Sentiment Analysis Project Report

Ena So, 0961375, Nov, 27th, 2019

***Abstract*—In this project, we explore various Natural Language Processing techniques to perform sentiment analysis on movie reviews. We work with 2 different data sets one with negative reviews and one with positive reviews. Using 2 different feature selection and 3 different classification model we compare the performance of each model.**

I. Introduction

MOVIE REVIEWS are frequently used to evaluate the quality of the movie and many people refer to these reviews to pick which movie fits their interest. Enormous amount of textual movie reviews available online provides us deeper qualitative information than numerical star ratings that gives quantitative information. But how can we analyze the textual reviews and conclude with whether ite objece objec meets readers expectation?

Sentiment Analysis is a well-known technique in the realm of natural language processing. It's objective is to determine the polarity of the data which is given as set of texts. In this project we classify the given data into 2 different cases: (+)positive, or (-)negative polarity.

In this project we aim to use Sentiment Analysis on a set of movie reviews given by reviewers and try to understand the overall reaction of the movie was.

II. Technique Description

Python has various modules that can be imported and use high level functionality libraries. Scikit learn is one of the most used library in machine learning field. It supports functionalities for feature selection and classification model. We use 2 different feature selection methods for extraction of meaningful features from the review text which could be used for training purposes. We apply three different classifiers to train the model.

First feature selection method is Bag of Words. we calculate the total term count for each token from entire document(review) list and use this data to create new feature representation. Maximum feature size being 500, 1000, 1500, 2000, 2500, and 3000. This converts words to number using the bag of word approach, where all the unique words in all the documents are converted to features. All the documents can contain tens of thousands of unique words. But the words that have a very low frequency of occurrences are unusually not a good parameter for classifying documents. Therefore, we test multiple max feature size and compare the performance differences. Also the minimum number of documents that should contain the feature is set to 5 which means we only include those words that occur in at least 5 documents.
In addition, maximum document frequency is set to 70 percent since words that occur in almost every document is usually not suitable for classification because they do not provide any unique information about the document.

Lastly, we remove the stop words from our text since, in the case of sentiment analysis, stop words may not contain any useful information.

Second feature selection method is TF-IDF. The feature representation is similar to Bag of Words model except that we use TF-IDF values for each word instead of their frequency counts. TF-IDF resolves the drawback of Bag of Words, which does not take account the face that the word might also be having a high frequency of occurrences in other documents as well. It resolves by multiplying the term frequency of a word by the inverse document frequency. The TF-IDF value for a word in a particular document is higher is the frequency of occurrence of that word is higher in that specific document but lower in all the other documents.

The overall task in this project is for classification of reviews as positive(+) or negative(-). Therefore, for this classification task we explored multiple classification models on above feature representations. We use model ranging from the Random Forest Classifier to Bernoulli Naive Bayes' classifier. We train the model with K-Fold cross validation to check if the data was over-fitting. We divide the data set into 85:15 ratio 85 for training for K-Fold cross validation and rest 15 for validating with top 4 classifiers. That 85 also divides into 80:20 ratio where 80 is for training and 20 for testing. By doing this we have 36 test results with 2 different feature method with 6 different feature sizes and 3 different classifiers. We choose 4 best classified models and test with 15 percent of data set we reserved.

III. Implementation & Design

We use Jupyter Notebook to implement the model and train/test the data using since unlike creating python program, with Jupyter Notebook you can have multiple cells in different segment and run each segment separately. Due to this characteristic of Jupyter Notebook we are able to display the executed scripts right underneath the code which makes easier to keep track of what is happening along the execution.

Also design wise it was clear and easy to follow the code since terminal and separate text editor is not needed. Everything is done in one place. Moreover, Jupyter Notebook is for python which has numerous modules that can be imported. We use sklearn module that contains various classifiers and vectorization model that can convert not numerical data set and represent it as numerical data set.

IV. Results & Analysis

As discussed above, we tried multiple classification models on various feature representations of the textual information in the reviews. First, we pre-process the input data set such as tokenizing, lemmanizing, and removing unnecessary characters then analyze the pre-processed data set containing the movie reviews just to have some insight on what are our data set looks like.
Using this pre-processed data we run K-Fold cross validation with 6 different feature size and Table II displays the result of the training and testing.

