TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN CUỐI KỲ MÔN XỬ LÝ NGÔN NGỮ TỰ NHIÊN

XÂY DỰNG HỆ THỐNG PHÂN LOẠI Ý KIẾN NGƯỜI DÙNG DỰA TRÊN CÁC MÔ HÌNH MÁY HỌC

Người hướng dẫn: **GV NGUYỄN TUẤN ĐĂNG**
Người thực hiện: **TRẦN MINH TRÍ – 52000815**
MAI QUỐC THẮNG – 52000802
NGÔ TRÁC HI - 52000755
Lớp : **200503401**
Khóa : **24**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN CUỐI KỲ MÔN XỬ LÝ NGÔN NGỮ TỰ NHIÊN

XÂY DỰNG HỆ THỐNG PHÂN LOẠI Ý KIẾN NGƯỜI DÙNG DỰA TRÊN CÁC MÔ HÌNH MÁY HỌC

Người hướng dẫn: **GV NGUYỄN TUẤN ĐĂNG**
Người thực hiện: **TRẦN MINH TRÍ – 52000815**
MAI QUỐC THẮNG – 52000802
NGÔ TRÁC HI - 52000755
Lớp : **200503401**
Khóa : **24**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024

LỜI CẢM ƠN

Nhóm xin chân thành cảm ơn thầy Nguyễn Tuấn Đăng đã soạn những giáo trình, bài tập về Xử lý ngôn ngữ tự nhiên. Thông qua những kiến thức cùng với sự phổ cập của các tài liệu liên quan đến môn học đã giúp nhóm dễ dàng hơn trong quá trình học. Trong quá trình thực hiện báo cáo này sẽ không tránh khỏi sai sót, mong thầy góp ý và bỏ qua, nhóm xin cảm ơn.

BÁO CÁO ĐƯỢC HOÀN THÀNH
TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong Khóa luận/Đồ án tốt nghiệp còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung Báo cáo cuối kỳ của mình. Trường Đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày 4 tháng 2 năm 2024

Tác giả

Trần Minh Trí

Mai Quốc Thắng

Ngô Trác Hi

TÓM TẮT

Ngày nay, với sự phát triển vượt bật của công nghệ kéo theo mức độ phổ biến của mạng xã hội cũng gia tăng nhanh chóng khiến cho con người càng ngày đắm chìm vào trong thế giới ảo này. Chúng ta sẵn sàng chia sẻ cảm xúc và ý kiến của mình trên các trang truyền thông, mạng xã hội, … để cho người khác biết mình đang nghĩ gì. Lợi dụng điều này, nhiều công ty đã và đang sử dụng dữ liệu từ các trang web khác nhau để tạo ra thông tin có ý nghĩa từ đó sử dụng vào mục đích kinh doanh của họ. Các dữ liệu văn bản khổng lồ có trên các trang như Google, Facebook, Amazon, IMDB,… là một nguồn tài nguyên vô cùng giá trị và việc phân tích chúng một cách thủ công là một lựa chọn vô cùng tồi tệ, vì nó quá chậm kèm theo nhiều rủi ro, sai sót,... Để giải quyết vấn đề đó, ta sẽ áp dụng các kỹ thuật lập trình để thu thập, trích xuất ý kiến của người dùng từ các phần bình luận, thảo luận, feedback,… và một trong số đó là kỹ thuật *Sentiment Analysis*. Đây là một nhánh con trong *Opinion Mining* mà trong đó việc phân tích sẽ tập trung vào giai đoạn khai thác, rút trích văn bản và ý kiến của người dùng dựa theo một chủ đề cụ thể. Trong đồ án này, ta sử dụng tập dữ liệu đánh giá về phim của IMDB và dùng các mô hình như Naïve Bayes, Support Vector Machine, và Logistic Regression để dự đoán liệu một phản hồi, bình luận của người dùng đánh giá phim là tiêu cực, tích cực hay trung lập. Ngoài ra, các phản hồi từ người dùng đa số sẽ là những câu văn mang thiên hướng sử dụng trong văn nói (gọi là *informal form*), do đó ta sẽ tiếp cận vấn đề bằng cách dùng *n-grams* và *count vectorizer*. Ta thực hiện *tokenization* để biến chuỗi đầu vào thành một véc-tơ từ, áp dụng *stemming* để xác định gốc của các từ, dùng *feature selection* để chọn các "đặc trưng" phù hợp và cuối cùng thực hiện phân loại (*classify*) dự đoán xem phản hồi người dùng mang khuynh hướng tích cực, tiêu cực hay trung lập.

MỤC LỤC

DANH MỤC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

CHƯƠNG 1 – TỔNG QUAN VỀ ĐỀ TÀI

1.1 Mục đích của đề tài

Xem phim là một trong những nhu cầu giải trí của con người nhưng chỉ có một số ít bộ phim là thành công và được đánh giá cao. Có nhiều trang chuyên về đánh giá phim giúp người xem chọn lọc ra những bộ phim mà họ nên xem và ngược lại. Một trong số đó chính là IMDB, việc đánh giá mức độ thành công của bộ phim dựa vào số điểm trên thang 10 thông qua số sao mà người dùng đánh giá. Tuy nhiên, chưa có phương pháp nào thực hiện đánh giá bộ phim qua những phản hồi, bình luận. Vì thế, trong đề tài này, ta sẽ áp dụng kỹ thuật *sentiment analysis* để hiện thực hóa mục tiêu trên.

1.2 Giới thiệu về Sentiment Analysis

Sentiment Analysis (phân tích quan điểm) là quá trình giải nghĩa, phân loại cảm xúc trong văn bản sử dụng kỹ thuật phân tích văn bản. Phân tích quan điểm cho phép các doanh nghiệp xác định cảm nghĩ của người dùng đối với sản phẩm, thương hiệu hoặc dịch vụ thông qua các cuộc trò chuyện trực tuyến, phản hồi (feedback),… Các mô hình phân tích quan điểm không chỉ tập trung vào kết quả (tích cực, tiêu cực, trung lập) mà còn dựa trên cảm giác và cảm xúc (buồn, vui, tức giận,…). Phân tích quan điểm dần trở thành xu thế trong những năm gần đây và ngày càng có nhiều công ty lớn sẵn sàng đầu tư tài nguyên vào để dự đoán kết quả cho doanh nghiệp của họ.

Quá trình phân tích quan điểm gồm: *tokenization, word filtering, stemming* và *classification*. Trong quá trình tokenization, văn bản sẽ được phân đoạn thành các mẫu nhỏ thành các từ, các con số, hoặc các dấu câu. Kế tiếp dữ liệu sẽ được thực thi stemming để loại bỏ các tiền tố và phụ tố để chuyển một từ cụ thể thành từ gốc. Toàn bộ những bước trên còn được gọi là quá trình tiền xử lý dữ liệu, sau quá trình này ta sẽ phân tích tập dữ liệu bằng cách thực hiện phân loại sử dụng các phương pháp học máy như Naïve Baiyes, Support Vector Machine, và Logistic Regression. Cuối cùng ta sẽ chọn ra mô hình tốt nhất dựa trên độ chính xác của chúng, vì vậy ta cần phân tích và tìm ra những đặc trưng mà có ảnh hướng đến kết quả đánh giá phim để phân loại chúng là tích cực, tiêu cực hay trung lập.

