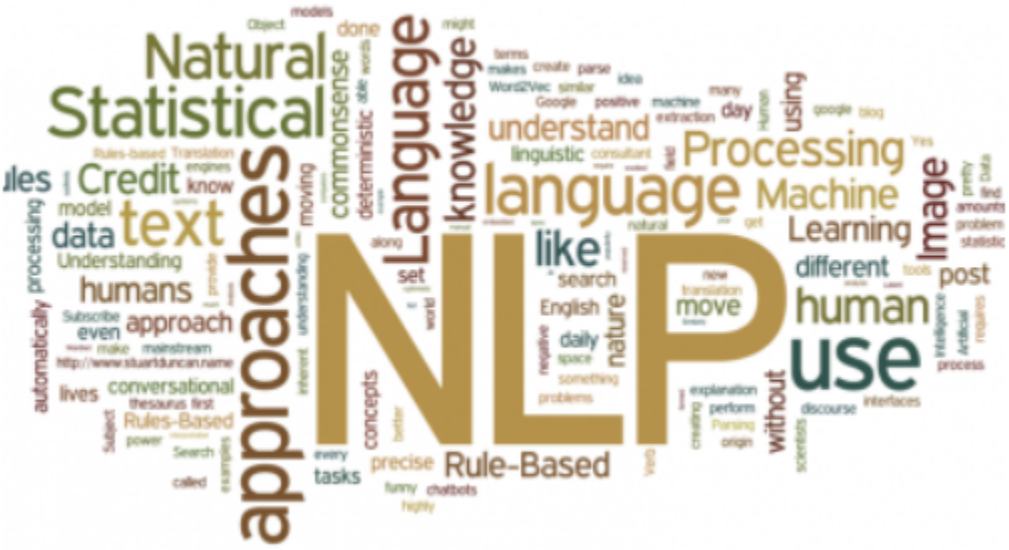
IST 664/CIS 668 - Movie Review Analysis



Name	SUID
Rutav Modi	989253064
Lu Yuchao	862356397
Deng Yicun	584579909
Ding Xiaminhao	813961016

May 15, 2021

Content

Abstract 3

Introduction 4

Data Description & Data Preprocessing 5

Tokenization: 7

Vectorization: 7

Implementation 8

Results 9

Multinomial Naïve Bayes 10

Benoulli Naïve Bayes 11

Input to the two models 11

Multinomial Naïve Bayes 12

Benoulli Naïve Bayes 12

SVM Classifier 15

Word Cloud: 21

Positive Word Cloud: 21

Conclusion 22

Future Work 23

Abstract

In this project, we will use the marked comments as a training set to train the model's ability to do sentiment analysis. We have another part of reviews as a test set to test the accuracy of the model. Based on establishing a good model, we hope to make the model more accurate. When traditional models perform sentiment analysis, it is difficult for them to correctly respond to similar emotional expressions. In other words, it will miss key emotional information.

In our process, first we will mark each row in the data set. We will further try various vectorization techniques, and then use each vectorization technique (for example, using Naive Bayes' Boolean value, using Naive Bayes' frequency vectorizer, and more combinations) to fit the model. After fitting the model, we will evaluate the data set on the test data and use accuracy, recall and F1 to determine which model works best. Finally, we will use CFG (Context Free Grammar) to analyze which are the most commonly used verbs, nouns, and words to express positive and negative emotions.

Introduction

In this project, we will process and analyze a large movie review dataset. This dataset contains a set of 25,000 polar movie reviews for training, and 25,000 for testing. There is also other unlabeled data available.

Our first goal is to build a model to do sentiment analysis. We hope to train the model to identify the positive and negative polarity of most words and phrases. Then the machine can judge whether a review is positive or negative based on the scores of different words. Therefore, we can use the model to automatically divide thousands of reviews into positive reviews and negative reviews.

We also want to make the model more accurate. When traditional models do sentiment analysis, it is difficult for them to respond correctly to similar emotional expressions. That is, it will miss key emotional information. For example, wonderful and amazing are close in semantics, but they express different meanings. The model we hope to achieve is to make words with similar emotions have similar vectors. We did not achieve this goal right now, but we are going to use Vender to let the machine learn these emotional vectors and correctly decide the pos/neg meaning of the sentence.

