LING406 Term Project
KangHong Zhao

Introduction

According to the different types of processed text, it can be divided into sentiment analysis based on news reviews(movie review) and sentiment analysis based on product reviews(yelp review). Among them, the former is mostly used for public opinion monitoring and information prediction, while the latter can help users understand the public praise of a product. There are many other applications other than yelp comments and movie reviews that can benefit from it. For example, product review(Amazon customer comments), comments on twitter to airline companies, comments on the presidential campaign on twitter. Sentimental analysis captures users' inner emotional information through various methods and means, and conduct in-depth analysis to obtain the insight of users' personal emotional differences and internal needs. Finally, with the help of these valuable users' emotional insights, enterprises can carry out emotional packaging, emotional promotion, product design, product design and product management.

Problem definition

Sentimental analysis is a process of analyzing, processing, inducing and reasoning the subjective text with emotional color. There are two main methods of sentiment polarity analysis: one is based on sentiment dictionary, the other is based on machine learning. In the context of computational linguistics, we give certain attributes to the input given text and assign values to evaluate the polarity of the text and with a sentiment as an output. The input text can be preprocessed in various ways and learned with various learning models and algorithms then. This assignment aims to give different approaches to build a sentimental analysis system.

Previous work

Sentimental analysis has been a long and intriguing topic these years. There are many previous work. For example, there are many emotional analysis dictionaries. [1]Rachael Tatman uploaded on Kaggle about sentimental lexicons for 81 languages, detecting whether a piece of text is positive or negative, generally relies on a hand-curated list of words with positive sentiment and negative sentiment. Another example might be SentiWordNet, This data set contains about 29000 words with an emotion score between 0 and 1. The third example is Emoticon Sentiment Lexicon, This dataset contains a list of 477 emoticons marked as positive, neutral, or negative. These previous dictionaries provide a series of word dictionaries, with tags that include emotions in various fields. Not only English words, but also different languages and even emojis. These really helps us to evaluate sentimental analysis system.

There are more topics on sentimental analysis. [2]Multiple Instance Learning Networks for Fine-Grained Sentiment Analysis conducted by Stefanos Angelidis and Mirella Lapata introduce how to use MIL method to analyze emotion. Due to the availability of fine-grained sentiment data (such as user comments + scoring), many supervised learning methods have achieved good results. However, fine-grained sentiment analysis, such as marking the sentiment score of each sentence in an article, still has great shortcomings, mainly because fine-grained data annotation is too difficult. In this paper, a method of MIL is proposed, which can predict the emotional polarity of continuous segments with the help of the label of document. The spot data set is constructed, which contains the granularity annotation of sentence or edu (elementary discovery unit), which can be used to test the model of MIL like function.

The second example is [3]Enhanced Twitter Sentiment Classification Using Contextual Information. By utilizing distant supervision to collect millions of labelled tweets from different locations, times and authors. They used the data to analyse the variation of tweet sentiments across different authors, times and locations. Then they use Bayesian to combine the variables with n-grams and other approaches to create a Twitter sentiment classifier.

The third example is [4]Sentiment Propagation via Implicature Constraints. How the good and bad entities may be exploited to improve sentiment analysis. And they find it has an 89% chance of propagating sentiments correctly by applying Loopy Belief Propagation to propagate sentiments among entities with the development of a graph-based model based on implicature rules to propagate sentiments among entities.

These three previous work all talk about contextual Sentiment Analysis. These three previous all did a great job analyzing the sentimental in different fields and applications. The third work is more fundamental, the second work considers different situations and variables to get a more broad statistics and combine them with linguistics model. The first work is more comprehensive that talks about the coarse-grained effect words.

It makes me think there are so many fields and questions remained in this area. For example, for the third work, there is still 10 percent chance that the implicatures not go through in context, under what condition might cause that. There are also other fields like how do we deal with sarcasm analysis, bias in Sentiment Analysis Systems(demographic language variations, gender and race bias). These are all very interesting topics.