CHƯƠNG 2 – KHAI PHÁ DỮ LIỆU

2.1 Mô tả dữ liệu

Bộ dữ liệu được thu thập và sử dụng trong đồ án này là bộ chứa 50000 reviews từ IMDB (Internet Movie Database), được chia đều thành hai tập training và testing (mỗi tập 25000 reviews). Cụ thể, trong mỗi phản hồi từ người xem sẽ chứa đựng ý kiến, nhận định và quan điểm của họ đối với một bộ phim cụ thể và với mỗi bộ phim ta chỉ có khoảng 30 reviews (lí do là vì các đánh giá trong cùng một bộ phim thường sẽ có độ tương quan nhất định). Ngoài ra, tập hợp các bộ phim của tập train và tập test là một tập không giao nhau (disjoint set), tức là trong quá trình huấn luyện mô hình thì có khả năng mô hình ghi nhớ một đoạn, một câu,… (gọi là specific terms) được liên kết với các nhãn nhất định sẽ không ảnh hưởng gì đến kết quả. Về việc gán nhãn ở bước cuối, ta sẽ dựa vào số điểm (score) sau khi dự đoán review để gán nhãn như sau:

$$\text{score} \leq 4\text{: negative}$$

$$\text{score} \geq 7\text{: positive}$$

$$4 < \text{score} < 7\text{: neutral}$$

2.2 Tiền xử lý dữ liệu

Để mô hình đạt được hiệu suất tốt, ta cần thực hiện một số phương pháp để sàn lọc dữ liệu nhằm loại bỏ những dữ liệu bị nhiễu (noise), tức những dữ liệu không cần thiết để giúp quá trình phân loại chính xác hơn. Cụ thể trong báo cáo này, đối với dữ liệu ta đang xử lý thì nó bao gồm các bước như sau: loại bỏ các ký tự HTML, lemmatization, loại bỏ stop words và các ký tự đặc biệt, và tokenize.

2.2.1 Loại bỏ các thẻ HTML

Trong hầu hết các trường hợp khi dữ liệu được thu thập (crawling) từ các trang web sẽ không tránh khỏi việc sẽ có những thẻ HTML dư thừa trong văn bản. Điều này có thể ảnh hướng đến kết quả, độ chính xác của mô hình. Bởi vì, các thẻ HTML như (, <p>, <a>, …) vốn dĩ không thuộc phạm trù về mặt ngôn ngữ tự nhiên (ta không sử dụng chúng trong văn nói, văn viết). Do đó, loại bỏ các ký tự này sẽ giúp văn bản "clean" hơn, và cũng đảm bảo cho các bước xử lý, phân tích phía sau trở nên "trơn tru" hơn vì chúng hoàn tập tập trung xử lý trên những văn bản có mang tính ngữ cảnh thay vì những văn bản có đánh dấu in đậm, in nghiêng, được format,…

Để thực hiện loại bỏ các thẻ HTML, ta sẽ dùng regex (regular expressions) để tạo một pattern khớp với các từ nằm trong hai dấu "<" và ">" như đoạn code sau đây:

```
def rmvhtmltags(text):
    remreg = re.compile('<.*?>')
    cleartext = re.sub(remreg, '', text)
    return cleartext
```

trong đó:

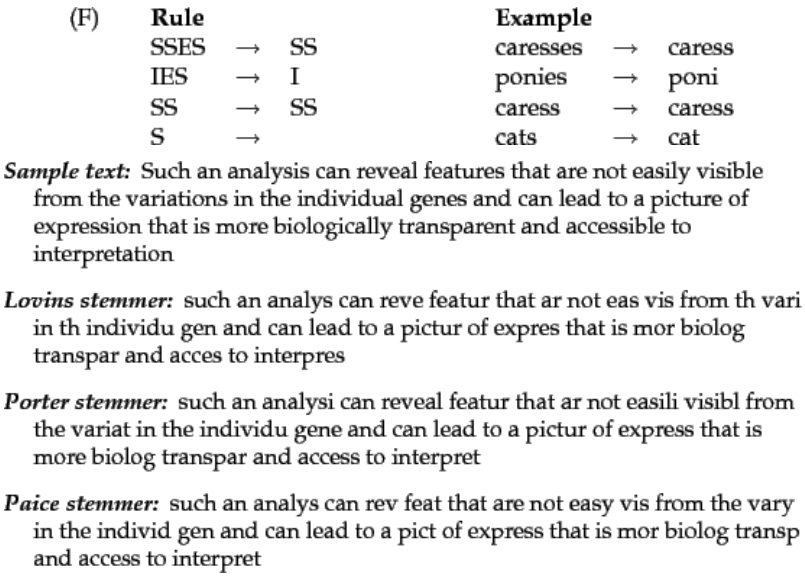
- "<.*?>": Tìm tất cả ký tự mà bắt đầu bằng dấu "<", kết thúc bằng dấu ">" và ".*?" nghĩa là tồn tại 0 hoặc vô số bất kỳ ký tự nào (trừ ký tự xuống dòng)
- Sau đó thay thế toàn bộ những ký tự khớp pattern trên bằng ký tự rỗng (empty string).

2.2.2 Lemmatization

Lemmatization (bổ đề hóa) là một quá trình rút gọn các từ về dạng gốc của chúng (được gọi là bổ đề). Ví dụ:

- Bổ đề của từ "running" là "run".
- Bổ đề của từ "better" là "good".

Mặc dù có cùng mục đích là để rút gọn các từ về dạng cơ bản của chúng. Song, đối với bài toán phân tích quan điểm, ta thường "ưa chuộng" lemmatization hơn là stemming là bởi vì ta cần quan tâm đến ngữ cảnh, ta muốn mô hình "hiểu" được ngữ cảnh của câu văn trong quá trình huấn luyện. Mà cơ chế của stemming là rút gọn bằng cách sử dụng các rules (thuật toán Porter năm 1980) như sau:


(F)

Rule			Example		
SSES	→	SS	caresses	→	caress
IES	→	I	ponies	→	poni
SS	→	SS	caress	→	caress
S	→		cats	→	cat

Sample text: Such an analysis can reveal features that are not easily visible from the variations in the individual genes and can lead to a picture of expression that is more biologically transparent and accessible to interpretation

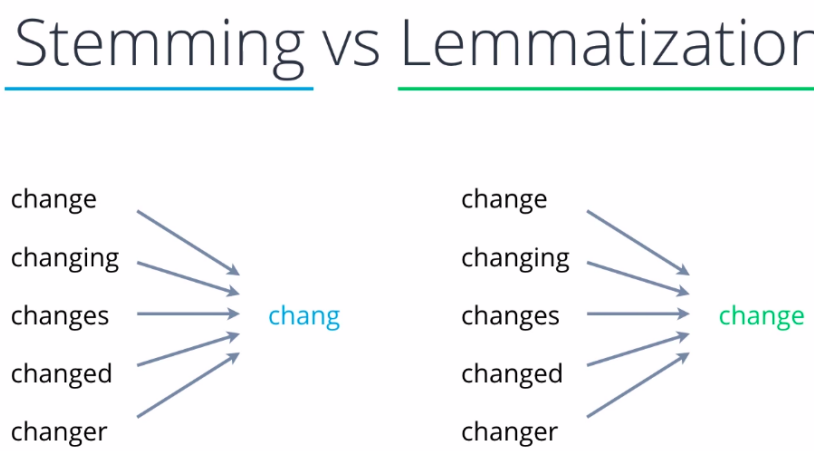
Lovins stemmer: such an analys can reve featur that ar not eas vis from th vari in th individu gen and can lead to a pictur of expres that is mor biolog transpar and acces to interpres

Porter stemmer: such an analysi can reveal featur that ar not easili visibl from the variat in the individu gene and can lead to a pictur of express that is more biolog transpar and access to interpret

Paice stemmer: such an analys can rev feat that are not easy vis from the vary in the individ gen and can lead to a pict of express that is mor biolog transp and access to interpret

Hình 2. 1 Minh họa kỹ thuật Stemming (nguồn: nlp.stanford.edu)

Ta thấy, stemming chỉ tập trung vào việc lược bỏ các prefix và suffix, điều này dẫn tới việc mất tính ổn định về mặt ngữ pháp (Part-of-Speech). Lấy ví dụ, từ "running" đóng hai vai trò vừa là động từ và danh từ, nhưng cơ chế của stemming đều sẽ đưa cả hai về dạng gốc là "run", còn đối với lemmatization động từ "running" sẽ được đưa về "run" trong khi danh từ "running" thì vẫn được giữ nguyên. Tóm lại, nếu cân nhắc về khía cạnh ngữ cảnh của từng từ thì sử dụng lemmatization sẽ tốt hơn



Hình 2. 2 Sự khác biệt giữa kỹ thuật Stemming và Lemmatization (nguồn: medium.com)

Trong đồ án này, ta sử dụng hàm WordNetLemmatizer từ thư viện nltk để thực hiện lemmatize các từ:

```
def lemmatize_words(text):
    lemmatized_words = [lemmatizer.lemmatize(word, 'v') for word in text.split()]
    return ('  '.join(lemmatized_words))
```

