BỘ GIÁO DỤC VÀ ĐÀO TẠO
ĐẠI HỌC KINH TẾ HỒ CHÍ MINH
KHOA CÔNG NGHỆ THÔNG TIN KINH DOANH

ĐỒ ÁN MÔN HỌC

XỬ LÝ NGÔN NGỮ TỰ NHIÊN

Đề tài: *PHÂN LOẠI ĐÁNH GIÁ CỦA KHÁN GIẢ VỀ BỘ PHIM*

Giảng viên: Đặng Ngọc Hoàng Thành

Mã lớp học phần: 23C1INF50907602

Thành viên: Trịnh Uyên Chi - 31211020738
Nguyễn Hoàng Hà My - 31211023184
Huỳnh Thị Ngọc Trâm - 31211027679

Thành phố Hồ Chí Minh , ngày 23 tháng 12 năm 2023

Link github: Sentiment-Analysis-with-IMDB-Dataset/NLP.ipynb at main · chiiko1403/Sentiment-Analysis-with-IMDB-Dataset · GitHub

Contents

CHƯƠNG 1: GIỚI THIỆU TỔNG QUAN

1. Xác định đề tài

Trong những năm gần đây với sự phát triển nhanh chóng của nền điện ảnh trên nhiều quốc gia và việc xem phim trực tuyến ngày càng phổ biến. Và sự tràn lan về phim ảnh cũng khiến khán giả hình thành thói quen đọc những đánh giá trước khi chọn xem bất kỳ bộ phim nào. Đồng thời, trong thị trường cạnh tranh gay gắt và nhiều vấn đề nhạy cảm, nhà sản xuất cũng cần biết được những ý nghĩa, thông điệp mà bộ phim truyền tải tới khán giả có đúng hay chưa, suy nghĩ của khán giả về bộ phim như thế nào cũng là những yếu tố ảnh hưởng tới mức độ thành công của bộ phim.

Nhận thức được tầm quan trọng của việc đánh giá của khán giả, nhóm tiến hành phân tích cảm xúc phân loại dựa trên các đánh giá ấy. Cụ thể, dựa trên bộ dữ liệu "IMDB dataset" có khoảng 50.000 đánh giá của khán giả về bộ phim.

2. Mục tiêu nghiên cứu

- Nhà sản xuất hiểu rõ về nhu cầu, cảm xúc của khán giả từ đó cải thiện những bộ phim có chủ đề tương tự.
- Theo dõi xu hướng cảm nhận của khán giả
- Hỗ trợ những khán giả khác trong quá trình chọn lựa phim
- Phân tích và phân lớp dựa trên các đánh giá của khách hàng về bộ phim
- Đánh giá hiệu suất của các mô hình.

3. Phương pháp nghiên cứu

- Biểu diễn văn bản thành vector dựa vào CountVectorizer và TF-IDF Vectorizer
- Áp dụng mô hình học máy Naive Bayes, mô hình MaxEnt Logistic Regression và mô hình học sâu Long short-term memory (LSTM) để phân lớp các đánh giá

4. Công cụ sử dụng

- Ngôn ngữ lập trình Python

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT VÀ CÁC VẤN ĐỀ LIÊN QUAN

1. Các thuật toán

a. Naive Bayes

Naive Bayes là một thuật toán phân lớp hoạt động dựa trên định lý Bayes trong xác suất thống kê. Trong đồ án này, nhóm sử dụng mô hình Multinomial Naive Bayes, một mô hình thường được áp dụng trong phân loại văn bản, để phân loại đánh giá tích cực hay tiêu cực của người dùng về các bộ phim. Đặc trưng đầu vào của mô hình này chính là tần suất xuất hiện của từ trong văn bản của bộ dữ liệu. Đặc biệt, mô hình này còn phù hợp cho các tác vụ phân loại văn bản với số lượng từ lớn.

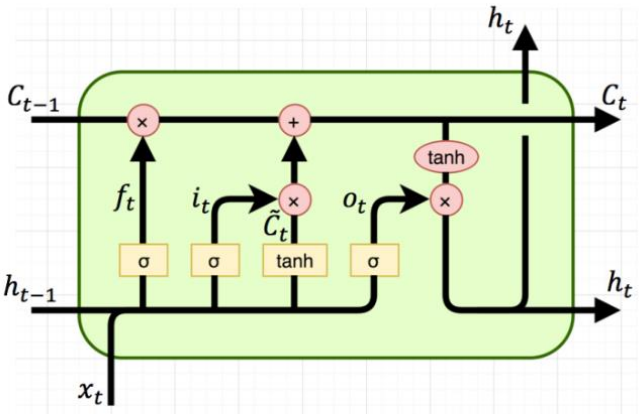
b. Logistic Regression

Logistic Regression là một mô hình học máy phân loại tuyến tính, sử dụng hàm logistic (sigmoid) để dự đoán xác suất thuộc về một lớp. Mô hình này phù hợp với các bài toán phân loại nhị phân trong đó mục tiêu là phân loại mẫu vào một trong hai lớp, hoạt động hiệu quả với các bài toán có số lượng mẫu lớn và không yêu cầu mức độ phức tạp cao, do đó nhóm sử dụng trong đồ án này, cụ thể là cho việc phân loại đánh giá tích cực hay tiêu cực dựa trên bộ dữ liệu.

c. Long short-term memory

Trước khi bàn đến LSTM, ta phải hiểu cơ chế hoạt động của RNN (Recurrent Neural Network). RNN là một loại thuật toán học sâu có giám sát. Ở đây, các nơron được kết nối với nhau theo thời gian. Ý tưởng đằng sau RNN là ghi nhớ thông tin nào đã có trong các nơron trước đó để các nơron này có thể truyền thông tin đến chính chúng trong tương lai để phân tích thêm. Điều đó có nghĩa là thông tin từ một phiên bản thời gian cụ thể (t1) được sử dụng làm đầu vào cho phiên bản thời gian tiếp theo (t2). Mô hình Long Short-Term Memory (LSTM) là một dạng của mô hình học sâu (deep learning) thuộc loại RNN. LSTM được thiết kế để giải quyết vấn đề biến mất gradient trong việc học từ dữ liệu chuỗi dài với khả năng lưu giữ thông tin trình tự trong "bộ nhớ" của nó.

Mô hình LSTM có một số đặc điểm chính như cổng quên (Forget Gate), cổng đầu vào (Input Gate), cổng đầu ra (Output Gate), trạng thái ẩn (Cell State). Nhờ vào khả năng học và duy trì mối quan hệ lâu dài trong dữ liệu chuỗi, LSTM đã trở thành một công cụ mạnh mẽ trong lĩnh vực học máy, đặc biệt là khi có liên quan đến xử lý ngôn ngữ tự nhiên (NLP) và các bài toán chuỗi thời gian. Nó thường được sử dụng trong các ứng dụng như máy dịch, tóm tắt văn bản, dự đoán chuỗi thời gian, và nhiều ứng dụng khác đòi hỏi khả năng xử lý thông tin chuỗi.



2. Các phương pháp biểu diễn văn bản thành dạng vector

a. CountVectorizer

CountVectorizer là một kỹ thuật tiền xử lý văn bản thường được sử dụng trong các tác vụ xử lý ngôn ngữ tự nhiên (NLP) để chuyển đổi một tập hợp tài liệu văn bản thành dạng biểu diễn số. Nó là một phần của thư viện scikit-learn, một thư viện máy học phổ biến trong Python. CountVectorizer hoạt động bằng cách mã hóa dữ liệu văn bản và đếm số lần xuất hiện của từng token. Sau đó, nó tạo ra một ma trận trong đó các hàng đại diện cho tài liệu và các cột đại diện cho token. Các giá trị ô cho biết tần suất của từng token trong mỗi tài liệu.

b. TF-IDF

Trong khai phá dữ liệu văn bản, thuật ngữ TF-IDF (Term Frequency - Inverse Document Frequency) là một phương thức thống kê được mọi người biết đến phổ biến nhất để xác định độ quan trọng của một từ trong đoạn văn bản trong một tập nhiều đoạn văn bản khác nhau. Nó thường được sử dụng như một trọng số trong việc khai phá dữ liệu văn bản. TF-IDF chuyển đổi dạng biểu diễn văn bản thành dạng không gian vector (VSM), hoặc thành những vector thưa thớt.