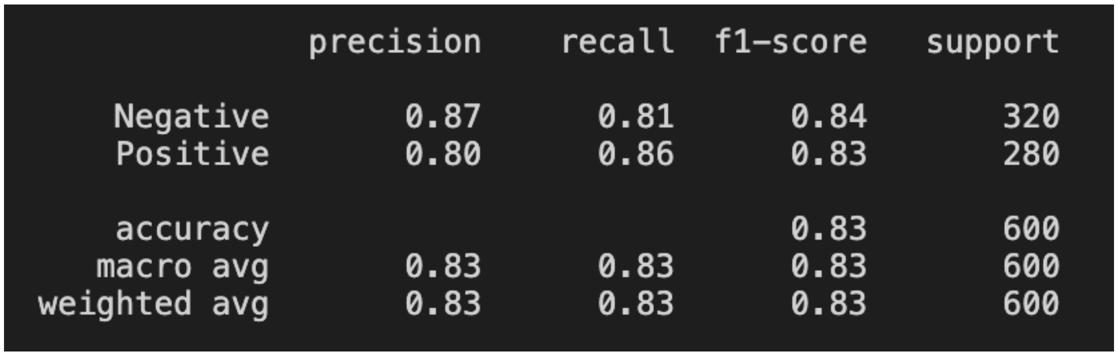
Approach

For baseline approach, in general, the Baseline Model- Multinomial Naive Bayes, decision tree and SVM using Count vectorizer and TFIDF vectorizer. First clean the data, remove all the punctuations, numbers and other redundant. A vectorizer helps me to convert text data into computer understandable numeric data. CountVectorizer counts the frequency of all words in our corpus, sorts them and grabs the most recurring features. TFIDF is a statistical measure said to have fixed the issues with CountVectorizer in some way.

These are the machine learning classifiers that I applied, the first one is naive Bayes, it’s based on the assumption that features are independent of each other (assuming that there is a feature in the class that has nothing to do with any other features). Even if these features depend on each other, or depend on the existence of other features, naive Bayes algorithm considers these features to be independent. The second one is decision tree, The decision tree constructs classification or regression model with tree structure. It makes the decision tree grow by dividing the data set into smaller subsets. Finally, it grows into a tree with decision nodes (including root nodes and internal nodes) and leaf nodes. The third one is support vector machine, Support vector machine can be used for both regression and classification. It is based on the decision plane that defines the decision boundary. Decision plane (hyperplane) can separate a group of objects belonging to different classes. With the help of support vector, SVM classifies by finding hyperplane and maximizes the boundary distance between two classes.

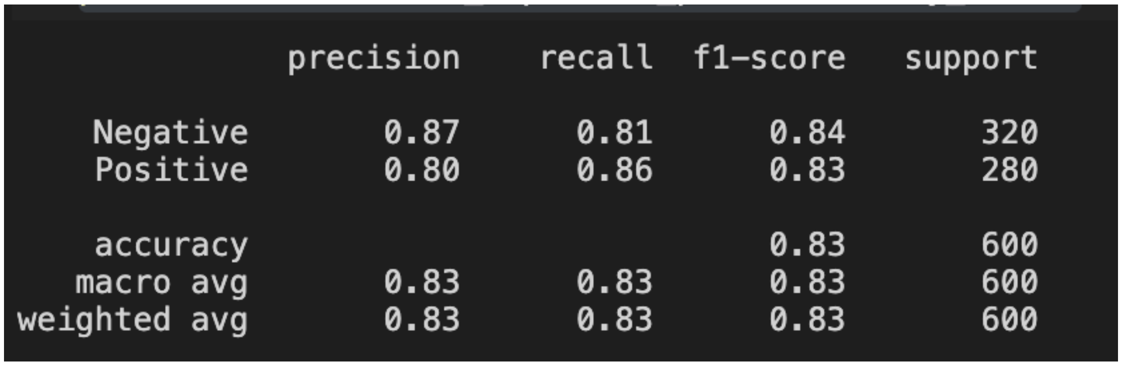
For improved approach, I introduced word vectors using gensim, pos-tagging, remove stopwords, negation. I also set different features sizes to see if the size of the positive/negative words make any difference to the results. And I also compare the performance of different feature sets under the same feature selection scenario and machine learning algorithm

Results
For baseline model, the model Accuracy using Count Vectorizer is 0.825 and Model accuracy using TFIDF is 0.81.