In the future, with the result, we can make statistics and analyze. We need to understand the relationship between the statistics data, such as the TOP most popular movies and the TOP greatest movies, why they are different, whether the number of reviews of a movie is related to whether its reviews are positive. We hope to get a conclusion on these issues.

Current application

We used the SVM model to do sentiment analysis on the IMDB dataset. Currently, sentiment analysis using SVM method is mainly used in three area:

Social Media:

1.Analyze the composition of users who like the activity

2.How people's attitude in an event

3.whether the advertisement style is in line with people’s preferences

E-Business:

1.Whether the new product is popular (the number of positive reviews is greater than the negative)

2.What features do people like about the product

3.What needs to be improved urgently

News distribution:

1.People's attitude toward a new policy

2.what is the point people generally support in a news

3.what is the point people generally criticized in a news

Literature review

R. R. Chowdhury, M. Shahadat Hossain, S. Hossain and K. Andersson had published a paper in 2019 proposed a new process of analyzing sentiment in Bangla language(R. R. Chodhury, M. Shahadat Hossain, S. Hossain and K. Andersson, 2019). This new process used the SVM algorithm and achieved 88.9% accuracy on the test dataset.

Data Description & Data Preprocessing

We have processed and analyzed a large movie review dataset. This dataset contains a set of 25,000 polar movie reviews for training, and 25,000 for testing. The data was available in folder format i.e. there were two folders train and test and additionally each of them contained folders labels as neg and pos and contains text files for respective labelled data. We first accessed our test folders then labelled folders and at last the text files in the dataset and we stored it in a dataframe.

Below are the dataset snapshot that we used for our analysis:

Out[17]:

	text	label
0	[For a movie that gets no respect there sure a...	pos
1	[Bizarre horror movie filled with famous faces...	pos
2	[A solid, if unremarkable film. Matthau, as Ei...	pos
3	[It's a strange feeling to sit alone in a thea...	pos
4	[You probably all already know this by now, bu...	pos
...	...	...
24995	[My comments may be a bit of a spoiler, for wh...	neg
24996	[The "saucy" misadventures of four au pairs wh...	neg
24997	[Oh, those Italians! Assuming that movies abou...	neg
24998	[Eight academy nominations? It's beyond belief...	neg
24999	[Not that I dislike childrens movies, but this...	neg

25000 rows × 2 columns

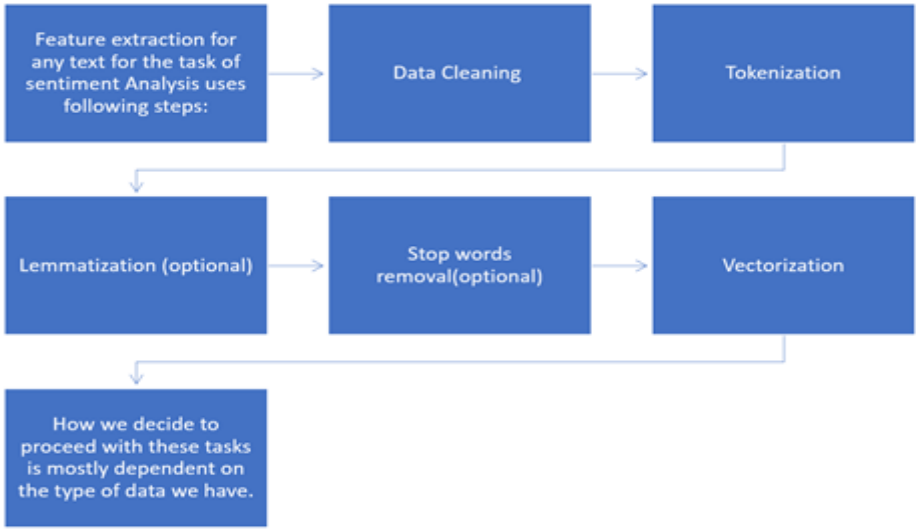
Out[20]:

	text	label
0	[Based on an actual story, John Boorman shows ...	pos
1	[This is a gem. As a Film Four production - th...	pos
2	[I really like this show. It has drama, romanc...	pos
3	[This is the best 3-D experience Disney has at...	pos
4	[Of the Korean movies I've seen, only three ha...	pos
...	...	...
24995	[With actors like Depardieu and Richard it is ...	neg
24996	[If you like to get a couple of fleeting glimp...	neg
24997	[When something can be anything you want it to...	neg
24998	[I had heard good things about "States of Grac...	neg
24999	[Well, this movie actually did have one redeem...	neg

25000 rows × 2 columns

The main motive to clean or preprocess the data is to transform the raw data into an understandable format. Mostly data which is available is incomplete, inconsistent and/or lacking in certain behaviours or trends and is likely to contain many errors. Another reason for preprocessing the data into an appropriate form to run and get more accurate understanding in the data mining process. The data can be noisy, and may contain special characters, which are not useful for the analysis purpose. We found that there are many URLs present in the tweets, so we cleaned and removed those URLs.

