

**BỘ GIÁO DỤC VÀ ĐÀO TẠO
ĐẠI HỌC KINH TẾ TP HỒ CHÍ MINH
TRƯỜNG CÔNG NGHỆ VÀ THIẾT KẾ**



ĐỒ ÁN MÔN HỌC

**ĐỀ TÀI: NHẬN BIẾT CẢM XÚC DỰA VÀO BỘ DỮ
LIỆU EMOTION DATASET FOR EMOTION
RECOGNITION TASKS**

Học phần: XỬ LÝ NGÔN NGỮ TỰ NHIÊN

Nhóm Sinh Viên:

BÙI LÊ KHANG - 31211027644

ĐẶNG CHÂU KỲ - 31211027647

PHẠM MINH PHƯỚC - 31211027663

Chuyên Ngành: KHOA HỌC DỮ LIỆU

Khóa: K47

Giảng Viên: TS. Đặng Ngọc Hoàng Thành

TP. Hồ Chí Minh, Ngày 21 tháng 12 năm 2023

MỤC LỤC

CHƯƠNG 1: TỔNG QUAN	3
1. Tổng quan về bài toán phân tích dựa vào bộ dữ liệu	3
2. Lý do chọn đề tài.....	3
CHƯƠNG 2. CƠ SỞ LÝ THUYẾT.....	4
1. Các phương pháp tiền xử lý dữ liệu	4
2. Mô hình DecisionTree Classification kết hợp Ensemble learning (Adaboost)	4
3. Mô hình MaxEnt cho bài toán Text Classification	5
4. Mô hình LSTM (Long short term memory) cho bài toán Text Classification	7
CHƯƠNG 3. CÁC KẾT QUẢ THỰC NGHIỆM.....	10
1. Bộ Dữ Liệu.....	10
a. Nguồn dữ liệu	10
b. Tiền xử lý dữ liệu.....	10
c. Vector hóa dữ liệu.....	14
2. Phân chia dữ liệu	15
3. Áp dụng bộ dữ liệu vào các mô hình	15
Mô hình Máy học	15
Mô hình Maximum Entropy	18
Mô hình Deep Learning LSTM.....	21
So sánh mô hình	27
Ứng dụng cho bài toán Text Classification	28
4. Phân tích và đánh giá	31
CHƯƠNG 4. KẾT LUẬN.....	33
1. Các Kết Quả Đạt Được	33
2. Những Hạn Chế và Hướng Phát Triển.....	33
Những hạn chế:.....	33
Hướng phát triển:.....	33
TÀI LIỆU THAM KHẢO.....	35
PHỤ LỤC.....	36

CHƯƠNG 1: TỔNG QUAN

1. Tổng quan về bài toán phân tích dựa vào bộ dữ liệu

Trong thời đại công nghệ ngày càng phát triển, các mạng xã hội ngày càng phổ biến kéo theo đó là số lượng dữ liệu tăng dần theo từng giây. Khái niệm về “Big data” cũng ngày càng được phổ biến rộng rãi. Tuy nhiên với lượng dữ liệu khổng lồ và phức tạp thì cũng đặt ra cho các doanh nghiệp bài toán để phân tích chúng. Những phương pháp chuyên biệt cũng từ đó ra đời để có thể phân tích lượng dữ liệu này. Một trong số đó là bài toán mà nhóm sẽ nghiên cứu là phân loại văn bản (Text Classification).

Text Classification là bài toán để dự đoán các thuộc tính của văn bản sau khi đã được phân loại. Một số ứng dụng thường được sử dụng trong thực tế như phân loại thư rác, phân loại tin tức, phân loại tình trạng giao hàng, ... Với đề tài này nhóm sẽ nghiên cứu để phân loại cảm xúc từ những tin nhắn bằng tiếng anh trên mạng xã hội twitter để có thể nghiên cứu về cảm xúc để quản lý và lọc nội dung phù hợp với người sử dụng mạng xã hội này.

2. Lý do chọn đề tài

Văn bản có thể đại diện cho một kho thông tin rất đa dạng, nhưng việc rút trích thông tin sâu rộng từ nó có thể gặp khó khăn và mất thời gian do tính chất tự nhiên của nó. Tuy nhiên, nhờ những tiến bộ trong lĩnh vực máy học (Machine Learning) và học sâu (Deep Learning) thì quá trình xử lý văn bản phức tạp để xây dựng các mô hình đã trở nên khả thi hơn.

Mục tiêu nhóm đặt ra cho bài toán này, là từ bộ dữ liệu của mọi người được trích từ tweet trên twitter, tìm ra nhóm cảm xúc mà từng đoạn trích trong bài tweet được tải lên và xây dựng các mô hình để dự đoán cho các đoạn trích sau này. Ứng dụng vào thực tế, bài toán này giúp các doanh nghiệp/ người bán hàng dự đoán được cảm xúc của khách hàng khi họ tương tác tới các bài đăng, quảng cáo, tin tức, sản phẩm hay các chủ đề cụ thể khác, từ đó đáp ứng nhu cầu của khách hàng và đưa ra được các chiến lược tốt hơn để tăng lại doanh thu cao hơn.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

1. Các phương pháp tiền xử lý dữ liệu

Loại bỏ Dấu câu:

- Phương pháp: `remove_punctuation(text)`
- Mục đích: Loại bỏ tất cả các dấu câu trong văn bản.

Tokenization:

- Phương pháp: `tokenization(text)`
- Mục đích: Phân mảnh văn bản thành các token (từ hoặc cụm từ).

Loại bỏ từ ngữ dừng (Stop-words Removal):

- Phương pháp: `remove_stopwords(text)`
- Mục đích: Loại bỏ các từ ngữ dừng (stop-words) từ văn bản, giúp tập trung vào các từ quan trọng hơn.

Lemmatization:

- Phương pháp: `lemmatizer(text)`
- Mục đích: Chuyển các từ về dạng gốc (lemma) để giảm sự biến thể và tăng đồng nhất trong dữ liệu.

Chạy các phương pháp tiền xử lý trên DataFrame:

- Phương pháp: Sử dụng `apply` và `lambda` để áp dụng các hàm tiền xử lý lên cột `'text'` của DataFrame (`df['text']`).

Chọn phần tử đầu tiên trong danh sách token:

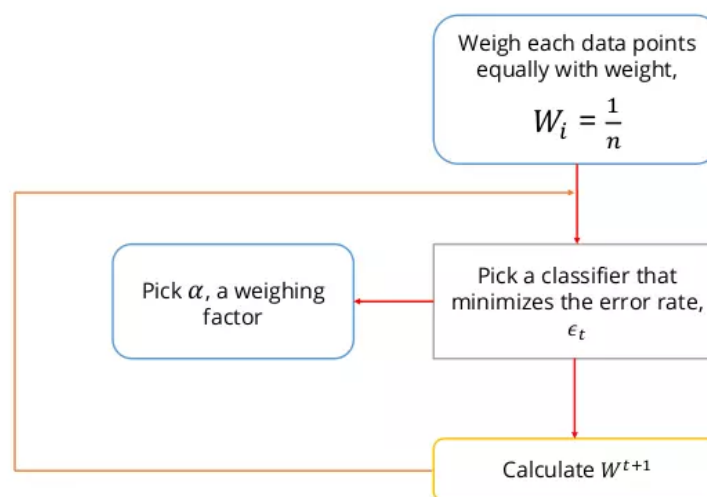
- Phương pháp: `df['text']=[item[0] for item in df['text']]`
- Mục đích: Lựa chọn phần tử đầu tiên của mỗi danh sách token. Nhằm loại bỏ cặp `[]` giữa các text trong cột `df['text']`

Tokenization sử dụng `RegexTokenizer`:

- Phương pháp: `RegexTokenizer(r'[a-zA-Z0-9]+')`
- Mục đích: Tokenization bằng cách chỉ giữ lại các ký tự chữ cái hoặc số.

2. Mô hình DecisionTree Classification kết hợp Ensemble learning (Adaboost)

- **Cây Quyết Định (Decision Tree):** Là một mô hình máy học có cấu trúc cây, mỗi nút trong cây đại diện cho một quyết định dựa trên một đặc trưng nào đó. Cây được xây dựng bằng cách chia dữ liệu thành các nhóm ngày càng nhỏ dựa trên các quyết định.
- **Ensemble learning:** là một kỹ thuật học máy kết hợp nhiều mô hình để cải thiện hiệu suất tổng thể. AdaBoost là một thuật toán Ensemble learning.
- **AdaBoost (Adaptive Boosting):** Là một thuật toán tập trung vào việc cải thiện hiệu suất của mô hình bằng cách tập trung vào các điểm dữ liệu bị lỗi. AdaBoost xây dựng nhiều cây quyết định liên tiếp, mỗi cây có trọng số dựa trên độ chính xác của cây trước đó. Các cây yếu (weak learners) sau đó được kết hợp thành một mô hình mạnh.
 - AdaBoost có thể hình dung đơn giản qua flowchart:



Có nghĩa là:

- Tập dữ liệu $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$, với $y_i \in \{-1, 1\}, i \in \{1, 2, \dots, n\}$
- Trọng số các điểm dữ liệu tại weak learner thứ t : $w_1^t, w_2^t, \dots, w_n^t, i \in \{1, 2, \dots, n\}$
- weak-learners $h : x \rightarrow \{-1, 1\}$
- Error function: $E(f(x), y, i) = e^{-y_i f(x_i)}$, trong đó $f(x_i) = \alpha h(x_i)$
- Output: $H_t(x)$, $H_t(x)$ còn được gọi là strong learner tại thời điểm t

3. Mô hình MaxEnt cho bài toán Text Classification

Mô hình Maximum Entropy (Roudi et al., 2015; Yeh et al., 2010) giả định rằng phân bố xác suất trạng thái mạng được đưa ra bởi hàm mũ của năng lượng mạng $E[S_i] =$

$E[(x_1, \dots, x_n)$ sao cho entropy được tối đa hóa đồng thời thỏa mãn mọi ràng buộc thống kê. Khi thống kê bậc một và bậc hai được đưa ra, phân bố xác suất trạng thái được đưa ra bởi:

$$P(x_1, \dots, x_n) = \frac{1}{Z} \exp \left(-E[(x_1, \dots, x_n)] \right) = \frac{1}{Z} \exp \left(\sum_i b_i x_i + \sum_{i \neq j} J_{ij} x_i x_j \right)$$


MaxEnt là viết tắt của mô hình “Maximum Entropy”, dự đoán sự xuất hiện của các loại bằng cách tìm ra sự phân bố trải rộng nhất hoặc gần đồng đều nhất, đồng thời tính được giới hạn của “biến môi trường” của các vị trí đã biết. Ví dụ biến môi trường trong bài toán này có thể là số các từ tích cực, tiêu cực trong văn bản; số từ thể hiện mức độ chắc chắn và không chắc chắn (“có lẽ”, “chắc chắn”, “rõ ràng”, ...); các từ khóa biểu thị cảm xúc, ...