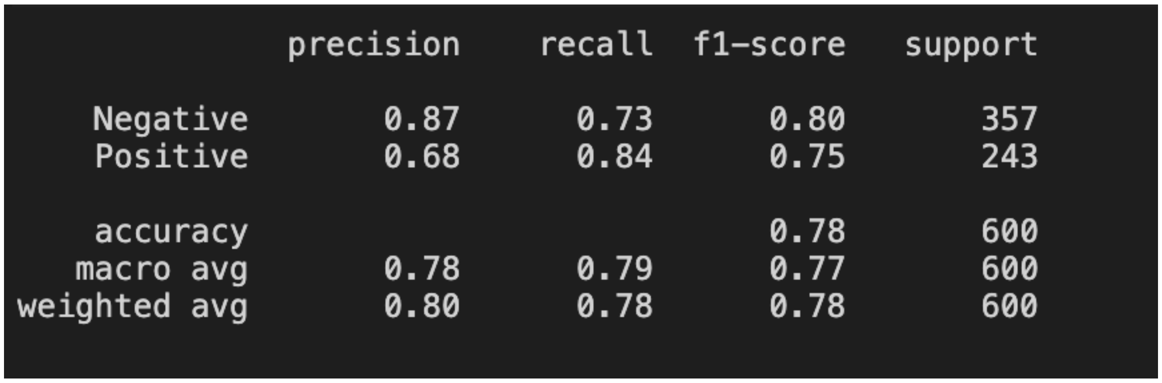
```
              precision    recall  f1-score   support

    Negative       0.87      0.81      0.84       320
    Positive       0.80      0.86      0.83       280

    accuracy                           0.83       600
   macro avg       0.83      0.83      0.83       600
weighted avg       0.83      0.83      0.83       600
```

When applied Naive Bayes,
the model Accuracy using Count Vectorizer is 0.81 and Model accuracy using TFIDF is 0.73.



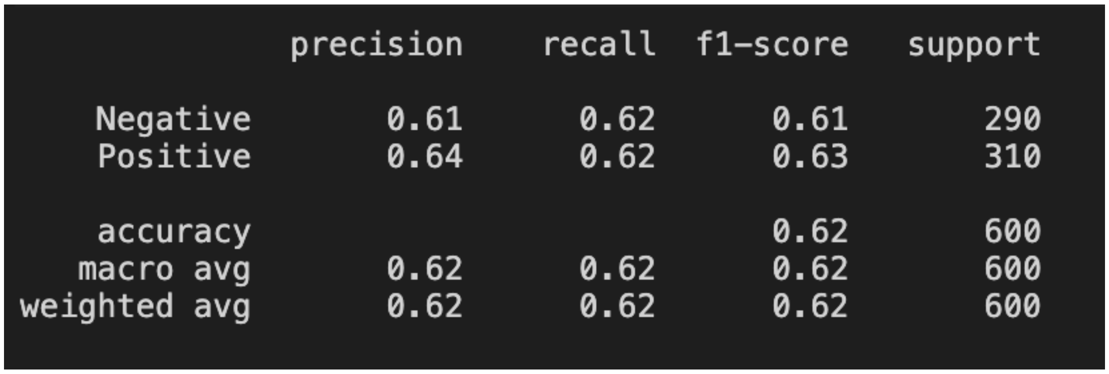
```
              precision    recall  f1-score   support

    Negative       0.87      0.81      0.84       320
    Positive       0.80      0.86      0.83       280

    accuracy                           0.83       600
   macro avg       0.83      0.83      0.83       600
weighted avg       0.83      0.83      0.83       600
```

When applied SVM,
the model Accuracy using Count Vectorizer is 0.72 and Model accuracy using TFIDF is 0.81.



```
              precision    recall  f1-score   support

    Negative       0.87      0.73      0.80       357
    Positive       0.68      0.84      0.75       243

    accuracy                           0.78       600
   macro avg       0.78      0.79      0.77       600
weighted avg       0.80      0.78      0.78       600
```

