3. LSTM

Ta thực hiện đánh giá mô hình LSTM cùng các chỉ số đánh giá với 2 mô hình trên để so sánh một cách công bằng về hiệu suất của cả ba mô hình.

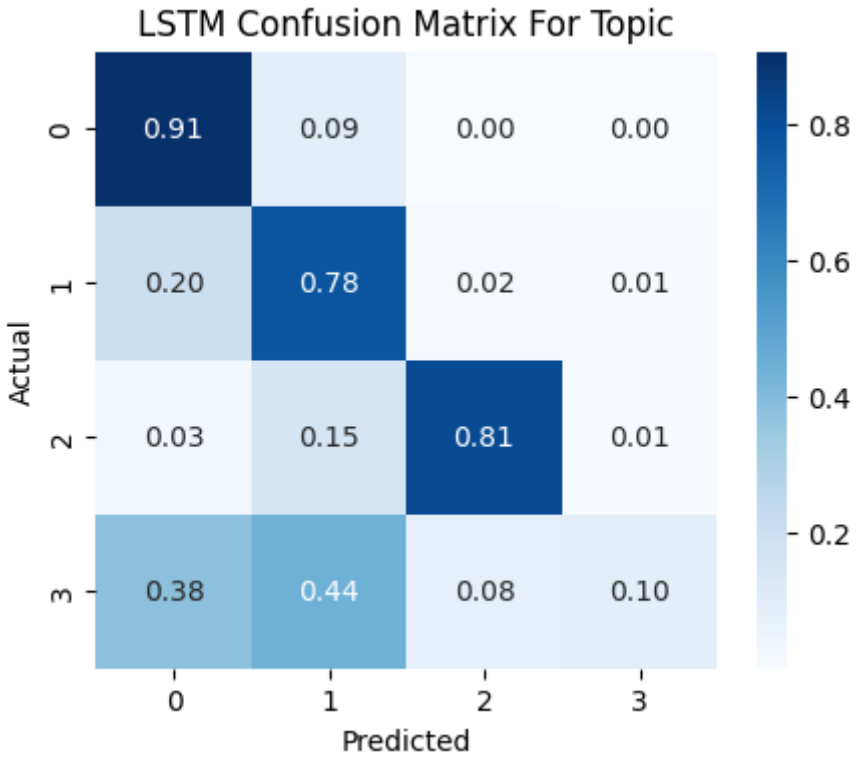
a) Phân lớp Topic:

- Accuracy: 0.8472824274758253
- Recall: 0.6473046248211057
- Precision: 0.7431469145309364
- F1 Score: 0.6452459800302058

Nhận xét: hiệu suất của mô hình LSTM đối với label topic nằm ở mức trung bình tốt. Trong đó:

- Độ chính xác (Accuracy) đạt 0.847, cho thấy mô hình dự đoán chính xác gần 85% các mẫu trong tập kiểm tra.
- Độ phủ (Recall) bằng 0.647, cho thấy mô hình đã phát hiện chính xác khoảng 65% các mẫu thuộc mỗi lớp trong tập kiểm tra.
- Độ chính xác (Precision) xấp xỉ 0.743, cho thấy khoảng 74% các mẫu được dự đoán thuộc một lớp thực sự thuộc lớp đó.
- Điểm F1 (F1 Score) là 0.645, là trung bình điều hòa của Precision và Recall, cho thấy giữa Precision và Recall của mô hình tương đối cân bằng.