2.2.3 Loại bỏ stop words và các ký tự đặc biệt

Trong xử lý ngôn ngữ tự nhiên, stop words là những từ không mang nhiều ý nghĩa, và việc có hay không có chúng cũng chẳng ảnh hưởng gì đến ý nghĩa của câu. Dĩ nhiên, đối với từng loại ngôn ngữ khác nhau thì stop words cũng sẽ khác nhau nhưng về mặt bản chất thì chúng thường là những mạo từ, dấu câu (ví dụ: trong Tiếng Anh thì các mạo từ là 'a', 'an', 'the', 'this', 'that', 'these', 'those',… là các stop words). Việc loại bỏ các stop words sẽ giúp mô hình khi huấn luyện sẽ chỉ quan tâm đến những từ có ý nghĩa, không bị ảnh hưởng bởi các dữ liệu nhiễu giúp tăng hiệu suất, cũng như độ chính xác của mô hình. Lấy ví dụ, ta có câu "The quick, brown fox jumps over the lazy dog!" có thể được viết rút gọn lại thành "quick brown fox jumps lazy dog". Tức ta có thể loại

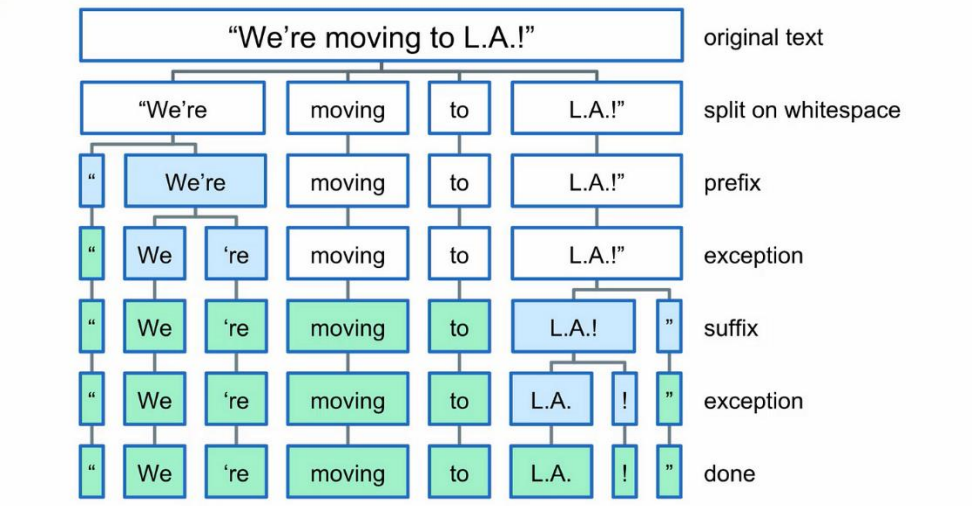
bỏ đi những mạo từ, dấu câu (“the”, “over”, “!”) mà vẫn giữ nguyên được ý nghĩa của câu.

Trong báo cáo này, ta sẽ dùng bộ stop words trong tiếng Anh được cung cấp bởi thư viện nltk. Sau đây là hàm dùng để loại bỏ các stop words và các ký tự đặc biệt:

```
def rmvspclcharacter(text):
    clearspcl = re.sub(r'[^A-Za-z0-9\s.]', r'', str(text).lower())
    clearspcl = re.sub(r'\n', r' ', clearspcl)
    clearspcl = " ".join([word for word in clearspcl.split() if word not in stopWords])
    return clearspcl
```

2.2.4 Text tokenization

Text tokenization (mã hóa văn bản) là quá trình chia nhỏ cụm từ, câu văn, hoặc đoạn văn,… thành các mẫu nhỏ hơn được gọi là token. Mỗi token có thể là một ký tự, một dấu câu, một từ, một câu văn,… tùy thuộc vào từng yêu cầu của bài toán cụ thể. Sau đây là ví dụ minh họa về quá trình mã hóa:



Hình 2. 3 Minh họa quá trình tokenize (nguồn: media.com)

Mục đích của việc mã hóa là để giúp mô hình hiểu cấu trúc của ngôn ngữ cấu thành văn bản. Việc chia nhỏ câu thành từng đơn vị để dễ dàng phân tích ngữ nghĩa, cấu trúc cú pháp và các mối quan hệ ngữ nghĩa từ đó giúp tăng độ hiệu quả trong việc indexing và searching (đối với bài toán information retrieval) và tiết kiệm nguồn tài nguyên của máy tính.

Ở đây ta sử dụng thư viện NLTK vì nó hỗ trợ rất nhiều ngôn ngữ như Tiếng Anh, Pháp, Tây Ban Nha,… Ngoài ra, trong NLTK, word tokenization là một hàm wrapper sử dụng treebank tokenization và nó phân tách cả các dấu câu (tức mỗi dấu câu cũng là một token, ngoại trừ dấu chấm).

Sau đây là quá trình thực hiện tiền xử lý dữ liệu đối với bài toán phân tích quan điểm với bộ dữ liệu IMDB, trong đó bước cuối ta sẽ thực hiện tokenize thông qua hàm có sẵn trong thư viện nltk:

```
def dataprocessing(x):
    x = rmvhtmltags(x)
    x = remove_urls(x)
    x = x.lower()
    x = rmvspclcharacter(x)
    x = remove_stopwords(x)
    x = strip_punctuation(x)
    x = strip_multiple_whitespaces(x)
    x = lemmatize_words(x)

    x = ' '.join([re.sub(r'\d+', '', i) for i in word_tokenize(x)])
    return x
```

2.3 Trích xuất đặc trưng

Cũng như bao lĩnh vực khác như thị giác máy tính, học sâu,… trích xuất đặc trưng là một bước vô cùng thiết yếu. Đối với các bài toán trong xử lý ngôn ngữ tự nhiên, đặc biệt là phân tích quan điểm người dùng thì mục tiêu của quá trình này là chuyển dữ liệu đầu vào về dạng biểu diễn số để đưa vào các mô hình huấn luyện. Một trong những hướng tiếp cận cho vấn đề này là sử dụng mô hình Bag-of-Words và kỹ thuật Count Vectorizer để triển khai mô hình này, và một số kỹ thuật cao hơn như TF-IDF(Term frequency-Inverse Document Frequency), các phương pháp word embeddings như Word2Vec, Glove,…

2.3.1 Mô hình túi từ sử dụng Count Vectorizer

Mô hình túi từ (Bag of Words) là một mô hình thể hiện một văn bản hoặc tài liệu dưới dạng một tập hợp các từ không có thứ tự, không quan tâm đến ngữ pháp mà chỉ quan tâm đến tần suất của các từ.

Cụ thể, ta sẽ tạo một tập hợp chứa toàn bộ các từ (mỗi từ là duy nhất) có trong tập ngữ liệu (corpus). Mỗi văn bản, tài liệu sẽ được biểu diễn dưới dạng véc-tơ, trong đó mỗi phần tử tương tứng với số lượng xuất hiện của phần tử đó trong văn bản, tài liệu đang xét. Lấy ví dụ, ta có hai câu như sau:

1. “I love NLP”
2. “NLP is fascinating”

thì túi từ của chúng sẽ có dạng như sau: [1, 1, 1, 1, 0, 0, 0, …], trong đó mỗi vị trí sẽ tương ứng với một từ độc lập trong tập từ vựng. Một số thư viện trong python có hỗ trợ ta thực hiện điều này, sau đây là một ví dụ sử dụng sklearn:

```
from sklearn.feature_extraction.text import CountVectorizer

# Example documents
documents = ["I love NLP.", "NLP is fascinating."]

# Create a Count Vectorizer instance
vectorizer = CountVectorizer()

# Fit and transform the documents
X = vectorizer.fit_transform(documents)

# Get the feature names (unique words in the documents)
feature_names = vectorizer.get_feature_names_out()

# Convert the sparse matrix to a dense array for better readability
dense_array = X.toarray()

# Create a DataFrame for better visualization
import pandas as pd
df = pd.DataFrame(dense_array, columns=feature_names)

print(df)
# output
# I   fascinating  is  love  nlp
# 0            0   0     1    1
# 1            1   1     0    1
```

2.3.2 TF-IDF

Cũng giống như BoW, TF-IDF là một kỹ thuật thống kê giúp phản ánh tầm quan trọng của một từ trong tập tài liệu (document) so với tập hợp các tập tài liệu (corpus). Nó là sự kết hợp giữa hai phần, term frequency và inverse document frequecy.