In the data preprocessing step the URLs and special characters were removed. The data also contained some missing values and such observations were omitted. The below image describes the steps and procedure that can be implemented for the data cleaning and preprocessing.



Tokenization:

Assembly language (the language of Machines) is a language of numbers and is very different from human languages. To analyze texts written in human languages it is important to represent them in computer-recognizable form i.e. numbers. The computer further counts these numbers. The process of grouping characters as tokens initiate the process of counting is known as Tokenization. Thus, tokenization prepares the data for counting. Consider the text :"Have a good day" the tokens for this text is ['Have','a','good','day']

Vectorization:

The processes of representing all the characters of a text document in the form of vectors is known as vectorization.Vectorization takes care of how we should proceed with counting the data.Vectorization creates a vocabulary of all the tokens present in set of documents and further represents text as numeric vector of all documents.

Types of Vectorizations:

- **Boolean Vectorizer:** The boolean vectorizer represents just the presence or absence of a word from vocabulary in the document
- **Countvectorizer:** The count vectorizer represents the number of times a word from vocabulary appears in the document
- **TfIDF Vectorizer:** The TFIDF(term frequency-inverse document frequency) vectorizer stands for is a statistical measure that evaluates how relevant a word is to a document in a collection of documents.
- This is done by multiplying two metrics: how many times a word appears in a document, and the inverse document frequency of the word across a set of documents
- TF-IDF (term frequency-inverse document frequency) was invented for document search and information retrieval. It works by increasing proportionally to the number of times a word appears in a document, but is offset by the number of documents that contain the word.
- So, words that are common in every document, such as this, what, and if, rank low even though they may appear many times, since they don't mean much to that document in particular.

Implementation

We import the os library to read the path of the data set, so as to extract the data of two polarities. Then use the pandas library to build a data frame to facilitate subsequent processing. We stitch texts with positive polarity together and do the same processing for texts with negative polarity, and then use union to set all stop words, and finally complete data visualization, so that different

texts generate their corresponding word clouds. Then we draw a histogram of the text distribution in the training set and the test set.

Next, we introduced three vectorized tokenizers from the sklearn library, namely CountVectorizer, BooleanVectorizer and Tf-idfVectorizer. The advantage of sklearn vectorization is that it will handle tokenization by itself when vectorizing, and we can also specify the n-gram range in the vectorizer. We use these tokenizers to perform the analysis using both unigrams and bigrams. Then we use these vectorizers for polynomial naive Bayes, Bernoulli naive Bayes and SVM models with different combinations of C values. Naive Bayes is simple to implement, has high learning efficiency, and has better performance under a large sample size. SVM is a type of classification algorithm that is widely used and has good results. So, we chose these two models. Among them, we used the linear SVC model for the SVM model. After fitting and predicting, we use three scores to verify the accuracy of the model: Precision, Recall and f1 score. We use this to perform model evaluation and get a more suitable model. For SVC, there is another parameter that can also affect the accuracy rate, that is, the C value. A large C value increases the penalty for misclassification, so that the accuracy rate when testing the training set will be high, but the generalization ability is weak. A small C value reduces the penalty for misclassification, allows fault tolerance, treats them as noise points, and has a strong generalization ability. By setting different C values, we found the model with the best performance parameters.