MaxEnt chỉ sử dụng dữ liệu hiện có và thuật toán này so sánh vị trí của một loại được tìm thấy thỏa mãn tất cả các môi trường sẵn có trong lĩnh vực nghiên cứu. Thuật toán xác định các môi trường có sẵn bằng cách lấy mẫu một lượng lớn các điểm trên toàn bộ khu vực nghiên cứu được gọi là **background points**. Vì **background points** có thể bao gồm các vị trí mà các loại được biết đến là xảy ra, nên **background points** không giống như các điểm **pseudo-absence** (*pseudo-absence là những điểm được tạo ngẫu nhiên hoặc chọn theo cách có tổ chức trong khu vực nghiên cứu nơi mà loài được giả định không có mặt. Những điểm này đóng vai trò như một tham chiếu để mô hình học về điều kiện môi trường không phù hợp với loài đó, chúng giúp máy học hiểu được điều kiện môi trường không phù hợp với loài đó, để mô hình có thể phân biệt giữa nơi loài có mặt và nơi loài có thể vắng mặt. Nếu không có pseudo-absences, mô hình có thể hiểu lầm và kết luận sai về sự vắng mặt của loài ở những nơi chưa được quan sát*). **Background points** xác định môi trường đã có sẵn.







Thuật toán được phát triển cho mô hình phân bố loại là một phương pháp học máy, vì vậy xây dựng lặp đi lặp lại nhiều mô hình. Thuật toán có 2 thành phần chính:

- Entropy: Mô hình được hiệu chỉnh để tìm ra phân bố trải rộng nhất hoặc gần đồng đều nhất trong lĩnh vực nghiên cứu.
- Ràng buộc (Constraints): Các quy tắc hạn chế phân phối dự đoán, các quy tắc này dựa vào giá trị của biến môi trường (được gọi là features) của vị trí mà loại được quan sát.

Trong bài toán này, MaxEnt xét “categorical features” với ràng buộc là “*Tỷ lệ các lần xuất hiện được dự đoán trong mỗi loại phải gần với tỷ lệ các lần xuất hiện được quan sát trong*

mỗi loại” cho ra hình dạng 

Ta có thể tham khảo các loại features mà MaxEnt có thể xét qua:

Feature type	Interpretation	Constraint	Shape
Linear	Continuous variable	The <i>mean</i> of each environmental variable at an unknown location should be close to the mean of that variable in known occurrence locations.	
Quadratic	Square of the variable	The <i>variance</i> of each environmental variable at an unknown location should be close to the variance of that variable in known occurrence locations.	
Product	Pairs of continuous variables – allows for interactions	The <i>co-variance</i> of two environmental variables at an unknown location should be close to the co-variance of those variables in known occurrence locations.	
Threshold	Conversion into binary response based on a threshold	The proportion of predicted occurrences with values above the threshold (binary response = 1) should be close to the proportion of known occurrences.	
Hinge	As threshold type, but response after the threshold (knot) is linear	The mean above the knot of each environmental variable at an unknown location should be close to the mean above the knot of that variable in known occurrence locations.	
Categorical	Categorical variable	The proportion of predicted occurrences in each category should be close to the proportion of observed occurrences in each category.	

Ưu điểm

- Chỉ yêu cầu dữ liệu hiện có
- Có thể sử dụng cả biến dự đoán liên tục và phân loại
- Bao gồm sự tương tác giữa các biến dự đoán
- Nhìn chung hiệu suất dự đoán tốt

Hạn chế

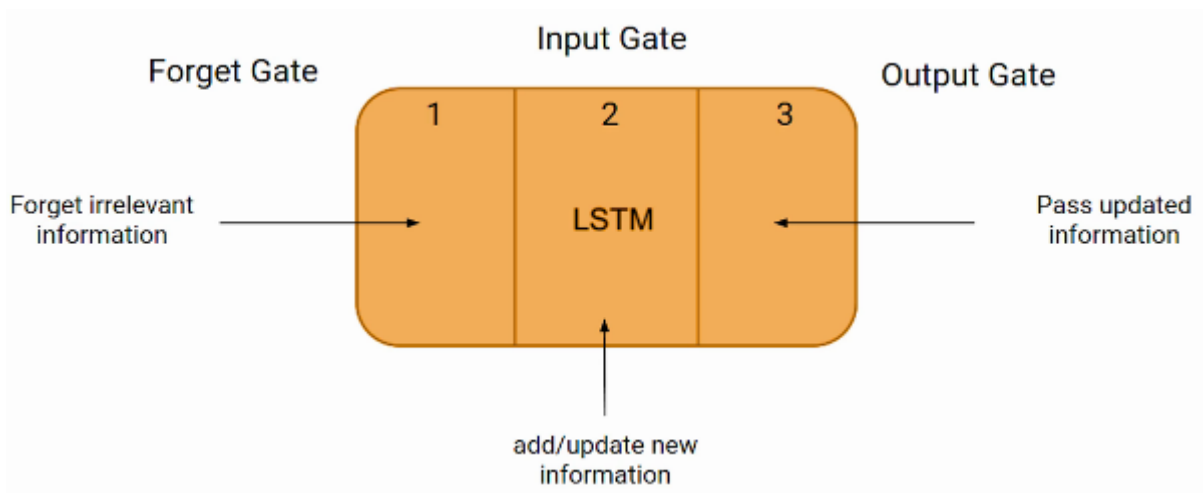
- Khó khăn trong việc so sánh output của các thuật toán khác vì output của MaxEnt mang lại sự phù hợp với môi trường hơn là xác suất xảy ra dự đoán.

4. Mô hình LSTM (Long short term memory) cho bài toán Text Classification

LSTM (**Long short term memory**) là một dạng của mô hình recurrent neural network (RNN) được sử dụng rộng rãi trong Deep Learning.

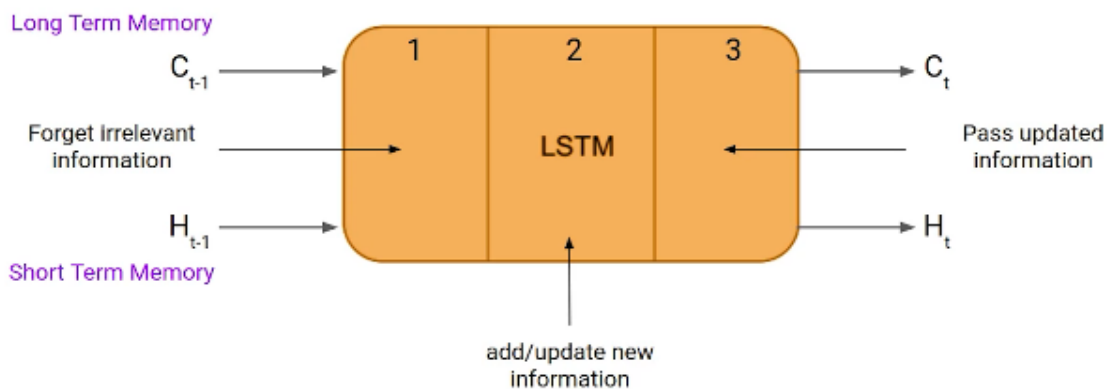
LSTM kết hợp kết nối phản hồi để xử lý toàn bộ chuỗi dữ liệu, không chỉ từng điểm dữ liệu riêng lẻ. Điều này giúp nó hiệu quả trong việc hiểu và dự đoán mẫu trong dữ liệu tuần tự như chuỗi thời gian, văn bản, và giọng nói, khác biệt so với các mô hình neural networks truyền thống.

Kiến trúc LSTM



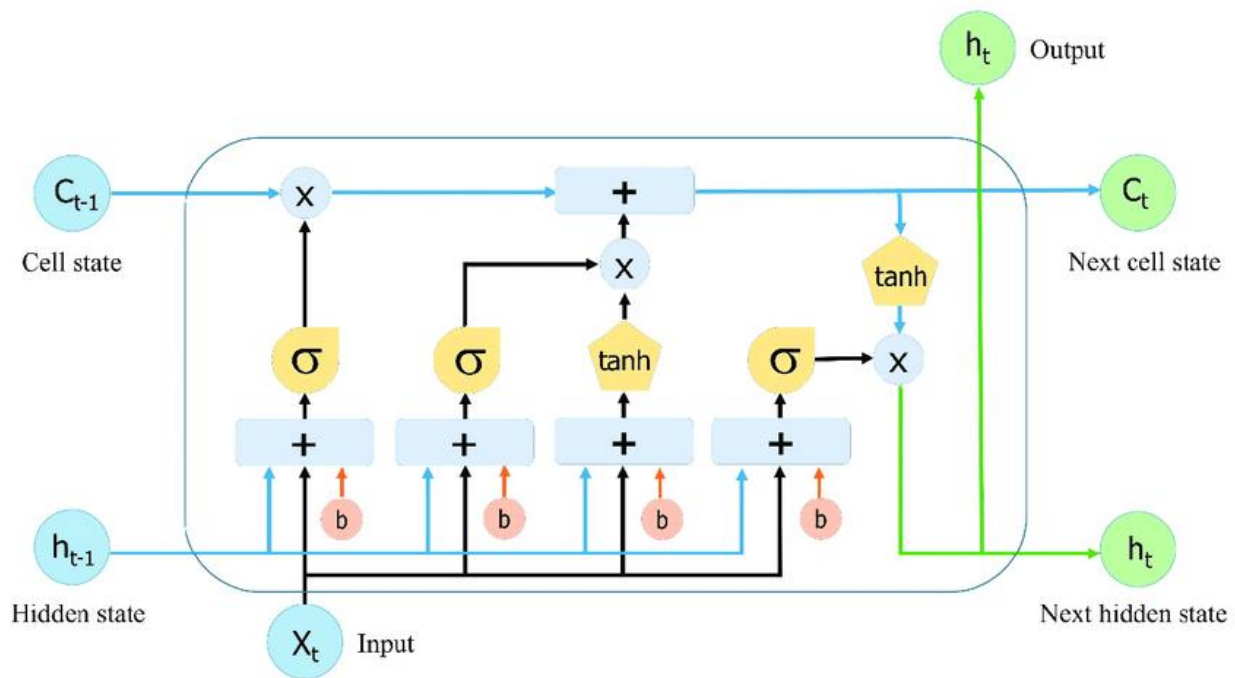
Ba phần này của đơn vị LSTM được gọi là cổng. Chúng kiểm soát luồng thông tin vào và ra khỏi ô nhớ hoặc ô lstm. Cổng đầu tiên gọi là **Forget gate**, cổng thứ hai gọi là **Input gate** và cổng cuối cùng là **Output gate**. Một đơn vị LSTM bao gồm ba cổng này và một ô nhớ hoặc ô LSTM có thể được coi là một lớp nơ-ron trong mạng nơ-ron truyền thẳng truyền thống, với mỗi nơ-ron có một lớp ẩn và trạng thái hiện tại.