Tần suất thuật ngữ (term frequency) thực chất là tính số lần xuất hiện của một thuật ngữ trong một tài liệu chia cho tổng số thuật ngữ trong tài liệu đó.

$$TF = \frac{\text{number of times the term appears in the document}}{\text{total number of terms in the document}}$$

Inverse document frequency dùng để đánh giá mức độ quan trọng của một từ trong văn bản, tuy nhiên trong văn bản thông thường thì những từ không quan trọng lại có xu hướng xuất hiện với tần suất cao (ví dụ: and, or, in, at, this, that,…). Do đó, ta cần giảm mức độ quan trọng của những từ đó bằng cách sử dụng IDF.

$$IDF = log(\frac{\text{number of the documents in the corpus}}{\text{number of documents in the corpus contain the term}})$$

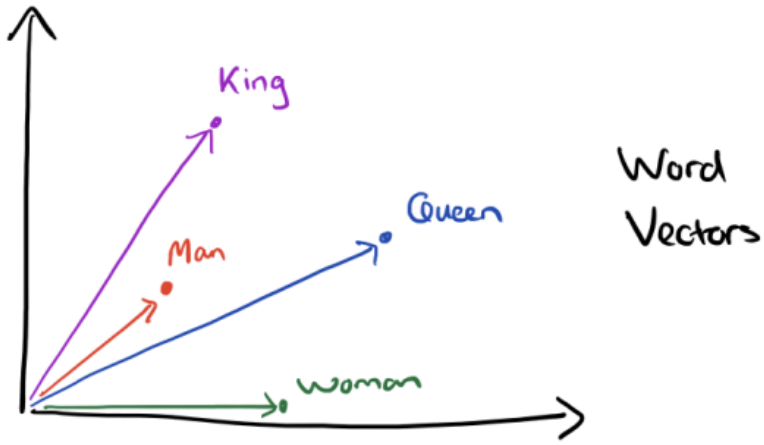
Những từ có TF-IDF cao là những từ xuất hiện trong văn bản này nhiều và xuất hiện ít trong văn bản khác. Việc này giúp lọc ra những từ ngữ phổ biến và giữ lại những từ có giá trị cao trong văn bản

$$TF\text{-}IDF = TF * IDF$$

Tuy nhiên, cũng giống như BoW thì TF-IDF chỉ quan tâm đến tần suất xuất hiện của các từ trong văn bản để đánh giá xem từ nào là thực sự quan trọng, từ nào không nhưng đối với bài toán phân tích quan điểm thì nó lại thiếu một yếu tố quan trọng đó là mối quan hệ ngữ nghĩa và điều này sẽ được khắc phục phần nào thông qua kỹ thuật Word2Vec.

2.3.3 Word2Vec

Đầu tiên ta cần hiểu về kỹ thuật nhúng từ (word embeddings), đây là kỹ thuật biểu diễn các từ ở dạng véc-tơ. Mục tiêu của việc nhúng từ là để điều chỉnh, xác định lại các word features mà có số chiều lớn nhằm giảm số chiều của chúng bằng cách duy trì sự tương đồng về mặt ngữ cảnh trong tập ngữ liệu. Bên dưới là ví dụ về cách hoạt động của véc-tơ từ



Hình 2. 4 Cách hoạt động của véc-tơ từ

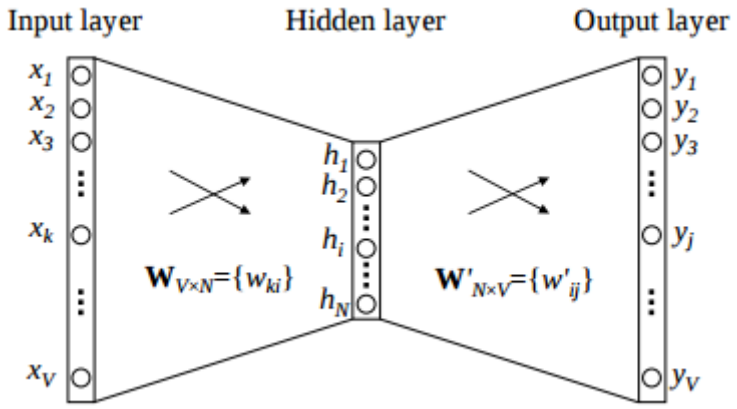
Ta có thể thấy, khi lấy tổng của các véc-tơ (King + (-Man) + Woman) ta thu được véc-tơ Queen, đây chính là sự "huyền diệu" (mind-blowing) của kỹ thuật này. Ưu điểm của kỹ thuật này khi so với các kỹ thuật BoW, TF-IDF có thể kể đến gồm:

- Giảm số chiều: tức ta sẽ giảm được một số lượng đáng kể features để đưa vào xây dựng mô hình.

- Nắm bắt được ý nghĩa của từ, mối quan hệ tương quan về mặt ngữ nghĩa, và ngữ cảnh của câu văn.

Tiếp theo, khi nói đến Word2Vec thực chất nó không phải là một thuật toán độc lập mà là sự kết hợp của hai kỹ thuật: CBOW (Continuous Bag of Words) và Skip-gram. Cụ thể thì cả hai đều là kiến trúc mạng nơ-ron thực hiện ánh xạ các từ tới các biến mục tiêu (cũng là một từ), chúng học cách điều chỉnh các trọng số (weight) đóng vai trò biểu diễn véc-tơ từ.

CBOW dùng để dự đoán xác suất của một từ trong một ngữ cảnh mà ngữ cảnh này có thể là một từ liền kề hay một nhóm các từ xung quanh. Ngược lại, đối với Skip-gram thì nó sẽ dự đoán ngữ cảnh của một từ nhất định. Sau đây là minh họa về mô hình Word2Vec biểu diễn ngữ cảnh của một từ:



Hình 2. 5 Mô hình Word2Vec biểu diễn ngữ cảnh của một từ (nguồn: researchgate.net)

Mô hình gồm 3 layer, trong đó input và output layer là một hot-encoded kích thước 1xV, với V là kích thước của tập từ vựng (số lượng các từ độc lập trong tập ngữ liệu). Tại output layter là một hàm softmax để đưa giá trị xác suất về đoạn [0,1]. Các trọng số W được cập nhật (học) bởi mô hình sẽ được sử dụng như một véc-tơ từ.

Mô hình Skip-gram có thể nắm bắt được hai ngữ nghĩa cho một từ. Lấy ví dụ ta có từ “apple” thì mô hình sẽ biểu diễn hai véc-tơ trong đó, một cái ám chỉ công ty Apple và cái còn lại ám chỉ cho quả táo ở trên cây.

CHƯƠNG 3 – XÂY DỰNG MÔ HÌNH PHÂN LOẠI

Sơ lược: Trong báo cáo này, ta sử dụng ba mô hình gồm: Naïve Bayes, Logistic Regression và Support Vector Machine. Ta sẽ xây dựng mô hình cho từng loại để thực hiện dự đoán xem bộ phim là tích cực, tiêu cực hay trung lập.