Results

Naïve Bayes Algorithm

Multinomial Naïve Bayes

Multinomial Naïve Bayes is one of the methods to train models for text categorization/ classification. It is based on the Bayes theorem which helps us compute the conditional probabilities of a word in text. It is referred to as Naïve as it assumes that the occurrence of each word in the text document is independent of the occurrence of other words, which is not really possible. But still this approach seems to work quite well. The multinomial naïve Bayes calculates the prior probabilities of each class and conditional probabilities of each word given each class to determine the posterior probabilities. The general formulas for these probabilities are as follows:

Prior Probability: Probability(class)

Conditional Probability (for Multinomial Naïve Bayes): $P(w|class) = nw + 1 / n + v$

Representations of variables:

w- represents a word from the text in the document

class – the class of the given text in the document

nw – The total number of times the word w occurs in this class of text

n – Total number of words in the class of text

v – vocabulary(number of distinct words in the whole document irrespective of their class)

Benoulli Naïve Bayes

As the name suggests it is another method based on Bayes theorem to train models for text categorization/classification. The Benoulli naïve Bayes calculates the prior probabilities of each class and conditional probabilities of each word given each class to determine the posterior probabilities. The general formulas for these probabilities are as follows:

- Prior Probability: Probability(class)
- Conditional Probability (for Benoulli Naïve Bayes): P(w|class) = nw +1 / n + cn
- Suppose the vocabulary has words w1, w2,w3, w4 and we want to classify a sentence with words w2 and w3.
- Posterior Probability = Prior Probability * P(w2|class1) *P(w3|class1) *P(1- w1|class1) *P(1- w4|class1)
- nw – The total number of texts in this class that contain word w
- n – Total number of texts in this class
- cn – Total number of classes.

As we can see, unlike multinomial models, this model uses the presence/absence of words while calculating conditional probabilities.

Input to the two models

The Benoulli Naïve Bayes accepts only the Boolean representation of the vectorized text documents. While on the other hand Multinomial Naïve Bayes accepts word frequencies or tfidf.

Multinomial Naïve Bayes

The data was split in 70:30 ratio for training the model and testing it. The training and testing set of input string data was vectorized using the term frequency vectorizer, Frequency vectorizer with n-gram tokenizer with max range 2 and min range 1 and tfidf vectorizer. It is important to note here that all the stop words were removed. In python this was achieved through using CountVectorizer() and TfidfVectorizer() objects. The reason for avoiding the use of Boolean Vectorizer is the approach that Multinomial Naïve Bayes algorithm uses to calculate posterior probabilities. Multinomial Naïve Bayes estimates the probability of the event that one of the N unique words occurs in a position and thus, we don't use the Boolean Vectorizer as it wouldn't be a good fit.

Benoulli Naïve Bayes

The data was split in 70:30ratio for training the model and testing it. The training and testing set of input string data was vectorized using Boolean Vectorizer and n-gram tokenizer without stop words and also using Boolean Vectorizer and word tokenizer without and with stopwords. The Benoulli Naïve Bayes calculates the probabilities based on the presence and absence of words in a class and thus we use Boolean vectorizer for this model.

For all the classification reports shared below the boolean vectorizer represents the use of Bernoulli Naïve Bayes while count and TFIDF Vectorizer represent the use of Multinomial Naïve Bayes.

Vectorizer: Boolean Vectorizer

Tokenizer: unigram

Classification Report :

	precision	recall	f1-score	support
neg	0.89	0.78	0.83	14176
pos	0.75	0.87	0.81	10824
accuracy			0.82	25000
macro avg	0.82	0.83	0.82	25000
weighted avg	0.83	0.82	0.82	25000

Vectorizer: Countvectorizer

Tokenizer: unigram

Classification Report :

	precision	recall	f1-score	support
neg	0.88	0.79	0.83	13911
pos	0.77	0.86	0.81	11089
accuracy			0.82	25000
macro avg	0.82	0.83	0.82	25000
weighted avg	0.83	0.82	0.82	25000

Vectorizer: TFIDF Vectorizer

Tokenizer: unigram

Classification Report :

	precision	recall	f1-score	support
neg	0.88	0.80	0.84	13705
pos	0.78	0.87	0.82	11295
accuracy			0.83	25000
macro avg	0.83	0.83	0.83	25000
weighted avg	0.83	0.83	0.83	25000

We can see the best performing model with unigram tokenizer was TFIDF vectorizer with accuracy of 86% and precision of almost 88% for label 1 and 84% for label 0

Vectorizer: Boolean Vectorizer

Tokenizer: biigram

Classification Report :

	precision	recall	f1-score	support
neg	0.89	0.83	0.86	13295
pos	0.82	0.88	0.85	11705
accuracy			0.85	25000
macro avg	0.85	0.86	0.85	25000
weighted avg	0.86	0.85	0.85	25000

