ĐẠI HỌC UEH
TRƯỜNG CÔNG NGHỆ VÀ THIẾT KẾ
KHOA CÔNG NGHỆ THÔNG TIN KINH DOANH

ĐỒ ÁN MÔN HỌC
NATURAL LANGUAGE PROCESSING

ĐỀ TÀI: Amazon Reviews for Sentiment Analysis

Giảng viên: TS. ĐẶNG NGỌC HOÀNG THÀNH

Mã lớp học phần: 23C1INF50907601

Sinh viên - MSSV: Ngô Gia Bảo - 31211024275

Vương Chí Binh - 31211020753

Trần Hoàng Trung Đức - 31211027635

Nguyễn Nhật Quang - 31211027664

TP. Hồ Chí Minh, ngày 23 tháng 12 năm 2023

MỤC LỤC

Chương 1. Tổng quan đề tài

1. Giới thiệu đề tài

Sentiment Analysis, hay còn gọi là Opinion Mining, là một lĩnh vực nghiên cứu trong xử lý ngôn ngữ tự nhiên (NLP) nhằm hiểu và phân loại ý kiến, cảm xúc từ văn bản ngôn ngữ tự nhiên. Trong bối cảnh mua sắm trực tuyến trở nên phổ biến, việc đánh giá sản phẩm trên các nền tảng thương mại điện tử như Amazon đang ngày càng trở nên quan trọng. Đây không chỉ là một nguồn thông tin quý giá cho người tiêu dùng khi quyết định mua sắm, mà còn là một cơ hội lớn cho các nhà nghiên cứu áp dụng công nghệ để tự động phân tích và hiểu ý kiến của người dùng.

Amazon, với hàng triệu đánh giá sản phẩm được đăng tải hàng ngày, là một kho dữ liệu lớn và đa dạng. Nghiên cứu Sentiment Analysis trên đánh giá sản phẩm trên Amazon đặt ra những thách thức lớn và đồng thời mang lại những cơ hội đáng kể.

Một trong những ứng dụng quan trọng của Sentiment Analysis trên Amazon là giúp doanh nghiệp hiểu rõ hơn về phản hồi của khách hàng. Bằng cách tự động phân loại đánh giá thành các phân khúc như tích cực hoặc tiêu cực, doanh nghiệp có thể theo dõi xu hướng cảm xúc của khách hàng đối với sản phẩm hoặc dịch vụ cụ thể. Điều này không chỉ giúp họ nắm bắt được những điểm mạnh và điểm yếu của sản phẩm mình cung cấp mà còn tạo ra cơ hội để nhanh chóng phản ứng và cải thiện chất lượng.

Ngoài ra, Sentiment Analysis cũng đóng vai trò quan trọng trong việc xây dựng hệ thống đề xuất sản phẩm (recommendation system). Dựa trên sự hiểu biết sâu sắc về ý kiến của người dùng, hệ thống có thể tối ưu hóa quá trình đề xuất sản phẩm, giúp người tiêu dùng dễ dàng tìm kiếm và chọn lựa những sản phẩm phù hợp với nhu cầu và mong muốn của họ.

Để thực hiện Sentiment Analysis trên đánh giá sản phẩm trên Amazon, các nhà nghiên cứu thường sử dụng các kỹ thuật máy học (machine learning) và học sâu (deep learning). Phương pháp này giúp họ tự động học từ dữ liệu lớn và phức tạp, từ đó tạo ra mô hình có khả năng phân loại ý kiến với độ chính xác cao.

Tuy nhiên, nghiên cứu Sentiment Analysis cũng đối mặt với những thách thức. Sự đa dạng ngôn ngữ, sự hiểu lầm, và sự thay đổi trong ngữ cảnh có thể làm cho việc phân tích trở nên phức tạp. Do đó, việc nghiên cứu và phát triển các mô hình phân loại ngày càng chính xác và linh hoạt là một hành trình không ngừng.

2. Mục tiêu nghiên cứu

a. Phát triển mô hình hiệu quả

Mục tiêu đầu tiên của nghiên cứu là phát triển mô hình Sentiment Analysis với khả năng phân loại ý kiến cao và linh hoạt. Điều này đòi hỏi sự nghiên cứu sâu rộng về các phương pháp máy học và học sâu, có khả năng xử lý đồng thời cả ngôn ngữ chính thống và các biến thể ngôn ngữ đặc biệt trong các đánh giá sản phẩm.

b. Giải quyết được vấn đề gán sai nhãn

Nhãn bị gán sai là một trường hợp không ai mong muốn, và với một bộ dữ liệu rất lớn thì ta lại càng không thể kiểm tra thủ công, do đó ta sẽ phát triển một cách thức mới để loại bỏ các mẫu có khả năng cao bị gán sai nhãn để mô hình có thể học tốt hơn.

c. **Giải quyết vấn đề Hiểu lầm và Ngữ cảnh**
Một khía cạnh quan trọng của nghiên cứu là giải quyết vấn đề hiểu lầm và ngữ cảnh trong đánh giá. Mô hình cần có khả năng nhận diện và xử lý các trường hợp này để đảm bảo kết quả phân loại chính xác và đáng tin cậy.

d. **Tích hợp Trí tuệ Nhân tạo (AI) vào các ứng dụng có thể sử dụng được.**
Mục tiêu cuối cùng là ứng dụng AI vào các ứng dụng có thể sử dụng được để chúng ta ngay lập tức thấy được mô hình chúng ta xây dựng có thể làm được gì, làm được tới đâu, có các hạn chế gì.

3. Phương pháp nghiên cứu

- **Thu Thập Dữ Liệu:** Các nghiên cứu của nhóm sử dụng dữ liệu đánh giá sản phẩm được thu thập từ Amazon, một nền tảng kho dữ liệu lớn và đa dạng. Dữ liệu này bao gồm thông tin đánh giá và cảm xúc tương ứng của từng đánh giá. Việc lựa chọn nguồn dữ liệu trên Kaggle là để tận dụng tài nguyên đa dạng và phong phú của cộng đồng nghiên cứu trực tuyến.
  Link kaggle: https://www.kaggle.com/datasets/bittlingmayer/amazonreviews
- **Tiền Xử Lý Dữ Liệu:** Trước khi áp dụng mô hình, quá trình tiền xử lý dữ liệu đang được thực hiện. Các bước này bao gồm loại bỏ các đường link URL, thẻ HTML, mở rộng các từ viết tắt, loại bỏ dấu chấm câu và chữ số, loại bỏ emoji, chuyển tất cả chữ sang chữ thường và cuối cùng là loại bỏ stopwords. Sau đó nhóm thực hiện thêm 1 bước là loại bỏ các mẫu có khả năng bị gán nhãn sai từ trước để mô hình hiệu quả hơn.
- **Trực Quan Dữ Liệu:** Trước khi có thể chọn ra mô hình một cách hiệu quả, nhóm sẽ trực quan dữ liệu để xem phân phối của các nhãn là thế nào để quyết định có các phương pháp riêng để xử lý cho việc dữ liệu bị bất cân xứng.
- **Xây Dựng Mô Hình:** Sau khi trực quan dữ liệu, nhóm đã có cơ sở để chọn ra các mô hình cần phải thử nghiệm, và nhóm sẽ chọn ra các siêu tham số phù hợp cho từng mô hình bằng cách sử dụng phương pháp GridSearch
- **Đánh Giá và So Sánh:** Chia tập dữ liệu thành tập huấn luyện và tập kiểm thử, mô hình được đánh giá hiệu suất bằng các thước đo như các thước đo accuracy, precision, f1,v.v…. Ngoài ra nhóm sẽ thực hiện thêm bằng confusion matrix và đường cong ROC.
- **Ứng dụng mô hình:** Nhóm thực hiện thêm xây dựng 2 demo khác nhau (mỗi demo có một điểm mạnh riêng) để người dùng có thể nhập vào một câu và trả ra loại cảm xúc cho câu đó.

4. Tài nguyên sử dụng

a. Nguồn Dữ Liệu Kaggle: Dữ liệu đánh giá sản phẩm được sử dụng để huấn luyện và đánh giá mô hình được thu thập từ Kaggle. Nguồn dữ liệu này cung cấp một tập hợp đa dạng và phong phú các đánh giá từ người dùng trên nhiều sản phẩm khác nhau.

b. Thư Viện và Frameworks

- Scikit-learn: Thư viện machine learning trong Python được sử dụng để xây dựng pipeline cho việc tiền xử lý dữ liệu và mô hình Machine Learning truyền thống(Multinomial Naive Bayes), mô hình MaxEnt (Logistic Regression).
- PyTorch: Framework deep learning được sử dụng để xây dựng và huấn luyện mô hình deep learning có khả năng học đặc trưng từ dữ liệu văn bản. Trong trường hợp nhóm thì PyTorch dùng để xây dựng mô hình Deep Learning (Transformer).
- Pandas và NumPy: Thư viện Pandas được sử dụng để quản lý và xử lý dữ liệu từ Kaggle, bao gồm việc chọn lọc cột chứa đánh giá và nhãn cảm xúc. NumPy hỗ trợ trong việc thực hiện các phép toán số học và xử lý mảng nhanh chóng.
- Các thư viện khác.
- Google Colab: Môi trường lập trình được triển khai trên nền tảng Google Colab, giúp nghiên cứu tiếp cận với tài nguyên máy học mạnh mẽ và đồng thời cung cấp sự linh hoạt cho việc lưu trữ và chia sẻ mã nguồn.

Chương 2. Thu thập và Tiền xử lý Dữ liệu

1. Mô tả nguồn dữ liệu: Amazon Reviews

Bộ dữ liệu này là một nguồn thông tin đáng kể, bao gồm hàng triệu đánh giá từ khách hàng trên trang web thương mại điện tử lớn Amazon. Nó chứa các đoạn văn bản đánh giá (text) kèm theo đánh giá tương ứng (label). Điều độc đáo của bộ dữ liệu này là nó không chỉ đơn giản là một "toy" trong thế giới học máy, mà thực sự là một tập dữ liệu doanh nghiệp có quy mô đủ lớn để phản ánh đa dạng và phức tạp của thông tin thực tế.

Hướng dẫn học máy giám sát fastText yêu cầu dữ liệu được chuẩn bị theo định dạng sau:

__label__<X> __label__<Y> ... <Text>

Trong đó X và Y là tên của các lớp. Không có dấu ngoặc kép, tất cả nằm trên một dòng. Trong trường hợp này, các lớp là __label__1 và __label__2, và sẽ có mỗi lớp khác nhau ứng với mỗi nhận xét khác nhau trên mỗi dòng khác nhau. Ý nghĩa của từng nhãn như sau:

- __label__1 tương ứng với đánh giá 1 và 2 sao.
- __label__2 tương ứng với đánh giá 4 và 5 sao.