Trước khi đi vào huấn luyện ta cần chia bộ dữ liệu thành các tập train và test, ở đây ta sẽ dùng hàm hỗ trợ từ thư viện sklearn để thực hiện điều này:

```
# separating them into lists
y_train_label = reviews_train['Label'].tolist()
x_train_review = reviews_train['review'].tolist()

y_test_label = reviews_test['Label'].tolist()
x_test_review = reviews_test['review'].tolist()

# Split to train and test data
X_train, X_test, y_train, y_test = train_test_split(x_train_review,
y_train_label, test_size=0.3, random_state=42)
```

trong đó:

- x_train_review: Là dữ liệu đầu vào gồm các bình luận, phản hồi của người dùng đã được tiền xử lý.
- y_train_label: Là nhãn tương ứng với dữ liệu đầu vào để xác định quan điểm người dùng (positive, negative hay neutral)
- test_size: Kích thước của tập test, cụ thể 0.3 ở đây tức tập test chiếm tỉ lệ 30% trên toàn bộ tập dữ liệu. Nói cách khác, ta sẽ dùng 30% tập dữ liệu để test và 70% còn lại để training.
- random_state: Là siêu tham số (hyperparameter) được dùng để thiết lập seed cho trình tạo ngẫu nhiên (random generator). Do bản chất của việc tách (split) dữ liệu thành tập dữ liêu huấn luyện (traning data) và tập dữ liệu thử nghiệm (testing set) là ngẫu nhiên, ta sẽ nhận được các dữ liệu khác nhau được gán cho tập dữ liệu huấn luyện và tập dữ liệu thử nghiệm trừ khi ta có thể kiểm soát yếu tố ngẫu nhiên. Với random state = 0, chúng ta nhận được cùng một tập dữ liệu huấn luyện và thử nghiệm trên các lần thực thi khác nhau. Với random state = 42, chúng ta nhận được cùng một tập dữ liệu huấn luyện và thử nghiệm trên các lần thực thi khác nhau, nhưng lần này tập dữ liệu huấn luyện và thử nghiệm khác với trường hợp trước với random state = 0

3.1 Naïve Bayes

3.1.1 Giới thiệu thuật toán

Naïve Bayes là một thuật toán phân loại dựa trên định lý Bayes với giả thuyết về tính độc lập giữa các yếu tố dự đoán (tức là các features). Trong xử lý ngôn ngữ tự nhiên, thuật toán này thường đường sử dụng cho các tác vụ như phân loại văn bản (*text classification*), lọc thư rác (*spam filtering*), phân tích quan điểm (*sentiment analysis*) và phân loại tài liệu (*document categorization*).

Từ "naive" có nghĩa là ngây thơ, tức muốn ám chỉ sự hiện diện của một đặc trưng cụ thể trong một lớp, độc lập với sự hiện diện của các đặc trưng khác. Naïve Bayes thường cho ra kết quả trong thực tế lẫn hiệu quả về mặt tính toán khá tốt…

Naïve Bayes được dùng trong các tác vụ xử lý ngôn ngôn tự nhiên khi ta cần phân loại văn bản thành nhiều loại khác nhau hoặc gán nhãn. Một vài ứng dụng điển hình như:

- Text classification: Gán các danh mục hoặc nhãn được xác định trước cho tài liệu dựa trên nội dung của chúng. Ví dụ: phân loại email là thư rác hoặc không phải thư rác, phân loại các bài báo theo chủ đề hoặc phân loại các bài đánh giá phim theo cảm tính.
- Spam filtering: Xác định email có phải là thư rác hay không dựa trên nội dung và tính năng của nó.
- Sentiment analysis: Phân tích quan điểm được thể hiện trong một đoạn văn bản, chẳng hạn như xác định xem đánh giá phim là tích cực, tiêu cực hay trung lập.
- Document categorization: Sắp xếp tài liệu thành các danh mục được xác định trước dựa trên nội dung của chúng.

Xét về mặt ưu điểm, Naïve Bayes là phương pháp mang tính đơn giản nhưng hiệu quả (performance tốt), có khả năng cải tiến, mở rộng. Cụ thể, việc triển khai thuật toán khá dễ dàng giúp những người không có chuyên môn sâu về lĩnh vực học máy cũng có thể tiếp cận được. Ngoài ra, thuật toán có hiệu quả về mặt tính toán, đặc biệt là so với các thuật toán phức tạp hơn, nó có thể xử lý các tập dữ liệu lớn và không gian đặc trưng nhiều chiều mà không tốn quá nhiều chi phí tính toán và tài nguyên của máy tính.

Về nhược điểm, hạn chế của Naïve Bayes là giả định về tính độc lập của đặc trưng. Mặc dù điều này giúp đơn giản hóa mô hình để tính toán dễ dàng nhưng nó có thể không đúng trong nhiều bộ dữ liệu thực tế (real-world datasets). Bởi vì, trong thực tế các đặc trưng thường tương quan với nhau, do đó đây là yếu điểm có thể kìm hãm hiệu suất của thuật toán khiến chúng không tối ưu. Chính vì thế, thuật toán này đòi hỏi chất lượng dữ liệu đầu vào rất khắt khe, đặc biệt là khi xử lý các đặc trưng nhiễu hoặc không liên quan vì chúng là nguyên nhân chính khiến giả định về tính độc lập bị vi phạm và ảnh hưởng đến hiệu suất.

3.1.2 Phân tích trên phương diện đại số

Xét bài toán classification với C classes 1,2,…, C. Giả sử có một điểm dữ liệu $x \in \mathbb{R}^d$. Hãy tính xác suất để điểm dữ liệu này rơi vào class c. Nói cách khác, hãy tính:

$$p(y = c \mid x) \quad (1)$$

hoặc viết gọn thành $p(c \mid x)$.

Tức tính xác suất để đầu ra là class c biết rằng đầu vào là vector x.

Biểu thức này, nếu tính được, sẽ giúp ta xác định được xác suất để điểm dữ liệu rơi vào mỗi class. Từ đó có thể giúp xác định class của điểm dữ liệu đó bằng cách chọn ra class có xác suất cao nhất:

$$c = \arg\max_{c \in \{1,\dots,C\}} p(c \mid x) \quad (2)$$

Biểu thức (2) thường khó được tính trực tiếp. Thay vào đó, quy tắc Bayes thường được sử dụng:

$$c = \arg\max_c p(c \mid x) \quad (3) = \arg\max_c \frac{p(x \mid c)p(c)}{p(x)} \quad (4) = \arg\max_c p(x \mid c)p(c) \quad (5)$$

Từ (3) sang (4) là vì quy tắc Bayes. Từ (4) sang (5) là vì mẫu số $p(x)$ không phụ thuộc vào c.

Tiếp tục xét biểu thức (5), $p(c)$ có thể được hiểu là xác suất để một điểm rơi vào class c. Giá trị này có thể được tính bằng Maximum Likelihood Estimation, tức tỉ lệ số điểm dữ liệu trong tập training rơi vào class này chia cho tổng số lượng dữ liệu trong tập training; hoặc cũng có thể được đánh giá bằng Maximum a Posteriori. Trường hợp thứ nhất thường được sử dụng nhiều hơn.

Thành phần còn lại $p(x|c)$, tức phân phối của các điểm dữ liệu trong class c, thường rất khó tính toán vì x là một biến ngẫu nhiên nhiều chiều, cần rất nhiều dữ liệu training để có thể xây dựng được phân phối đó. Để giúp cho việc tính toán được đơn giản, người ta thường giả sử một cách đơn giản nhất rằng các thành phần của biến ngẫu nhiên x là độc lập với nhau, nếu biết c (c được cho trước). Tức là:

$$p(x|c) = p(x_1, x_2, ..., x_d|c) = \prod_{i=1}^{d} p(x_i|c) \quad (6)$$

Giả thiết các chiều của dữ liệu độc lập với nhau, nếu biết c, là quá chặt và ít khi tìm được dữ liệu mà các thành phần hoàn toàn độc lập với nhau. Tuy nhiên, giả thiết ngây ngô này lại mang lại những kết quả tốt bất ngờ. Giả thiết về sự độc lập của các chiều dữ liệu này được gọi là Naive Bayes. Cách xác định class của dữ liệu dựa trên giả thiết này có tên là Naive Bayes Classifier (NBC).

NBC, nhờ vào tính đơn giản một cách *ngây thơ*, có tốc độ training và test rất nhanh. Việc này giúp nó mang lại hiệu quả cao trong các bài toán large-scale.

Ở bước training, các phân phối $p(c)$ và $p(x_i|c), i = 1, ..., d$ sẽ được xác định dựa vào training data. Việc xác định các giá trị này có thể dựa vào Maximum Likelihood Estimation hoặc Maximum A Posteriori.

Ở bước test, với một điểm dữ liệu mới x, class của nó sẽ được xác định bởi:

$$c = \arg\max_{c \in \{1,...,C\}} p(c) \prod_{i=1}^{d} p(x_i|c) \quad (7)$$

Khi d lớn và các xác suất nhỏ, biểu thức ở vế phải của (7) sẽ là một số rất nhỏ, khi tính toán có thể gặp sai số. Để giải quyết việc này, (7) thường được viết lại dưới dạng tương đương bằng cách lấy $\log$ của vế phải:

$$c = \arg\max_{c \in \{1,...,C\}} = \log\left(p(c)\right) + \sum_{i=1}^{d} \log\left(p(x_i|c)\right) \quad (7.1)$$

Việc này không ảnh hưởng tới kết quả vì $\log$ là một hàm đồng biến trên tập các số dương.