Vectorizer: Countvectorizer

Tokenizer: biigram

Classification Report :

	precision	recall	f1-score	support
neg	0.88	0.82	0.85	13454
pos	0.81	0.87	0.84	11546
accuracy			0.84	25000
macro avg	0.84	0.85	0.84	25000
weighted avg	0.85	0.84	0.84	25000

Vectorizer: TFIDF Vectorizer

Tokenizer: biligram

Classification Report :

	precision	recall	f1-score	support

neg	0.88	0.84	0.86	13181
pos	0.83	0.87	0.85	11819
accuracy			0.85	25000
macro avg	0.85	0.85	0.85	25000
weighted avg	0.86	0.85	0.85	25000

Introducing bigrams improves model accuracy and precision.However, the best performing model for Naïve Bayes turns out to be Bernoulli Naïve Bayes with bigram Tokenizer and Boolean Vectorizer. It gives us accuracy of 85%and precision of almost 89% for label 0 and 82% for label 1.

SVM Classifier

Usually, Linear SVM is used to do text based classifications. For this project I have experimented with different values of C representing the regularization parameter greater the value of C less the strength of regularization. It should strictly be positive.

Tokenizer: unigram

C : 0.05

Classification Report :

CountVectorizer , c = 0.05

	precision	recall	f1-score	support
neg	0.86	0.85	0.86	12756
pos	0.84	0.86	0.85	12244
accuracy			0.85	25000
macro avg	0.85	0.85	0.85	25000
weighted avg	0.85	0.85	0.85	25000

TfidfVectorizer , c = 0.05

	precision	recall	f1-score	support

neg	0.87	0.88	0.88	12308
pos	0.88	0.87	0.88	12692
accuracy			0.88	25000
macro avg	0.88	0.88	0.88	25000
weighted avg	0.88	0.88	0.88	25000

Tokenizer: unigram

C : 0.1

Classification Report :

CountVectorizer , c = 0.1

	precision	recall	f1-score	support
neg	0.86	0.84	0.85	12787
pos	0.83	0.85	0.84	12213
accuracy			0.85	25000
macro avg	0.85	0.85	0.85	25000
weighted avg	0.85	0.85	0.85	25000

TfidfVectorizer , c = 0.1

	precision	recall	f1-score	support
neg	0.88	0.88	0.88	12432
pos	0.88	0.88	0.88	12568
accuracy			0.88	25000
macro avg	0.88	0.88	0.88	25000
weighted avg	0.88	0.88	0.88	25000

Tokenizer: unigram

C : 0.5

Classification Report :

CountVectorizer , c = 0.5

	precision	recall	f1-score	support
neg	0.85	0.83	0.84	12824
pos	0.82	0.84	0.83	12176
accuracy			0.83	25000

macro avg	0.83	0.83	0.83	25000
weighted avg	0.83	0.83	0.83	25000

TfidfVectorizer , c = 0.5

	precision	recall	f1-score	support
neg	0.88	0.87	0.87	12756
pos	0.86	0.88	0.87	12244
accuracy			0.87	25000
macro avg	0.87	0.87	0.87	25000
weighted avg	0.87	0.87	0.87	25000

Tokenizer: unigram

C : 1

Classification Report :

CountVectorizer , c = 1

	precision	recall	f1-score	support
neg	0.84	0.82	0.83	12832
pos	0.82	0.84	0.83	12168
accuracy			0.83	25000
macro avg	0.83	0.83	0.83	25000
weighted avg	0.83	0.83	0.83	25000

TfidfVectorizer , c = 1

	precision	recall	f1-score	support
neg	0.88	0.85	0.87	12832
pos	0.85	0.87	0.86	12168
accuracy			0.86	25000
macro avg	0.86	0.86	0.86	25000
weighted avg	0.86	0.86	0.86	25000

We can see for all values of C TFIDF Vectorizer does a better job than the Frequency/Count Vectorizer. The highest accuracy with unigram tokenizer was obtained with C=0.1 and C=0.05 which was approximately 88%. The precision for label 1 by both the classifiers was the same

88% however, c=0.1 did a slightly better job with precision for label 0 and thus, it can be seen as the best model for unigram tokenizer.