Kết quả confusion matrix cho thấy:



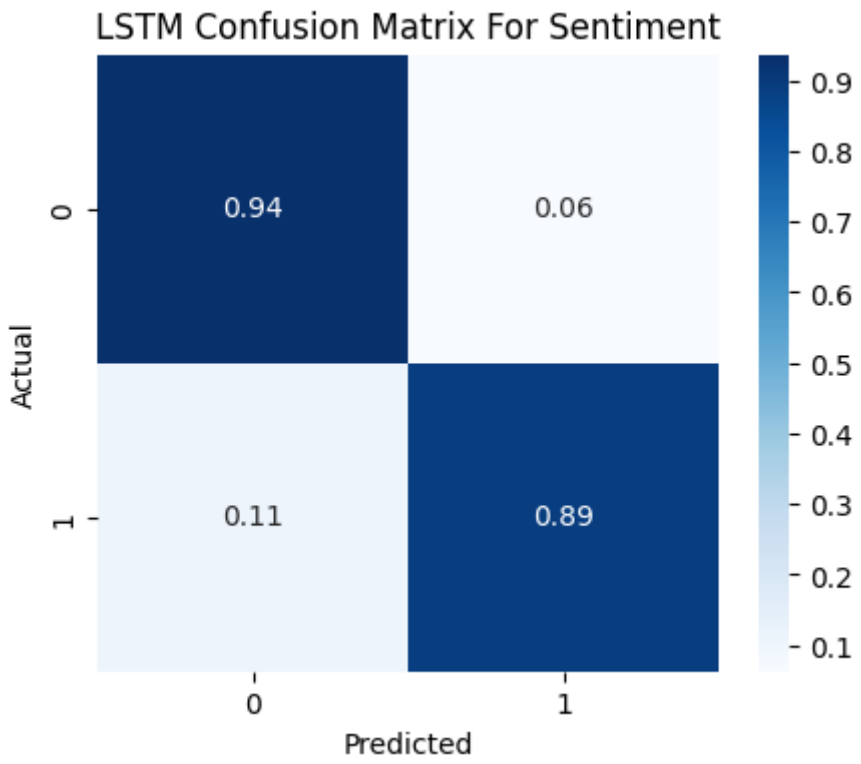
Nhận xét: Lớp “0” tức lecturer được dự đoán rất tốt với 91%, tiếp theo là lớp “2” tức facility cũng tốt với 81% và lớp “1” với 78%. Tuy nhiên lớp “3” là others được dự đoán sai lệch rất nhiều, với độ chính xác chỉ 10%, và chúng đa phần bị nhầm sang lớp “0” và “1”.

b) Phân lớp Sentiment:

- Accuracy: 0.9129709903301101
- Recall: 0.9143292669318086
- Precision: 0.91283038403447
- F1 Score: 0.9128779594223186

Nhận xét: Hiệu suất của mô hình LSTM đối với label topic nằm ở mức rất tốt. Trong đó các chỉ số đều hơn 91%.

Kết quả confusion matrix cho thấy:



Nhận xét: Cả 2 lớp đều có chỉ số dự đoán chính xác cao và các sai lầm chỉ dưới 11%.

3.4. Phân Tích và Đánh Giá

Mô hình	Thuộc tính	Accuracy (%)	Recall (%)	Precision (%)	F1 Score (%)
Random Forest	**Topic**	84.50	84.50	83.44	83.41
	Sentiment	89.80	89.80	90.04	89.81
Logistic Regression	**Topic**	85.13	85.13	84.14	84.01
	Sentiment	90.33	90.33	90.46	90.33
LSTM	**Topic**	84.47	64.73	74.31	64.52
	Sentiment	91.30	91.43	91.28	91.28

Nhận xét:

- Dự đoán 'topic': Mô hình Random Forest và Logistic Regression có số liệu khá cao, với hơn 80% cho mỗi chỉ số đánh giá. Mô hình học sâu LSTM có số liệu cao ở chỉ số Accuracy nhưng ở các chỉ số như Recall, Precision và F1 Score chỉ đạt khoảng từ 64 - 74%, thấp hơn hẳn hai mô hình còn lại. Lý do là vì dữ liệu của nhóm bị mất cân bằng (imbalance) nên Accuracy không phản ánh đúng hiệu suất và có xu hướng học cách dự đoán lớp thiểu số nhiều hơn.
- Dự đoán 'sentiment': ngược lại với 'topic', ở đây mô hình học sâu LSTM phát huy hiệu suất cao hơn cả hai mô hình còn lại. Theo thống kê thì label sentiment không hề bị imbalance nên từ đó mà chỉ số đánh giá của cả ba mô hình đều cao hơn 90%.