Giống như một RNN đơn giản, LSTM cũng có trạng thái ẩn trong đó $H(t-1)$ biểu thị trạng thái ẩn của dấu thời gian trước đó và H_t là trạng thái ẩn của dấu thời gian hiện tại. Ngoài ra, LSTM còn có trạng thái ô được biểu thị lần lượt là $C(t-1)$ và $C(t)$ cho dấu thời gian trước đó và hiện tại.



- **Forget Gate:** Trong một ô của mạng nơ-ron LSTM, bước đầu tiên là quyết định xem chúng ta nên giữ thông tin từ bước thời gian trước đó hay quên nó đi.
- **Input Gate:** được sử dụng để định lượng tầm quan trọng của thông tin mới được mang theo đầu vào.
- **Output Gate:** truyền thông tin cập nhật

Đây là sơ đồ trực quan hơn của LSTM:



Inputs:

- x_t Current input
- C_{t-1} Memory from last LSTM unit
- h_{t-1} Output of last LSTM unit

Outputs:

- C_t New updated memory
- h_t Current output

Nonlinearities:

- σ Sigmoid layer
- \tanh Tanh layer
- b Bias

Vector operations:

- \times Scaling of information
- $+$ Adding information

CHƯƠNG 3. CÁC KẾT QUẢ THỰC NGHIỆM

1. Bộ Dữ Liệu

a. Nguồn dữ liệu

Bộ dữ liệu là tập hợp các tin nhắn từ API Twitter (bây giờ được gọi là X) bằng tiếng Anh với sáu cảm xúc cơ bản của con người: giận dữ, sợ hãi, vui vẻ, yêu thương, buồn bã và bất ngờ. Tập dữ liệu này được sử dụng để phân loại cảm xúc dựa trên các đoạn văn bản.

- Truy cập vào bộ dữ liệu:

```
data_train = pd.read_csv('/content/training.csv')
data_test = pd.read_csv('/content/test.csv')
data_validation = pd.read_csv('/content/validation.csv')
```

b. Tiền xử lý dữ liệu

- Nhóm sử dụng phương thức “.shape” để tìm số dòng số cột của bộ dữ liệu:

```
print('Dataset information:')
print(f'Training data: {data_train.shape}')
print(f'Validation data: {data_validation.shape}')
print(f'Test data: {data_test.shape}')

Dataset information:
Training data: (16000, 2)
Validation data: (2000, 2)
Test data: (2000, 2)
```

- Để có thể tăng độ mức độ quan sát cho tập dữ liệu và thuận tiện hơn cho việc tiền xử lý dữ liệu, ta sẽ gộp 2 tập ‘training’ và ‘test’ lại để tiến hành xử lý và xây dựng mô hình. Điều này sẽ giúp cho việc dự đoán sẽ chính xác hơn và tiết kiệm thời gian cho việc tiền xử lý dữ liệu hơn.

```
df = pd.concat([data_train, data_test], axis = 0)
```

- Sau khi ghép 2 tập, ta có bộ dữ liệu như sau:

	text	label
0	i didnt feel humiliated	0
1	i can go from feeling so hopeless to so damned...	0
2	im grabbing a minute to post i feel greedy wrong	3
3	i am ever feeling nostalgic about the fireplac...	2
4	i am feeling grouchy	3
...
1995	i just keep feeling like someone is being unki...	3
1996	im feeling a little cranky negative after this...	3
1997	i feel that i am useful to my people and that ...	1
1998	im feeling more comfortable with derby i feel ...	1
1999	i feel all weird when i have to meet w people ...	4

18000 rows × 2 columns

- Xử lý dữ liệu bị thiếu:

```
missing_value = df.isnull().sum()
missing_value

text      0
label     0
dtype: int64
```

- Không có dữ liệu bị thiếu trong bộ dữ liệu.
- Thêm một cột “emotion” và vẽ biểu đồ cột để so sánh số lượng giữa các “emotion”:
 - Xác định label và thêm cột:

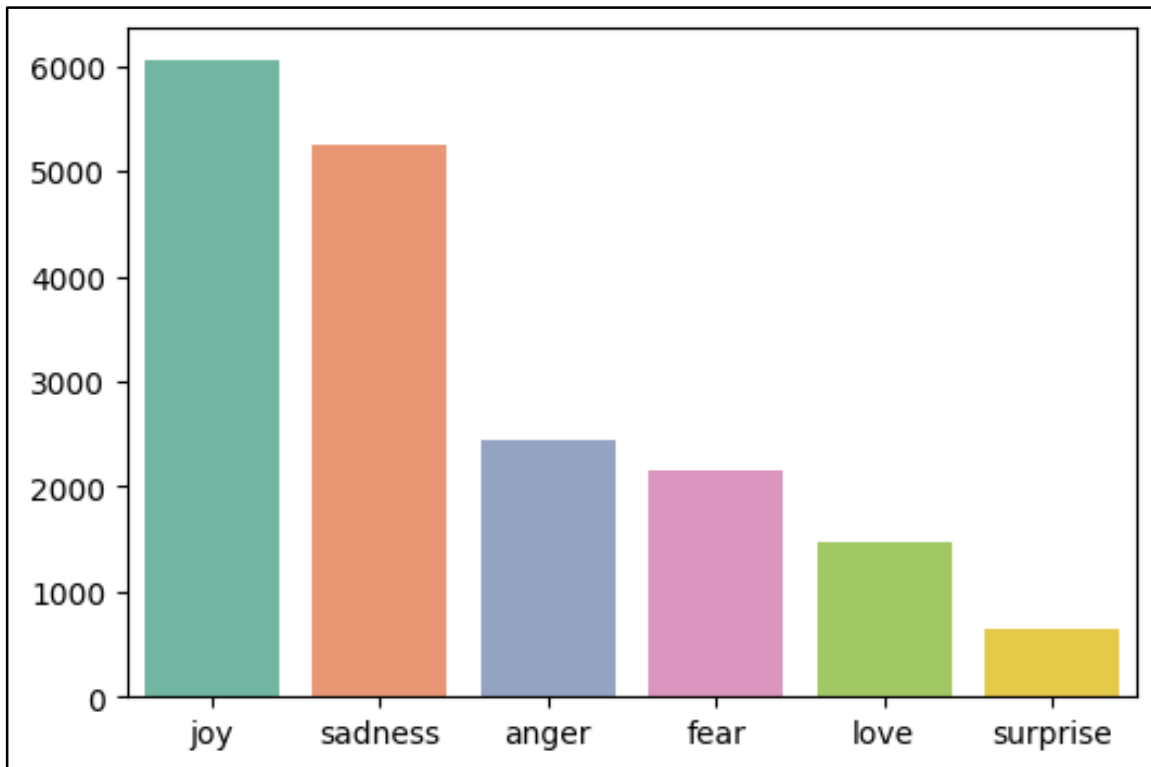
```
labels_dict = {0:'sadness', 1:'joy', 2:'love', 3:'anger', 4:'fear', 5:'surprise'}
df['emotion'] = df['label'].map(labels_dict )
df.head()
```

	text	label	emotion
0	i didnt feel humiliated	0	sadness
1	i can go from feeling so hopeless to so damned...	0	sadness
2	im grabbing a minute to post i feel greedy wrong	3	anger
3	i am ever feeling nostalgic about the fireplac...	2	love
4	i am feeling grouchy	3	anger

- Vẽ biểu đồ:

```
plt.figure(figsize=(6, 4))
sns.barplot(x=df['emotion'].value_counts().index, y=df['label'].value_counts().values, palette='Set2')

# displaying chart
plt.show()
```



- **Nhận xét:** Qua biểu đồ trên, ta nhận thấy được rằng lớp biểu cảm “joy” chiếm nhiều nhất (gần 6000), ngược lại lớp “surprise” chiếm ít nhất (dưới 1000). Nhìn chung lại, lớp “sadness” và “joy” vượt trội hơn về số lượng so với các lớp còn lại. 3 lớp “love”, “fear” và “anger” có sự chênh lệch nhau nhưng không đáng kể.
- Ta sử dụng thư viện nltk để lấy danh sách “stop-words” trong tiếng Anh. “stop-words” thường là những từ phổ biến như "is", "the", "and", không mang nhiều ý nghĩa trong việc phân tích ngôn ngữ tự nhiên và thường được loại bỏ để tăng hiệu suất và độ chính xác trong xử lý ngôn ngữ tự nhiên.

```
#Stop words present in the library
stopwords = nltk.corpus.stopwords.words('english')
wordnet_lemmatizer = WordNetLemmatizer()
```

- Sau đó ta sử dụng hàm **WordNetLemmatizer()** để thực hiện **lemmatization** trên các từ trong văn bản, **lemmatization** là quá trình chuyển đổi các từ về dạng gốc

(lemma).

- Tiếp theo ta xây dựng các hàm để tiến hành tiền xử lý dữ liệu văn bản:

- Xây dựng hàm để tạo một chuỗi mới bằng cách lựa chọn từng ký tự trong **text** mà không phải là dấu câu.

```
def remove_punctuation(text):  
    punctuationfree="".join([i for i in text if i not in string.punctuation])  
    return punctuationfree
```

- Xây dựng hàm để tách văn bản thành các token dựa trên khoảng trắng và ký tự không phải chữ cái hoặc số ('\\W+').

```
def tokenization(text):  
    tokens = re.split('\\W+',text)  
    return tokens
```

- Xây dựng hàm để tạo một danh sách mới chỉ chứa các token không thuộc danh sách “stop-words”.

```
def remove_stopwords(text):  
    output= [i for i in text if i not in stopwords]  
    return output
```

- Xây dựng hàm để áp dụng lemmatization cho mỗi từ trong danh sách sử dụng **WordNetLemmatizer**

```
def lemmatizer(text):  
    lemm_text = [wordnet_lemmatizer.lemmatize(word) for word in text]  
    return lemm_text
```

- Áp dụng các hàm vừa tạo vào tập data

```
df['text'] = df['text'].apply(lambda x:remove_punctuation(x))  
df['text'] = df['text'].apply(lambda x: tokenization(x))  
df['text'] = df['text'].apply(lambda x:remove_stopwords(x))  
df['text'] = df['text'].apply(lambda x:lemmatizer(x))
```

Ta gán lại giá trị cho cột 'text' trong DataFrame (df) từ một cột 'text' hiện tại:

```
df['text'] = [item[0] for item in df['text']]
```