CHƯƠNG 3: ĐỀ XUẤT MÔ HÌNH

1. Ứng dụng mô hình phân lớp

a. Xây dựng và đánh giá mô hình bằng các thuật toán máy học

Để sử dụng mô hình máy học ta sử dụng thư viện Scikit-learn (Sklearn) để tiến hành chia tập dữ liệu thành hai phần là tập huấn luyện và tập kiểm tra. Để vector hóa văn bản ta sử dụng hai phương pháp là TF-IDF và CountVectorizer. Sau cùng, để xây dựng một hệ thống phân loại, chúng ta cần một chỉ số để đánh giá hiệu suất của mô hình đã xây dựng. Nhóm đã tính chỉ số accuracy cho những mô hình được xây dựng sau đó trực quan hóa về các dự đoán của từng mô hình trên tập huấn luyện và tập kiểm tra.

b. Xây dựng và đánh giá mô hình bằng cách thuật toán học sâu

Xây dựng mô hình BiLSTM bằng TensorFlow/Keras để thực hiện phân loại văn bản. Mô hình này chứa một lớp LSTM hoạt động ở cả hai chiều (trước và sau) để học thông tin từ cả hai hướng của dữ liệu chuỗi. Điều này giúp mô hình hiểu rõ hơn ngữ cảnh xung quanh từng từ trong văn bản, đặc biệt hữu ích trong các tác vụ liên quan đến ngôn ngữ tự nhiên. Thực hiện huấn luyện mô hình bằng việc sử dụng tập dữ liệu đã được xử lý, tùy chỉnh các tham số mô hình như số lớp, số nơ-ron, hệ số học và số vòng lặp để tối ưu hóa độ chính xác của mô hình. Cuối cùng, ta thực hiện kiểm tra và đánh giá mô hình sử dụng tập dữ liệu kiểm tra để đánh giá độ chính xác của mô hình

2. Kết quả mong muốn dự báo đạt được

Mô hình Machine Learning sẽ so sánh về mức độ chính xác giữa dữ liệu thực tế phân loại đánh giá tích cực và tiêu cực đã được gán nhãn cho thấy và dữ liệu dự báo khi được áp dụng mô hình máy học của nhóm khi phân loại đánh giá của khán giả. Trong khi mô hình Deep Learning sẽ thu được các từ mang ý nghĩa phân loại đánh giá thường xuyên dùng nhất từ đó dự đoán những thể loại phim có thể được yêu thích. Deep Learning sẽ lược bỏ một số thao tác thủ công và thu được kết quả tốt hơn trên bộ dữ liệu lớn hơn.

CHƯƠNG 4: TRIỂN KHAI MÔ HÌNH VÀO BỘ DỮ LIỆU VÀ ĐÁNH GIÁ

1. Thực hiện tiền xử lý

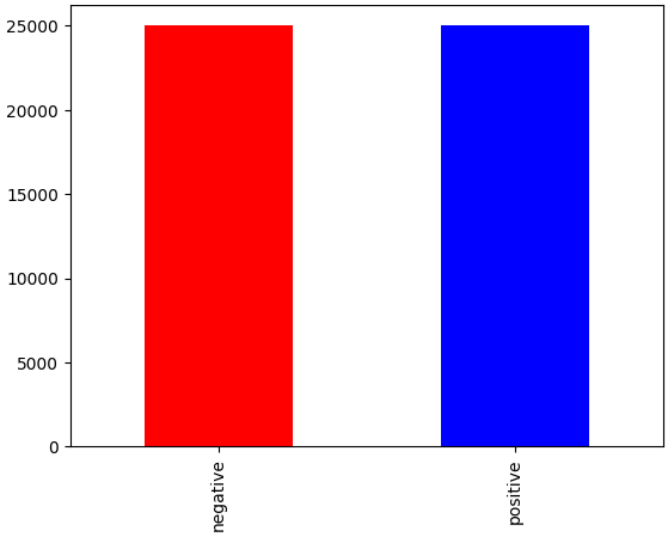
- Kiểm tra trùng lặp: có tổng cộng 418 dòng trùng lặp, tiến hành loại bỏ những dòng đó để không ảnh hưởng đến hiệu quả các mô hình

```
# Check if there are any duplicates
dup = data.duplicated().sum()
print(f'Có {dup} giá trị bị trùng lặp')

# Drop the duplicates
data = data.drop_duplicates(subset = 'review')
data
```

- Kích thước và phân phối bộ dữ liệu

```
# Data distribution
rating_counts = data['sentiment'].value_counts()
print(rating_counts)

data['sentiment'].value_counts().sort_index().plot.bar(color=['red',
'blue'])
plt.title('Biểu đồ thể hiện phân phối giữa 2 nhóm trong biến
sentiment')
plt.xticks(rotation = 0)
# The shape of data
data.shape
```

```
(50000, 2)

positive    25000
negative    25000
```



Bộ dữ liệu gồm có 50000 dòng tương ứng với 50000 đánh giá của khán giả về bộ phim. Trong đó có tỉ lệ đánh giá tích cực và tiêu cực là ngang bằng nhau, không có những ý kiến trung lập (neutral). Tức bộ dữ liệu đang được cân bằng tỉ lệ giữa 2 lớp sentiment là Positive và Negative.

- Tạo danh sách các stop word:

```
# Stopwords in English
stop_words = stopwords.words('english')
len(stop_words)
```

Kết quả: 179

```
# Remove the negative words in stopwords
negative_words=['no','not',"don't","aren't","couldn't",
                "didn't","doesn't","hadn't","hasn't","haven't",
                "isn't","mightn't","mustn't","needn't","shouldn't",
                "wasn't","weren't","won't","wouldn't"]
for negative_word in negative_words:
    if negative_word in stop_words:
        stop_words.remove(negative_word)

print("Stopwords sau khi loại bỏ những từ negative:",len(stop_words))
```

Sau khi loại bỏ những từ negative thì stopwords có 160 từ

Stopwords là các từ phổ biến trong câu, được xuất hiện nhiều lần nhưng không mang lại ý nghĩa quan trọng để phân tích về cảm xúc của khán giả về bộ phim. Tuy nhiên thì trong một số trường hợp như những từ mang ý nghĩa phủ định negative có thể bị đưa vào danh sách stop word nhưng thực tế đó là những từ trọng tâm của câu, có ý nghĩa quan trọng và cần được giữ lại. Ví dụ như là “This film wasn’t too good”. Do đó nhóm tiến hành loại bỏ những từ negative ra khỏi danh sách stop word

- Xử lý văn bản và loại bỏ các ký tự không mong muốn

```
REPLACE_BY_SPACE_RE = re.compile('[/(){}—[]|@,;‘?|!-ll–’-]')

def clean_text(sample):
    if isinstance(sample, str):  # Check if sample is a string or not
        sample = sample.lower()
        sample = sample.replace("<br /><br />", "")
        sample = REPLACE_BY_SPACE_RE.sub(' ', sample)
        sample = re.sub("[^a-z]+", " ", sample)
        sample = sample.split(" ")
        sample = [word for word in sample if word not in stop_words]
        sample = " ".join(sample)
        return sample
    else:
        return ''
```

```
# Process text, remove unwanted characters and unnecessary words
data['review'] = data['review'].apply(clean_text)
data.head()
```

	review	sentiment
0	one reviewers mentioned watching oz episode ho...	positive
1	wonderful little production filming technique ...	positive
2	thought wonderful way spend time hot summer we...	positive
3	basically family little boy jake thinks zombie...	negative
4	petter mattei love time money visually stunnin...	positive

Tiến hành xử lý văn bản với các tác vụ sau

- Chuyển đổi văn bản thành chữ thường
- Thay thế các ký tự trong biểu thức regular expression thành các khoảng trắng
- Loại bỏ các ký tự không phải là chữ cái
- Tách văn bản thành các danh sách
- Loại bỏ các stop word khỏi danh sách các từ rồi nối lại thành chuỗi văn bản mới

Áp dụng hàm clean_text với các tác vụ kể trên cho cột review nhằm tiến hành chuẩn hóa văn bản, loại bỏ các yếu tố như kí tự đặc biệt, stop words. Điều đó giúp cải thiện và đồng nhất văn bản, nâng cao hiệu quả khi áp dụng các mô hình học máy.

- Tạo hàm để loại bỏ biểu tượng cảm xúc

```
def remove_emoji(text):
    return re.sub(r'[^\w\s,]', '', str(text)) if isinstance(text, str)
else text

# Apply remove_emoji function for 'review' column in DataFrame
data['review'] = data['review'].apply(remove_emoji)
```

Vì đây là dữ liệu về những đánh giá của khán giả nên sẽ có những trường hợp bao gồm cả những biểu tượng cảm xúc (emoji), điều đó khiến cho dữ liệu không đồng

nhất và ảnh hưởng tới hiệu quả mô hình. Nên nhóm tiến hành tạo hàm remove_emoji để loại bỏ những biểu tượng đó trong văn bản, và áp dụng cho cột review

- Tiến hành tách từ đối với dữ liệu cột review

```
# Tokenize and lemmatize text
w_tokenizer = nltk.tokenize.WhitespaceTokenizer()
lemmatizer = nltk.stem.WordNetLemmatizer()
def lemmatize_text(text):
    st = ""
    for w in w_tokenizer.tokenize(text):
        st = st + lemmatizer.lemmatize(w) + " "
    return st
data['review'] = data.review.apply(lemmatize_text)
data
```

	review	sentiment
0	one reviewer mentioned watching oz episode hoo...	positive
1	wonderful little production filming technique ...	positive
2	thought wonderful way spend time hot summer we...	positive
3	basically family little boy jake think zombie ...	negative
4	petter mattei love time money visually stunnin...	positive
...	...	...
49995	thought movie right good job creative original...	positive
49996	bad plot bad dialogue bad acting idiotic direc...	negative
49997	catholic taught parochial elementary school nu...	negative
49998	going disagree previous comment side maltin on...	negative
49999	no one expects star trek movie high art fan ex...	negative

50000 rows × 2 columns

Sử dụng hàm từ thư viện nltk để tách từ dựa vào khoảng trắng (tokenize) và thực hiện lemmatize xử lý các ký tự trên cột review.