Nếu xét về tổng thể thì Random Forest và Logistic Regression có hiệu suất ổn định hơn hẳn LSTM. Tuy nhiên, vì đây là bài toán phân loại nhiều lớp nên mô hình Random Forest sẽ có tính ứng dụng cao hơn hẳn Logistic Regression.

Vậy nên kết luận theo tổng thể thì mô hình Random Forest tối ưu nhất, tiếp đến là Logistic Regression và cuối cùng là LSTM.

Tuy nhiên đó cũng chỉ là tổng thể, nhóm đã nhập và dự đoán thử nhiều câu văn bản. Kết quả cho thấy, nếu câu dài thì cả ba mô hình có thể đoán chính xác cao, ngược lại câu ngắn thì Logistic Regression lộ điểm yếu của nó.

Đây là ví dụ:

Topic

```
In [401…  text_to_predict = input("Nhập đoạn văn bản cần dự đoán: ")

Nhập đoạn văn bản cần dự đoán: bài hay

In [402…  predicted_topic_lr = predict_topic(text_to_predict, w2v_model, lr_model=logistic_model_topic)
          predicted_topic_lstm = predict_topic(text_to_predict, w2v_model, lstm_model=model_LSTM_topic)
          predicted_topic_rf = predict_topic(text_to_predict, w2v_model, rf_model=classifier_topic)

          print(f"Random Forest - Dự đoán Topic: {predicted_topic_rf}")
          print(f"Logistic Regression - Dự đoán Topic: {predicted_topic_lr}")
          print(f"LSTM - Dự đoán Topic: {predicted_topic_lstm}")

1/1 [==============================] - 0s 115ms/step
Random Forest - Dự đoán Topic: training_program
Logistic Regression - Dự đoán Topic: lecturer
LSTM - Dự đoán Topic: training_program
```

Sentiment

```
In [403…  predicted_sentiment_rf = predict_sentiment(text_to_predict, w2v_model, rf_model=classifier_sentiment)
          predicted_sentiment_lr = predict_sentiment(text_to_predict, w2v_model, lr_model=logistic_model_sentiment)
          predicted_sentiment_lstm = predict_sentiment(text_to_predict, w2v_model, lstm_model=model_LSTM_sentiment)

          print(f"Random Forest - Dự đoán Sentiment: {predicted_sentiment_rf}")
          print(f"Logistic Regression - Dự đoán Sentiment: {predicted_sentiment_lr}")
          print(f"LSTM - Dự đoán Sentiment: {predicted_sentiment_lstm}")

1/1 [==============================] - 0s 156ms/step
Random Forest - Dự đoán Sentiment: positive
Logistic Regression - Dự đoán Sentiment: positive
LSTM - Dự đoán Sentiment: positive
```

Nhận xét: cả Random Forest và LSTM đều dự đoán chính xác “bài hay” thuộc chủ đều training program nhưng Logistic Regression lại đoán sai.

CHƯƠNG IV: KẾT QUẢ ĐẠT ĐƯỢC

4.1. Các Kết Quả Đạt Được

Thông qua quá trình nghiên cứu và xây dựng các mô hình trên, nhóm đã xây dựng được mô hình phân lớp đánh giá của sinh viên theo cảm xúc (tiêu cực, tích cực) và theo chủ đề (giảng viên, chương trình dạy, cơ sở vật chất và những thứ khác). Kết quả là theo các chỉ số đánh giá tổng thể thì mô hình Random Forest cho kết quả dự đoán tốt nhất, tiếp theo là Logistic Regression và cuối cùng là mô hình học sâu LSTM.