Lúc này ta có bộ dữ liệu sau khi được tiền xử lý

	text	label
0	i didnt feel humiliated	0
1	i can go from feeling so hopeless to so damned...	0
2	im grabbing a minute to post i feel greedy wrong	3
3	i am ever feeling nostalgic about the fireplac...	2
4	i am feeling grouchy	3
...
1995	i just keep feeling like someone is being unki...	3
1996	im feeling a little cranky negative after this...	3
1997	i feel that i am useful to my people and that ...	1
1998	im feeling more comfortable with derby i feel ...	1
1999	i feel all weird when i have to meet w people ...	4

[18000 rows x 2 columns]

Sau đó, ta phân mảnh văn bản thành các token bắt đầu từ ký tự chữ cái hoặc số, và mỗi token có thể chứa một hoặc nhiều ký tự chữ cái hoa, chữ cái thường hoặc số bằng hàm sau:

```
token = RegexpTokenizer(r'[a-zA-Z0-9]+')
```

c. Vector hóa dữ liệu

- Đầu tiên, xây dựng một ma trận thuật ngữ bằng CountVectorizer từ dữ liệu đã xử lý. Với:

- **lowercase = True**: tất cả các từ trong văn bản sẽ được chuyển thành chữ thường. Điều này giúp giảm sự phân loại giữa các từ viết hoa và viết thường.
- **stop_words='english'**: Tham số này xác định danh sách các từ dừng (stop words) mà bạn muốn loại bỏ khỏi văn bản. Trong trường hợp này, các stop words được chọn là tiếng Anh ('english').
- **ngram_range = (1,1)**: Tham số này chỉ định kích thước của các "n-grams" mà bạn muốn xem xét. Trong trường hợp này, chỉ sử dụng các từ đơn

(ngram_range=(1,1)), tức là chỉ xem xét các từ riêng lẻ.

- **tokenizer = token.tokenize**: Đối số này xác định một hàm hoặc đối tượng tokenizer để sử dụng để chia văn bản thành các từ. Trong trường hợp này, có vẻ như 'token' là một đối tượng tokenizer và "token.tokenize" là hàm chia từ.

```
vect = CountVectorizer(lowercase=True, stop_words='english', ngram_range = (1,1), tokenizer = token.tokenize)
```

2. Phân chia dữ liệu

Từ bộ dữ liệu quan sát trên, ta tiến hành chia thành các tập train test. Với:

- Biến feature sử dụng là cột "**text**"– Cột dữ liệu chứa các câu có những từ chỉ cảm xúc (X)
- Biến target là cột "**label**" (y)

```
X = df['text']  
y = df['label']
```

- Hàm `train_test_split` của thư viện `sklearn` để chia tách dữ liệu thành 2 tập train test với tỷ lệ tập test là 20% trên tổng bộ dữ liệu.
- **random_state** là một số nguyên dùng để đảm bảo rằng việc phân chia dữ liệu sẽ được thực hiện theo cùng một cách mỗi lần chạy mã. Điều này giúp đảm bảo tính nhất quán và tái tạo kết quả.

```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2, random_state=123)
```

3. Áp dụng bộ dữ liệu vào các mô hình

Mô hình Máy học

- Đầu tiên, nhóm tiến hành nhập các thư viện cần thiết cho mô hình học máy - Decision Tree Classification kết hợp với kỹ thuật Adaboost
- Tiếp theo, tiến hành chuẩn hóa dữ liệu và phân chia dữ liệu với tỷ lệ tập test là 20%
- Khởi tạo mô hình phân loại `AdaBoostClassifier` và chỉ định công cụ ước tính cơ sở là `DecisionTreeClassifier()`. Tham "n_estimators" số được đặt thành 100, cho biết

rằng chúng tôi muốn sử dụng 100 bộ phân loại yếu

- Mô hình phân loại AdaBoost được huấn luyện dựa trên dữ liệu huấn luyện (X_train và y_train)
- Cuối cùng, dùng tập test để đánh giá mô hình

```
X_ml = vect.fit_transform(X)
```

```
X_train_ml, X_test_ml, y_train, y_test = train_test_split(X_ml, y, test_size=0.2, random_state=123)
```

```
classifier = AdaBoostClassifier(base_estimator=DecisionTreeClassifier(), n_estimators=100)
```

```
classifier.fit(X_train_ml, y_train)
```

```
y_pred_adaboost = classifier.predict(X_test_ml)
```

```
accuracy_Ada = accuracy_score(y_test, y_pred_adaboost)
```

```
print(f"Accuracy: {accuracy_Ada*100:.2f}%")
```

```
print("Classification Report:\n", classification_report(y_test, y_pred_adaboost))
```

Và thu được kết quả khi chạy mô hình:

Accuracy: 83.86%

Classification Report:

	precision	recall	f1-score	support
0	0.87	0.86	0.86	1060
1	0.86	0.84	0.85	1225
2	0.76	0.73	0.74	300
3	0.80	0.87	0.83	459
4	0.86	0.82	0.84	443
5	0.65	0.78	0.71	113
accuracy			0.84	3600
macro avg	0.80	0.82	0.81	3600
weighted avg	0.84	0.84	0.84	3600

- **Nhận xét:**

- Độ chính xác (Accuracy) của tổng thể mô hình Decision Tree kết hợp với Adaboost là 83.86%. Đây là một độ chính xác khá cao, cho thấy mô hình có thể dự đoán chính xác loại cảm xúc của một tweet trong 83,86% trường hợp.
- Lớp cảm xúc "0" có độ chính xác cao nhất, đạt 87%. Lớp cảm xúc "5"

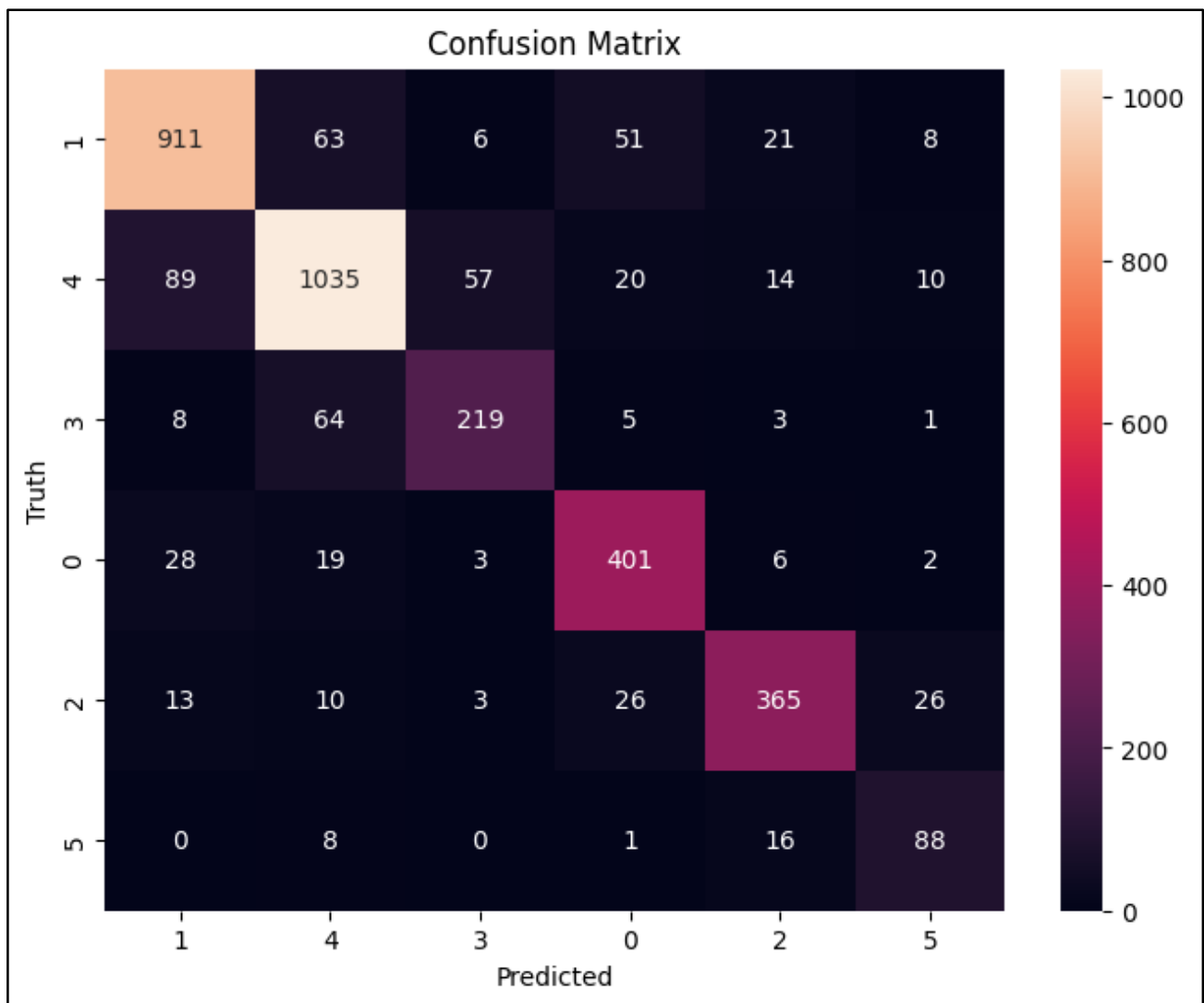
có độ chính xác thấp nhất, đạt 65%.

