Programming Assignment 1: Sentiment Analysis using

Hanying Shao

Abstract

In this project, I investigated the performance of 4 classification models: logistic regression, naïve Bayes, support vector machine and decision tree on the binary classification of positive and negative comments. The proposed question is "*What machine learning model works well for sentence-level sentiment classification*?" From the experiment, I found that the naïve Bayes algorithm generally resulted in better accuracy on the unigram count features that I extracted from the raw data. I also experimented on different preprocessing techniques: lemmatization, stemming and removing stopwords to clean the textual data and test the cleaned data on the four algorithms mentioned above. The results suggested that these methods do affect the performance of the classifiers. Meanwhile, I also experimented on different size of features which has also an impact on the accuracy of models. The analysis of results is carried out based on the accuracy of different models and the confusion matrix plot. A conclusion and possible improvement ideas were driven as well.

Parameter Settings

The dataset used for the experiment is 5331 positive and negative snippets authored by Bo Pang and Lillian Lee stored in 2 separate files. I loaded all the texts into a data frame in a column named "Reviews" and labeled them by "neg" or "pos." I first removed all the punctuations and transformed texts into lower-cased letters. Then, I plotted a Word Cloud graph to visualize the words that appear frequently in each positive and negative comments.

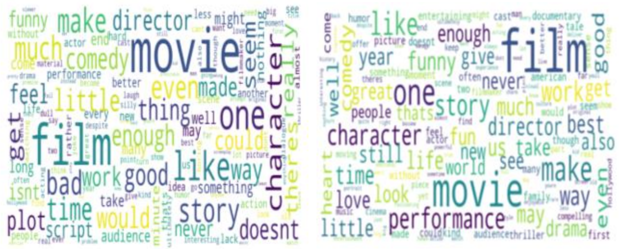
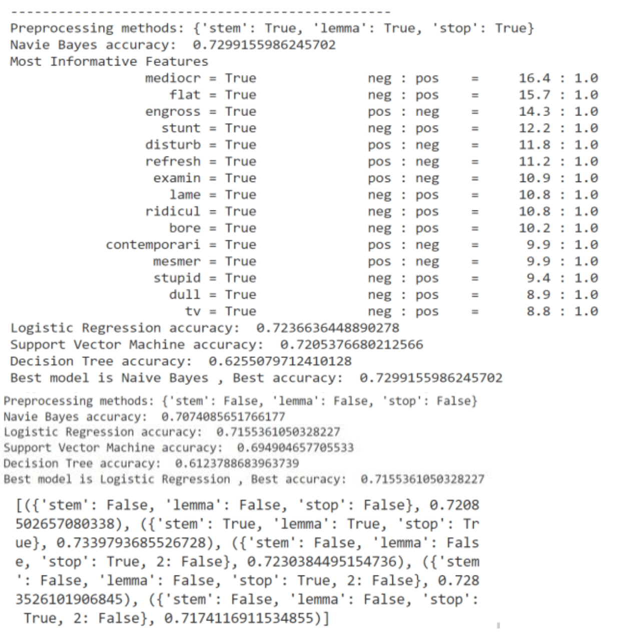


Figure 1: Word Cloud for negative and positive comments respectively

From the graphs shown above used the raw datasets that are not processed by lemmatization and stemming. From the Word Cloud, we can observe that some adjectives like good, funny occur frequently in positive comments whereas they have a smaller space in negative comments.