Tokenizer: bigram

C : 0.05

Classification Report :

CountVectorizer , c = 0.05

	precision	recall	f1-score	support
neg	0.88	0.86	0.87	12696
pos	0.86	0.87	0.87	12304
accuracy			0.87	25000
macro avg	0.87	0.87	0.87	25000
weighted avg	0.87	0.87	0.87	25000

TfidfVectorizer , c = 0.05

	precision	recall	f1-score	support
neg	0.87	0.88	0.88	12274
pos	0.89	0.87	0.88	12726
accuracy			0.88	25000
macro avg	0.88	0.88	0.88	25000
weighted avg	0.88	0.88	0.88	25000

Tokenizer: bigram

C : 0.1

Classification Report :

CountVectorizer , c = 0.1

	precision	recall	f1-score	support
neg	0.87	0.86	0.86	12688
pos	0.86	0.87	0.86	12312
accuracy			0.86	25000
macro avg	0.86	0.86	0.86	25000

weighted avg	0.86	0.86	0.86	25000

TfidfVectorizer , c = 0.1

	precision	recall	f1-score	support
neg	0.88	0.89	0.88	12426
pos	0.89	0.88	0.88	12574
accuracy			0.88	25000
macro avg	0.88	0.88	0.88	25000
weighted avg	0.88	0.88	0.88	25000

Tokenizer: bigram

C : 0.5

Classification Report :

CountVectorizer , c = 0.5

	precision	recall	f1-score	support
neg	0.87	0.85	0.86	12696
pos	0.85	0.87	0.86	12304
accuracy			0.86	25000
macro avg	0.86	0.86	0.86	25000
weighted avg	0.86	0.86	0.86	25000

TfidfVectorizer , c = 0.5

	precision	recall	f1-score	support
neg	0.89	0.88	0.88	12679
pos	0.88	0.89	0.88	12321
accuracy			0.88	25000
macro avg	0.88	0.88	0.88	25000
weighted avg	0.88	0.88	0.88	25000

Tokenizer: bigram

C : 1

Classification Report :

CountVectorizer , c = 1

	precision	recall	f1-score	support

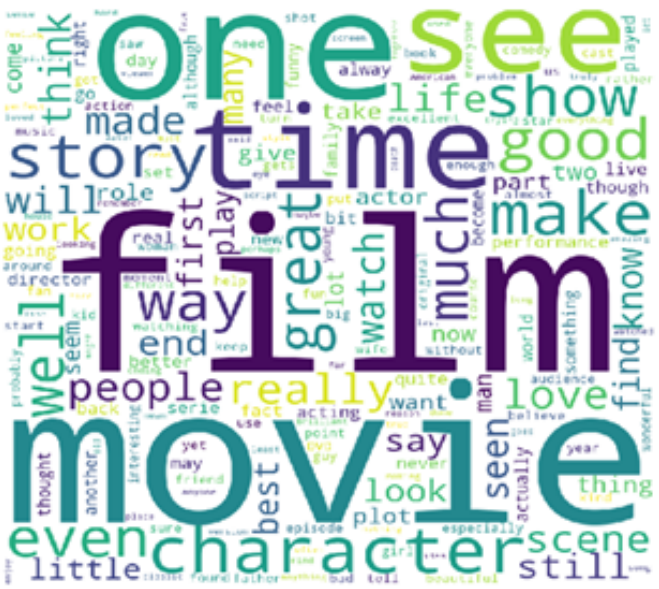
neg	0.87	0.85	0.86	12684
pos	0.85	0.86	0.86	12316
accuracy			0.86	25000
macro avg	0.86	0.86	0.86	25000
weighted avg	0.86	0.86	0.86	25000

TfidfVectorizer , c = 1

	precision	recall	f1-score	support
neg	0.89	0.87	0.88	12768
pos	0.87	0.89	0.88	12232
accuracy			0.88	25000
macro avg	0.88	0.88	0.88	25000
weighted avg	0.88	0.88	0.88	25000

For bigram tokenizer almost all models did give accuracies around 88% and precision around 89 % for both labels. However, the precision and accuracy of the best performing model for bigram (C=0.5,TFIDF vectorizer). Additionally, bigrams are computationally expensive as they increase the number of features. **Thus the overall best performing model remains SVM with unigram tokenizer and with C=0.1.**

Word Cloud:



This is the word cloud which is made on the whole movie review dataset. We can see words like film, movies,film as most of the reviews contain the words related to movies. So no sentiments are found in it.