Analysis	Value
Min Sentence	1
Max Sentence	112
Avg Sentence	32.36
Min Token	16
Max Token	2305
Avg Token	633.7365
Avg Sentence Length for Collection	39.60853125

TABLE I
Data Analysis on Movie Review data sets.

Feature/Feature Sz	Random Forest	Multinomial	Bernoulli
Bag of Word/500	76.9	76.4	75.8
Bag of Word/1000	77.9	82.3	77.17
Bag of Word/1500	80.41	81.23	78.4
Bag of Word/2000	80.11	81.17	79.82
Bag of Word/2500	81.29	82.29	79.58
Bag of Word/3000	80.4 7	81.41	79.41
TFIDF/500	76.41	76.58	75.88
TFIDF/1000	77.88	78.66	77.17
TFIDF/1500	79.94	81.58	78.47
TFIDF/2000	78.58	82.05	79.82
TFIDF/2500	78.05	82.82	79.58
TFIDF/3000	79.88	82.64	79.41

TABLE II
Average Accuracy Score for 5 Folds

Base on the result on Table II we pick 4 best scored average classified model with corresponding feature and feature size we validate the model with development set that we split in the beginning. But before we move on to validating the development set let's take a look at the result from cross validation with K-Folding. We can see that smaller the feature size is the lower the accuracy score is. This is because the less data you feed in to the model to train with the less data to train and test against it. However, high feature size does not necessarily gives the highest score. This is because there is a threshold where model score starts to plateau after certain point. Not only it can plateau but also the score can be lower that the slightly smaller feature size.

etc	percision	recall	f1-score	support
0	0.74	0.76	0.75	154
1	0.74	0.72	0.73	146
Accuracy			0.74	300
Macro Avg	0.74	0.74	0.74	300
Weighted Avg	0.74	0.74	0.74	300

TABLE III
Development Set MultilNomial with TFIDF 2500

	precision	recall	f1-score	support
0	0.77	0.78	0.77	154
1	0.76	0.75	0.76	146
Accuracy			0.77	300
Macro Avg	0.77	0.77	0.77	300
Weighted Avg	0.77	0.77	0.77	300

TABLE IV
Development Set MultilNomial with TFIDF 3000

	precision	recall	f1-score	support
0	0.76	0.76	0.76	154
1	0.75	0.75	0.75	146
Accuracy			0.75	300
Macro Avg	0.75	0.75	0.75	300
Weighted Avg	0.75	0.75	0.75	300

TABLE V
Development Set MultilNomial with Bag of Words 2500

	precision	recall	f1-score	support
0	0.80	0.84	0.82	154
1	0.82	0.78	0.80	146
Accuracy			0.81	300
Macro Avg	0.81	0.81	0.81	300
Weighted Avg	0.81	0.81	0.81	300

TABLE VI
Development Set Random Forest with TFIDF 2500

Above 4 tables were the best performed in cross validation tested with development set.

V. Future Improvement

Unfortunately, our development test result was not above 0.85 but for the future imporvement there are couple things we can do. We can add hyper parameters for the classifiers and add POStag during the preprocessing stage. Trying 10 fold cross validation might also improve out test accuracy.

VI. Build Guide

We soley worked on Jupyter Notebook therefore, there is nothing to build just open up the notebook and run all the cells. The result of each cell will be displayed bottom of the cell. It was developed with Python version 3.

Refernece

Citations :

[1] Text Classification with Python and Scikit-Learn , 2018,https://stackabuse.com/text-classification-with-python-and-scikit-learn/

[2]Predicting Reddit News Sentiment with Naive Bayes and Other Text Classifiers, 2017, B. Martin, N. Koufos, https://www.learndatasci.com/tutorials/predicting-reddit-news-sentiment-naive-bayes-text-classifiers/

[3] Cross Validation and Grid Search for Model Selection in Python, 2018, U. Malik, https://stackabuse.com/cross-validation-and-grid-search-for-model-selection-in-python/