- Theo bảng dữ liệu, F1-score trung bình của mô hình là 84%. Đây là một F1-score khá cao, cho thấy mô hình có cả precision và recall cao.
 - Nhìn chung, mô hình được đánh giá là có chất lượng tốt. Tuy nhiên, mô hình cũng còn một số điểm cần cải thiện.
- Để có thể nhìn rõ kết quả của mô hình phân lớp giữa các lớp, nhóm tiến hành vẽ ma trận nhầm lẫn của mô hình:

```
cm_ml = confusion_matrix(y_test, y_pred_adaboost)
labels = y_test.unique()

plt.figure(figsize=(8,6))
sns.heatmap(cm_ml, annot=True, fmt='d', xticklabels=labels, yticklabels=labels)
plt.xlabel('Predicted')
plt.ylabel('Truth')
plt.title('Confusion Matrix')
plt.show()
```

- Và thu được kết quả:



Mô hình Maximum Entropy

Ta tiến hành nhập các thư viện cần thiết:

```
import nltk

from nltk.classify import MaxentClassifier

from nltk.tokenize import word_tokenize
```

Sau khi thêm các hàm từ các thư viện cần thiết ta tiến hành huấn luyện dữ liệu cho mô hình Maximum Entropy theo các bước:

- Đầu tiên ta trích xuất hàm đặc trưng với tên là `extract_features`. Hàm này sẽ giúp tách văn bản thành các từ riêng lẻ với mỗi từ là một đặc trưng với giá trị là `True`.

```
# Hàm trích xuất đặc trưng từ văn bản
def extract_features(text):
    words = word_tokenize(text)
    return dict([(word, True) for word in words])
```

- Chuẩn bị dữ liệu huấn luyện (train_data) và dữ liệu kiểm thử (test_data):
 - Sử dụng hàm zip để kết hợp danh sách X_train (văn bản đầu vào) và y_train (nhãn đầu ra) thành các cặp tương ứng và tạo danh sách train_data.
 - Tương tự, thực hiện tương tự cho X_test và y_test để tạo danh sách test_data.

```
train_data = list(zip(X_train, y_train))
test_data = list(zip(X_test, y_test))
```

- Trích xuất đặc trưng và tạo dữ liệu đầu vào cho MaxEnt:

```
# Trích xuất đặc trưng và tạo dữ liệu đầu vào cho MaxEnt
train_features = [(extract_features(text), label) for text, label in train_data]
test_features = [(extract_features(text), label) for text, label in test_data]
```

- Xây Dựng và Huấn Luyện Mô Hình bằng MaxentClassifier.train: Hàm này được sử dụng để xây dựng và huấn luyện mô hình MaxEnt trên dữ liệu huấn luyện với thuật toán Improved Iterative Scaling (iis) là một phương pháp tối ưu hóa. Quá trình này sẽ diễn ra 20 lần lặp, và sẽ dừng lại khi mô hình khi sự thay đổi của mô hình nhỏ hơn 0.001.

```
# Xây dựng và huấn luyện mô hình MaxEnt
maxent_classifier = MaxentClassifier.train(train_features, trace=0,
                                          algorithm='iis', max_iter = 20, min_lldelta=0.001)
```

- Đánh giá mô hình trên tập kiểm tra: Dữ liệu kiểm tra được sử dụng để đánh giá hiệu suất của mô hình trên tập test bằng cách sử dụng độ chính xác (làm tròn đến 2 chữ số thập phân). Sau đó in ra **Classification Report** của các thể loại cảm xúc với các chỉ số về độ chính xác, độ nhạy cũng như là F1-score.

```
# Đánh giá mô hình trên tập kiểm tra
predictions = [maxent_classifier.classify(extract_features(text)) for text, _ in test_data]

# Đánh giá mô hình
accuracy = accuracy_score([label for _, label in test_data], predictions)
print(f"Accuracy: {accuracy*100:.2f}%")
print("Classification Report:\n", classification_report([label for _, label in test_data], predictions))
```

- Nhóm thu được kết quả sau khi chạy mô hình như sau:

Accuracy: 74.61%				
Classification Report:				
	precision	recall	f1-score	support
0	0.77	0.88	0.82	1060
1	0.72	0.91	0.81	1225
2	0.66	0.28	0.39	300
3	0.77	0.56	0.65	459
4	0.79	0.61	0.69	443
5	0.76	0.22	0.34	113
accuracy			0.75	3600
macro avg	0.74	0.58	0.62	3600
weighted avg	0.75	0.75	0.73	3600

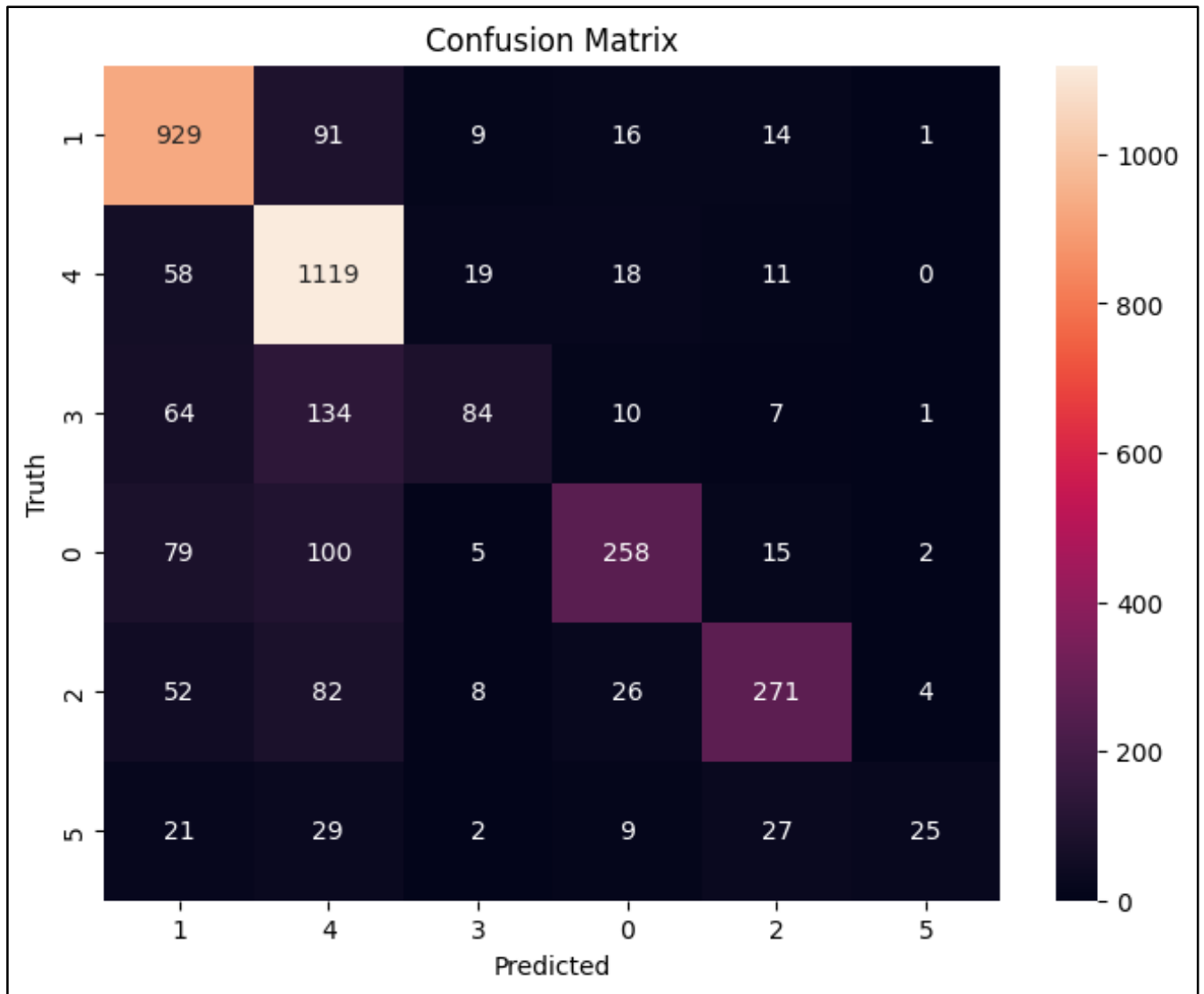
- **Nhận xét:**

- Mô hình có độ chính xác tổng thể là 74.61%, đây là một mức độ chấp nhận được.
- Lớp 0 (Class 0) có hiệu suất tốt với precision là 0.77, recall là 0.88, và F1-score là 0.82, có thể nói rằng khả năng dự đoán của mô hình cho các mẫu ở lớp 0 khá chính xác
- Tuy nhiên các chỉ số ở lớp 2 và 5 có hiệu suất khá thấp, chỉ số recall chỉ có 0.28 và 0.22. Mô hình có thể gặp khó khăn trong việc nhận biết và dự đoán các mẫu trong các lớp này.

- Tương tự, nhóm có ma trận nhầm lẫn của mô hình MaxEnt:

```
cm_maxent = confusion_matrix([label for _, label in test_data], predictions)
labels = y_test.unique()

plt.figure(figsize=(8,6))
sns.heatmap(cm_maxent, annot=True, fmt='d', xticklabels=labels, yticklabels=labels)
plt.xlabel('Predicted')
plt.ylabel('Truth')
plt.title('Confusion Matrix')
plt.show()
```



Mô hình Deep Learning LSTM

- Import thư viện cần thiết để xây dựng mô hình LSTM:

```
import keras

import tensorflow

from keras.preprocessing.text import Tokenizer

from tensorflow.keras.preprocessing.sequence import pad_sequences

from sklearn.preprocessing import LabelEncoder

from keras.models import Sequential

from keras.layers import Embedding, Flatten, Dense, LSTM
```

- Đầu tiên, nhóm tiến hành mã hóa dữ liệu văn bản bằng phương thức Tokenizer:

```
# Tokenize the text data
tokenizer = Tokenizer()
tokenizer.fit_on_texts(df['text'])
```

- Chuyển dữ liệu văn bản thành chuỗi bằng cách sử dụng từ vựng do bộ mã thông báo tạo ra. Đệm các chuỗi để đảm bảo rằng tất cả các chuỗi có cùng độ dài (max_sequence_length):

```
# Chuyển văn bản thành chuỗi
train_sequences = tokenizer.texts_to_sequences(df['text'])

val_sequences = tokenizer.texts_to_sequences(data_validation['text'])

# Đệm các chuỗi để đảm bảo rằng tất cả các chuỗi có cùng độ dài
max_sequence_length = max(len(seq) for seq in train_sequences)

train_sequences = pad_sequences(train_sequences, maxlen=max_sequence_length)
val_sequences = pad_sequences(val_sequences, maxlen=max_sequence_length)
```

- Tạo mô hình tuần tự với lớp embedding, lớp LSTM và lớp dense có kích hoạt softmax:

```
# Define the model architecture
model = Sequential()
model.add(Embedding(input_dim=len(tokenizer.word_index) + 1,
                    output_dim=128, input_length = max_sequence_length))

model.add(LSTM(128))
model.add(Dense(6, activation='softmax'))
```

- Biên dịch mô hình bằng cách sử dụng hàm mất mát sparse categorical cross-entropy, trình tối ưu hóa Adam và độ chính xác làm thước đo đánh giá:

```
# Compile the model

model.compile(loss='sparse_categorical_crossentropy',
              optimizer='adam', metrics=['accuracy'])
```

- Huấn luyện mô hình trên dữ liệu huấn luyện (train_sequences) với các nhãn tương ứng (df['label']):

```
# Train the model
history = model.fit(train_sequences, df['label'], epochs=10, batch_size=32,
                    validation_data=(val_sequences, data_validation['label']))
```

- Đánh giá mô hình:

```
# Evaluate the model on the validation set
val_loss, val_accuracy = model.evaluate(val_sequences, data_validation['label'], verbose=0)
print(f'Validation Accuracy: {val_accuracy*100:.2f}%')
```

```
# Predict probabilities for each class on the validation set
val_probabilities = model.predict(val_sequences)

# Get predicted classes based on the highest probability
val_predictions = val_probabilities.argmax(axis=-1)

# Print classification report
class_report = classification_report(data_validation['label'], val_predictions)
print("Classification Report:\n", class_report)
```