(Đánh giá 3 sao, tức là đánh giá có tâm trạng trung lập, không được bao gồm trong ban đầu),

Giữa nhãn của câu và câu sẽ phân tách với nhau bằng một khoảng trắng. Cụ thể ví dụ như hình sau đây:

```
__label__1 Batteries died within a year ...: I bought this charger in Jul 2003 and it worked OK for a while. The design is nice and
convenient. However, after about a year, the batteries would not hold a charge. Might as well just get alkaline disposables, or look
elsewhere for a charger that comes with batteries that have better staying power.
__label__2 works fine, but Maha Energy is better: Check out Maha Energy's website. Their Powerex MH-C204F charger works in 100 minutes for
rapid charge, with option for slower charge (better for batteries). And they have 2200 mAh batteries.
```

2. Các bước thu thập dữ liệu

Nhóm đã tải bộ dữ liệu từ kaggle về và sau đó đẩy lên google drive, sau đó từ colab dùng lệnh !gdown và tải trực tiếp xuống workspace làm việc của colab

```
# Data train
#https://drive.google.com/file/d/1ZWnEccAMmKl9JGLjbr39e9AUHn0_r4e_/view?usp=sharing
!gdown 1ZWnEccAMmKl9JGLjbr39e9AUHn0_r4e_
```

```
Downloading...
From: https://drive.google.com/uc?id=1ZWnEccAMmKl9JGLjbr39e9AUHn0_r4e_
To: /content/train.ft.txt
100% 1.60G/1.60G [00:20<00:00, 79.5MB/s]
```

```
# Data test
#https://drive.google.com/file/d/1vrM3twfe387IEPPwTF3fSv-CGSSwZf66/view?usp=sharing
!gdown 1vrM3twfe387IEPPwTF3fSv-CGSSwZf66
```

```
Downloading...
From: https://drive.google.com/uc?id=1vrM3twfe387IEPPwTF3fSv-CGSSwZf66
To: /content/test.ft.txt
100% 177M/177M [00:02<00:00, 85.3MB/s]
```

Sau đó nhóm đọc qua từng file train vào một list content_train, làm tương tự cho file test và lưu vào list content_test.

Kế đến ta cần data phải ở dạng mà ta có thể cho vào DataFrame để tiện lợi cho việc phục vụ tạo dữ liệu train và dữ liệu test như sau:

```
# Cho dữ liệu train_content vào trong DataFrame
data = {'text': [], 'label': []}

for line in content_train:
    if line.startswith('__label__1'):
        label = 0
        text = line.replace('__label__1', '').strip()
        data['text'].append(text)
        data['label'].append(label)
    elif line.startswith('__label__2'):
        label = 1
        text = line.replace('__label__2', '').strip()
        data['text'].append(text)
        data['label'].append(label)
```

Data sẽ là một dictionary chứa text và label, nhóm duyệt qua từng list trong list content_train, nếu một câu bắt đầu bằng __label__1 thì nhóm sẽ cho nhãn của câu đó là 0, ngược lại thì cho nhãn là 1. Sau đó nhóm loại bỏ phần __label__ đó đi để lấy phần còn lại là phần text, sau cùng nhóm thu được một dictionary chứa đầy đủ text và label (lúc này ở dạng nhị phân). Làm tương tự cho data_test.

Kết thúc quá trình trên ta thu được 2 dictionaries, sau đó cho vào DataFrame, ta thu được hai DataFrame mới là df và test_df với độ dài như hình sau:

```
print(f'len of the train data is {len(df)}')
print(f'len of the test data is {len(test_df)}')
# Quá lớn -> Phải reduce
```

```
len of the train data is 3600000
len of the test data is 400000
```

Có thể thấy rằng trong data ban đầu của nhóm, ta sẽ thu được 3.600.000 bản ghi cho dữ liệu huấn luyện và 400.000 bản ghi cho dữ liệu kiểm tra. Và đương nhiên là quá lớn, do đó nhóm thực hiện lấy mẫu data từ bộ huấn luyện và bộ kiểm tra và thu được bộ mới như sau:

```
# Sau khi trích mẫu
print(f'len of the train data is {len(df_sampled)}')
df_sampled
```

```
len of the train data is 40000
```

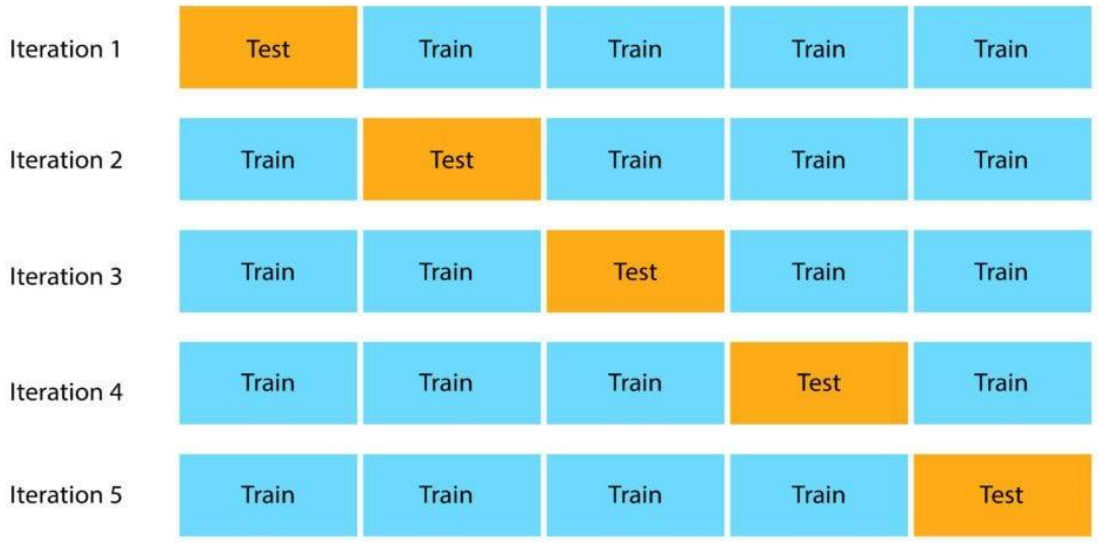
```
print(f'len of the test data is {len(test_df)}')
test_df
```

```
len of the test data is 1000
```

3. Xử lý nhãn lỗi

Nhãn lỗi có thể nói là một cái gì đó hết sức nhạy cảm bởi vì đó là lỗi do người gán nhãn, và mô hình sẽ học từ cái nhãn đó. Nhưng nếu dữ liệu của chúng ta quá lớn thì việc kiểm tra lại xem người gán nhãn có gán lỗi hay không dường như là không thể (như ví dụ về 3.600.000 bản ghi lúc nãy, việc kiểm tra thủ công là rất tốn kém). Nên chúng ta cần phương pháp để xử lý các trường hợp này một cách ít tốn kém hơn.

Nhóm đề xuất sử dụng K-Fold để làm điều này với ý tưởng như sau:



Ý tưởng cốt lõi là nếu một mô hình rất tự tin vào kết quả dự đoán của nó (ví dụ mô hình dự đoán hơn 95% là cái nhãn trả ra sẽ là tích cực nhưng cái nhãn thực ra lại là tiêu cực thì có khả năng là nhãn do người gán có khả năng là bị sai) thì ta nên tin vào mô hình, và như vậy thì đối với các nhãn có khả năng bị sai đó, ta nên cho ra khỏi bộ dữ liệu để mô hình có thể chạy tốt hơn.

Cụ thể hơn, ta có thể chia bộ dữ liệu huấn luyện sang 5 phần bằng nhau (4 phần train, 1 phần test) và làm như vậy 5 lần, sao cho hợp của tất cả 5 bộ test thì ra được 1 bộ train ban đầu. Như vậy, sẽ có tổng cộng 5 lần train riêng biệt, và ứng với mỗi lượt train, ta sẽ bỏ đi cái mẫu có khả năng bị gán sai nhãn ở lượt test tương ứng. Và sau khi làm vậy đủ 5 lần, ta sẽ hợp tất cả các bộ test lại và ta sẽ có một bộ train mới đã loại bỏ đi các nhãn có khả năng bị gán sai, như hình sau:

```
6 df_sampled
```

	text	label	preprocess_sentence
0	dare you to finish this book: I dare you to fi...	0	dare finish book dare finish book simple inter...
1	Amazon should not be promoting "hate" stuff li...	0	amazon promoting hate stuff like hate jews hom...
2	Herbie plays it easy-listening: As an HH great...	0	herbie plays easy listening hh great fan disap...
3	Freddie Will Live 4Ever: As long as there are ...	1	freddie live 4ever long humans electricity con...
4	Cord description 100% incorrect: New in origin...	0	cord description 100 incorrect new original pa...
...	...	...	...
39995	Powdered wax is a bad idea: My daughter receiv...	0	powdered wax bad idea daughter received one tw...
39996	WHAT was the Network Thinking??: Answer...they...	1	network thinking answer terrific show yanked 4...
39997	After Shave Cream: This product is a great fol...	1	shave cream product great follow close shave v...
39998	Zzzzzzzzzzzzzzzzzz: I thought I was gonna fa...	0	zzzzzzzzzzzzzzzzzz thought going fall asleep...
39999	Put together by a blind man: I received the bo...	0	put together blind man received book quickly h...

40000 rows × 3 columns

```
1 df_new_train
```

	text	label
0	dare finish book dare finish book simple inter...	0
1	amazon promoting hate stuff like hate jews hom...	0
2	herbie plays easy listening hh great fan disap...	0
3	freddie live 4ever long humans electricity con...	1
4	cord description 100 incorrect new original pa...	0
...	...	...
39429	powdered wax bad idea daughter received one tw...	0
39430	network thinking answer terrific show yanked 4...	1
39431	shave cream product great follow close shave v...	1
39432	zzzzzzzzzzzzzzzzzz thought going fall asleep...	0
39433	put together blind man received book quickly h...	0

39434 rows × 2 columns

Còn cụ thể làm sao để ra được preprocess_sentence thì nhóm sẽ trình bày phía dưới. Và sau khi đã có được DataFrame mới, ta lưu nó về dưới dạng file csv, sau đó tải ngược lại lên drive, và sau này dùng !gdown để làm lại từ đầu cho nó không bị trường hợp tràn RAM. 2 file mới sẽ là ‘train.csv’ và ‘test.csv’.

4. Tiền xử lý dữ liệu

Sẽ có 7 bước tiền xử lý dữ liệu chính mà nhóm sẽ sử dụng như sau:

- Bước 1: Loại bỏ các đường link URL.
- Bước 2: Loại bỏ các thẻ HTML.
- Bước 3: Mở rộng các từ viết tắt.
- Bước 4: Loại bỏ dấu chấm câu .