Mặc dù giả thiết mà Naïve Bayes Classifiers sử dụng là quá phi thực tế, chúng vẫn hoạt động khá hiệu quả trong nhiều bài toán thực tế, đặc biệt là trong các bài toán phân loại văn bản, ví dụ như lọc tin nhắn rác hay lọc email spam. Cả việc training và

test của NBC là cực kỳ nhanh khi so với các phương pháp classification phức tạp khác. Việc giả sử các thành phần trong dữ liệu là độc lập với nhau, nếu biết class, khiến cho việc tính toán mỗi phân phối $p(x_i \mid c)$ trở nên cực kỳ nhanh.

Mỗi giá trị $p(c), c = 1,2,...,C$ có thể được xác định như là tần suất xuất hiện của class c trong training data. Việc tính toán $p(x_i \mid c)$ phụ thuộc vào loại dữ liệu. Có ba loại được sử dụng phổ biến là: Gaussian Naïve Bayes, Multinomial Naïve Baiyes và Bernouli Naïve.

3.1.3 Áp dụng vào bài toán

Trong báo cáo này, ta sẽ chọn Multinomial Naive Bayes làm mô hình để thực hiện *sentiment analysis*, lí do là vì ta đang phân loại văn bản mà feature véc-tơ được tính bằng Bags of Words. Lúc này, mỗi văn bản được biểu diễn bởi một vector có độ dài d chính là số từ trong từ điển. Giá trị của thành phần thứ i trong mỗi vector chính là số lần từ thứ i xuất hiện trong văn bản đó.

Khi đó, $p(x_i \mid c)$ tỉ lệ với tần suất từ thứ i xuất hiện trong các văn bản của class c, nó được tính là tổng của tất cả thành phần thứ i của các feature vectors ứng với class c. Giá trị này có thể được tính bằng cách:

$$\lambda_{ci} = p(x_i \mid c) = \frac{N_{ci}}{N_c} \quad (10)$$

Trong đó:

- N_{ci} là tổng số lần từ thứ i xuất hiện trong các văn bản của class c, nó được tính là tổng của tất cả các thành phần thứ i của các feature fectors ứng với class c.
- N_c là tổng số từ (kể cả lặp) xuất hiện trong class c. Nói cách khác, nó bằng tổng độ dài của toàn bộ các văn bản thuộc vào class c. Có thể suy ra rằng $N_c = \sum_{i=1}^{d} N_{ci}$, từ đó $\sum_{i=1}^{d} \lambda_{ci} = 1$.

Cách tính này có một hạn chế là nếu có một từ mới chưa bao giờ xuất hiện trong class c thì biểu thức (10) sẽ bằng 0, điều này dẫn đến vế phải của (7) bằng 0 bất kể các giá trị còn lại có lớn thế nào. Việc này sẽ dẫn đến kết quả không chính xác.

Để giải quyết việc này, một kỹ thuật được gọi là *Laplace smoothing* được áp dụng:

$$\hat{\lambda}_{ci} = \frac{N_{ci} + \alpha}{N_c + d\alpha} \quad (11)$$

Với a là một số dương, thường bằng 1, để tránh trường hợp tử số bằng 0. Mẫu số được cộng với da để đảm bảo tổng xác suất $\sum_{i=1}^{d} \hat{\lambda}_{ci} = 1$

Như vậy, mỗi class c sẽ được mô tả bởi bộ các số dương có tổng bằng 1:

$$\hat{\lambda}_c = \left\{ \hat{\lambda}_{c1}, \ldots, \hat{\lambda}_{cd} \right\}$$

Sau đây là đoạn code thực hiện quá trình training mô hình Naïve Bayes biến thể Multinomial Naïve Bayes sử dụng Bag of Words:

```
# Naive Bayes with Bag of Words
model_nb_bow = Pipeline([
    ('bow', CountVectorizer()),
    ('clf', MultinomialNB()),
])

# Fit the model
model_nb_bow = model_nb_bow.fit(X_train, y_train)

print("Cross Validation for Naive Bayes on Bag of Words")
cross_val_score(model_nb_bow, X_train, y_train, cv=3)
```

```
Cross Validation for Naive Bayes on Bag of Words

array([0.8416181 , 0.85376307, 0.85222013])
```

Ta có thể thấy, kết quả thu được khá tốt qua 3 lần thực hiện fold và đạt kết quả cao nhất ở lần fold thứ 2. Cụ thể, cross-validation là phương pháp đánh giá độ chính xác của mô hình khi được training qua các bộ nhỏ (subset) trên toàn bộ tập training, do đó nó còn được gọi là validation set.

3.2 Logistic Regression

3.2.1 Giới thiệu thuật toán

Logistic regression là một phương pháp thống kê được sử dụng trong binary classification, trong đó mục tiêu là dự đoán xác suất một thể hiện thuộc về một lớp cụ thể. Mặc dù mang tên hồi quy (regression) nhưng thuật toán được sử dụng để thực hiện phân loại, hàm ý của chữ hồi quy ở đây là thuật toán sử dụng mô hình hồi quy tuyến tính để dự đoán tỉ lệ của xác suất.

Trong kỹ thuật này, biến đầu ra hoặc biến phụ thuộc là nhị phân, đại diện cho hai lớp (ví dụ: tích cực/tiêu cực, thư rác/không phải thư rác, ...). Thuật toán mô hình hóa

mối quan hệ giữa các biến (đặc điểm) độc lập và xác suất của lớp dương bằng cách sử dụng hàm logistic, còn được gọi là hàm sigmoid.

Mô hình này thường được sử dụng trong các bài toán phân loại, cụ thể trong lĩnh vực xử lý ngôn ngữ tự nhiên gồm:

- Sentiment analysis: Phân tích quan điểm (tích cực, tiêu cực, trung lập) được thể hiện trong dữ liệu văn bản, chẳng hạn như đánh giá sản phẩm, bài đăng trên mạng xã hội hoặc đánh giá phim.
- Spam detection: Phân loại email hoặc tin nhắn văn bản là thư rác hoặc không phải thư rác dựa trên nội dung và tính năng của chúng.
- Text categorization: Gán tài liệu hoặc đoạn văn bản vào các danh mục hoặc nhãn được xác định trước dựa trên nội dung của chúng.
- Name entity recognition (ner): Xác định và phân loại các thực thể được đặt tên (chẳng hạn như tên người, tổ chức, địa điểm, v.v.) trong dữ liệu văn bản.
- Topic modeling: Gán chủ đề cho tài liệu hoặc phân đoạn văn bản dựa trên nội dung của chúng.

Các mô hình Logistic Regression mang lại khả năng diễn giải đơn giản hơn khi so với các mô hình học sâu. Các hệ số liên quan đến từng đặc trưng có thể cung cấp cái nhìn sâu sắc về tầm quan trọng về mối quan hệ giữa các đặc trưng và biến mục tiêu. Bên cạnh đó, mô hình cho khả năng tính toán chi phí thấp và được huấn luyện nhanh chóng, phù hợp với các tập dữ liệu lớn và ứng dụng thời gian thực. Ngoài ra, thuật toán xử lý tương đối tốt dữ liệu nhiễu và các đặc trưng không liên quan, đặc biệt khi áp dụng các kỹ thuật regularization như L1 hoặc L2 để ngăn chặn bị overfitting. Tuy nhiên, hạn chế của thuật toán là chỉ có thể dùng để thực hiện binary classification, do đó nó không phải là lựa chọn tốt cho các bài toán như multi-class classification.