- Thu được kết quả chạy mô hình như sau:

```
Validation Accuracy: 92.85%
63/63 [=====] - 1s 3ms/step
Classification Report:
```

	precision	recall	f1-score	support
0	0.96	0.97	0.97	550
1	0.95	0.93	0.94	704
2	0.83	0.88	0.86	178
3	0.91	0.95	0.93	275
4	0.90	0.86	0.88	212
5	0.82	0.85	0.84	81
accuracy			0.93	2000
macro avg	0.90	0.91	0.90	2000
weighted avg	0.93	0.93	0.93	2000

- **Nhận xét:**

- Mô hình Deep Learning LSTM cho thấy hiệu suất khá tốt trên tập kiểm tra với độ chính xác tổng thể là 92.85%.
- Trong lớp 0, mô hình đạt độ chính xác (precision) lên đến 96% và recall là 97%, cùng với F1-score đạt 97%. Mô hình có khả năng dự đoán chính

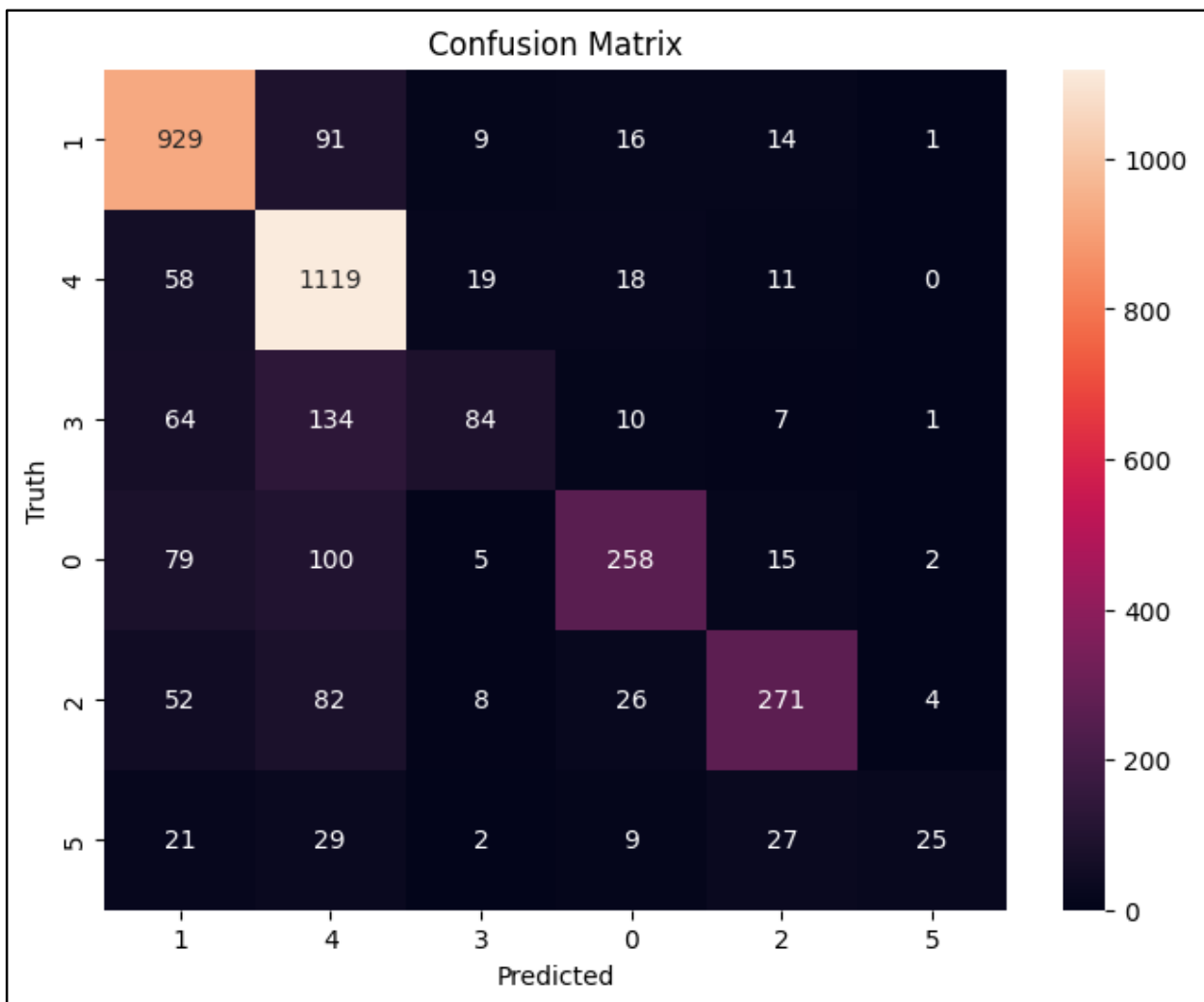
xác và nhận biết tốt trên lớp này.

- Còn với lớp 1, mô hình cũng đạt hiệu suất cao precision là 95%, recall là 93%, và F1-score là 94%. Điều này cũng cho thấy khả năng dự đoán và nhận biết tốt của mô hình.
- Với các lớp còn lại ta cũng có thể thấy rằng mặc dù có hiệu suất thấp hơn so với lớp 0 và lớp 1, nhưng mô hình vẫn có khả năng dự đoán tốt.

- Ma trận nhầm lẫn của mô hình:

```
cm_LSTM = confusion_matrix(data_validation['label'], val_predictions)
labels = y_test.unique()

plt.figure(figsize=(8,6))
sns.heatmap(cm_maxent, annot=True, fmt='d', xticklabels=labels, yticklabels=labels)
plt.xlabel('Predicted')
plt.ylabel('Truth')
plt.title('Confusion Matrix')
plt.show()
```



- Đồ thị biểu diễn mức độ chính xác và độ mất mát của mô hình

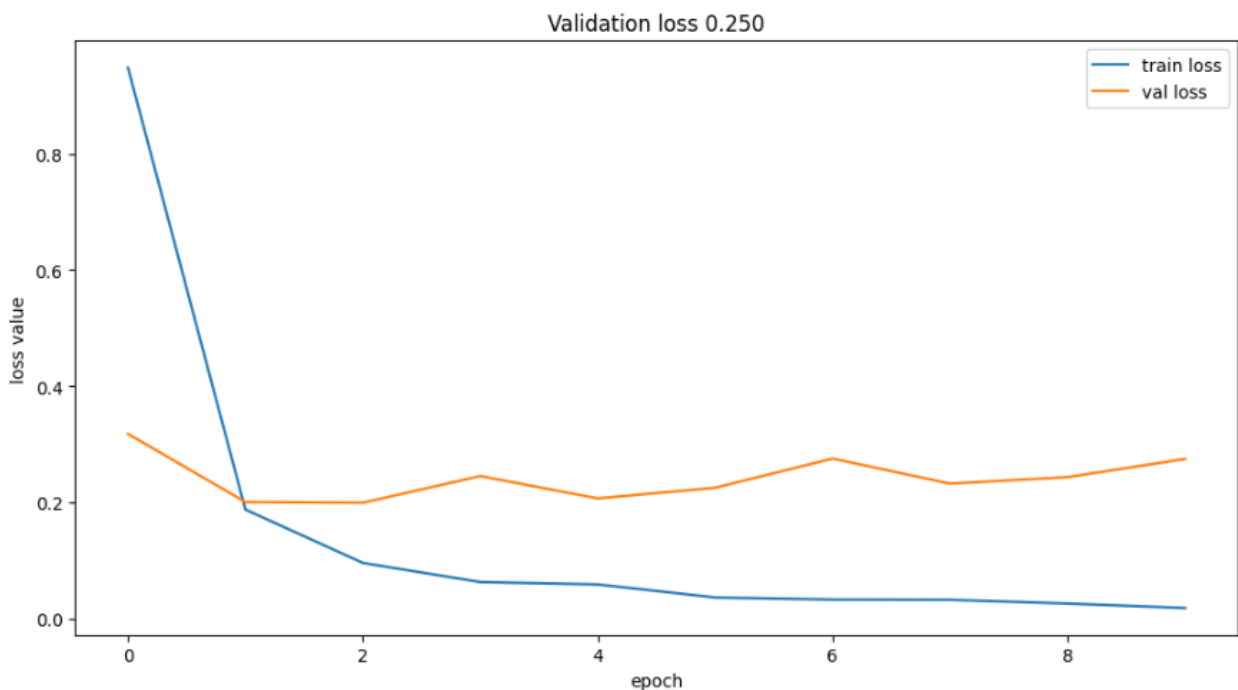
- Đồ thị biểu diễn độ mất mát:

```
# Create two plots: one for the loss value, one for the accuracy
fig, (ax1, ax2) = plt.subplots(nrows=1, ncols=2, figsize=(20, 6))

plt.suptitle('Model training/validation curves', size=15)
```

```
# Plot loss values
ax1.plot(history.history["loss"], label="train loss")
ax1.plot(history.history["val_loss"], label="val loss")
ax1.set_title(
    "Validation loss {:.3f}".format(
        np.mean(history.history["val_loss"][-3:]) # last three values
    )
)
ax1.set_xlabel("epoch")
ax1.set_ylabel("loss value")
ax1.legend()
```

- Biểu đồ tương ứng:



- **Nhận xét:**

- Biểu đồ thể hiện sự thay đổi của độ mất mát huấn luyện (train loss) và độ mất mát xác thực (validation loss) của mô hình. Độ mất mát huấn luyện là một thước đo mức độ sai lệch giữa dự đoán của mô hình và giá trị thực của các mẫu trong tập huấn luyện. Độ mất mát xác thực là một thước đo mức độ sai lệch giữa dự đoán của mô hình và giá trị thực của

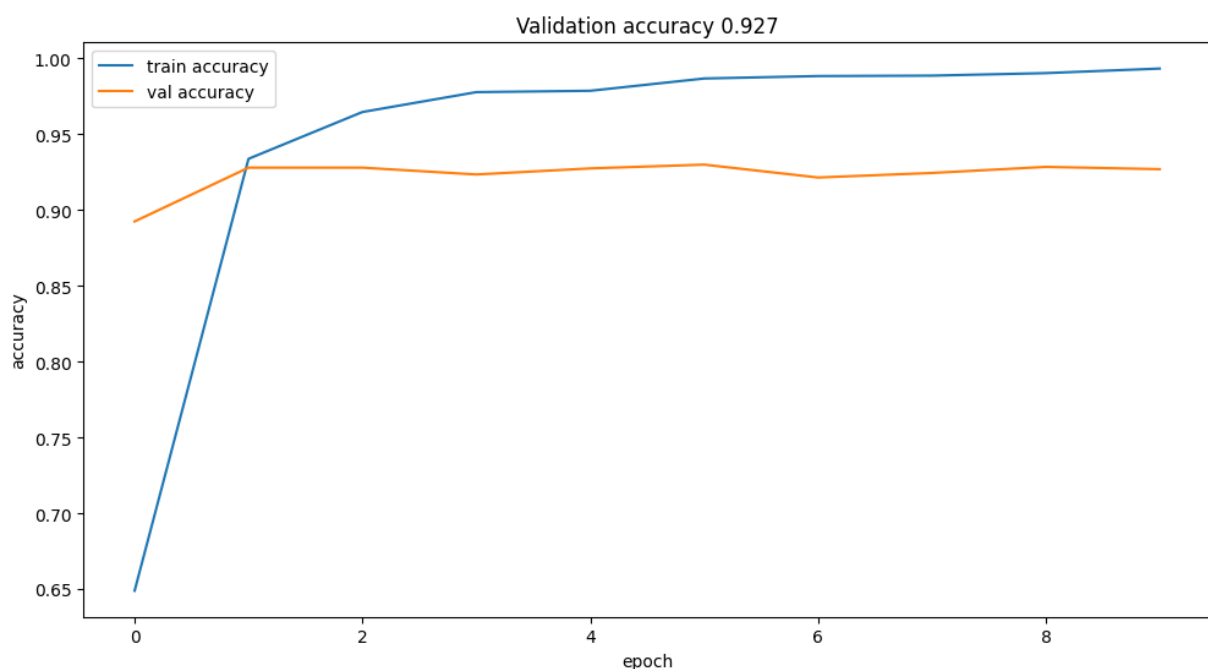
các mẫu trong tập xác thực.