I merged the 2 datasets and shuffle all the comments to split the dataset into training and test sets later. Then, I implemented a function which includes the preprocessing methods lemmatization, stemming and removing stopwords to test automatically the impact of each technique on the performance of different models.



```
------------------------------------------------
Preprocessing methods: {'stem': True, 'lemma': True, 'stop': True}
Navie Bayes accuracy:  0.7299155986245702
Most Informative Features
                 mediocr = True              neg : pos    =     16.4 : 1.0
                    flat = True              neg : pos    =     15.7 : 1.0
                 engross = True              pos : neg    =     14.3 : 1.0
                   stunt = True              neg : pos    =     12.2 : 1.0
                 disturb = True              pos : neg    =     11.8 : 1.0
                 refresh = True              pos : neg    =     11.2 : 1.0
                  examin = True              pos : neg    =     10.9 : 1.0
                    lame = True              neg : pos    =     10.8 : 1.0
                 ridicul = True              neg : pos    =     10.8 : 1.0
                    bore = True              neg : pos    =     10.2 : 1.0
             contemporari = True              pos : neg    =      9.9 : 1.0
                  mesmer = True              pos : neg    =      9.9 : 1.0
                  stupid = True              neg : pos    =      9.4 : 1.0
                    dull = True              neg : pos    =      8.9 : 1.0
                      tv = True              neg : pos    =      8.8 : 1.0
Logistic Regression accuracy:  0.7236636448890278
Support Vector Machine accuracy:  0.7205376680212566
Decision Tree accuracy:  0.6255079712410128
Best model is Naive Bayes , Best accuracy:  0.7299155986245702
```

```
Preprocessing methods: {'stem': False, 'lemma': False, 'stop': False}
Navie Bayes accuracy:  0.707408565176617
Logistic Regression accuracy:  0.7155361050328227
Support Vector Machine accuracy:  0.694904657705533
Decision Tree accuracy:  0.6123788683963739
Best model is Logistic Regression , Best accuracy:  0.7155361050328227
```

```
[({'stem': False, 'lemma': False, 'stop': False}, 0.7208
502657080338), ({'stem': True, 'lemma': True, 'stop': Tr
ue}, 0.7339793685526728), ({'stem': False, 'lemma': Fals
e, 'stop': True, 2: False}, 0.7230384495154736), ({'stem
': False, 'lemma': False, 'stop': True, 2: False}, 0.728
3526101906845), ({'stem': False, 'lemma': False, 'stop':
 True, 2: False}, 0.7174116911534855)]
```

Figure 2: output results for different combinations of preprocessing methods

From the above results we can observe that the accuracy of all models has shown an improvement of around 3% after stemming, lemmatizing words and removing stopwords. I observed from the results that naïve Bayes algorithm generally gives a better performance on different tests. Using the nltk package, it is possible to display the 15 most informative features as well as the ratio of occurrence of the word in positive and negative comments. From Figure 2, we can observe that the adjectives tend to be most significant features in the sentiment analysis and they have a high *neg:pos* ratio. On top of the method mentioned above, I used the POS tags to classify all the words and used only adjectives, nouns and verbs when selecting features. I tested on different size of features whose results confirmed my hypothesis that conjunctions, adverbs, modals, determiners, etc. are less significant in displaying the sentiment or emotion of sentences. The models showed generally an increase of around 4% in accuracy after using the POS tags and eliminating the less significant words. I finally used 1000 most frequently occurred words as features to train the models.

Conclusion

I used the confusion matrix analyze the results of different models and normalize the number to see proportions instead of numbers.

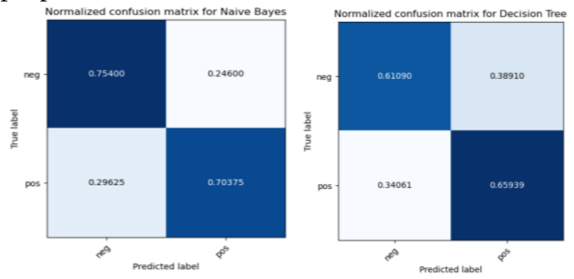


Figure 3: Normalized Confusion Matrix for naïve Bayes and Decision Tree models

From the confusion matrix of naïve Bayes model, we can see that it gives a higher accuracy in classifying negative comments than positive comments. Therefore, the FN is lower than FP. The naïve Bayes shows a best performance compared to all other 3 algorithms mentioned above. Naïve Bayes performs generally well on the categorical variables and scalable with the number of predictors and data points which may explain why it is the best model in the experiment. The decision tree shows instead a worst performance in different tests. decision tree algorithm is sensitive to small changes in the data and a single decision tree is often a weak classifier whereas random forest may give a better result.

In brief, from the experiment, we can observe that lemmatizing, stemming words and removing stopwords improve the performance of algorithms on textual datasets. Moreover, the naïve Bayes gives a highest accuracy in general compared to 3 other algorithms used in experiment.

References

Harvard Business Review "Decision Trees for Decision Making". Retreived from https://hbr.org/1964/07/decision-trees-for-decision-making

Jackie CK Cheung. Class Notes on linear and non-linear classifications.