Positive Word Cloud:

In the above word cloud we can see the positive sentiments of IMDB movie reviews. We can observe words like great, good, well and many more which shows the positive nature of the reviews.

Conclusion

In this project, we mainly studied the results of different combinations of various models and vectorizers. We have observed that comparing Precision, Recall and F1 scores of all combinations, the combination of SVM model and TF-IDF vectorizer is the best. Compared with other vectorizers, the TF-IDF vectorizer calculates a score (Term Frequency (TF) × Inverse Document Frequency (IDF)) through a weighting method, which can effectively extract keywords in sentences. It is more direct help for the sentiment analysis of the entire sentence.

It is not very difficult to construct a high-precision or high-recall classifier, but it is not so simple to ensure that both are at a high level at the same time. The naive Bayes algorithm is relatively simple, and the correlation analysis between words is not clear, so it will affect the correct classification. The SVM can adjust the penalty parameter C, which can improve the accuracy of text classification and also has a certain generalization ability. In addition, SVM performs well on small sample sizes.

Through combination and evaluation, we have obtained the best classifier and vectorizer for this movie review collection. There may be other higher-quality models, but for data sets where the amount of text is not large, our current method is more suitable. Through sentiment analysis of the text, it can be obtained whether people's evaluation of a certain movie is mostly good or bad, and this could provide suggestions and guidance when the website pushes high-quality movies to

people or when people choose good movies. Therefore, a correct classification result is very important at this time.

Future Work

We'd like to use Vander for more accurate sentiment analysis. With the emotional dictionary of Vender, its scoring mechanism can help us better identify semantically similar phrases and sentences. Therefore, we will be able to get a more accurately mark to modify the neutral part of sentences.

Also, we want to apply this project to the following areas in the future:

First is Social Media sites (twitter, Instagram), They can use it to Analyze the composition of users who like an activity. We can distinguish positive and negative reviews, and then we will use positive parts to analyze the age and gender composition of people who like this activity. In future activities, we can make improvements by meeting the characteristics of these people. Also, we can get how people's attitude in an event We can count the proportion of positive and negative comments in an event and find out whether people support or oppose it.

Another important thing is that we can know whether the advertisement style is in line with people's preferences. As we know Advertising is an important revenue component of social media website. If the website cannot get feedback, they may not be able to improve the unpopular advertisement in time. we can analyze the user's evaluation of this advertisement, whether it is positive than negative, and the reasons why people like or hate it. Then we can improve this advertisement.

And Second, for Business site (Amazon, eBay). By using review analysis, they will know whether the new product is popular (I mean whether the number of positive reviews is greater than the negative one). So that they can judge whether this new product is worth enough to promote. Also, they can get what features do people like about the product. When we separate the positive reviews, we can further analyze the most popular characteristics in these reviews, which is the best features of this product. So that they can keep these features in next generation. Then they will know what needs to be improved urgently. we can analyze the negative reviews to find out what should be improved.

For News site (Tencent news), we will know People's attitude toward a new policy. Before the new policy is implemented, we can analyze its support rate on the Internet, Therefore, we can treat unsupported policies more cautiously and make appropriate changes in advance like what is the point people generally support in an event, what is the point people generally criticized in an event. After we divide the evaluation into positive and negative through this procedure, we can use the positive evaluation to analyze the reasons for the popularity of this policy. Similarly, we can analyze its negative areas for improvement.

References

1. Dataset Link: https://ai.stanford.edu/~amaas/data/sentiment/ https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47
2. https://monkeylearn.com/sentiment-analysis/
3. R. R. Chowdhury, M. Shahadat Hossain, S. Hossain and K. Andersson, "Analyzing Sentiment of Movie Reviews in Bangla by Applying Machine Learning Techniques," 2019 International Conference on Bangla Speech and Language Processing (ICBSLP), 2019, pp. 1-6, doi: 10.1109/ICBSLP47725.2019.201483.
4. https://ieeexplore.ieee.org/abstract/document/9084036

5. https://towardsdatascience.com/imdb-reviews-or-8143fe57c825
6. https://towardsdatascience.com/sentiment-analysis-introduction-to-naive-bayes-algorithm-96831d77ac91