- Sử dụng hàm WhitespaceTokenizer() để tách từ dựa trên khoảng trắng. Và hàm WordNetLemmatizer() để tìm từ gốc của các từ trong văn bản dựa vào từ điển WordNet.
- Khởi tạo hàm lemmatize_text để chuẩn hóa các từ về dạng gốc giá trị trong cột review.

2. Xây dựng mô hình học máy và đánh giá

a. Chia tập dữ liệu

```
# Input
x = data['review'].values
# Output
y = data['sentiment'].values

# Train and test set
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size =
0.2, random_state = 0, stratify = y)

print(f"Kích thước của dữ liệu huấn luyện: {X_train.shape[0]}.")
print(f"Kích thước của dữ liệu thử nghiệm: {X_test.shape[0]}.")
```

```
Kích thước của dữ liệu huấn luyện: 39665.
Kích thước của dữ liệu thử nghiệm: 9917.
```

Sử dụng thư viện Scikit-learn để chia tập dữ liệu thành tập training set và test set với tỉ lệ (80/20). Trong đó x dữ liệu đầu vào là nội dung của cột review, y kết quả đầu ra là giá trị cột sentiment.

b. Xây dựng mô hình học máy Naive Bayes

- Nhóm lựa chọn mô hình học máy là Naive Bayes để phân loại đánh giá của bộ dữ liệu. Thực hiện vector hóa văn bản dựa trên phương pháp trong xử lý ngôn ngữ tự nhiên là CountVectorizer và TF-IDF

Count vectorizer:

- Phương pháp Count Vectorizer: phương pháp này sẽ đếm số lần xuất hiện của từ trong văn bản. Nó có ưu điểm là đơn giản và hiệu quả khi mà tập trung chủ yếu vào những từ phổ biến. Song, nhược điểm của nó là sẽ khiến cho trọng số của những từ phổ biến cao hơn (ví dụ a/an/the) nhưng mức độ quan trọng của nó với việc phân loại văn bản thấp. Ngược lại, những từ ít xuất hiện nhưng quan trọng thì có thể có trọng số thấp, cũng ảnh hưởng tới kết quả phân loại của mô hình

Tạo pipeline để vector hóa văn bản và tìm tham số alpha tối ưu của mô hình Naive Bayes bằng cách sử dụng GridSearchCV.

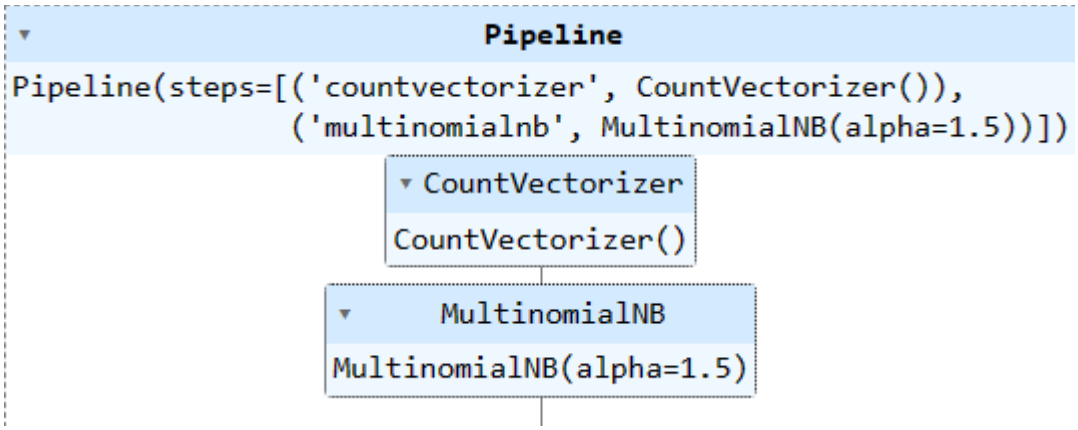
```
# Make pipeline with CountVectorizer and MultinomialNB
pipeline = make_pipeline(CountVectorizer(), MultinomialNB())
# Experience with different alpha values
parameters = {'multinomialnb__alpha': [0.1, 0.5, 1.0, 1.5, 2.0]}
# Perform GridSearchCV
grid = GridSearchCV(pipeline, param_grid=parameters, cv=5)
grid.fit(X_train, y_train)

# Get the best alpha value
best_alpha_cv = grid.best_params_['multinomialnb__alpha']
print(f'Best alpha with Count Vectorizer: {best_alpha_cv}')
```

Kết quả:

```
Best alpha with Count Vectorizer: 1.5
```

- Áp dụng giá trị alpha tối ưu vừa tìm được vào mô hình Naive Bayes

```
# Make pipeline with CountVectorizer and MultinomialNB with optimized alpha
nb_model_cv = make_pipeline(CountVectorizer(), MultinomialNB(alpha=best_alpha_cv))

# Train the model with training dataset
nb_model_cv.fit(X_train, y_train)
```



- Sau khi hoàn tất xây dựng mô hình, đánh giá mô hình thông qua các chỉ số như accuracy, precision, recall,... thông qua classification_report trong thư viện scikit learn

```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt
```

```
# Predict on test set
nb_predictions = nb_model_cv.predict(X_test)

# Calculate and print the evaluation scores
accuracy = accuracy_score(y_test, nb_predictions)
print(f'Accuracy: {accuracy:.3f}')

# Print classification report
print('Classification Report:')
print(classification_report(y_test, nb_predictions))
```

Kết quả:

```
Accuracy: 0.862
Classification Report:
              precision    recall  f1-score   support

    negative       0.85      0.88      0.86      4940
    positive       0.88      0.84      0.86      4977

    accuracy                           0.86      9917
   macro avg       0.86      0.86      0.86      9917
weighted avg       0.86      0.86      0.86      9917
```

Với kết quả như trên, có thể thấy mô hình Naive Bayes đạt được mức độ chính xác tổng thể là 86,2%, con số này cho thấy mô hình có hiệu quả khá tốt trên bộ dữ liệu này.

Chỉ số Precision và Recall đều nằm trong khoảng 0.85 đến 0.88 con số tương đối cao. Thông qua đó có thể thấy mô hình có khả năng phân biệt tốt giữa 2 lớp negative và positive trong biến sentiment

TF-IDF:

- Phương pháp TF-IDF: có phần tốt hơn CountVectorizer vì không chỉ tập trung vào tần suất của các từ có trong văn bản mà cũng quan tâm đến tầm quan trọng của từ đó.

Tương tự với phương pháp trên thì phương pháp TF-IDF cũng thực hiện tạo pipeline chuyển văn bản thành vector và tìm alpha tối ưu nhất thông qua hàm GridSearchCV

```
# Make pipeline with TfidfVectorizer and MultinomialNB
pipeline = make_pipeline(TfidfVectorizer(), MultinomialNB())
# Experience with different alpha values
parameters = {'multinomialnb__alpha': [0.1, 0.5, 1.0, 1.5, 2.0]}
# Perform GridSearchCV
grid = GridSearchCV(pipeline, param_grid=parameters, cv=5)
grid.fit(X_train, y_train)
```

```
# Get the best alpha value
best_alpha_tf = grid.best_params_['multinomialnb__alpha']
print(f'Best alpha with TF-IDF Vectorizer: {best_alpha_tf}')
```

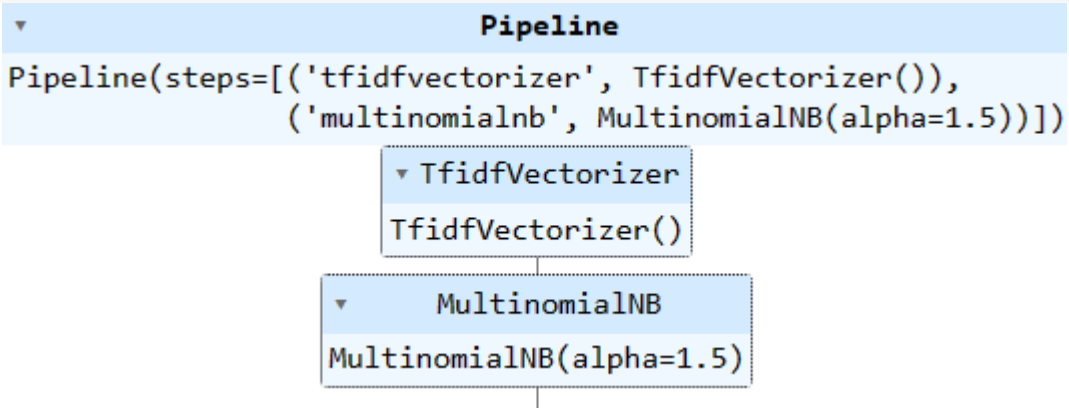
Kết quả:

```
Best alpha with TF-IDF Vectorizer: 1.5
```

Có thể thấy cả hai phương pháp vector hóa văn bản là Count Vectorizer và TF-IDF đều có chỉ số alpha tối ưu là 1.5