3.2.2 Áp dụng vào bài toán

Trong báo cáo này, ta sẽ huấn luyện mô hình Logistic Regression sử dụng TF-IDF để thu được kết quả tốt (so với Bag of Words), quá trình huấn luyện như sau:

```
# Logistic Regression
model_lr = Pipeline([('tfidf', TfidfVectorizer()),
                     ('clf', LogisticRegression()), ])
model_lr = model_lr.fit(X_train, y_train)

print("Cross Validation for Logistic Regression on TF-IDF")
cross_val_score(model_lr, X_train, y_train, cv=3)
```

```
Cross Validation for Logistic Regression on TF-IDF

array([0.87418581, 0.88033602, 0.87793588])
```

Ta có thể thấy kết quả đánh giá cross-validation cho ra khá cao, mô hình cho hiệu suất cao nhất ở subset thứ 2 với 88.03% độ chính xác.

3.3 Support Vector Machine

3.3.1 Giới thiệu thuật toán

Support vector machine (SVM) là một thuật toán học máy sử dụng các mô hình học có giám sát để giải quyết các vấn đề classification, regression, and outlier detection bằng cách thực hiện các phép biến đổi dữ liệu tối ưu nhằm xác định ranh giới giữa các điểm dữ liệu dựa trên các lớp, nhãn hoặc đầu ra được xác định trước.

Giải thích các thông số của hàm SVM:

- Hàm mất mát (Loss function): SVM sử dụng hàm mất mát được gọi là "hinge loss", mục tiêu là tối thiểu hóa sự cực đại hoá khoảng cách từ các điểm dữ liệu đến ranh giới phân chia.
- Tối ưu hóa (Optimization): SVM giải quyết bài toán tối ưu hóa để tìm ra ranh giới phân chia tối ưu bằng cách sử dụng kỹ thuật tối ưu hóa lồi (convex optimization).
- Kernel trick: SVM có thể được mở rộng để xử lý các bài toán phân loại phi tuyến tính bằng cách sử dụng kernel trick. Kernel trick cho phép ánh xạ dữ liệu từ không gian ban đầu sang một không gian đặc trưng cao hơn, nơi mà việc phân loại tuyến tính trở nên khả thi.
- Tham số điều chỉnh (Hyperparameters): SVM có một số tham số quan trọng cần được điều chỉnh như tham số regularization (C), loại kernel, và các siêu tham số khác tùy thuộc vào kernel được sử dụng.
- Hạt nhân (Kernel): Kernel trong SVM là một phần quan trọng, vì nó ảnh hưởng trực tiếp đến khả năng phân loại của mô hình. Một số hạt nhân phổ

biến bao gồm Linear Kernel, Polynomial Kernel và Radial Basis Function (RBF) Kernel.

SVM là một trong những mô hình tốt nhất khi được sử dụng để có thể predict, và phân loại tốt nhất. Sau đây là các phương pháp của mô hình SVM:

Hard Margin SVM: Trong trường hợp này, SVM cố gắng tìm một đường biên phân chia sao cho không có điểm dữ liệu nào rơi vào khoảng cách giữa các lớp (đường biên) và tất cả các điểm dữ liệu đều được phân loại chính xác. Tuy nhiên, điều này chỉ khả thi khi dữ liệu là tuyến tính phân loại được và không bị nhiễu.

***Soft Margin SVM*:** Trong thực tế, dữ liệu thường không phải lúc nào cũng tuyến tính phân loại được và có thể có nhiễu. Soft Margin SVM cho phép một số điểm dữ liệu rơi vào khoảng cách giữa các lớp (đường biên), được gọi là lề mềm (soft margin). Mục tiêu là tối ưu hóa ranh giới phân loại và đồng thời giảm thiểu sự vi phạm lề mềm.

Linear SVM: Là trường hợp đơn giản nhất của SVM, sử dụng một đường biên tuyến tính để phân loại giữa các lớp dữ liệu. Thường được áp dụng cho các tập dữ liệu có thể phân loại tuyến tính được.

Kernel SVM: Khi dữ liệu không phân loại tuyến tính được, SVM có thể sử dụng các hàm kernel để ánh xạ dữ liệu từ không gian đặc trưng ban đầu sang không gian đặc trưng cao hơn mà phân loại tuyến tính trở nên khả thi. Các hàm kernel phổ biến bao gồm Kernel tuyến tính (Linear Kernel), Kernel đa thức (Polynomial Kernel), và Kernel RBF (Radial Basis Function Kernel).

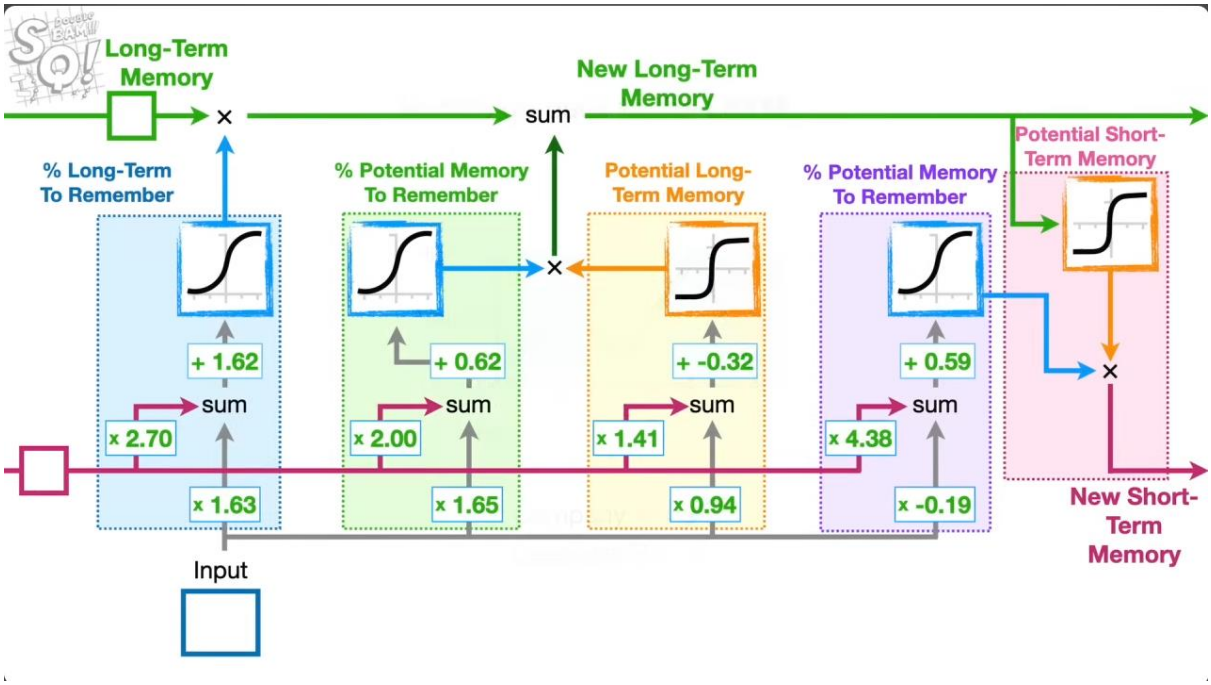
Multi-Class SVM: SVM có thể được mở rộng để giải quyết các bài toán phân loại đa lớp bằng cách sử dụng các phương pháp như One-vs-One hoặc One-vs-All. Sequential Minimal Optimization (SMO): Là một phương pháp tối ưu hóa dành riêng cho SVM, SMO được sử dụng để giải quyết bài toán tối ưu hóa convex lớn kích thước mà không cần phải xây dựng toàn bộ ma trận kernel.

Support Vector Regression (SVR): SVM cũng có thể được sử dụng trong bài toán hồi quy, được gọi là Support Vector Regression (SVR), trong đó mục tiêu là tìm một ranh giới mà tối đa hóa sự cực đại hoá khoảng cách giữa các điểm dữ liệu và đường biên hồi quy.

3.4 LSTM

Long Short-Term Memory (LSTM) là một kiến trúc mạng nơ-ron nhân tạo được thiết kế để giải quyết vấn đề biến mất gradient (Gradient Vanishing/Explode) trong mô hình Recurrent Neural Networks (RNNs). Đối với mô hình xử lý chuỗi dữ liệu có quan hệ về thứ tự, đặc biệt là dữ liệu liên ngôn ngữ, LSTM thường được ưa chuộng do khả năng của nó trong việc duy trì và học các mối quan hệ dài hạn của các phần tử trong dữ liệu chuỗi.