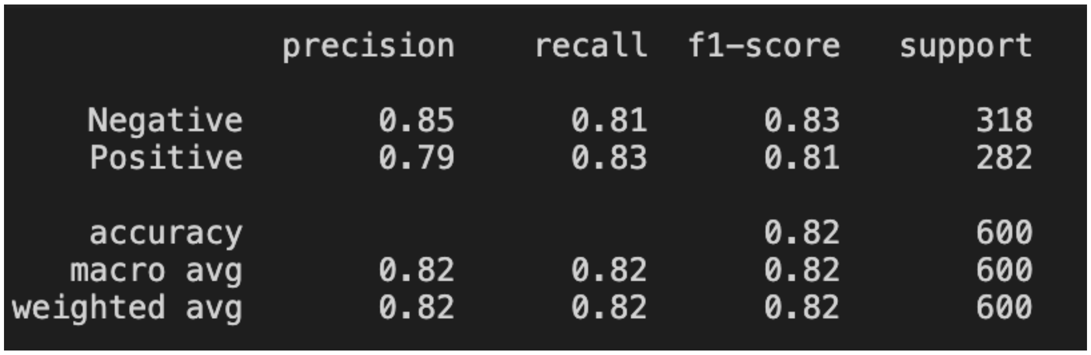
When applied Decision Tree,
the model Accuracy using Count Vectorizer is 0.63 and Model accuracy using TFIDF is 0.60.



```
              precision    recall  f1-score   support

    Negative       0.61      0.62      0.61       290
    Positive       0.64      0.62      0.63       310

    accuracy                           0.62       600
   macro avg       0.62      0.62      0.62       600
weighted avg       0.62      0.62      0.62       600
```

Naive Bayes therefore has a better performance.

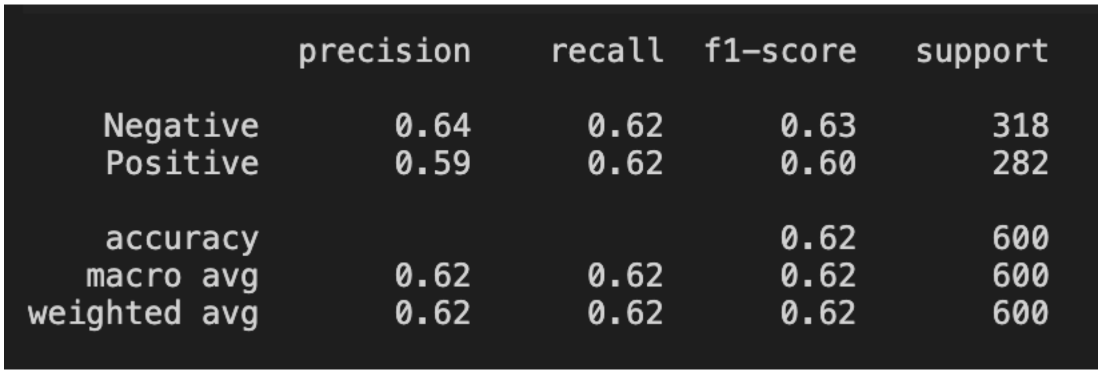
While the Naive Bayes when add negation,
the model Accuracy using Count Vectorizer is 0.78 and Model accuracy using TFIDF is 0.77.



```
              precision    recall  f1-score   support

    Negative       0.85      0.81      0.83       318
    Positive       0.79      0.83      0.81       282

    accuracy                           0.82       600
   macro avg       0.82      0.82      0.82       600
weighted avg       0.82      0.82      0.82       600
```