- Xây dựng mô hình với chỉ số alpha tối ưu theo phương pháp TF-IDF

```
# Make pipeline with TF-IDF and MultinomialNB with optimized alpha
nb_model_tf = make_pipeline(TfidfVectorizer(),
MultinomialNB(alpha=best_alpha_tf))

# Train the model with training dataset
nb_model_tf.fit(X_train, y_train)
```



- Độ chính xác (accuracy) tổng quan của mô hình với phương pháp vector hóa này

```
nb_accuracy = nb_model_tf.score(X_test, y_test)
print(f'Độ chính xác của mô hình Naive Bayes với TF-IDF:
{round(nb_accuracy*100)}%')
```

Kết quả:

```
Độ chính xác của mô hình Naive Bayes với TF-IDF: 87%
```

```
# Predict on test set
nb_pre = nb_model_tf.predict(X_test)
# Calculate and print the evaluation scores
accuracy_tf = accuracy_score(y_test, nb_pre)
print(f'Accuracy: {accuracy_tf:.3f}')

# Print classification report
print('Classification Report:')
print(classification_report(y_test, nb_pre))
```

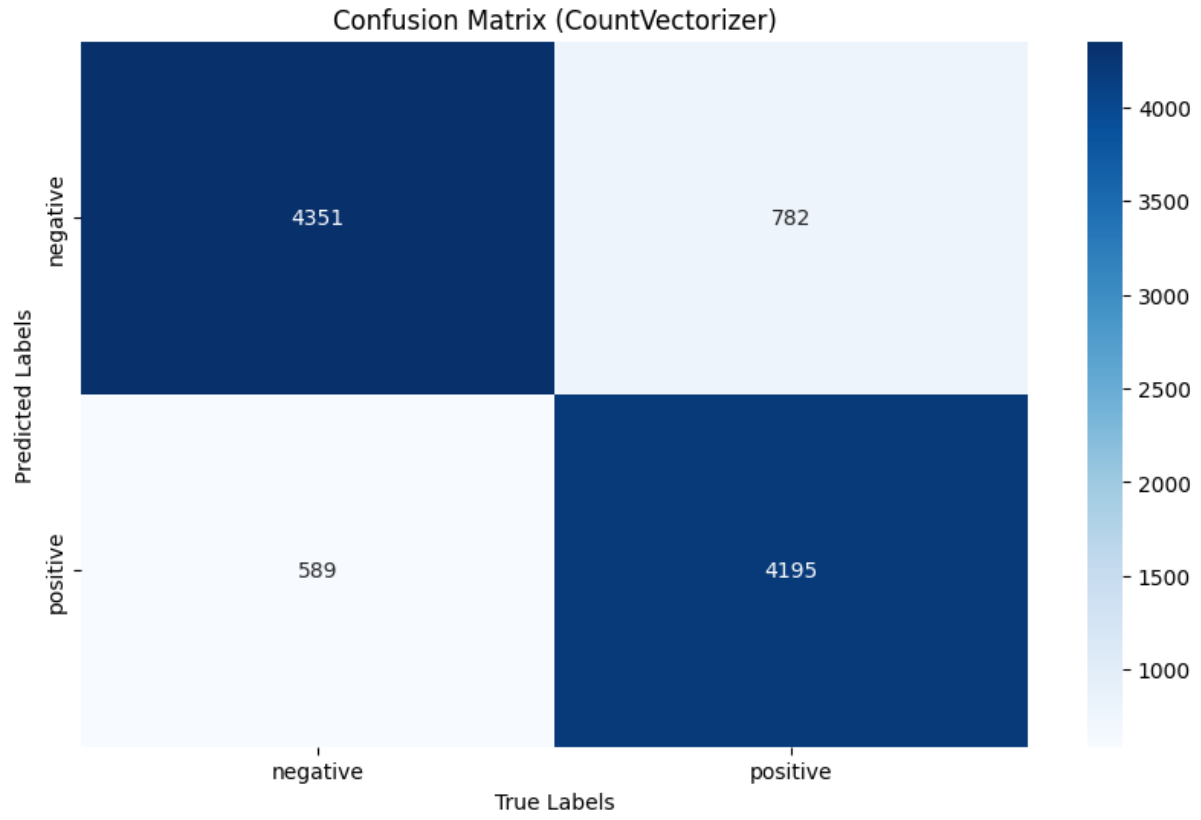
```
Accuracy: 0.868
Classification Report:
              precision    recall  f1-score   support

    negative       0.86      0.88      0.87      4940
    positive       0.88      0.85      0.87      4977

    accuracy                           0.87      9917
   macro avg       0.87      0.87      0.87      9917
weighted avg       0.87      0.87      0.87      9917
```

Dựa vào kết quả trên cho thấy độ chính xác tổng thể của mô hình khi chạy trên bộ dữ liệu là khoảng 86.8%. Tổng quát mô hình hoạt động khá tốt và hiệu suất ổn định trên cả hai lớp của biến sentiment là "negative" và "positive" với các chỉ số precision, recall và f1-score đều cao ở mức 0.86-0.88.

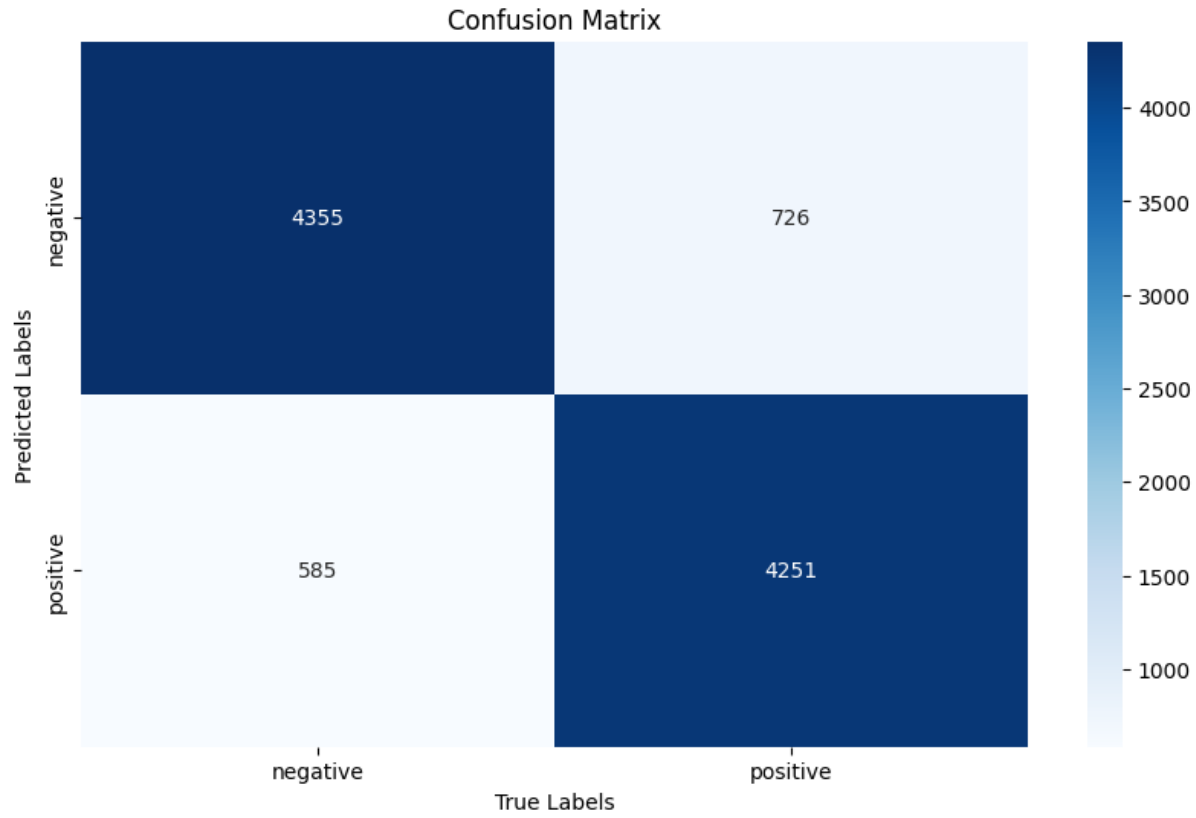
c. Đánh giá kết quả

- Các chỉ số đánh giá mô hình ở phần trên cho thấy mô hình hoạt động khá hiệu quả. Tuy nhiên để đưa ra cái nhìn chi tiết hơn thì ta tiến hành xem xét các ma trận nhầm lẫn của từng phương pháp trên mô hình Naive Bayes

```
# Confusion matrix
import seaborn as sns
matrix = confusion_matrix(nb_predictions,y_test)
matrix_df = pd.DataFrame(matrix, index=np.unique(y),
columns=np.unique(y))
plt.figure(figsize=(10,6))
plt.tight_layout(pad=2.0)
sns.heatmap(matrix_df, annot=True, cmap="Blues", fmt="d")
plt.xlabel("True Labels")
plt.ylabel("Predicted Labels")
plt.title("Confusion Matrix (CountVectorizer)")
plt.show()
```




```
# Confusion matrix
import seaborn as sns
matrix_tf = confusion_matrix(nb_pre,y_test)
matrix_tf      =      pd.DataFrame(matrix_tf,      index=np.unique(y),
columns=np.unique(y))
plt.figure(figsize=(10,6))
plt.tight_layout(pad=2.0)
sns.heatmap(matrix_tf, annot=True, cmap="Blues", fmt="d")
plt.xlabel("True Labels")
plt.ylabel("Predicted Labels")
plt.title("Confusion Matrix")
plt.show()
```



Nhìn vào ma trận nhầm lẫn có thể thấy mô hình Naive Bayes hoạt động khá hiệu quả. Đều có mức độ sai lầm loại I và sai lầm loại II trên 500 trường hợp nhưng đó không phải là con số đáng kể, và mức độ nghiêm trọng cũng thấp trong trường hợp đây chỉ là phân loại đánh giá phim. Tuy nhiên có thể thấy, với phương pháp TF-IDF lại có mức độ chính xác cao hơn so với phương pháp còn lại là Count Vectorizer. Mặc dù không có sự chênh lệch quá lớn nhưng cũng cho thấy mức độ hiệu quả của phương pháp này.