- Bước 5: Loại bỏ các emoji phổ biến.
- Bước 6: Chuyển các chữ sang dạng chữ thường.
- Bước 7: Loại bỏ các stopword ra khỏi câu.

Cụ thể hàm được cài như sau:

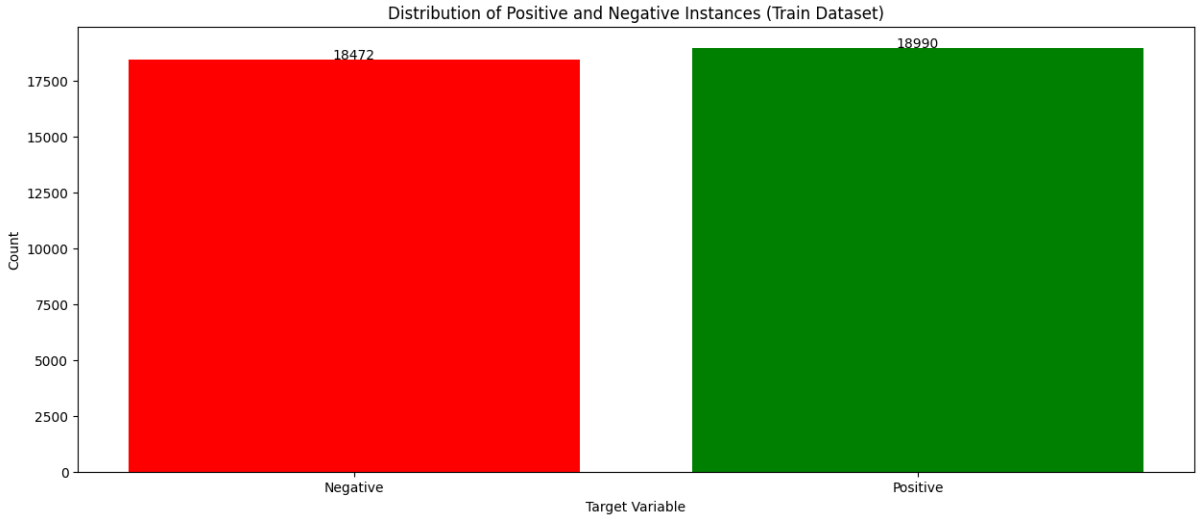
```
def preprocess_text(text):
    # Remove URLs
    url_pattern = re.compile(r'https?://\S+')
    text = url_pattern.sub(' ', text)

    # Remove HTML Tags
    html_pattern = re.compile(r'<[^<>]+>')
    text = html_pattern.sub(' ', text)

    # Expand contractions
    text = ' '.join([contractions_dict.get(word, word) for word in
text.split()])

    # Remove punctuation and digits
    text = re.sub(r'[^\w\s]', ' ', text)

    # Remove emojis
    emoji_pattern = re.compile("["
        u"\U0001F600-\U0001F64F"
        u"\U0001F300-\U0001F5FF"
        u"\U0001F680-\U0001F6FF"
        u"\U0001F1E0-\U0001F1FF"
        u"\U0001F1F2-\U0001F1F4"
        u"\U0001F1E6-\U0001F1FF"
        u"\U0001F600-\U0001F64F"
        u"\U00002702-\U000027B0"
        u"\U000024C2-\U0001F251"
        u"\U0001f926-\U0001f937"
        u"\U0001F1F2"
        u"\U0001F1F4"
        u"\U0001F620"
        u"\u200d"
        u"\u2640-\u2642"
        "]+", flags=re.UNICODE)
    text = emoji_pattern.sub(' ', text)

    # Convert to lowercase
    text = text.lower()
```

```
    # Tokenize and remove stopwords
    stop_words = set(stopwords.words('english'))
    tokens = word_tokenize(text)
    tokens = [token for token in tokens if token not in stop_words]

    # Join tokens back into text
    text = ' '.join(tokens)

    return text
```

Và thật ra trong DataFrame df_sampled ở trên, nhóm đã tạo ra cột mới preprocess_sentence là cột text, là cột mà được cho qua hàm preprocess_text vừa thiết lập ở trên.

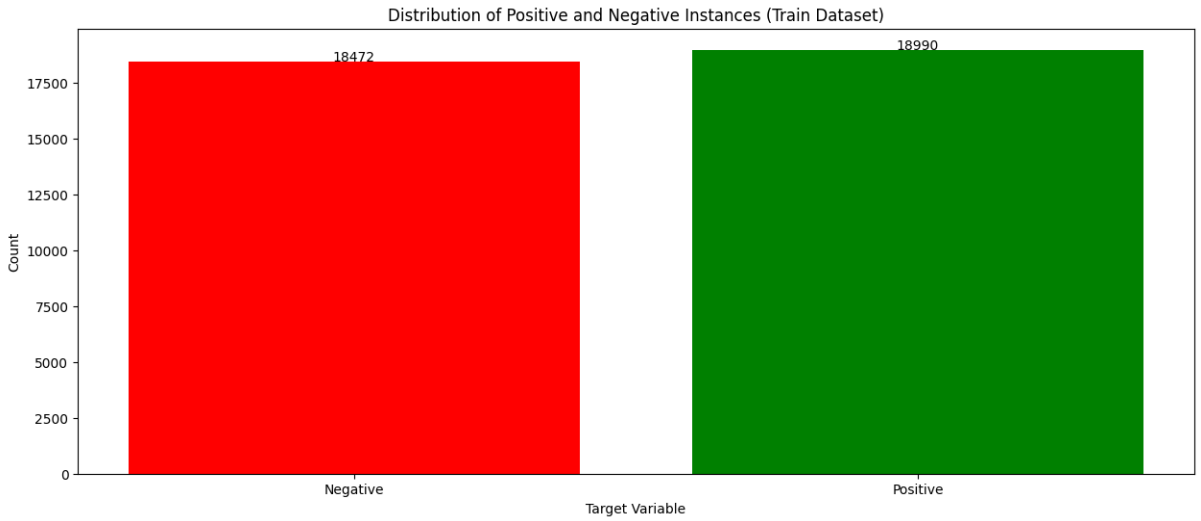
5. Biểu diễn trực quan dữ liệu

Với mục tiêu là để phân loại cảm xúc khách hàng, một bước ta cần thực hiện là để xem phân phối của các nhãn đang thế nào, các nhãn có bị lệch hay không, bởi vì nếu dữ liệu không cân bằng thì ta cần phải có hướng tiếp cận khác. Đầu tiên ta xem thử phân phối của nhãn tích cực và nhãn tiêu cực trong tập huấn luyện:

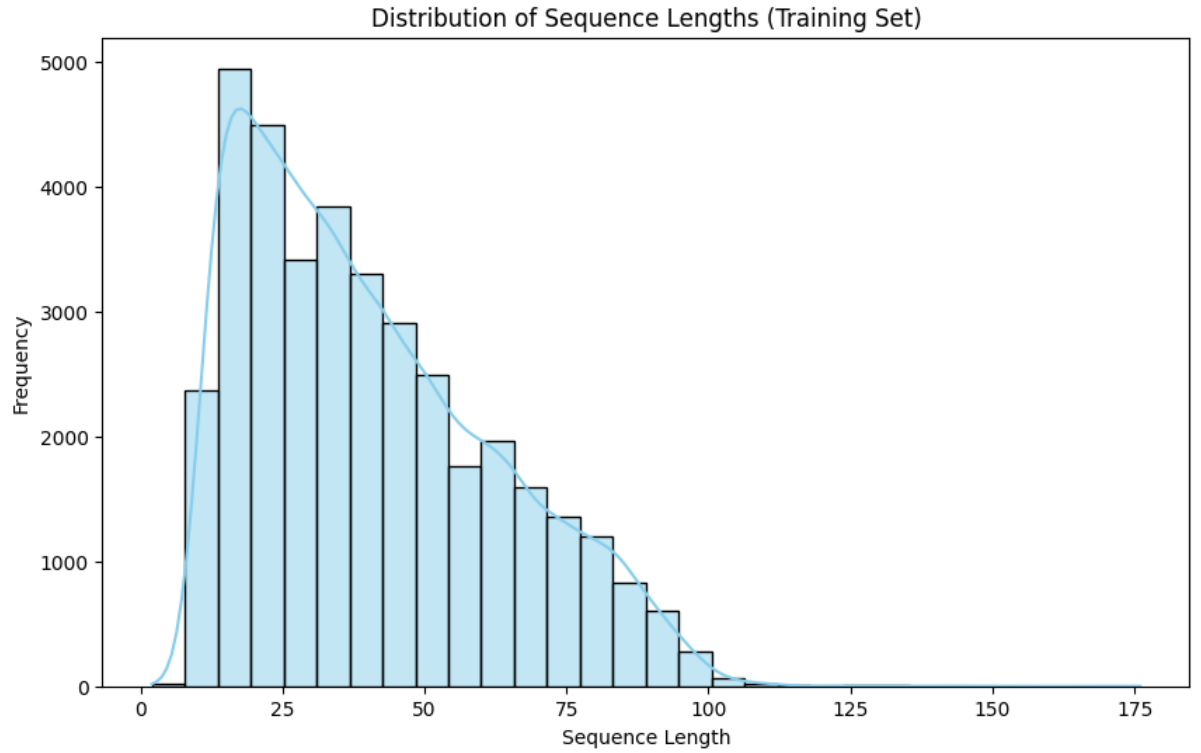


Có thể thấy rằng số lượng nhãn tiêu cực và nhãn tích cực là ngang bằng nhau, do đó ta không cần phải có thêm các bước xử lý như cân bằng lại mẫu bằng các phương pháp oversampling,