- Theo biểu đồ, độ mất mát huấn luyện của mô hình giảm dần đều theo số lượng epoch. Điều này cho thấy mô hình đang học hỏi và cải thiện khả năng dự đoán của mình. Độ mất mát xác thực của mô hình cũng giảm dần trong giai đoạn đầu, nhưng sau đó bắt đầu chững lại và dao động xung quanh giá trị 0.26. Điều này cho thấy mô hình đã đạt đến giới hạn về khả năng dự đoán của mình trên tập dữ liệu xác thực.

- Đồ thị biểu diễn độ chính xác:

```
# Plot accuracy values
ax2.plot(history.history["accuracy"], label="train accuracy")
ax2.plot(history.history["val_accuracy"], label="val accuracy")
ax2.set_title(
    "Validation accuracy {:.3f}".format(
        np.mean(history.history["val_accuracy"][-3:]) # last three values
    )
)
ax2.set_xlabel("epoch")
ax2.set_ylabel("accuracy")
ax2.legend()
plt.tight_layout()
plt.show()
```

- Đồ thị tương ứng:



- Nhận xét:

- Biểu đồ thể hiện sự thay đổi của độ chính xác huấn luyện (train accuracy) và độ chính xác xác thực (validation accuracy) của một mô

hình dự đoán. Độ chính xác huấn luyện là tỷ lệ các mẫu được mô hình dự đoán chính xác trong quá trình huấn luyện, trong khi độ chính xác thực là tỷ lệ các mẫu được mô hình dự đoán chính xác trên tập dữ liệu mới, chưa từng thấy trước đây.

- Theo biểu đồ, độ chính xác huấn luyện của mô hình tăng dần đều theo số lượng epoch (vòng lặp huấn luyện). Điều này cho thấy mô hình đang học hỏi và cải thiện khả năng dự đoán của mình. Độ chính xác thực của mô hình cũng tăng dần trong giai đoạn đầu, nhưng sau đó bắt đầu chững lại và dao động xung quanh giá trị 0.92. Điều này cho thấy mô hình đã đạt đến giới hạn về khả năng dự đoán của mình trên tập dữ liệu xác thực.
- Nhìn chung, các biểu đồ cho thấy mô hình đang hoạt động tốt. Độ chính xác huấn luyện và độ chính xác thực của mô hình đều cao, đồng thời độ mất mát huấn luyện và độ mất mát thực của mô hình đều thấp. Điều này cho thấy mô hình có khả năng dự đoán tốt trên cả tập dữ liệu huấn luyện và tập dữ liệu xác thực.

So sánh mô hình

- Đầu tiên, tạo một dict để lưu trữ chỉ số “accuracy” của ba mô hình
- Chọn ra mô hình có độ chính xác cao nhất trong ba mô hình
- Trả về kết quả độ chính xác cao nhất và mô hình tương ứng

```
# Tạo một dictionary lưu trữ độ chính xác của từng mô hình
accuracy_dict = {
    'Decision Tree': accuracy_Ada,
    'MaxEnt': accuracy_maxent,
    'Deep Learning LSTM': val_accuracy
}

# Chọn ra mô hình có độ chính xác cao nhất
best_model = max(accuracy_dict, key=accuracy_dict.get)

# In ra mô hình có độ chính xác cao nhất và giá trị độ chính xác tương ứng
print("Best Model:", best_model)
print("Accuracy:", accuracy_dict[best_model])
```

- Thu được kết quả:

Best Model: Deep Learning LSTM
Accuracy: 0.9265000224113464

- **Nhận xét:** Dựa vào kết quả trên, nhóm rút ra được mô hình Deep Learning (LSTM) là mô hình tốt nhất trong ba mô hình. Vì vậy, nhóm sẽ sử dụng mô hình này vào bộ dữ liệu để tiến hành dự đoán kết quả.

Ứng dụng cho bài toán Text Classification

- **Ứng dụng 1: In ra tất cả các dự đoán của tập dữ liệu thử nghiệm (validation)**
 - Khởi tạo LabelEncoder và gắn với “emotion_labels”:

```
# Khởi tạo LabelEncoding và gắn nó với emotion_labels
label_encoder = LabelEncoder()
emotion_labels = {0: 'sadness', 1: 'joy', 2: 'love', 3: 'anger', 4: 'fear', 5: 'surprise'}
label_encoder.fit(list(emotion_labels.keys()))
```

- Ở đây, LabelEncoder được sử dụng để chuyển đổi các emotion_labels từ dạng chuỗi văn bản sang dạng số nguyên. “emotion_labels” định nghĩa một ánh xạ giữa các số nguyên và cảm xúc tương ứng.
- Tạo một cột mới có tên là 'predict' trong DataFrame data_validation để lưu trữ kết quả dự đoán cảm xúc từ mô hình.

```
# Tạo cột 'predict' mới trong DataFrame data_validation
data_validation['predict'] = np.nan
```

- Sử dụng vòng lặp duyệt qua từng mẫu văn bản trong tập validation. Mỗi văn bản được chuẩn bị bằng cách chuyển đổi thành dãy số, sau đó đưa vào mô hình để đưa ra dự đoán về cảm xúc. Kết quả dự đoán được chuyển ngược từ dạng số sang chuỗi cảm xúc và sau đó được cập nhật vào cột 'predict' tương ứng với mỗi mẫu văn bản trong DataFrame “data_validation”:

```
# Lặp qua tất cả các hàng trong data_validation
for index, row in data_validation.iterrows():
    input_text = row['text']

    # Xử lý trước văn bản đầu vào
    input_sequence = tokenizer.texts_to_sequences([input_text])
    padded_input_sequence = pad_sequences(input_sequence, maxlen = max_sequence_length)

    # Đưa ra dự đoán
    prediction = model.predict(padded_input_sequence)

    # Chuyển đổi ngược the predicted label
    predicted_label = label_encoder.inverse_transform([np.argmax(prediction[0])])

    # Cập nhật cột "predict" với predicted label
    data_validation.at[index, 'predict'] = predicted_label[0]

data_validation['predict'] = data_validation['predict'].astype(int)
```

- Ta thu được dataframe data_validation sau khi chạy mô hình và thêm cột dự đoán:

```
print(data_validation)
```

	text	label	predict
0	im feeling quite sad and sorry for myself but ...	0	0
1	i feel like i am still looking at a blank canv...	0	0
2	i feel like a faithful servant	2	2
3	i am just feeling cranky and blue	3	3
4	i can have for a treat or if i am feeling festive	1	1
...
1995	im having ssa examination tomorrow in the morn...	0	0
1996	i constantly worry about their fight against n...	1	1
1997	i feel its important to share this info for th...	1	1
1998	i truly feel that if you are passionate enough...	1	1
1999	i feel like i just wanna buy any cute make up ...	1	1

2000 rows × 3 columns

- **Nhận xét:** Qua kết quả trên, ta thấy sẽ so sánh được kết quả dự đoán cảm xúc với cột label của bộ dữ liệu validation. Tuy nhiên, vì đây là mô hình dự đoán nên trường hợp dự đoán sai là điều không thể tránh khỏi. Đây cũng là một trong những hạn chế của mô hình.

• Ứng dụng 2: Nhập vào câu và trả về kết quả dự đoán

- Với trường hợp này, nhóm sẽ tạo ra một giao diện cơ bản để nhập câu muốn dự đoán cảm xúc và trả về kết quả dự đoán. Do notebook (ipynb) không hỗ trợ hiển thị giao diện vì vậy nhóm sẽ thực hiện trên tập tin có thuộc tính “.py”.
- Đầu tiên, khởi tạo một hàm “predict_emotion” để dự đoán cảm xúc từ một đoạn văn bản được nhập vào.
 - Kết quả dự đoán được hiển thị trong một GUI sử dụng thư viện Tkinter.
 - Giao diện người dùng bao gồm một ô nhập văn bản, một ô hiển thị kết quả, và một nút "Predict Emotion" để gọi hàm dự đoán.

```
# Chức năng dự đoán emotion
def predict_emotion(input_text, output_text_widget):
    input_text = input_text.strip()
    if input_text:
        input_sequence = tokenizer.texts_to_sequences([input_text])
        padded_input_sequence = pad_sequences(input_sequence, maxlen=max_sequence_length)

        prediction = rnn_model.predict(padded_input_sequence)

        # Xác định emotion_labels
        emotion_labels = {0: 'sadness', 1: 'joy', 2: 'love', 3: 'anger', 4: 'fear', 5: 'surprise'}

        # Điều chỉnh LabelEncode với các đối số phù hợp
        label_encoder = LabelEncoder()
        label_encoder.fit(list(emotion_labels.keys()))

        # Chuyển đổi ngược predicted label
        predicted_label = label_encoder.inverse_transform([np.argmax(prediction[0])])
        emotion_labels = emotion_labels[label_encoder.inverse_transform([np.argmax(prediction[0])])[0]]

        # Cập nhật output text widget
        output_text_widget.config(state="normal")
        output_text_widget.delete(1.0, "end")
        output_text_widget.insert("end", f"Predicted Emotion: {predicted_label}: {emotion_labels}")
        output_text_widget.config(state="disabled")
```

- Đối với giao diện người dùng Tkinter:
 - Một cửa sổ Tkinter được tạo với các phần tử như nhãn, ô nhập, ô hiển thị kết quả, thanh cuộn, và nút dự đoán.
 - Khi người dùng nhập văn bản và nhấn nút "Predict Emotion," mô hình sẽ dự đoán cảm xúc và hiển thị kết quả trong ô hiển thị.