3. Xây dựng mô hình trong nhóm MaxEnt và đánh giá

a. Xây dựng mô hình MaxEnt Logistic Regression

Ở mô hình này, nhóm cũng sẽ xây dựng mô hình dựa trên 2 phương pháp biểu diễn từ thành vector là Count Vectorzier và TF-IDF Vectorizer như mô hình học máy Naive Bayes trên.

Count Vectorizer:

Để tìm được tham số C tối ưu nhất cho mô hình Logistic Regression, ở đây nhóm tạo pipeline với CountVectorizer() và LogisticRegression với max_iter là 1000 sau đó dùng GridSearchCV với các giá trị C khác nhau.

```
# Make pipeline with CountVectorizer and Logistic Regression
pipeline = make_pipeline(CountVectorizer(),
LogisticRegression(max_iter=1000))

# Experience with different values of C
parameters = {'logisticregression__C': [0.001, 0.01, 0.1, 1, 10, 100]}
```

```
# Perform GridSearchCV
grid = GridSearchCV(pipeline, param_grid=parameters, cv=5)
grid.fit(X_train, y_train)

# Get the best C value
best_C_cv = grid.best_params_['logisticregression__C']
print(f'Best C with Count Vectorizer: {best_C_cv}')
```

Kết quả cho thấy giá trị C tốt nhất đối với mô hình này là 0.1

```
Best C with Count Vectorizer: 0.1
```

Sau đó ta tiến hành huấn luyện mô hình với tham số C lấy được ở trên.

```
# Train model with best C value
lr_model_cv = make_pipeline(CountVectorizer(),
LogisticRegression(C=best_C_cv, max_iter=1000))
lr_model_cv.fit(X_train, y_train)
```

TF-IDF Vectorizer:

Cũng giống như trên, tiến hành tạo pipeline và GridSearchCV với các giá trị tham số C khác nhau để tối ưu hóa mô hình.

```
# Make pipeline with CountVectorizer and Logistic Regression
pipeline = make_pipeline(TfidfVectorizer(),
LogisticRegression(max_iter=1000))

# Experience with different values of C
parameters = {'logisticregression__C': [0.001, 0.01, 0.1, 1, 10, 100]}
# Perform GridSearchCV
grid = GridSearchCV(pipeline, param_grid=parameters, cv=5)
grid.fit(X_train, y_train)

# Get the best C value
best_C_tf = grid.best_params_['logisticregression__C']
print(f'Best C with TF-IDF Vectorizer: {best_C_tf}')
```

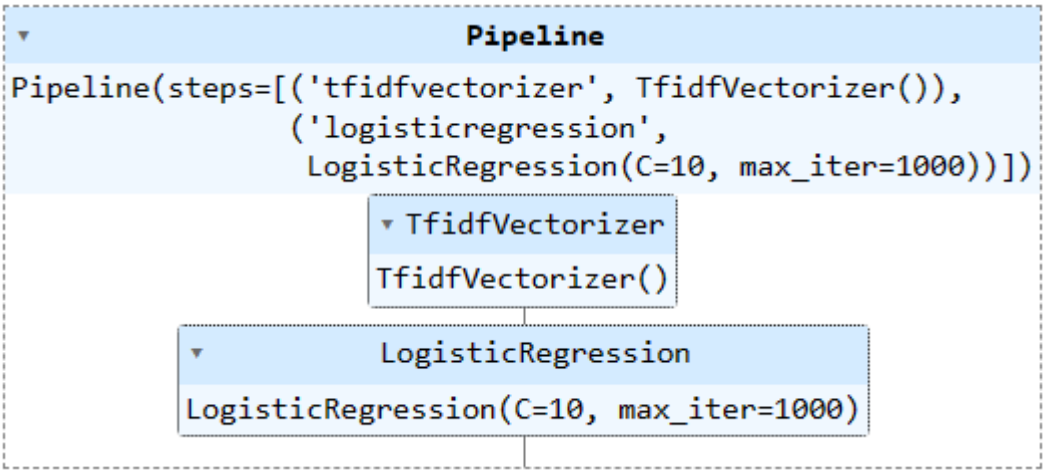
Kết quả cho thấy tham số C tốt nhất đối với mô hình này là 10, khác với mô hình sử dụng Count Vectorizer.

```
Best C with TF-IDF Vectorizer: 10
```

Sau đó tiến hành huấn luyện mô hình với tham số C tối ưu như trên.

```
# Train model with best C value
```

```
lr_model_tf = make_pipeline(TfidfVectorizer(),
LogisticRegression(C=best_C_tf, max_iter=1000))
lr_model_tf.fit(X_train, y_train)
```




```
# Predict on test set
predictions_tf = lr_model_tf.predict(X_test)

# Print classification report
print(classification_report(y_test, predictions_tf))
```

```
              precision    recall  f1-score   support

    negative       0.90      0.89      0.90      4940
    positive       0.89      0.90      0.90      4977

    accuracy                           0.90      9917
   macro avg       0.90      0.90      0.90      9917
weighted avg       0.90      0.90      0.90      9917
```

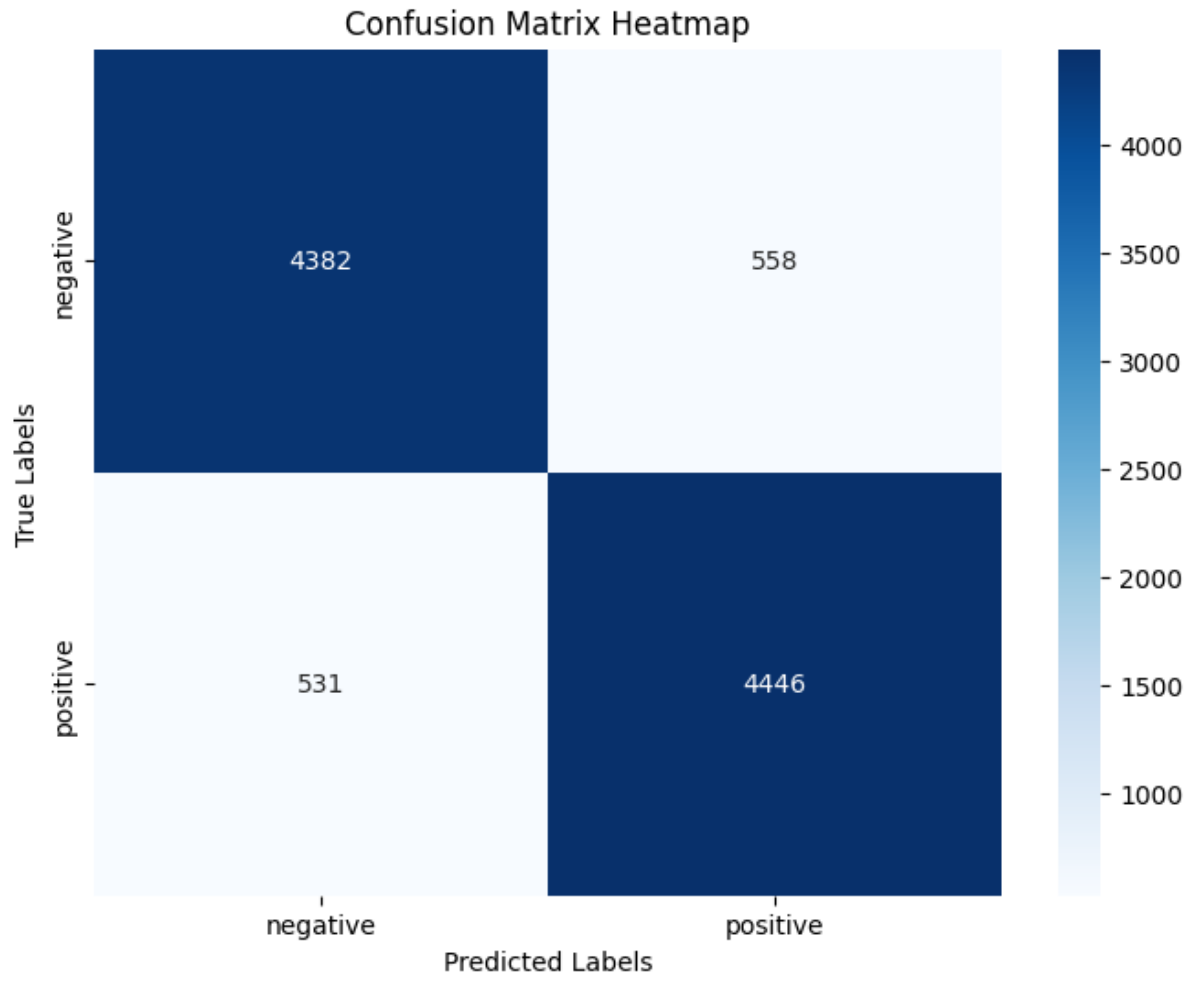
Mô hình Logistic với phương pháp vector hóa văn bản TF-IDF hoạt động tốt trên bộ dữ liệu này. Các chỉ số đánh giá có phần cao hơn so với mô hình Naive Bayes phía trên khi dao động từ 0.89-0.9.