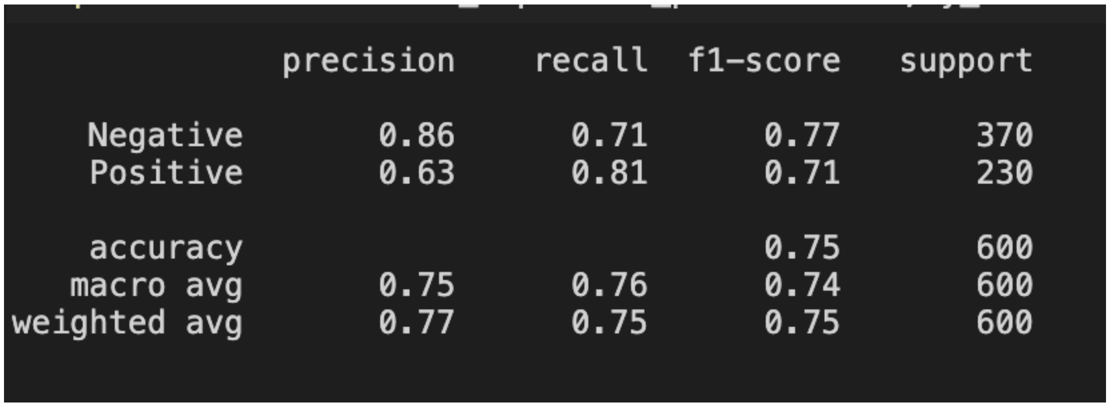
When Decision Tree applied,

the model Accuracy using Count Vectorizer is 0.57 and Model accuracy using TFIDF is 0.56.



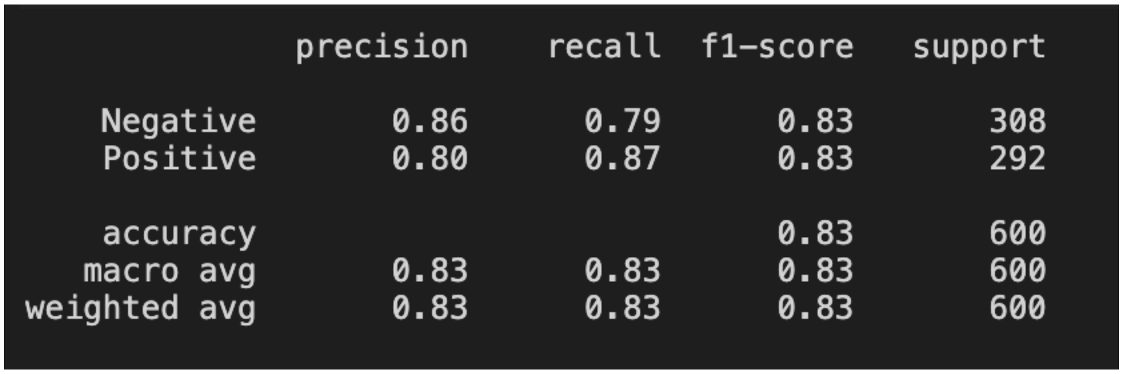
```
              precision    recall  f1-score   support

    Negative       0.64      0.62      0.63       318
    Positive       0.59      0.62      0.60       282

    accuracy                           0.62       600
   macro avg       0.62      0.62      0.62       600
weighted avg       0.62      0.62      0.62       600
```

When SVM applied,
the model Accuracy using Count Vectorizer is 0.71 and Model accuracy using TFIDF is 0.81.

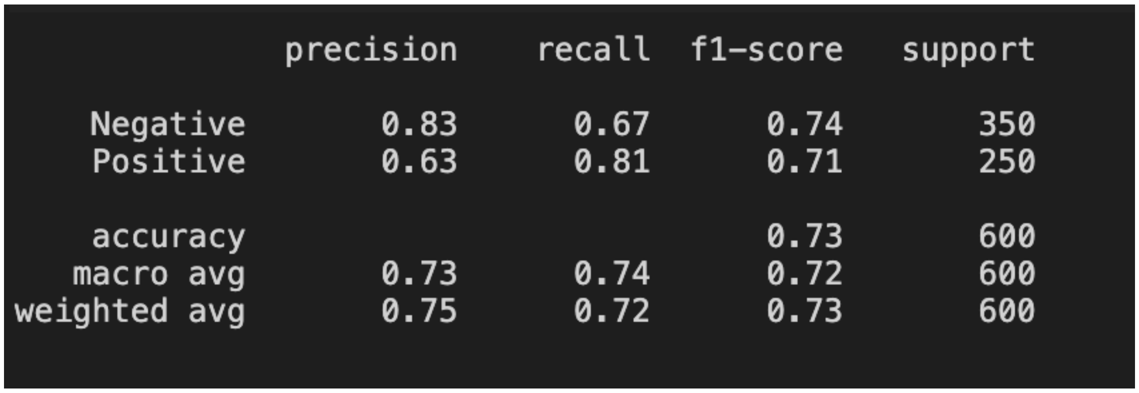


```
              precision    recall  f1-score   support

    Negative       0.86      0.71      0.77       370
    Positive       0.63      0.81      0.71       230

    accuracy                           0.75       600
   macro avg       0.75      0.76      0.74       600
weighted avg       0.77      0.75      0.75       600
```