4.2. Những Hạn Chế và Hướng Phát Triển

Hạn chế đầu tiên phải nói đến là bộ từ vựng còn thiếu rất nhiều, dẫn đến gặp lỗi mỗi khi nhập từ không có trong bộ từ vựng. Để giải quyết vấn đề này thì nhóm dự định sẽ nghiên cứu và tìm bộ dữ liệu lớn hơn giúp thu thập bộ từ vựng nhiều hơn.

Hạn chế kế tiếp là có những từ sai chính tả, teencode và những kí tự đặc biệt còn chưa xử lý hết do thời gian khi làm báo cáo này thời gian khá gấp rút. Hướng giải quyết vấn đề này mà nhóm nghĩ ra được là tạo một bộ từ vựng tiếng Việt lớn để train mô hình, so sánh và chỉnh sửa từ sai chính tả và teencode.

Hạn chế tiếp theo là bộ dữ liệu của nhóm bị mất cân bằng, nguyên do là kích thước của bộ dữ liệu không lớn và thiếu tính đa dạng. Để xử lý vấn đề này ta có thể sử dụng phương pháp SMOTE hoặc GANs, hai phương pháp này cải thiện hiệu suất bằng cách tăng độ đa dạng và chất lượng của các mẫu tổng hợp.

Hạn chế cuối cùng là bộ dữ liệu hiện tại chỉ có thể phân loại vài nhãn chung chung. Sau khi thu thập bộ dữ liệu lớn hơn, nhóm có thể chia các nhãn đa dạng hơn, ví dụ ở nhãn cảm xúc từ tiêu cực, tích cực có thể mở rộng thành hài lòng, bất mãn, lo lắng, giận dữ, hứng thú…

PHỤ LỤC

BẢNG PHÂN CÔNG

Họ và tên	MSSV	Lớp	Đóng góp
Văn Ngọc Như Quỳnh	31211027669	DS001	100%
Võ Ngọc Mỹ Kim	31211027646	DS001	100%
Nguyễn Nhật Thảo Vy	31211025542	DS001	100%

LINK GITHUB

Link cả bài: https://github.com/kimvo646/NLP.git

Link colab train model:
https://github.com/kimvo646/NLP/blob/main/NLP_Source_code.ipynb

TÀI LIỆU THAM KHẢO

[1]Tin Kam Ho, "Random decision forests" Proceedings of 3rd International Conference on Document Analysis and Recognition, Montreal, QC, Canada, 1995, pp. 278-282 vol.1, doi: 10.1109/ICDAR.1995.598994.

[2] Breiman, L. (n.d.). Random Forests | Machine Learning. *SpringerLink*. Truy cập vào 20/12/2023 từ https://link.springer.com/article/10.1023/A:1010933404324#Abs1

[3] Shafi, A. (2023, February). *Random Forest Classification with Scikit-Learn*. DataCamp. Truy cập vào 21/12/2023 từ https://www.datacamp.com/tutorial/random-forests-classifier-python

[4] Sharma, A. (2023, September 26). *Random Forest vs Decision Tree | Which Is Right for You?*Analytics Vidhya. Truy cập vào 21/12/2023 từ https://www.analyticsvidhya.com/blog/2020/05/decision-tree-vs-random-forest-algorithm/

[5] *What is Random Forest?* (n.d.). IBM. Truy cập vào 21/12/2023, từ https://www.ibm.com/topics/random-forest

[6] What is logistic regression? (n.d.) IBM. Truy cập vào 20/12/2023, từ https://www.ibm.com/topics/logistic-regression

[7] Farhad Mortezapour Shiri, Thinagaran Perumal, Norwati Mustapha, Raihani Mohamed (University Putra Malaysia (UPM)), "A Comprehensive Overview and Comparative Analysis on Deep Learning Models: CNN, RNN, LSTM, GRU"