b. Đánh giá

- Ta sẽ đánh giá cả 2 mô hình dựa trên confusion matrix.
- Count Vectorizer

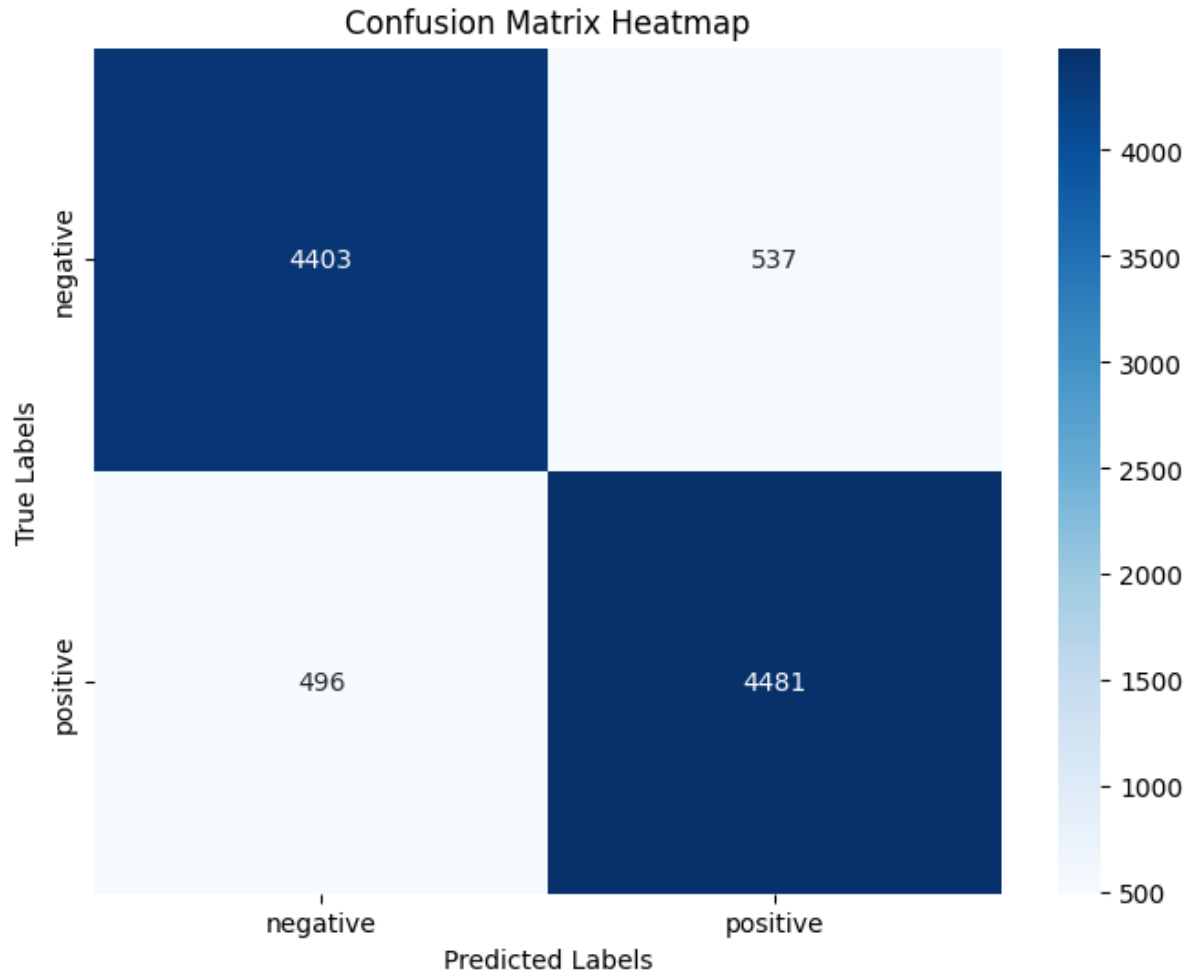
```
# Confusion matrix
conf_matrix = confusion_matrix(y_test, predictions_cv)

# Create heatmap from confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix,     annot=True,     fmt='d',     cmap='Blues',
xticklabels=lr_model_cv.classes_, yticklabels=lr_model_cv.classes_)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix Heatmap')
```

```
plt.show()
```



- TF-IDF

```
# Confusion matrix
conf_matrix = confusion_matrix(y_test, predictions_tf)

# Create heatmap from confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix,     annot=True,     fmt='d',     cmap='Blues',
xticklabels=lr_model_tf.classes_, yticklabels=lr_model_tf.classes_)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix Heatmap')
plt.show()
```



Hiệu suất mô hình Logistic Regression trên bộ dữ liệu cao hơn nên số trường hợp dự đoán đúng và sai cũng cao hơn so với Naive Bayes. Tuy nhiên thì giữa 2 phương pháp vector hóa văn bản thì TF-IDF vẫn đưa ra kết quả đúng hơn và hiệu quả hơn.

4.Xây dựng mô hình học sâu và đánh giá

2.3 Xây dựng và đánh giá mô hình bằng các thuật toán học sâu (Chi)

2.2.1 Xây dựng mô hình học sâu: BiLSTM

Trước khi xây dựng mô hình, chúng ta cần phải xác định các hyperparameters để tối ưu hóa mô hình trước. Ở đây ta sẽ xác định kích thước của từ vựng, các token được sử dụng cho các từ ngoài từ vựng, số chiều không gian nhúng, độ dài tối đa của câu sau chuẩn hóa, và việc đệm hay cắt sẽ được thực thi ở đầu hay cuối mỗi chuỗi.

```
# Hyperparameters of the model
vocab_size = 3000
oov_tok = ''
embedding_dim = 200
max_length = 200
padding_type='post'
```

```
trunc_type='post'
```

Kích thước của từ vựng được đặt ở giá trị 3000, mặc dù nhóm đã đếm được số lượng từ vựng lên đến hơn 90,000 từ nhưng giá trị này được giữ ở mức tương đối nhỏ để giảm độ phức tạp của mô hình và giữ cho quá trình huấn luyện mô hình diễn ra một cách nhanh chóng.

Tham số oov_tok xác định token được sử dụng cho các từ nằm ngoài từ vựng. Ở đây, một chuỗi trống được sử dụng, cho biết rằng các từ không có từ vựng sẽ được biểu thị bằng một token chuỗi trống.

`embedding_dim` thể hiện số chiều không gian của của việc nhúng cho mỗi từ. Kích thước cao hơn cho phép mô hình nắm bắt được các mối quan hệ phức tạp hơn giữa các từ nhưng cũng làm tăng thời gian tính toán. Giá trị 200 là lựa chọn phổ biến cho việc nhúng từ nhờ vào việc mang lại sự cân bằng tốt giữa việc nắm bắt ngữ nghĩa và hiệu quả tính toán.

Tham số tiếp theo đặt độ dài tối đa của chuỗi đầu vào. Các chuỗi dài hơn sẽ bị cắt bớt và các chuỗi ngắn hơn sẽ bị đệm. Giá trị 200 được chọn dựa trên số liệu thống kê (có thể thấy độ dài trung bình của đánh giá là khoảng 119.96 từ) nhằm mục đích nắm bắt đủ bối cảnh và tránh thời gian tính toán bị quá tải.

2 tham số cuối cùng xác định việc đệm hoặc cắt bớt xảy ra ở đầu hay cuối mỗi chuỗi. Trong trường hợp này, cả phần đệm và phần cắt bớt đều được đặt thành 'post', nghĩa là phần đệm sẽ được thêm vào hoặc việc cắt bớt sẽ xảy ra ở cuối chuỗi.

Sau khi xác định được các hyperparameters, đoạn code sau được dùng để chuẩn bị dữ liệu văn bản trước khi đưa vào mô hình học sâu LSTM bằng cách biểu diễn dữ liệu train và dữ liệu test dưới dạng sequence và pad sequence.

```
# tokenize sentences
tokenizer = Tokenizer(num_words = vocab_size, oov_token=oov_tok)
tokenizer.fit_on_texts(train_sentences)
word_index = tokenizer.word_index

# convert train dataset to sequence and pad sequences
train_sequences = tokenizer.texts_to_sequences(train_sentences)
train_padded = pad_sequences(train_sequences, padding='post',
maxlen=max_length)

# convert test dataset to sequence and pad sequences
test_sequences = tokenizer.texts_to_sequences(test_sentences)
test_padded = pad_sequences(test_sequences, padding='post',
maxlen=max_length)
```

Ở đây dữ liệu được chuyển về sequence vì mô hình được sử dụng trong keras là Sequential. Nhóm chọn Sequential cho BiLSTM vì Sequential thích hợp cho các mô hình với một chồng lớp đơn giản trong đó mỗi lớp có chính xác một tenxơ đầu vào và một tenxơ đầu ra. Bằng cách mã hóa các câu, chuyển đổi chúng thành chuỗi số nguyên và đệm các chuỗi để đảm bảo độ dài đồng đều, train_padded và test_padded đã sẵn sàng để sử dụng làm đầu vào cho BiLSTM để đào tạo và thử nghiệm mô hình.