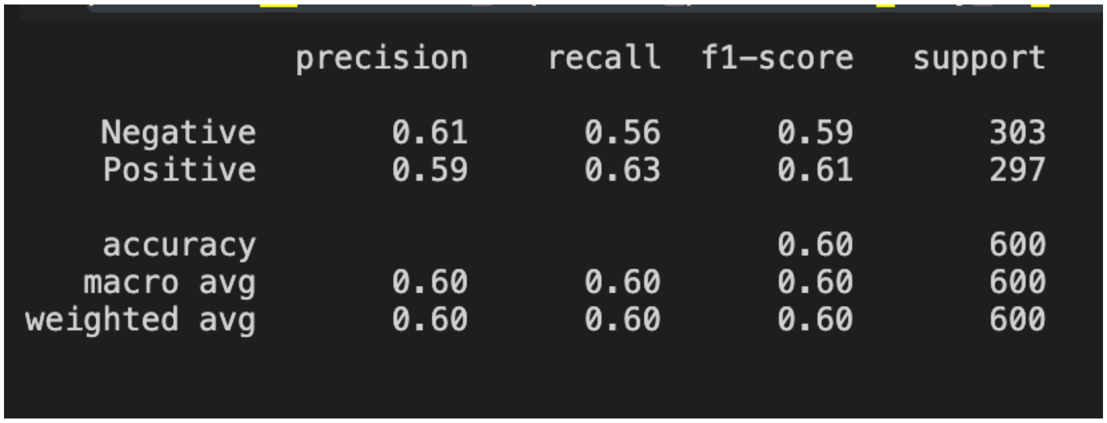
When applied stopwords with Naive Bayes, Decision Tree and SVM,



```
              precision    recall  f1-score   support

    Negative       0.86      0.79      0.83       308
    Positive       0.80      0.87      0.83       292

    accuracy                           0.83       600
   macro avg       0.83      0.83      0.83       600
weighted avg       0.83      0.83      0.83       600
```



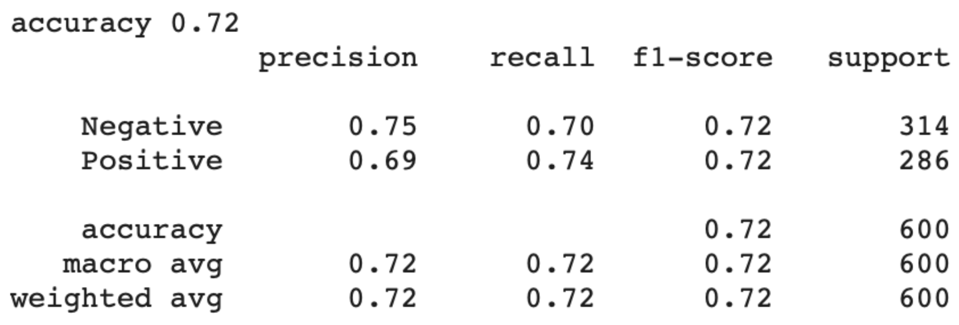
```
              precision    recall  f1-score   support

    Negative       0.83      0.67      0.74       350
    Positive       0.63      0.81      0.71       250

    accuracy                           0.73       600
   macro avg       0.73      0.74      0.72       600
weighted avg       0.75      0.72      0.73       600
```



```
              precision    recall  f1-score   support

    Negative       0.61      0.56      0.59       303
    Positive       0.59      0.63      0.61       297

    accuracy                           0.60       600
   macro avg       0.60      0.60      0.60       600
weighted avg       0.60      0.60      0.60       600
```

After that, I tried to Introduce Word vectors using gensim.