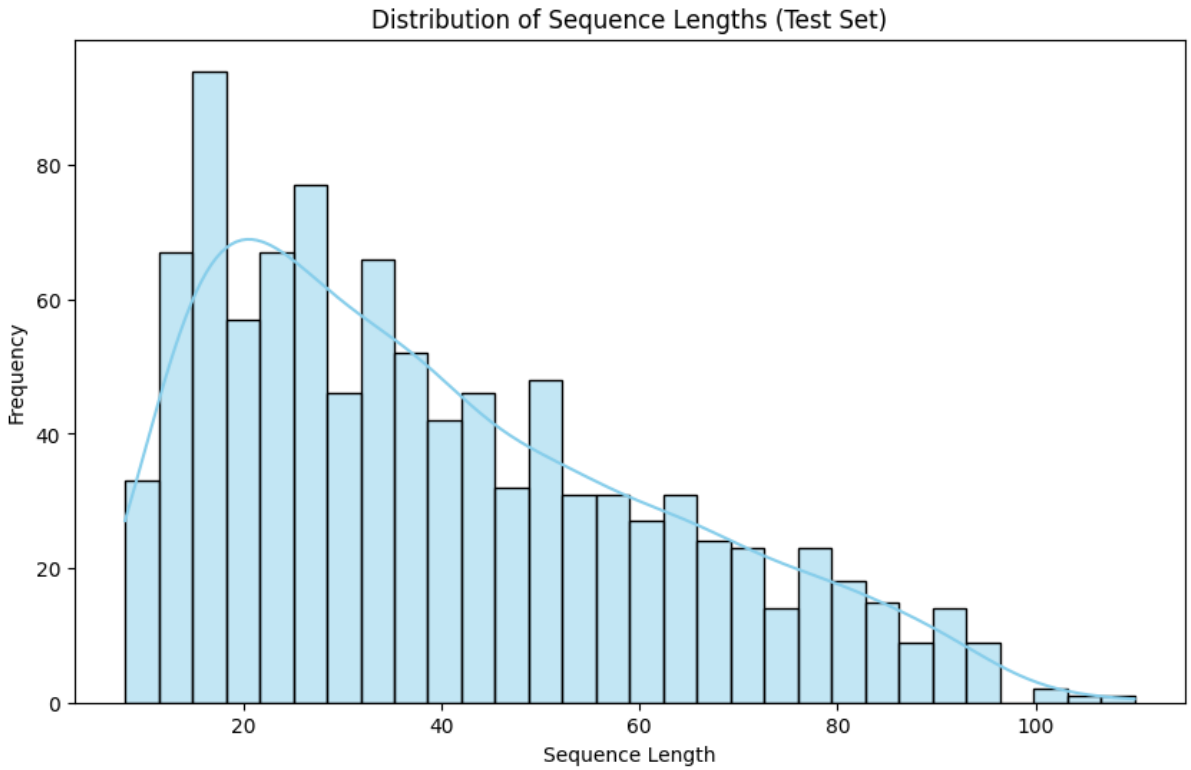
undersampling,v.v.… Kế đến ta xem xét thử phân phối của bộ dữ liệu kiểm thử:



Bộ dữ liệu kiểm thử cũng không bị trường hợp mất cân bằng. Sau đó ta cũng tiến hành xem phân phối độ dài chuỗi của cả bộ huấn luyện và bộ kiểm thử để phục vụ cho mô hình học sâu mà nhóm thực hiện ở sau:



Nếu phải chọn một độ dài chuỗi tối đa để phục vụ cho mô hình học sâu ở dưới, nhóm sẽ chọn một độ dài chuỗi tối đa là 100 dựa vào hình trên.



Có thể thấy đối với bộ dữ liệu kiểm thử, một độ dài chuỗi tối đa bằng 100 là hợp lý.

Chương 3. Xây dựng mô hình

3.0 Phương pháp trích xuất đặc trưng văn bản TF-IDF

a. Mô tả phương pháp

TF-IDF là một kỹ thuật trích xuất đặc trưng từ văn bản. Nó tính toán mức độ quan trọng của một từ trong một văn bản bằng cách nhân tần số xuất hiện của từ trong văn bản với nghịch đảo của tần số xuất hiện của từ trong toàn bộ tập dữ liệu.

b. Lý do chọn phương pháp

- **Đơn giản và dễ hiểu:** TF-IDF là một kỹ thuật đơn giản và dễ hiểu, có thể được triển khai một cách hiệu quả. Phương pháp này chỉ phụ thuộc vào hai thông số là tần số xuất hiện của từ trong văn bản và tần số xuất hiện của từ trong toàn bộ tập dữ liệu.
- **Hiệu quả:** TF-IDF đã được chứng minh là hiệu quả trong nhiều ứng dụng xử lý ngôn ngữ tự nhiên, bao gồm phân loại văn bản, tìm kiếm thông tin và kết nối văn bản.
- **Tính phổ biến:** TF-IDF là một kỹ thuật phổ biến được sử dụng trong nhiều ứng dụng xử lý ngôn ngữ tự nhiên. Điều này có nghĩa là có nhiều tài liệu tham khảo và tài nguyên có sẵn để giúp chúng ta hiểu và sử dụng TF-IDF.

3.1 Mô hình Multinomial Naive Bayes

a. Lý do chọn mô hình Multinomial Naive Bayes

- Rất phù hợp với các tác vụ phân loại văn bản, kể cả khi dữ liệu có chiều cao với mỗi từ là một thuộc tính.
- Có hiệu suất tính toán cao và có thể xử lý tốt các bộ dữ liệu lớn.
- Cung cấp xác suất cho các phân loại, giúp hiểu rõ hơn về mức tin cậy của mô hình trong các dự đoán của mình.

b. Nguyên lý hoạt động của Multinomial Naive Bayes

- Hoạt động dựa trên định lý Bayes, trong đó thuật toán Naive Bayes dựa trên hai giả định chính:
    - **Độc lập**: Mỗi đặc trưng/ biến của cùng một lớp có ảnh hưởng độc lập đối với kết quả.
        - **Note:** *Trong thực tế, điều này hiếm khi đúng đối với các thuật ngữ trong văn bản. Thường thì sự thật lại ngược lại, nhiều trường hợp các thuật ngữ có mối liên kết mật thiết với nhau* (Stanford NLP, 2009). Sự đơn giản hóa này khiến cho thuật toán Naive Bayes vừa có điểm mạnh về hiệu quả tính toán, vừa có điểm yếu về khả năng áp dụng vào thực tế, đặc biệt là trong lĩnh vực xử lý ngôn ngữ tự nhiên, nơi mà sự phụ thuộc ngữ cảnh giữa các từ là điều không thể bỏ qua.
    - **Đóng góp đồng đều**: Mỗi đặc trưng đều góp phần như nhau vào kết quả cuối cùng.
- Mô hình Naive Bayes được xây dựng dựa trên cơ sở dữ liệu ma trận X cùng với vector mục tiêu y. Ta đặt vấn đề dưới dạng tìm xác suất của y khi biết X, có thể được biểu diễn qua công thức:

$$P(y \mid X) = \frac{P(X \mid y)P(y)}{P(X)}$$

Trong đó, y là biến lớp và X là vector đặc trưng phụ thuộc với kích thước d, tức là X = (x1, x2, ..., xd) và d là số lượng biến/đặc trưng của mẫu.

- P(y|X) biểu thị xác suất quan sát được lớp y khi có mẫu X.
- P(X|y) là xác suất quan sát được X khi biết y.
- P(y) là xác suất tiên nghiệm của y, không phụ thuộc vào X.
- P(X) là xác suất của X, không phụ thuộc vào y.

Chúng ta giả định tất cả các đặc trưng trong X đều độc lập lẫn nhau khi đã biết y. Điều này cho phép chúng ta tách rời xác suất chung thành tích của các xác suất riêng lẻ:

$$P(y \mid x_1, \ldots, x_d) = \frac{P(y)\prod_{i=1}^{d} P(x_i|y)}{P(x_1)P(x_2)\ldots P(x_d)}$$

Cuối cùng, để tìm ra lớp mà mẫu đang xét thuộc vào, ta chỉ cần xác định đầu ra có xác suất cao nhất:

$$y = \text{argmax}_y P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

c. Cài đặt mô hình:

Sau khi đã thực hiện kiểm nghiệm với các siêu tham số khác nhau bằng GridSearch (chi tiết trong code) thì nhóm rút ra được mô hình Multinomial Naive Bayes như sau:

```
# Cài đặt Pipeline với tham số chọn từ GridSearch
model_NB = Pipeline([
    ('tfidf', TfidfVectorizer(min_df=1, ngram_range=(1, 2))),
    ('nb', MultinomialNB())
])

# Huấn luyện mô hình trên tập huấn luyện
model_NB.fit(X_train, y_train)

# Dự đoán nhãn trên tập kiểm thử
y_pred = model_NB.predict(X_test)
predicted_probabilities = model_NB.predict_proba(X_test)

# Đánh giá mô hình
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

print(f"Accuracy: {accuracy}")
print(report)
```

3.2. Mô hình Logistic Regression:

a. Lý do chọn mô hình Logistic Regression

- Hiệu Suất với Dữ Liệu Có Chiều Cao: Logistic Regression, một hình thức cụ thể của mô hình MaxEnt, hiệu quả trong việc xử lý các bộ dữ liệu có nhiều đặc trưng, đặc biệt là trong xử lý ngôn ngữ tự nhiên, nơi mỗi từ có thể được xem là một đặc trưng. Điều này giúp mô hình phù hợp với các bài toán phân loại văn bản và các tác vụ NLP khác.
- Xử Lý Tốt Dữ Liệu Phụ Thuộc: Khác biệt so với các mô hình truyền thống như Naive Bayes, Logistic Regression không yêu cầu giả định về sự độc lập của các đặc trưng. Điều này cho phép mô hình xử lý tốt hơn các trường hợp có mối quan hệ phụ thuộc giữa các đặc trưng, thường thấy trong xử lý ngôn ngữ tự nhiên.
- Cung Cấp Xác Suất Dự Đoán: Tương tự như Naive Bayes, Logistic Regression cũng cung cấp xác suất cho mỗi lớp phân loại, giúp hiểu rõ hơn về mức độ tự tin của mô hình trong các dự đoán của nó.

b. Nguyên Lý Hoạt Động của Logistic Regression:

- Logistic Regression dựa trên nguyên lý của Maximum Entropy, tức là chọn phân phối xác suất sao cho thông tin mà nó cung cấp là lớn nhất, trong khi vẫn tuân thủ các ràng buộc từ dữ liệu.
- Đặc Trưng và Xác Suất Điều Kiện: Trong Logistic Regression, xác suất của một lớp cụ thể khi biết một bộ đặc trưng được tính toán dựa trên tổ hợp tuyến tính của các trọng số và đặc trưng, sau đó được chuyển đổi qua hàm logistic.
- Tối Ưu Hóa: Mô hình được tối ưu hóa thông qua các phương pháp như Gradient Descent để tìm ra bộ trọng số tối ưu, nhằm cực đại hóa hàm entropy đồng thời tuân thủ các ràng buộc từ dữ liệu.

c. Cài đặt mô hình:

Sau khi đã thực hiện kiểm nghiệm với các siêu tham số khác nhau bằng GridSearch (chi tiết trong code) thì nhóm cài đặt mô hình Logistic Regression như sau:

```
# Các siêu tham số
best_parameters = {'logisticregression__C': 100, 'tfidfvectorizer__ngram_range': (1, 2)}

# Cài đặt Pipeline với tham số chọn từ GridSearch

model_LR = Pipeline([
    ('tfidfvectorizer', TfidfVectorizer(ngram_range=best_parameters['tfidfvectorizer__ngram_range'])),
    ('logisticregression', LogisticRegression(C=best_parameters['logisticregression__C']))
])

# Huấn luyện mô hình trên tập huấn luyện
model_LR.fit(X_train, y_train)

# Dự đoán nhãn trên tập kiểm thử
y_pred = model_LR.predict(X_test)

# Đánh giá mô hình
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

print(f"Accuracy: {accuracy}")
print(report)
```

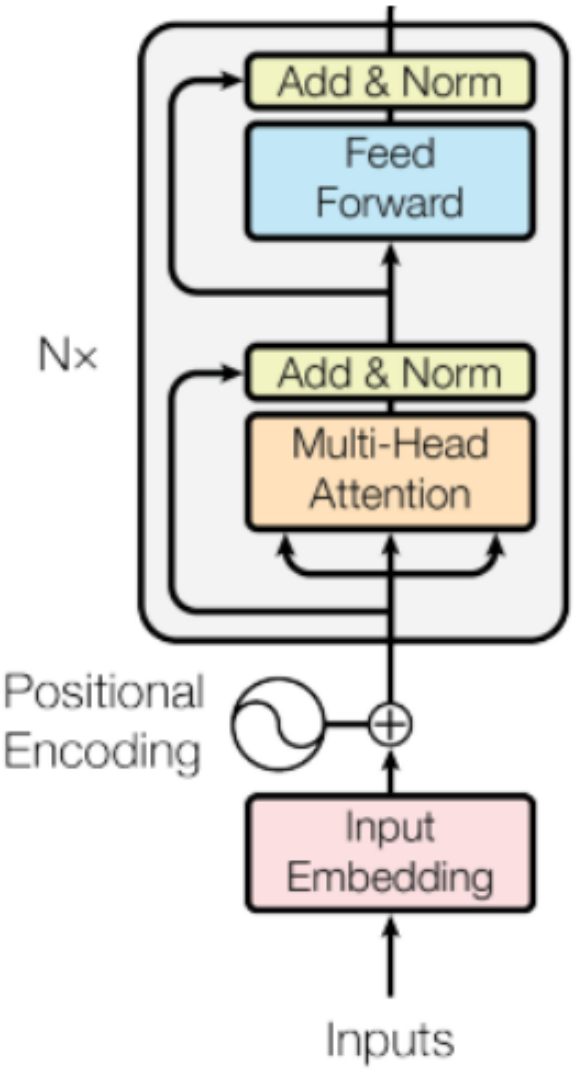
3.3. Mô hình Transformer (Encoder):

Mô hình transformer là một loại mô hình học sâu được sử dụng trong xử lý ngôn ngữ tự nhiên (NLP). Nó được giới thiệu lần đầu tiên trong bài báo "Attention is All You Need" (Vaswani, 2017) của Vaswani và cộng sự vào năm 2017.

a. Lý do chọn mô hình Transformer:

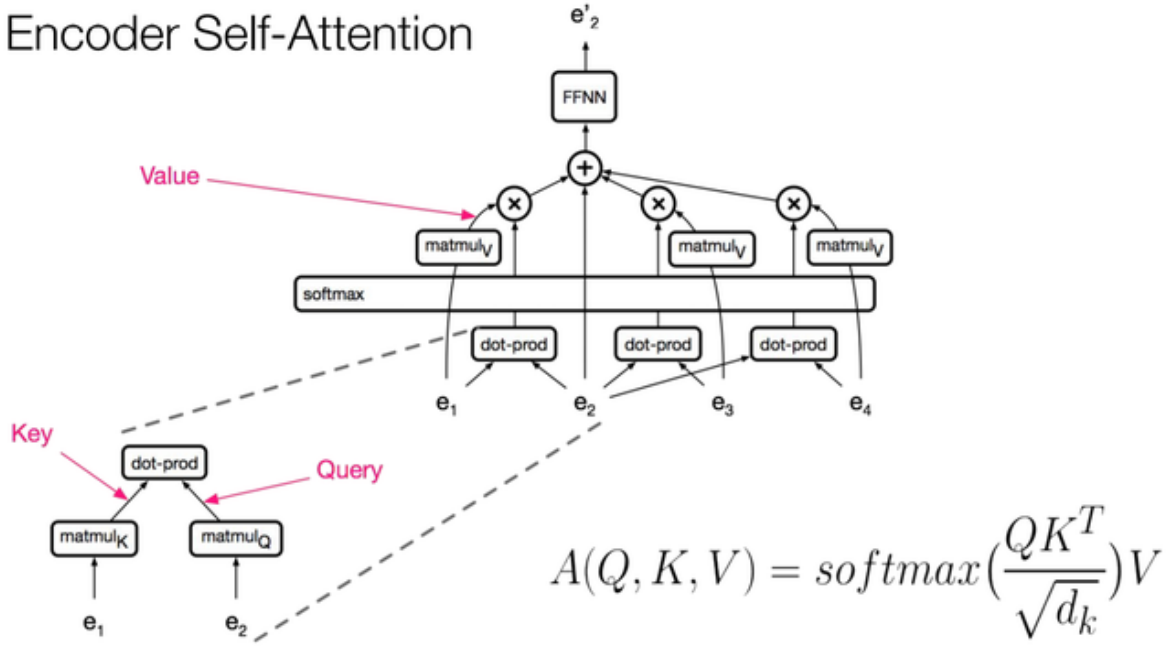
- Khả năng xử lý song song: Trong khi các mô hình truyền thống như RNN hoặc LSTM phải xử lý các từ trong một chuỗi theo thứ tự tuần tự, thì mô hình transformer có thể xử lý tất cả các từ cùng một lúc. Điều này cho phép mô hình transformer học được các mối quan hệ phức tạp giữa các từ trong một câu, ngay cả khi chúng nằm cách xa nhau.
- Khả năng học các biểu diễn từ có ý nghĩa: Mô hình transformer sử dụng cơ chế self-attention để học các biểu diễn từ có tính đến ngữ cảnh của chúng trong một câu. Điều này cho phép mô hình transformer hiểu được ý nghĩa của các từ trong một ngữ cảnh cụ thể.

b. Nguyên lý hoạt động của Transformer (Encoder):



Đầu vào của phần Encoder là các Embedding (Embedding này là tổng của Embedding đầu vào cộng cho Embedding vị trí, lý do cho việc ta cần Embedding vị trí là do mô hình Transformer xử lí các từ song song với nhau mà không quan tâm đến vị trí của từ trong khi vị trí từ đóng vai trò quan trọng trong ngữ nghĩa của từ, và các Embedding sau cùng này sẽ có cùng số chiều với Embedding đầu vào, nhưng lúc này nó còn chứa thêm nhiều thông tin ý nghĩa tùy theo ngữ cảnh, từ loại, vị trí từ, v.v…). Và đầu ra của Encoder sau khi nhận vào các Embedding này gọi là hidden state (trạng thái ẩn). Và thông thường, các vector hidden state này sẽ được cho vào bộ Decoder (bộ giải mã) để thực hiện các tác vụ của chúng. Trong trường hợp của nhóm, nhóm chỉ dùng phần encoder thôi nên là nhóm sẽ cho vector hidden state vào các lớp linear rồi sau đó phân loại luôn. Cụ thể hơn như dưới đây:

- **Embeddings**: Embedding là một kỹ thuật mã hóa thông tin từ văn bản thành các vector số. Các vector này có thể được sử dụng để đại diện cho các từ, cụm từ, hoặc thậm chí cả các câu. Positional embedding là một kỹ thuật bổ sung cho embedding để mã hóa thông tin về vị trí của các từ trong câu. Điều này giúp mô hình hiểu được ý nghĩa của các từ trong ngữ cảnh của chúng.
- **Self-attention**: Vấn đề là input embedding là tĩnh (chỉ chứa một phần thông tin nhất định). Lớp self-attention xem xét ngữ cảnh của các từ hoặc token trong câu hoặc tập hợp câu, và mã hóa thêm thông tin ngữ cảnh đó vào các embedding tương ứng.Self-attention sử dụng ba tensor (Query, Key và Value) và các phép toán liên quan để mã hóa ngữ cảnh. Công thức của Self-attention có thể tham khảo hình dưới đây:



- **Multi-head Attention:** Multi-Head Attention là một lớp bao gồm nhiều đơn vị self-attention, cho phép mô hình tập trung vào nhiều khía cạnh khác nhau của văn bản. Multi-Head Attention giúp mô hình học được các đặc trưng khác nhau của văn bản, từ đó dẫn đến kết quả tốt hơn.
- **Feed-Forward**: Một lớp đơn giản gồm 2 lớp tuyến tính (linear layer).

Vừa được trình bày ở trên là những lý thuyết quan trọng của phần Encoder trong mô hình Transformer. Cụ thể mô hình sẽ được đề cập ở phần sau:

c. Cài đặt mô hình:

Nhóm sẽ cài đặt mô hình từ ban đầu, không finetune, không pre-train, nhóm sẽ cài đặt mô hình Transformer (Encoder) lại từ đầu để tiện so sánh với 2 mô hình còn lại. Cụ thể code cài đặt mô hình ở link github nhóm để ở cuối bài. Dưới đây là tổng quan của mô hình.

```
TransformerEncoderCls(
  (embd_layer): TokenAndPositionEmbedding(
    (word_emb): Embedding(10000, 200)
    (pos_emb): Embedding(100, 200)
  )
  (transformer_layer): TransformerEncoder(
    (attn): MultiheadAttention(
      (out_proj): NonDynamicallyQuantizableLinear(in_features=200, out_features=200, bias=True)
    )
    (ffn): Sequential(
      (0): Linear(in_features=200, out_features=128, bias=True)
      (1): ReLU()
      (2): Linear(in_features=128, out_features=200, bias=True)
    )
    (layernorm_1): LayerNorm((200,), eps=1e-06, elementwise_affine=True)
    (layernorm_2): LayerNorm((200,), eps=1e-06, elementwise_affine=True)
    (dropout_1): Dropout(p=0.1, inplace=False)
    (dropout_2): Dropout(p=0.1, inplace=False)
  )
  (pooling): AvgPool1d(kernel_size=(100,), stride=(100,), padding=(0,))
  (fc1): Linear(in_features=200, out_features=20, bias=True)
  (fc2): Linear(in_features=20, out_features=2, bias=True)
  (dropout): Dropout(p=0.1, inplace=False)
  (relu): ReLU()
)
```