Tiếp đến, nhóm tiến hành xây dựng mô hình bằng keras với phương thức Sequential() bao gồm các thuộc tính về các lớp như Embedding, Bidirectional, Dropout, và cuối cùng là Dense.

```
# model initialization
model = keras.Sequential([
    keras.layers.Embedding(vocab_size, embedding_dim, input_length =
max_length),
    keras.layers.Bidirectional(keras.layers.LSTM(64)),
    keras.layers.Dropout(0.2),
    keras.layers.Dense(24, activation='relu'),
    keras.layers.Dense(1, activation='sigmoid')])
# compile model
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

# model summary
model.summary()
```

Trong phương thức Sequential, đầu tiên ta sẽ tiến hành tạo 1 lớp chứa dữ liệu được nhúng, với kích thước từ vựng, số chiều không gian nhúng và độ dài đầu vào đã được xác định ở phần hyperparameters. Kế tiếp, ta tạo 1 lớp mô hình BiLSTM xử lý chuỗi đầu vào theo cả hướng tiến và lùi, thu thập thông tin từ cả bối cảnh trong quá khứ và tương lai. Nó có 64 đơn vị, là số lượng ô nhớ trong lớp LSTM. Lớp tiếp theo sẽ là Dropout với giá trị là 0.2 nhằm ngăn chặn vấn đề overfitting trong mô hình. Lớp Dense đầu tiên có 24 đơn vị và sử dụng hàm kích hoạt là đơn vị tuyến tính được chỉnh lưu (ReLU). Các lớp Dense là các lớp được kết nối đầy đủ và ReLU đưa tính phi tuyến tính vào mô hình, cho phép nó tìm hiểu các mối quan hệ phức tạp trong dữ liệu. Lớp Dense cuối cùng có một đơn vị duy nhất có hàm kích hoạt sigmoid. Lớp này được sử dụng trong các bài toán phân loại nhị phân. Nó tạo ra một đầu ra trong khoảng từ 0 đến 1, thể hiện dự đoán của mô hình về xác suất chuỗi đầu vào thuộc về lớp positive. Nhóm chọn những giá trị như trên dựa vào bài nghiên cứu “Analysis of hyperparameters in Sentiment Analysis of Movie Reviews using Bi-LSTM”.

Tiếp theo, phương thức compile() được sử dụng để cấu hình quá trình học tập của mô hình, bao gồm xác định hàm mất mát, trình tối ưu hóa và thang đo sẽ được sử dụng trong quá trình đào tạo. Ở đây hàm mất mát được xác định là Binary Cross Entropy, thường được sử dụng cho bài toán phân loại nhị phân, dùng để đo lường sự khác biệt giữa các nhãn thực tế và xác suất dự đoán. Thuật toán tối ưu hóa được chọn ở đây là Adam, một thuật toán phổ biến giúp điều chỉnh tốc độ học trong quá trình huấn luyện mô hình. Và cuối cùng, thang đo được dùng để đánh giá mô hình là 'accuracy', là độ chính xác của các dự đoán so với thực tế. Sau bước này, mô hình đã sẵn sàng để được huấn luyện.

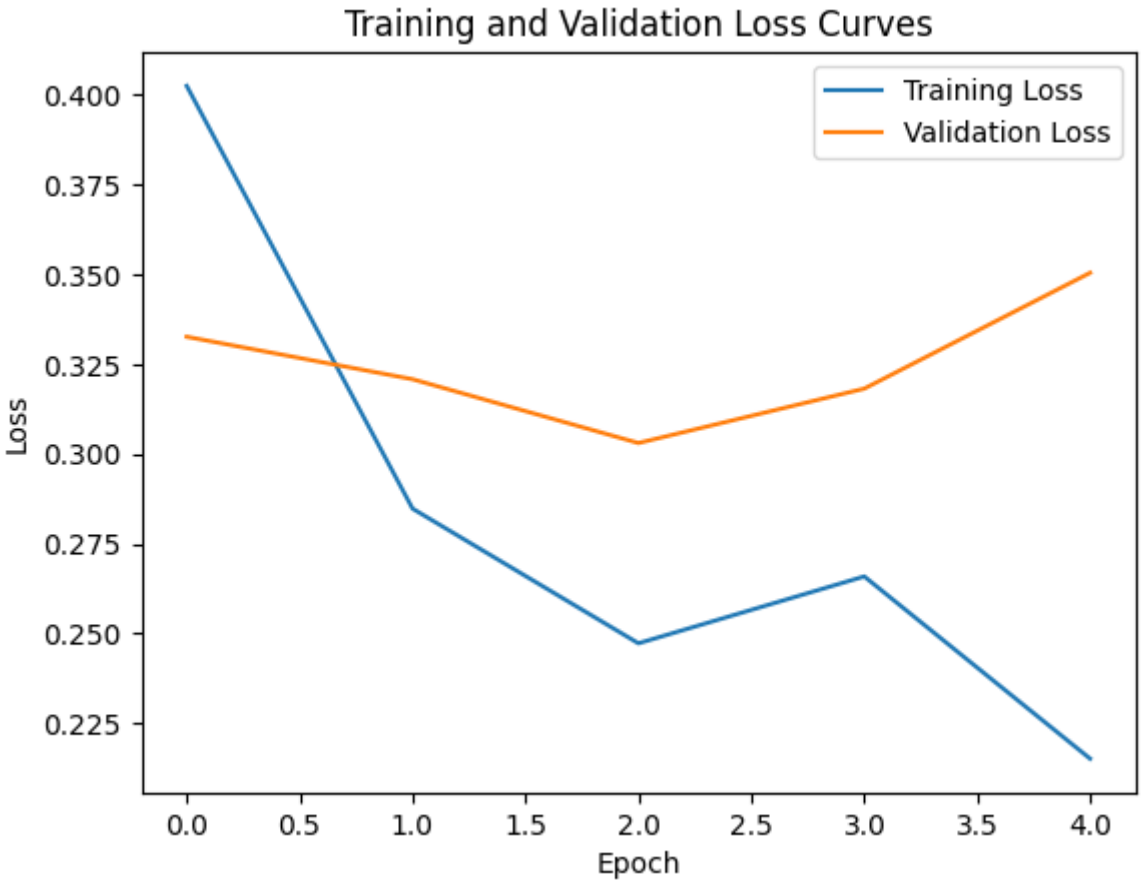
Cuối cùng, nhóm tiến hành huấn luyện mô hình với số lượng epochs là 10. Một epoch là một lần mô hình hoàn thành đi qua toàn bộ tập dữ liệu huấn luyện. Trong trường hợp này, mô hình sẽ được huấn luyện trong 5 epochs.

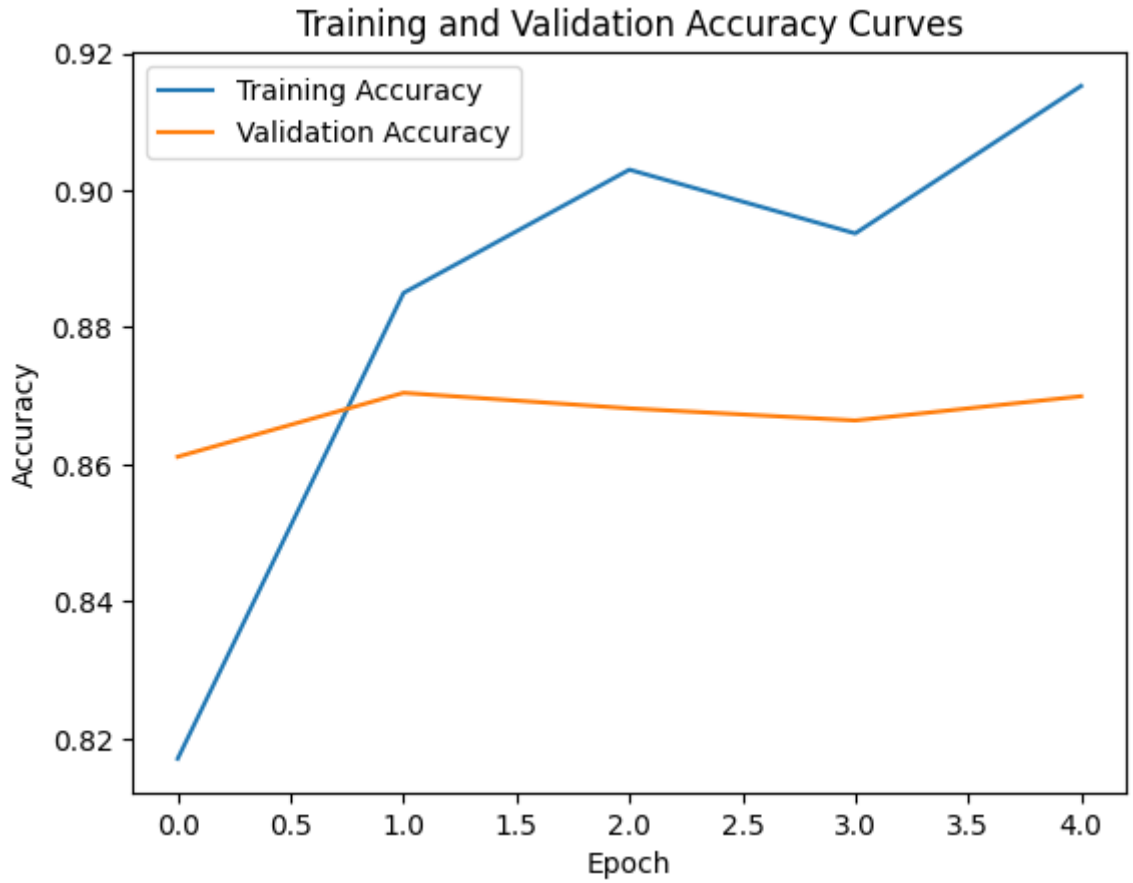
```
num_epochs = 5
history = model.fit(train_padded, train_labels,
                    epochs=num_epochs, verbose=1,
                    validation_split=0.1)
```