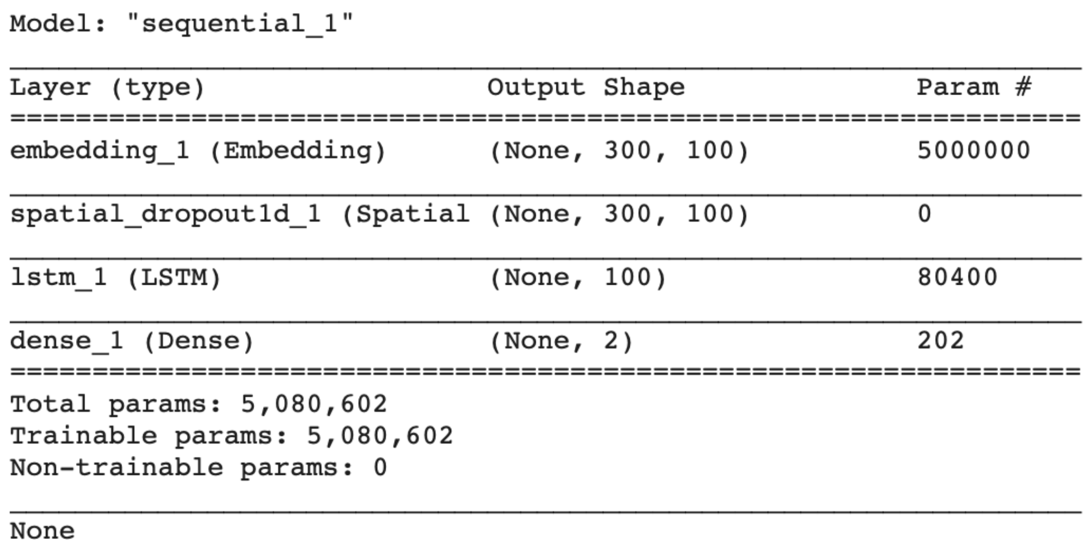
The result of Naive Bayes is following,
the model Accuracy using Count Vectorizer is 0.83 and Model accuracy using TFIDF is 0.81.



```
accuracy 0.72
              precision    recall  f1-score   support

    Negative       0.75      0.70      0.72       314
    Positive       0.69      0.74      0.72       286

    accuracy                           0.72       600
   macro avg       0.72      0.72      0.72       600
weighted avg       0.72      0.72      0.72       600
```

These indicates that removing the non alphabets and stop words, introduce word vectors doesn't improve the accuracy of the sentimental analysis system, it actually decreases the accuracy a little.