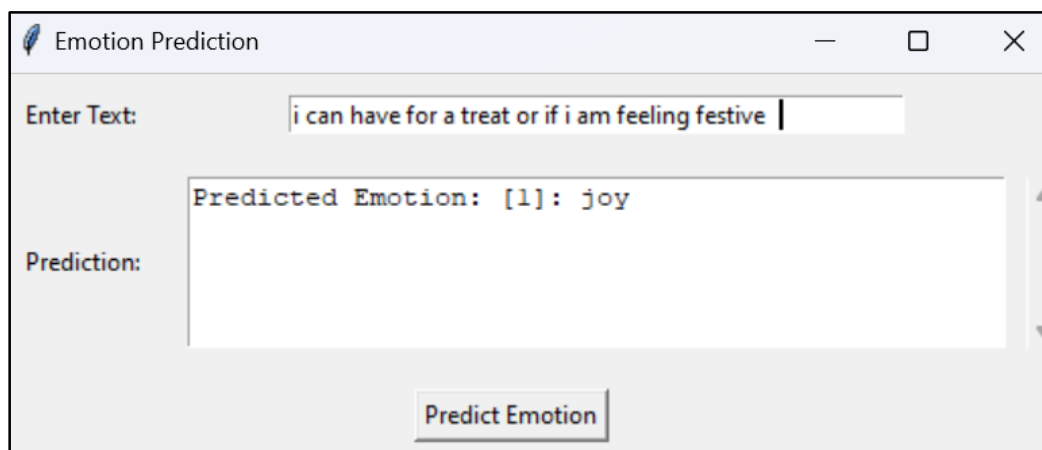
```
# Tạo GUI
root = tk.Tk()
root.title("Emotion Prediction")
```

```
# Định nghĩa các phần tử GUI
input_label = Label(root, text="Enter Text:")
input_entry = Entry(root, width=50)
output_label = Label(root, text="Prediction:")
output_text = Text(root, height=5, width=50, state="disabled")
scrollbar = Scrollbar(root, command=output_text.yview)
output_text.config(yscrollcommand=scrollbar.set)
predict_button = Button(root, text="Predict Emotion", command=lambda: predict_emotion(input_entry.get(), output_text))
```

```
# Đặt các phần tử GUI trên cửa sổ
input_label.grid(row=0, column=0, padx=10, pady=10, sticky="w")
input_entry.grid(row=0, column=1, padx=10, pady=10)
output_label.grid(row=1, column=0, padx=10, pady=10, sticky="w")
output_text.grid(row=1, column=1, padx=10, pady=10)
scrollbar.grid(row=1, column=2, pady=10, sticky="ns")
predict_button.grid(row=2, column=0, columnspan=2, pady=10)

# Chạy vòng lặp chính của GUI
root.mainloop()
```

- Nhóm thu được kết quả như sau:



4. Phân tích và đánh giá

Từ kết quả đánh giá 3 mô hình trên có thể thấy, độ chính xác của cả ba mô hình có sự chênh lệch rõ ràng, với mô hình có độ chính xác cao nhất là mô hình Deep Learning LSTM với độ chính xác là 92.85%, cao hơn rất nhiều so với 2 mô hình là cây quyết định với 83.86% và mô hình MaxEnt với 74.61%. Sự chênh lệch này là do LSTM có khả năng mô hình hóa mối quan hệ phức tạp và lâu dài trong dữ liệu chuỗi văn bản và việc đào tạo mô hình LSTM trên lượng dữ liệu đa dạng và sử dụng tiền xử lý dữ liệu chính xác cũng đóng góp vào hiệu suất của nó. Trong khi mô hình cây quyết định và MaxEnt có thể gặp khó khăn trong việc mô hình hóa các mối quan hệ phức tạp và dài hạn trong dữ liệu văn bản.

Đối với những ứng dụng mà nhóm áp dụng, cũng có những khía cạnh có lợi và những hạn chế nhất định đối với bài toán trên. Về mặt tích cực, nhóm đã ứng dụng được mô hình và đưa ra kết quả dự đoán, cũng như là áp dụng được một giao diện đơn giản để hiển thị kết quả dự đoán. Về mặt hạn chế, nhóm nhận thấy rằng mô hình dự đoán không chính xác hoàn toàn. Và phần giao diện chỉ là giao diện đơn giản, vẫn chưa thể đưa vào áp dụng thực tế cho các doanh nghiệp. Tuy nhiên, dự án này vẫn còn nhiều hướng phát triển và có thể phát triển hơn nữa để có thể áp dụng vào thực tiễn.

CHƯƠNG 4. KẾT LUẬN

1. Các Kết Quả Đạt Được

Trong quá trình nghiên cứu đồ án xử lý ngôn ngữ tự nhiên, nhóm nhận thấy thành lĩnh vực NLP là một lĩnh vực vô cùng rộng lớn. Vì vậy trong đề tài này nhóm tập trung tìm hiểu và nghiên cứu tổng quan về các lĩnh vực trong môn học xử lý ngôn ngữ tự nhiên như một số thuật toán phân tích cú pháp và cũng như tìm hiểu về các phương pháp gán nhãn từ loại các phương pháp học máy. Từ đó xây dựng các phương pháp gán nhãn từ loại.

Do thời gian cũng như kiến thức còn hạn chế nên nhóm vẫn còn nhiều thiếu sót và chỉ tìm hiểu và áp dụng những bước cơ bản trong quá trình xử lý ngôn ngữ tự nhiên. Trong thời gian tới, nhóm sẽ tìm hiểu thêm về các khía cạnh khác của xử lý ngôn ngữ tự nhiên, các bước để tiến xử lý dữ liệu để có thể cải thiện độ chính xác của các mô hình khi áp dụng bài toán vào thực tế.

2. Những Hạn Chế và Hướng Phát Triển

Những hạn chế:

- Bài toán giới hạn bộ dữ liệu cảm xúc chỉ đến 5 loại, điều này tạo ra một hạn chế lớn trong việc phân loại và dự đoán cảm xúc. Thực tế, môi trường xã hội phong phú và đa dạng về cảm xúc, và giới hạn chỉ đến 5 loại không thể hiện đủ sự đa dạng và phức tạp của cảm xúc con người. Điều này gây ra khả năng bỏ qua, hoặc không phân loại chính xác, những cảm xúc flexible và phong phú hơn mà con người có thể trải nghiệm trong thực tế.
- Một vấn đề khác đối mặt trong bài toán này là thời gian chạy mô hình, đặc biệt là khi xử lý lượng dữ liệu lớn. Việc đưa ra các dự đoán cảm xúc yêu cầu sự xử lý số liệu lớn và phức tạp, điều này có thể dẫn đến thời gian chạy mô hình tăng cao. Thời gian chờ đợi lâu có thể làm giảm hiệu suất và khả năng áp dụng mô hình trong các ứng dụng thời gian thực hoặc yêu cầu độ trễ thấp. Điều này trở thành một thách thức đặc biệt khi cần xử lý dữ liệu liên tục hoặc trong môi trường yêu cầu phản ứng nhanh.

Hướng phát triển:

Mở rộng số lượng cảm xúc

Tăng cường số lượng cảm xúc được phân loại để bao quát đầy đủ độ đa dạng của trạng thái tinh thần và cảm xúc con người. Các nghiên cứu có thể mở rộng từ 5 cảm xúc cơ bản ban đầu thành một tập lớn hơn, có thể bao gồm cảm xúc phức tạp, đa chiều và đặc biệt hơn.

Sử dụng dữ liệu đa nguồn

Sử dụng dữ liệu từ nhiều nguồn khác nhau như mạng xã hội, diễn đàn trực tuyến, và các nền tảng khác để cải thiện độ đa dạng và đại diện của dữ liệu cảm xúc.

Phát triển công nghệ dựa trên trí tuệ nhân tạo

Nghiên cứu về sự kết hợp giữa các phương pháp cảm xúc và các công nghệ AI khác, như học tăng cường và tự học, để tạo ra các hệ thống thông minh có khả năng tương tác tốt hơn với con người và hiểu rõ hơn về cảm xúc của họ.

Ứng dụng thực tiễn cho doanh nghiệp/ người bán hàng

Dự đoán cảm xúc của khách hàng trong các tương tác trực tuyến mang lại nhiều ứng dụng quan trọng cho doanh nghiệp và người bán hàng. Việc này giúp tối ưu hóa chiến lược quảng cáo, quản lý dịch vụ khách hàng, đánh giá hiệu suất sản phẩm, và phân loại khách hàng dựa trên phản hồi. Nó cũng hỗ trợ trong việc tối ưu hóa nội dung truyền thông và đo lường hiệu suất chiến lược truyền thông, đồng thời mang lại cơ hội để doanh nghiệp điều chỉnh chiến lược dựa trên sự thay đổi của cảm xúc khách hàng qua thời gian.

TÀI LIỆU THAM KHẢO

What is LSTM? Introduction to Long Short-Term Memory <https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/>

Slide bài giảng môn xử lý ngôn ngữ tự nhiên

<https://onedrive.live.com/?authkey=%21ANB%5FtsCZjUVIKG4&id=13E7A03477E24249%21907494&cid=13E7A03477E24249>

What is MaxEnt? <https://support.bccvl.org.au/support/solutions/articles/6000083216->

[maxent#:~:text=Maxent%2C%20which%20stands%20for%20maximum,environmental%20variables%20of%20known%20locations.](https://support.bccvl.org.au/support/solutions/articles/6000083216-maxent#:~:text=Maxent%2C%20which%20stands%20for%20maximum,environmental%20variables%20of%20known%20locations.)

BẢNG ĐÁNH GIÁ THÀNH VIÊN

Họ và tên	MSSV	Công việc	Mức độ hoàn thành
Đặng Châu Kỳ	31211027647	Chương 1 - Tổng quan Chương 2 - Mô hình LSTM cho bài toán Chương 3.2, 3.3 - Xây dựng mô hình Chương 3.3 - Ứng dụng bài cho bài toán Chương 5 - Hạn chế	100%
Bùi Lê Khang	31211027644	Chương 2 - Mô hình decision tree kết hợp với Ensemble learning (Adaboost) Chương 3.1 - Bộ dữ liệu Chương 4 - Phân tích đánh giá Chương 5 - Các kết quả đạt được	100%
Phạm Minh Phước	31211027663	Chương 2 - Các phương pháp tiền xử lý dữ liệu Chương 2 - Mô hình MaxEnt cho bài toán Chương 3.2 - Đánh giá và nhận xét các mô hình, so sánh mô hình Chương 4 - Phân tích đánh giá Chương 5 - Hướng phát triển	100%

PHỤ LỤC

Link Github: https://github.com/chaukydang/NLP_Project