Sau đó, phương thức fit() được sử dụng để tiến hành huấn luyện mô hình LSTM. Trong phương thức này ta có train_padded là dữ liệu đầu vào, train_labels là các nhãn tương ứng, số lần mô hình sẽ lặp lại trên toàn bộ dữ liệu là 5, verbose = 1 tức là sẽ hiển thị thanh tiến trình và tham số cuối cùng sẽ chỉ định tỷ lệ dữ liệu huấn luyện sẽ được sử dụng làm dữ liệu xác thực. Trong trường hợp này, 10% dữ liệu huấn luyện được sử dụng để xác thực nhằm đánh giá hiệu suất của mô hình trên dữ liệu mà mô hình chưa thấy trong quá trình đào tạo và giúp giám sát tình trạng overfitting.

2.2.2 Đánh giá

Sau khi xem xét các kết quả từ quá trình huấn luyện và trực quan hóa nó, ta được 2 biểu đồ như sau:


Training and Validation Loss Curves
Training Loss
Validation Loss
Loss
0.400
0.375
0.350
0.325
0.300
0.275
0.250
0.225
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Epoch

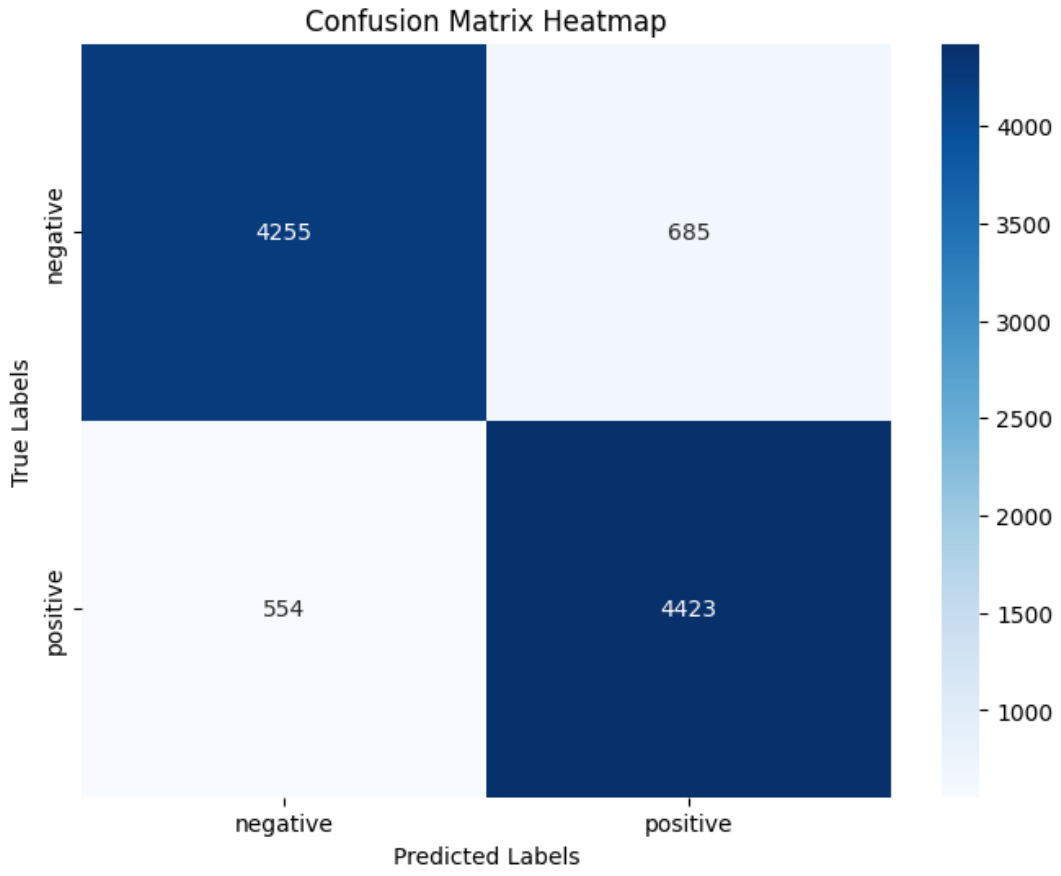

Training and Validation Accuracy Curves
Training Accuracy
Validation Accuracy
Accuracy
0.92
0.90
0.88
0.86
0.84
0.82
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Epoch

2 biểu đồ trên thể hiện quá trình huấn luyện của mô hình BiLSTM qua 5 epochs. Ta có thể thấy rằng sự mất mát trên tập huấn luyện có xu hướng giảm mạnh và sự chính xác trên tập huấn luyện tăng một cách rõ rệt, có nghĩa là mô hình đang học từ dữ liệu huấn luyện. Và cũng có thể thấy khoảng cách hơn giữa chỉ số của tập huấn luyện và tập xác thực không chênh nhau quá nhiều. Điều này có nghĩa là mô hình có thể không bị overfitting. Các giá trị trên tập xác thực tương đối tốt, cho thấy rằng mô hình có khả năng khái quát hóa tốt đối với dữ liệu mới.

Sau khi xem xét quá trình huấn luyện, ta sẽ đánh giá mô hình dựa trên tập test và có được kết quả như sau:

```
Accuracy of prediction on test set :  0.8750630230916607
```

	precision	recall	f1-score	support
0	0.88	0.86	0.87	4940
1	0.87	0.89	0.88	4977
accuracy			0.88	9917
macro avg	0.88	0.88	0.88	9917
weighted avg	0.88	0.88	0.88	9917



Kết quả cho thấy rằng mô hình này có độ chính xác khoảng 88% trên tập test, precision, recall và f1-score cũng khá tương đồng với nhau ở mức 86% đến 88%. Ta kết luận rằng kết quả báo cáo phân loại cho thấy mô hình có khả năng phân loại tốt và độ tin cậy cao.

So sánh chỉ số của các thuật toán máy học với nhau, ta có thể thấy Tfidf Vectorizer + Multinomial Logistic Regression là mô hình đánh giá có kết quả tốt nhất.

CHƯƠNG 5: KẾT LUẬN

Các mô hình được sử dụng trong bài báo cáo là:

- Mô hình học máy Naive Bayes
- Mô hình trong nhóm MaxEnt Logistic Regression
- Mô hình học sâu BiLSTM

Kết quả cho thấy mô hình Logistic Regression sử dụng TF-IDF Vectorizer đạt hiệu quả cao nhất, với độ chính xác (accuracy) trên tập test set là 89.6%.

Cho thấy mô hình được áp dụng khá tốt trên bộ dữ liệu và cũng có hiệu quả cao trong việc phân loại những đánh giá phim.

Trong tương lai, việc phân loại văn bản dựa trên các mô hình máy học sẽ được phát triển và ứng dụng rộng rãi hơn nữa trong nhiều lĩnh vực khác nhau. Đây là kỹ thuật quan trọng mang lại nhiều lợi ích cho doanh nghiệp, cũng như người tiêu dùng.

BẢNG PHÂN CÔNG

Tên thành viên	**Công việc**	**Mức độ tham gia**
Trịnh Uyên Chi	- Chương 2 - Mô hình Deep Learning - Power Point	100%
Nguyễn Hoàng Hà My	- Chương 3 - Mô hình MaxEnt - Power Point	100%
Huỳnh Thị Ngọc Trâm	- Chương 1, chương 5 - Mô hình Machine learning - Power Point	100%

TÀI LIỆU THAM KHẢO

[1] sentiment analysis using SVM.
[2] sentiment analysis with LSTM.
[3] Analysis of hyperparameters in Sentiment Analysis of Movie Reviews using Bi-LSTM
[4] CountVectorizer in NLP.
[5] TF-IDF