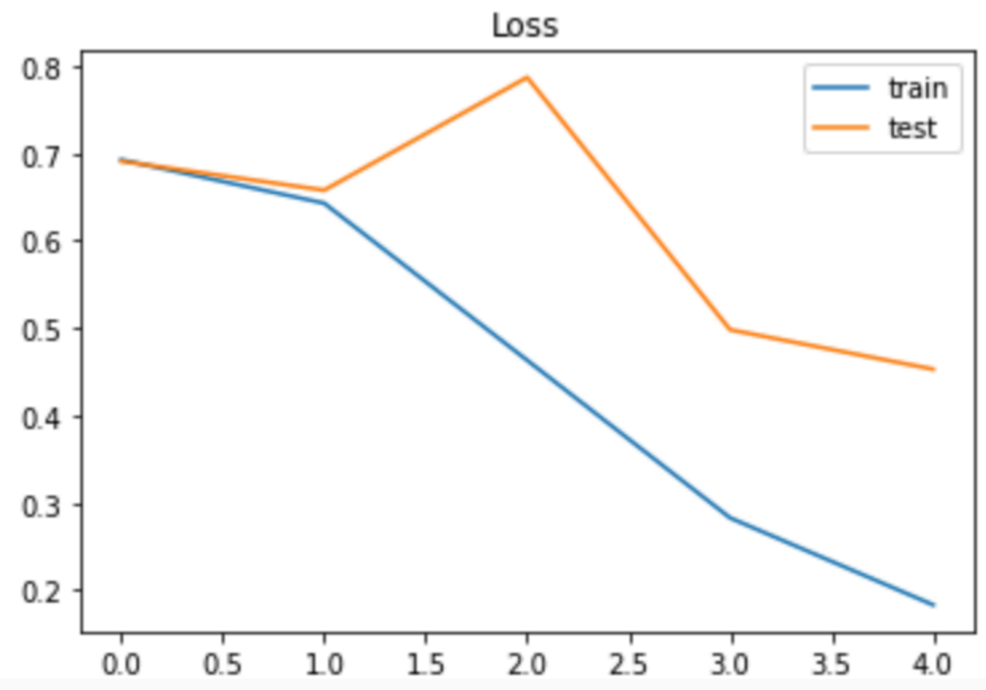
Also, by using LSTMS For Text Classification,



```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
embedding_1 (Embedding)      (None, 300, 100)          5000000   
_________________________________________________________________
spatial_dropout1d_1 (Spatial (None, 300, 100)          0         
_________________________________________________________________
lstm_1 (LSTM)                (None, 100)               80400     
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 202       
=================================================================
Total params: 5,080,602
Trainable params: 5,080,602
Non-trainable params: 0
_________________________________________________________________
None
```

```
Epoch 1/5
26/26 [==============================] - 20s 597ms/step - loss: 0.6931 - accuracy: 0.4960 - val_loss: 0.6910 -
val_accuracy: 0.5444
Epoch 2/5
26/26 [==============================] - 15s 580ms/step - loss: 0.6624 - accuracy: 0.7556 - val_loss: 0.6578 -
val_accuracy: 0.6611
Epoch 3/5
26/26 [==============================] - 15s 591ms/step - loss: 0.5196 - accuracy: 0.8368 - val_loss: 0.7870 -
val_accuracy: 0.6611
Epoch 4/5
26/26 [==============================] - 15s 571ms/step - loss: 0.3144 - accuracy: 0.8969 - val_loss: 0.4981 -
val_accuracy: 0.7444
Epoch 5/5
26/26 [==============================] - 15s 582ms/step - loss: 0.2031 - accuracy: 0.9692 - val_loss: 0.4530 -
val_accuracy: 0.7778
```

```
7/7 [==============================] - 1s 73ms/step - loss: 0.4594 - accuracy: 0.8200
Test set
  Loss: 0.459
  Accuracy: 0.820
```



The best accuracy is 0.820

Discussion and Conclusions

I learned from this project how interesting this topic can be. There are drawbacks and advantages in different machine learning classifiers. support vector machine: the computational cost is relatively high. SVM maps low dimensional disordered data to high dimensional space through kernel function (RBF, poly, linear, sigmoid), and separates them through hyperplane but SVM is classified by support surface, that is to say, it does not need to calculate all samples, only a small number of samples need to be removed in high-dimensional data, which saves memory. Naive Bayes. The model cannot learn the interaction between features. But it’s based on its independent assumption, the probability calculation is greatly simplified, saving memory and time. Decision tree: It’s a time consuming in training data but for the model with the lowest data requirement, data can be missing, can be nonlinear, can be of different types, and the model closest to human logical thinking has good interpretability. The best accuracy I can get is no more than 90 percent, so there’s a lot to work on. Like there are some rare words that the model hasn’t seen in its training process, I might use some lexical resources like the ones I mentioned in previous work next time. Also, I will do some researches on sarcasm in sentimental analysis because it’s such an interesting topic.

Citation:

1. Sentiment Lexicons for 81 Languages. (2017, September 13). Kaggle. https://www.kaggle.com/rtatman/sentiment-lexicons-for-81-languages

2. Angelidis, S. (2017b, November 27). Multiple Instance Learning Networks for Fine-Grained Sentiment Analysis. Stefanos Angelidis. https://arxiv.org/abs/1711.09645

3. Vosoughi, S. (2015). Enhanced Twitter Sentiment Classification Using Contextual Information. ACL Anthology. https://www.aclweb.org/anthology/W15-2904/

4. University of Pittsburgh, & Deng, L. (2014, January). https://www.aclweb.org/anthology/E14-1040.pdf.