3.4. Mô hình DistilBERT (Fine Tuning).

DistilBERT (Sanh et al., 2019) là một mô hình ngôn ngữ tiền huấn luyện (pretrained language model) dựa trên kiến trúc transformer, mô hình này được sử dụng rộng rãi trong các tác vụ NLP như phân loại văn bản,v.v…

a. Lý do chọn mô hình DistilBERT:

- Kích thước nhỏ hơn: DistilBERT có số lượng tham số ít hơn 40% so với BERT. Điều này khiến DistilBERT trở thành một lựa chọn tốt cho các ứng dụng NLP có yêu cầu về tài nguyên hạn chế, chẳng hạn như các ứng dụng chạy trên thiết bị di động hoặc các ứng dụng có ngân sách hạn chế như google colab bản thường.
- Hiệu suất tương đương: DistilBERT đạt được hiệu suất tương đương BERT trên một số vài nhiệm vụ NLP nhưng với một số lượng tham số ít hơn, bao gồm phân loại văn bản. Điều này khiến DistilBERT trở thành một lựa chọn tốt cho các ứng dụng NLP cần hiệu suất cao nhưng không cần phải sử dụng một mô hình lớn.
- Tốc độ tính toán nhanh hơn: DistilBERT có tốc độ tính toán nhanh hơn BERT (nhanh hơn khoảng 60%). Điều này khiến DistilBERT trở thành một lựa chọn tốt cho các ứng dụng NLP cần xử lý lượng dữ liệu lớn hoặc cần phản hồi nhanh (tính real-time trong ứng dụng)

b. Nguyên lý hoạt động của DistilBERT:

Mô hình DistilBERT học từ mô hình BERT (học theo kiểu teacher forcing) và cập nhật tham số bằng cách dùng một hàm Loss có chứa 3 thành phần là:

- Masked Language Modeling (MLM) Loss: Mô hình DistilBERT học bằng cách dự đoán các từ bị che đi trong nhiệm vụ dự đoán từ bị che. Sau khi đoán xong thì kết quả xác suất đó được so sánh với kết quả của mô hình giáo viên (BERT trong trường hợp này), sử dụng Cross-Entropy loss và lan truyền ngược để cập nhật tham số mô hình DistilBERT
- Distillation Loss: Thật ra ta chỉ cần dùng hàm loss cho mô hình học sinh DistilBERT thôi là đủ, tuy nhiên vấn đề nằm ở hàm Softmax với tham số nhiệt độ T được cho là bằng 1 (không phù hợp với bài toán có hai hoặc nhiều nhãn phân loại hợp lệ cho một input nào đó), với T = 1 thì kết quả trả ra qua hàm Softmax kém đa dạng hơn. Do đó ta cần Distillation Loss để mô hình học sinh trả ra nhiều câu trả lời hơn.
- Similarity Loss: Thêm vào hàm Cosine similarity loss cũng khiến mô hình DistilBERT học tốt hơn bởi vì nhờ vào hàm loss này mà mô hình có thể tái tạo các embedding giống với BERT

Sau cùng, một tổ hợp tuyến tính của 3 hàm Loss vừa đề cập chính là hàm Loss để huấn luyện mô hình DistilBERT. Và dựa vào giá trị Loss, lan truyền ngược được thực hiện để mô hình DistilBERT cập nhật lại tham số của nó.

c. Cài đặt mô hình:

Đối với mô hình DistilBERT, nhóm sẽ thực hiện Fine Tuning trên bộ dữ liệu của nhóm để mô hình chạy tốt hơn. Cụ thể hơn nhóm sẽ train với batch size bằng 4, epoch bằng 1, learning rate = 0.00001. Còn cụ thể hơn nữa mô hình được cài đặt thế nào thì nhóm để trong code ở mục 5 (Phần Extra). Cấu trúc mô hình như sau:

```
DistilBERTClass(
  (l1): DistilBertModel(
    (embeddings): Embeddings(
      (word_embeddings): Embedding(30522, 768, padding_idx=0)
      (position_embeddings): Embedding(512, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (transformer): Transformer(
      (layer): ModuleList(
        (0-5): 6 x TransformerBlock(
          (attention): MultiHeadSelfAttention(
            (dropout): Dropout(p=0.1, inplace=False)
            (q_lin): Linear(in_features=768, out_features=768, bias=True)
            (k_lin): Linear(in_features=768, out_features=768, bias=True)
            (v_lin): Linear(in_features=768, out_features=768, bias=True)
            (out_lin): Linear(in_features=768, out_features=768, bias=True)
          )
          (sa_layer_norm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
          (ffn): FFN(
            (dropout): Dropout(p=0.1, inplace=False)
            (lin1): Linear(in_features=768, out_features=3072, bias=True)
            (lin2): Linear(in_features=3072, out_features=768, bias=True)
            (activation): GELUActivation()
          )
          (output_layer_norm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
        )
      )
    )
  )
  (pre_classifier): Linear(in_features=768, out_features=768, bias=True)
  (dropout): Dropout(p=0.1, inplace=False)
  (classifier): Linear(in_features=768, out_features=1, bias=True)
)
```

Chương 4. Phân tích và đánh giá kết quả

1. Phương pháp đánh giá:

a. Precision

Precision là một chỉ số của hiệu suất mô hình học máy - chất lượng của một dự đoán được thực hiện bởi mô hình. Precision đề cập đến số lượng dự đoán tích cực đúng chia cho tổng số dự đoán tích cực (tức là số lượng dự đoán tích cực đúng cộng với số lượng dự đoán tích cực sai).

$$\text{Công thức: } Precision = \frac{TP}{TP + FP}$$

Trong đó:

- TP (True Positives): Số lượng trường hợp dự đoán tích cực đúng, tức là mô hình đúng khi dự đoán một điểm dữ liệu là tích cực.
- FP (False Positives): Số lượng trường hợp dự đoán tích cực sai, tức là mô hình sai khi dự đoán một điểm dữ liệu không phải là tích cực.

Nếu precision là 1, tức là không có false positives nào (tất cả các dự đoán tích cực đều đúng). Nếu precision là 0, tức là không có true positives nào (tất cả các dự đoán tích cực đều sai).

b. Recall

Recall là một độ đo đánh giá tần suất mà một mô hình học máy xác định đúng các trường hợp tích cực (true positives) so với tất cả các mẫu tích cực thực sự trong bộ dữ liệu. Bạn có thể tính recall bằng cách chia số lượng true positives cho số lượng trường hợp tích cực

$$\text{Công thức: } Recall = \frac{TP}{TP + FN}$$

Trong đó:

- TP (True Positives) là số lượng trường hợp dự đoán tích cực đúng, tức là mô hình đúng khi dự đoán một điểm dữ liệu là tích cực.
- FN (False Negatives) là số lượng trường hợp dự đoán tiêu cực sai, tức là mô hình sai khi dự đoán một điểm dữ liệu là tiêu cực, trong khi thực tế nó là tích cực.

Nếu giá trị của Recall là cao (gần 1), điều này nghĩa là mô hình của bạn có xu hướng bao quát và nhận diện được hầu hết hoặc tất cả các trường hợp tích cực thực sự. Tuy nhiên, có một khía cạnh cần lưu ý: mô hình có thể dự đoán nhầm một số trường hợp tiêu cực, đặc biệt nếu nó quá cảnh báo và hiểu sai nhiều trường hợp là tích cực.

c. Accuracy

Accuracy là một thước đo đánh giá đo lường tần suất mà mô hình học máy dự đoán đúng kết quả. Bạn có thể tính độ chính xác bằng cách chia số lần dự đoán đúng cho tổng số lần dự đoán.

$$\text{Công thức: } Accuracy = \frac{\text{Số lần dự đoán đúng}}{\text{Tổng số lần dự đoán}} = \frac{TP + TN}{TP + TN + FP + FN}$$

Trong đó:

- TP (True Positives) là số lượng trường hợp dự đoán tích cực đúng, tức là mô hình đúng khi dự đoán một điểm dữ liệu là tích cực.
- TN (True Negatives) là số lượng trường hợp dự đoán tiêu cực đúng, tức là mô hình đúng khi dự đoán một điểm dữ liệu là tiêu cực.
- FN (False Negatives) là số lượng trường hợp dự đoán tiêu cực sai, tức là mô hình sai khi dự đoán một điểm dữ liệu là tiêu cực, trong khi thực tế nó là tích cực.
- FP (False Positives): Số lượng trường hợp dự đoán tích cực sai, tức là mô hình sai khi dự đoán một điểm dữ liệu không phải là tích cực.

d. F1 Score

F1 Score là một đo lường của trung bình điều hòa giữa độ chính xác (precision) và độ bao quát (recall). Thường được sử dụng làm tiêu chí đánh giá trong các bài toán phân loại nhị phân và đa lớp, F1 Score tích hợp độ chính xác và độ bao quát thành một thước đo duy nhất để có cái nhìn toàn diện hơn về hiệu suất của mô hình.

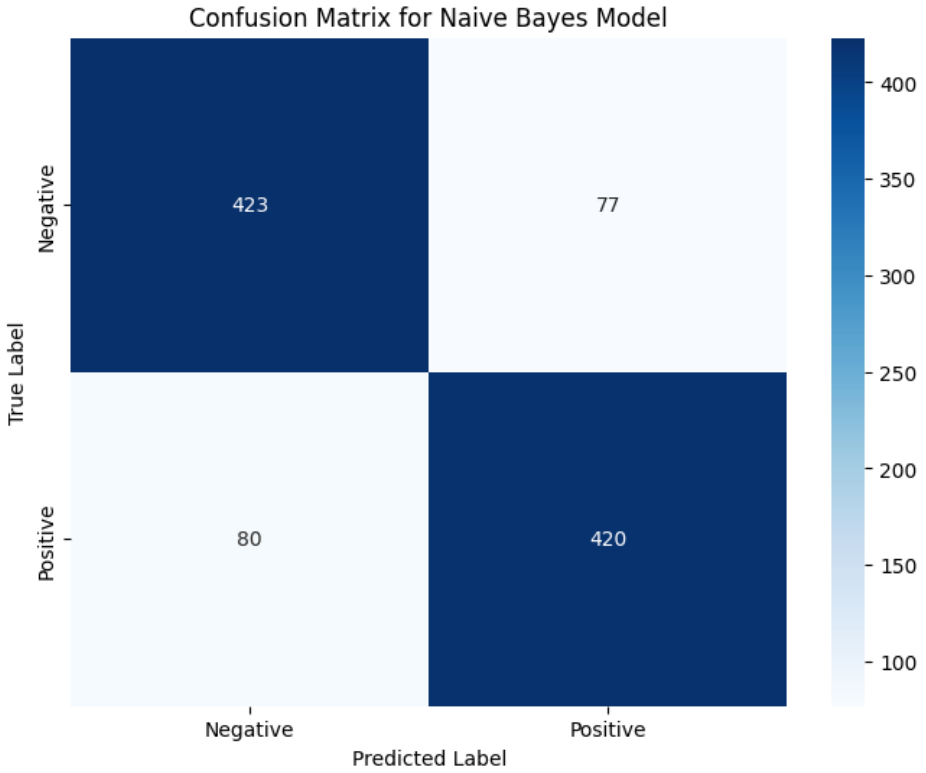
Công thức: $\frac{2}{F1} = \frac{1}{Precision} + \frac{1}{Recall}$