3.4.1 Cấu trúc cơ bản của LSTM



Hình 3. 1 Kiến trúc của mô hình LSTM (nguồn: lightning.ai)

Cổng Quên (Forget Gate): Cổng này quyết định thông tin nào từ trạng thái trước đó sẽ được "quên" hoặc không được chuyển đến trạng thái hiện tại. Nó giúp giải quyết vấn đề biến mất gradient bằng cách cho phép mô hình quyết định loại bỏ hoặc giữ lại thông tin từ quá khứ.

Cổng Đầu Vào (Input Gate): Cổng này quyết định thông tin mới nào sẽ được thêm vào trạng thái hiện tại. Nó giúp mô hình xác định những điểm quan trọng trong dữ liệu mới và giữ lại chỉ những thông tin quan trọng.

Cổng Đầu Ra (Output Gate): Cổng này quyết định trạng thái ẩn mới sẽ làm thế nào ảnh hưởng đến đầu ra. Nó quyết định trạng thái ẩn tiếp theo và là **kết quả dự đoán của Cell hiện tại**.

3.4.2 Công thức cập nhật của LSTM

Dựa vào sơ đồ trên (Figure 1.), ta có thể tổng quát công thức như sau:

Cổng Quên (Forget Gate):

$\mathbf{F_t}$ = σ(Wf * [ht−1, xt] + bf)

Cổng Đầu Vào (Input Gate):

$\mathbf{I_t}$ = σ(Wi * [ht−1, xt] + bi)

Ċt = tanh(WC * [ht−1, xt] + bC)

Cập Nhật Trạng Thái (LONG TERM MEMORY_CELL STATE):

Ct = $\mathbf{F_t}$ * Ct−1 + it * Ċt

Cổng Đầu Ra (Output Gate):

$\mathbf{O_t}$ = σ(Wo * [ht−1, xt] + bo)

Cập Nhật Trạng Thái Ẩn (SHORT TERM MEMORY_HIDDEN STATE):

ht = ot * tanh(Ct)

Trong đó:

- $\mathbf{F_t}$, $\mathbf{I_t}$, $\mathbf{O_t}$ là các hàm toán học tương ứng với các cổng
- b là tham số học Bias.
- σ là hàm kích hoạt sigmoid.
- tanh là hàm kích hoạt tanh.
- Wf, Wi, WC, Wo là các ma trận trọng số và là tham số học.
- [ht−1, xt] là ghép nối của trạng thái ẩn trước đó và đầu vào tại bước thời gian t.

3.4.3 Áp dụng vào bài toán.

Sử dụng thư viện Keras để tạo mô hình LSTM cơ bản:

```
embedding_dim = 100
model_lstm = Sequential()
model_lstm.add(Embedding(input_dim=max_words, output_dim=embedding_dim, input_length=max_len))
model_lstm.add(SpatialDropout1D(0.2))
model_lstm.add(LSTM(100, dropout=0.2))
model_lstm.add(Dense(1, activation='sigmoid'))
print(model_lstm.summary())
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 200, 100)          500000

 spatial_dropout1d (Spatial  (None, 200, 100)          0
 Dropout1D)

 lstm (LSTM)                 (None, 100)               80400

 dense (Dense)               (None, 1)                 101

=================================================================
Total params: 580501 (2.21 MB)
Trainable params: 580501 (2.21 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
```

Mô hình LSTM trên bao gồm:

- Embedding Layer: kỹ thuật Word Embedding từ thư viện keras với
- input_dim: kích thước từ vựng của tập dữ liệu
- output_dim: chiều dài vector của mỗi từ
- input_length: kích thước đầu vào của mỗi câu
- SpatialDropout1D: hoạt động như Dropout nhưng sẽ loại bỏ cả 1 vector thay vì từng phần tử
- LSTM: một block LSTM với kích thước cell + hidden_state là 100
- Dense: một lớp neuron với kích thước 1 phần tử tổng hợp từ output của LSTM

```
# Compile the model
model_lstm.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# Fit the model
batch_size = 64
epochs = 5
model_lstm.fit(X_train_pad, encoded_y_train, batch_size=batch_size, epochs=epochs, validation_data=(X_test_pad, encoded_y_test))
```

```
Epoch 1/5
274/274 [==============================] - 11s 32ms/step - loss: 0.4279 - accuracy: 0.7973 - val_loss: 0.3065 - val_accuracy: 0.8763
Epoch 2/5
274/274 [==============================] - 8s 30ms/step - loss: 0.2535 - accuracy: 0.8993 - val_loss: 0.3591 - val_accuracy: 0.8469
Epoch 3/5
274/274 [==============================] - 9s 31ms/step - loss: 0.2094 - accuracy: 0.9190 - val_loss: 0.3272 - val_accuracy: 0.8652
Epoch 4/5
274/274 [==============================] - 8s 30ms/step - loss: 0.1724 - accuracy: 0.9376 - val_loss: 0.3578 - val_accuracy: 0.8673
Epoch 5/5
274/274 [==============================] - 8s 30ms/step - loss: 0.1281 - accuracy: 0.9557 - val_loss: 0.4037 - val_accuracy: 0.8585

<keras.callbacks.History at 0x1effa6b84c0>
```

```
# Evaluate the model
loss, accuracy = model_lstm.evaluate(X_test_pad, encoded_y_test, verbose=1)
print("Test Accuracy:", accuracy)
```

```
  1/235 [..............................] - ETA: 10s - loss: 0.6055 - accuracy: 0.8750
235/235 [==============================] - 2s 10ms/step - loss: 0.4037 - accuracy: 0.8585
Test Accuracy: 0.8585333228111267
```

Mô hình cho độ chính xác 0.85 tương ứng với 85%.

3.5 Đánh giá và so sánh

Method	Accuracy	Precision	Recall	F1-score
Logistic Regression + TF-IDF	**0.885467**	**0. 885589**	**0.885467**	**0.885448**
SVM + TF-IDF	0.879067	0.879069	0.879067	0.879064
LSTM	0.858533	0.841761	0.885654	0.863150
Naive Bayes + Bag of Words	0.855867	0.856675	0.855867	0.855815

Bảng 3. 1 Đánh giá hiệu của các mô hình

Ta sẽ chọn hai mô hình có hiệu suất cao nhất lần lượt là LR + TF-IDF và SVM + TF-IDF để thực hiện so sánh.

Đầu tiên là về accuracy, ta thấy LR cho kết quả 88.54% trong khi con số là 87.90% ở SVM, điều này cho thấy mô hình LR thực hiện tốt hơn trong việc phân loại dữ liệu test.

Thứ hai là về precision và recall, ta thấy LR cho kết quả 88.55% và SVM cho kết quả là 87.90% cho thấy khả năng nhận diện positive, negative giữa các instances khá giống nhau. Mô hình LR cho kết quả tốt hơn một tí chứng tỏ nó có khả năng ít false positive so với mô hình SVM.

Cuối cùng là F1-score, đây là giá trị trung bình nhằm cân bằng hai chỉ số precision và recall, mô hình LR cho kết quả cao hơn cho thấy nó cân bằng hai giá trị precision và recall tốt hơn so với mô hình SVM.

Nhìn chung, mặc dù cả hai mô hình đều cho kết quả tương đối tốt, nhưng mô hình LR có lợi thế hơn một chút về độ accuracy, precision, recall và F1-score trên tập dữ liệu thử nghiệm. Ngược lại, mô hình SVM cho thấy hiệu suất tốt hơn một chút về độ chính xác trung bình trong quá trình cross-validation.

TÀI LIỆU THAM KHẢO

Tiếng Anh:

[1]: Sentiment Analysis using SVM – Vasista Reddy – 12/11/2018

[2]: Sentiment Analysis using Logistic Regression and Naive Baiyes - Atharva Mashalkar – 28/11/2020

[3]: Sentiment Analysis using LSTM - Samarth Agrawal – 13/03/2019

[4]: An overview of Machine Learning Optimization Techniques - Yulia Gavrilova – 02/12/2020

Tiếng Việt:

[1]: Các bước xây dựng và phương pháp đánh giá mô hình – Huy Bui – 25/04/2022

[2]: Kỹ thuật phân tích quan điểm – VinBigData – 18/05/2022

[3]: Các bài viết về Logistic Regression, SVM, Naive Bayes của tác giả Tiep Vu Huu