2. Phân tích kết quả thu được từ mô hình

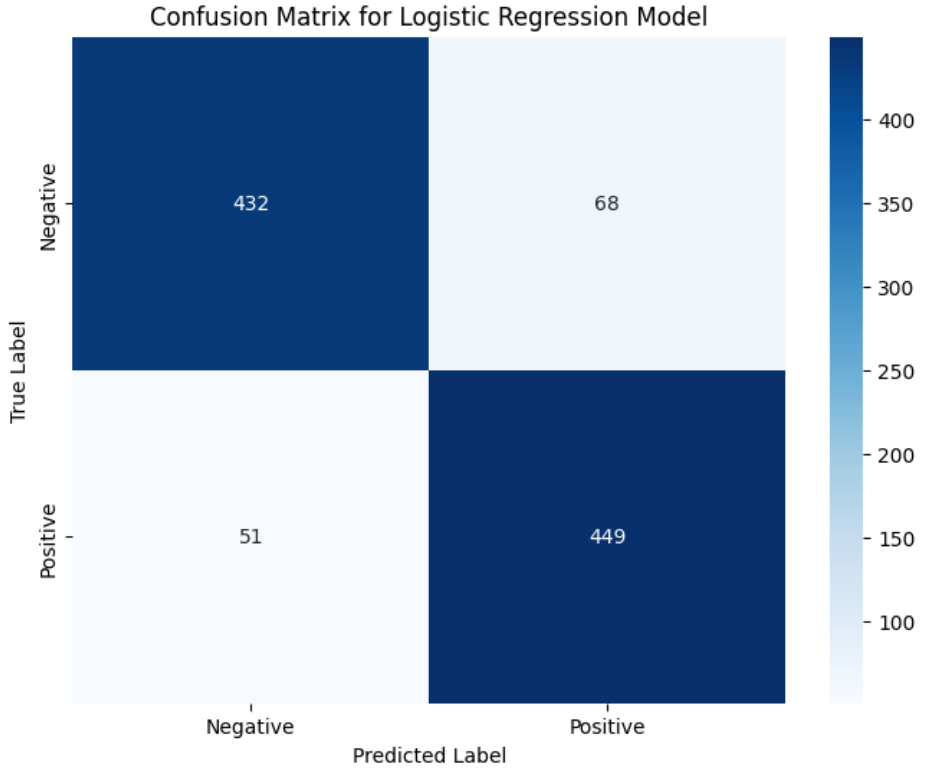
a. Bằng các thước đo hệ số:

Model	Accuracy	Precision	Recall	F1-Score
Model DB	0.90	0.94	0.86	0.90
Model LR	0.88	0.87	0.90	0.88
Model NB	0.84	0.85	0.84	0.84
Model TC	0.75	0.69	0.91	0.78

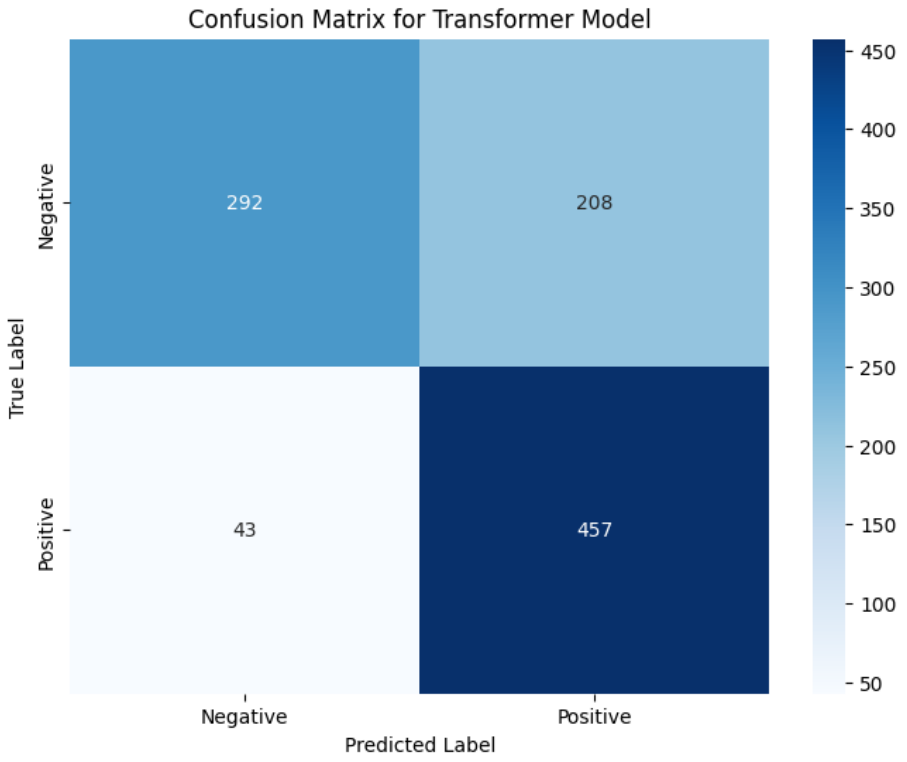
b. Bằng confusion matrix:



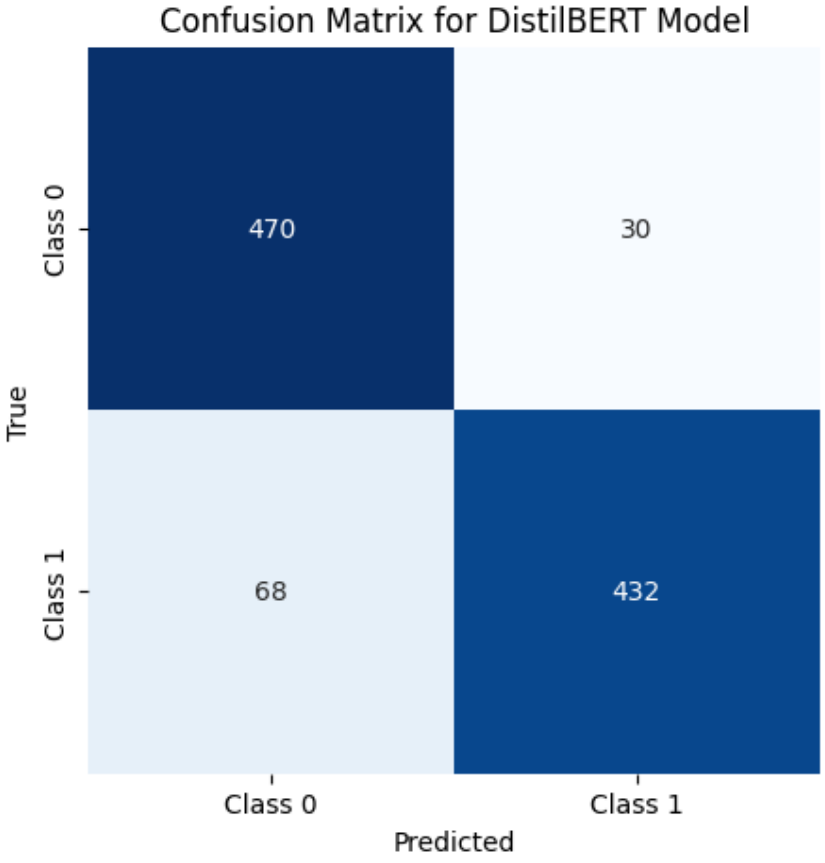
Với mô hình Naive Bayes, ta thấy tỷ lệ sai lầm loại I và sai lầm loại II của mô hình là không cao, có thể thấy rằng mô hình khá là đáng tin cậy.



Với mô hình Logistic Regression, tỷ lệ mắc sai lầm loại I và loại II là thấp hơn của Naive Bayes, tính tới thời điểm hiện tại thì mô hình này mô hình đáng tin nhất.



Mô hình Transformer thì không được tốt như 2 mô hình kia khi mà tỷ lệ mắc sai lầm loại I của mô hình này là cao nhất. Do đó mô hình này là mô hình ít được tin cậy nhất.



Mô hình DistilBERT là mô hình tốt nhất của nhóm tính tới thời điểm hiện tại khi có cả sai lầm loại I và sai lầm loại II đều có tỷ lệ thấp. Do đó đây là mô hình đáng tin tưởng nhất.

3. **Đánh giá hiệu suất của mô hình:**

Dựa vào các kết quả trên, ta có thể thấy rằng mô hình MaxEnt (Logistic Regression) là mô hình hoạt động hiệu quả nhất trong các mô hình mà nhóm đề xuất, điều này không đồng nghĩa với việc những mô hình khác không tốt.

Thực tế, 3 mô hình chênh lệch với nhau không quá nhiều nhưng điểm chung của mô hình này là đều có các chỉ số trên 0.75 (một dấu hiệu tích cực). Tuy nhiên mô hình Transformer (Model TC) lại là mô hình có thời gian huấn luyện lâu nhất nhưng lại không cho ra kết quả như mong muốn. Các mô hình còn lại như mô hình Logistic Regression hay mô hình Naive Bayes có thời gian huấn luyện thấp hơn nhưng kết quả là rất tốt. Lấy ví dụ như Naive Bayes với giả định độc lập như đã nói ở trên là vốn không phù hợp cho các bài toán về xử lý ngôn ngữ tự nhiên, song vẫn đạt được độ hiệu quả cao. Còn về mô hình Transformer, ta có thể nhận xét rằng mô hình Transformer không hiệu quả với dữ liệu tương đối nhỏ, nhưng Transformer có thể làm rất tốt nếu ta tăng dữ liệu lên. Nhóm đã có thử với cũng từ bộ dữ liệu này, nhưng thay vì rút xuống còn 40.000 samples cho tập train thì trước đó nhóm rút xuống còn 180.000 sample và nhận thấy chỉ sau 3 epoch, accuracy của Transformer từ 51.4% đã lên 89.1% và sẽ còn lên nữa nếu nhóm có đủ tài nguyên cũng như sử dụng toàn bộ dữ liệu gốc (3.600.000 bản ghi).

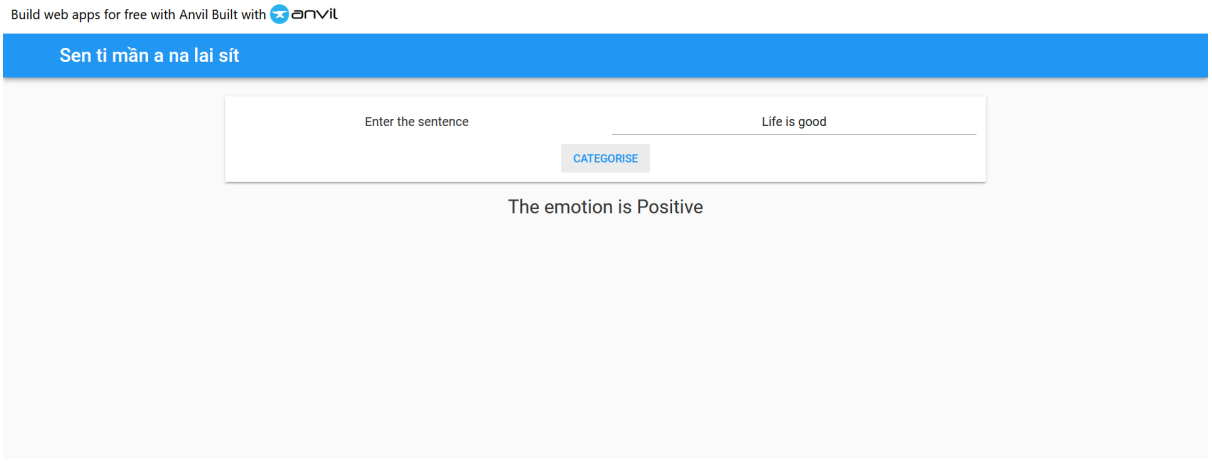
Nhóm sẽ không so sánh mô hình DistilBERT chung với 3 mô hình kia bởi vì nhóm không chạy lại từ đầu mô hình này, tuy nhiên ta có thể thấy rằng hiệu quả của fine tuning là không thể chối cãi. Trong quá trình fine tune mô hình DistilBERT, có một quan sát thú vị là giá trị training loss của DistilBERT không giảm dần, mà thay vào đó giá trị này tăng giảm một cách bất thường. Do đó nhóm sẽ lưu lại giá trị của model nếu như giá trị loss giảm xuống dưới mức threshold mà nhóm chọn là 0.07. Như vậy, để so sánh thì DistilBERT nếu tính thời gian fine tune với hết 1 epoch hoặc đạt điều kiện dừng thì nhanh hơn nhiều so với việc ta train lại từ đầu bằng mô hình Transformer. Trong khi đó DistilBERT lại cho ra kết quả tốt hơn (mô hình DistilBERT sau khi qua bước fine tune là mô hình tốt nhất của nhóm).

Chương 5. Ứng dụng vào thực tế

1. Ứng dụng 1:

Với ứng dụng đầu tiên, nhóm sẽ host trên Anvil, một nền tảng cho phép thiết kế GUI dựa trên file colab của nhóm. Ứng dụng số 1 của nhóm sẽ sử dụng kết quả dự đoán từ cả 3 mô hình mà nhóm thiết kế ở trên để cùng nhau đưa ra kết quả cuối cùng. Ý tưởng cốt lõi là "tin vào số đông", tức là nếu ⅔ mô hình cùng nhau vote rằng câu đầu vào sẽ là một câu mang ý nghĩa tích cực, thì có nghĩa là câu đó là tích cực, và ngược lại, nếu ⅔ mô hình tin câu đầu vào là tiêu cực thì câu đó là tiêu cực.

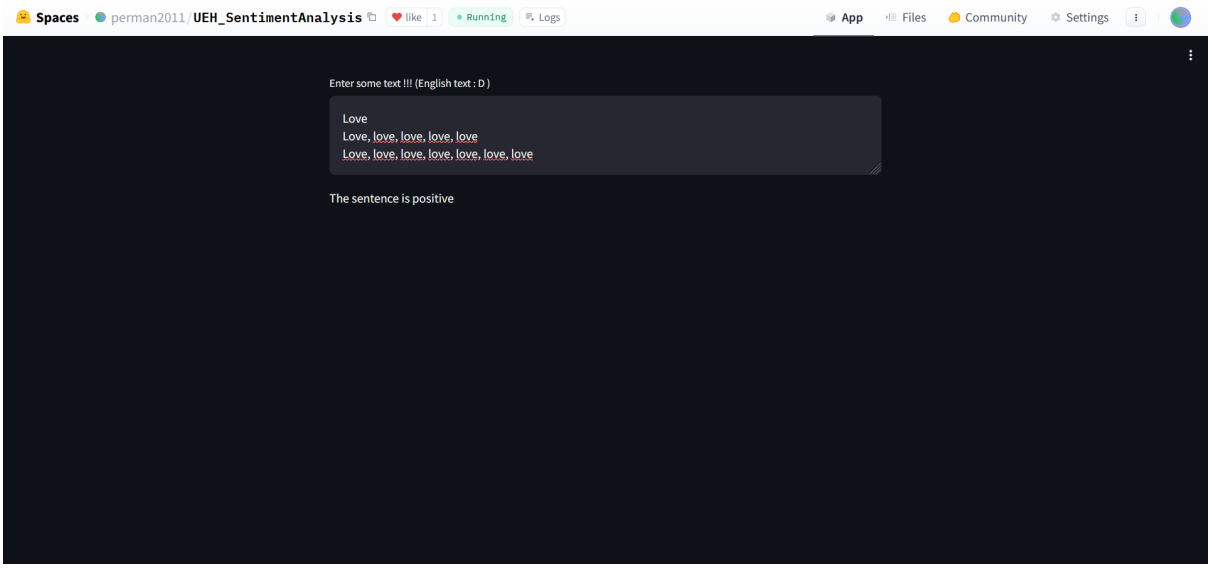
Với việc sử dụng phương pháp major voting, có thể thấy rằng kết quả trả ra rất đáng tin cậy. Tuy nhiên ứng dụng 1 có một hạn chế, do Anvil sẽ link server từ colab. Nên nếu như sau một thời gian, colab sẽ disconnect và delete runtime, điều này sẽ làm cho app tạm thời sập, và sập tới khi nào mà nhóm bật lại cái colab. Vậy trường hợp với ứng dụng 1 là ta sẽ đánh đổi độ chính xác, tốc độ của việc dự đoán với thời gian mà cái app tồn tại. Demo của app như ở dưới đây (Link app):



Ứng dụng này yêu cầu người dùng nhập vào trong ô textbox, sau khi đã nhập xong, người dùng click vào ô "CATEGORISE" và sẽ trả ra kết quả là cảm xúc của câu đó. Và để đạt được kết quả tốt nhất thì người dùng nên nhập vào tiếng anh

2. Ứng dụng 2:

Sau một khoảng thời gian, colab ngắt kết nối, ứng dụng 1 cũng ngắt kết nối theo. Mặc dù chạy rất tốt, rất nhanh, nhưng không vĩnh viễn chính là nhược điểm rất lớn của ứng dụng 1. Hiểu được điều này nên nhóm đã xây dựng thêm một ứng dụng số 2, ứng dụng này sẽ được nhóm tải lên huggingface spaces để ứng dụng sống lâu hơn. Tuy nhiên ứng dụng số 2 này thì lại không được mạnh như ứng dụng 1 (thời gian chạy lâu hơn, không thể biểu diễn major voting được như ứng dụng 1). Một phần bởi vì với ứng dụng số 2 này, nhóm sẽ chỉ sử dụng mô hình MaxEnt để đưa ra kết quả cuối cùng (bởi vì mô hình này đang là mô hình tốt nhất). Vậy trường hợp với ứng dụng 2 là ta sẽ đánh đổi thời gian mà cái app tồn tại với độ chính xác, tốc độ của việc dự đoán. Demo của app như ở dưới đây (Link app):



Ứng dụng này yêu cầu người dùng nhập vào một câu và ấn tổ hợp Ctrl + Enter để chạy. Và để đạt được kết quả tốt nhất thì người dùng nên nhập vào tiếng anh.

BẢNG PHÂN CÔNG

STT	Họ Tên	Công việc phụ trách	Mức độ hoàn thành
1	Trần Hoàng Trung Đức	- C3 - Naive Bayes - C4 - Naive Bayes - C5	**100%**
2	Nguyễn Nhật Quang	- C2-Xử lý nhãn lỗi - C2-EDA - C3-Transformer - C3-DistilBERT - C4-Transformer - C4-DistilBERT - C5	**100%**
3	Ngô Gia Bảo	- C3-Logistic Regression - C4-Logistic Regression - C5	**100%**
4	Vương Chí Bình	- C1 - C2- Tiền xử lý - C4- Phương pháp đánh giá	**100%**

Note: Trong repo github không có file ‘model_DB_1’ do file này nặng quá (200MB) nên không tải lên github được. Nhưng cũng không sao vì mọi phần data, weight của model đã được nhóm up lên google drive và tải xuống bằng lệnh !gdown nên về cơ bản khi chạy code chỉ cần file .ipynb là đủ.

Link github: https://github.com/ngnquang/NLP_Final

Trích dẫn

Accuracy vs. precision vs. recall in machine learning: what's the difference? (n.d.). Evidently AI. Retrieved December 10, 2023, from https://www.evidentlyai.com/classification-metrics/accuracy-precision-recall

Accuracy vs. precision vs. recall in machine learning: what's the difference? (n.d.). Evidently AI. Retrieved December 10, 2023, from https://www.evidentlyai.com/classification-metrics/accuracy-precision-recall

Loukas, S. (2020, October 12). *Text Classification Using Naive Bayes: Theory & A Working Example*. Towards Data Science. Retrieved December 10, 2023, from https://towardsdatascience.com/text-classification-using-naive-bayes-theory-a-working-example-2ef4b7eb7d5a

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019, October 2). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv*. https://arxiv.org/abs/1910.01108

Sharma, N. (2023, June 10). *Understanding and Applying F1 Score: A Deep Dive with Hands-On Coding*. Arize AI. Retrieved December 10, 2023, from https://arize.com/blog-course/f1-score/

Smolic, H. (2022, September 16). *Precision Versus Recall - Essential Metrics in Machine Learning*. Graphite Note. Retrieved December 10, 2023, from https://graphite-note.com/precision-versus-recall-machine-learning

Stanford NLP. (2009, 7 4). *Properties of Naive Bayes*. Stanford NLP Group. Retrieved December 10, 2023, from https://nlp.stanford.edu/IR-book/html/htmledition/properties-of-naive-bayes-1.html

Vaswani. (2017, June 12). [1706.03762] Attention Is All You Need. *arXiv*. https://arxiv.org/abs/1706.03762