Manmeet Singh
Khanna

CS 584
Homework 1

HW 1 (584-Rangwala): Movie Review Classification

User Name: Deviant
Rank
:Accuracy Score:
0.80

Approach:

In this Classification system, there are two major parts:

- Preprocessing Data
- Classification using Knn

These two processes have their own significance and are equally important as theyare dependent on each other.

Data Description: The rows in the data were in the following form:

Train Data

“ <Rating as +1/-1> + </t> + <Review>

Test Data

“<Review>

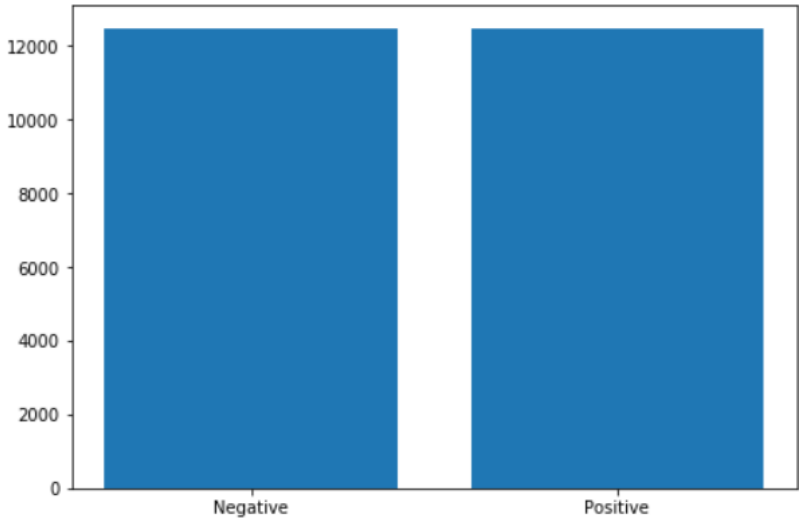
Each having 25000 reviews i.e. number of rows. The data seemed to be a web scraped text, containing html tags too.

Preprocessing Data:

Initially, the files were downloaded directly and saved as a “Csv” file as “Train.csv” and “Test.csv”. After which the train and test data were read using “pd.read_csv”, separating it with the delimiter #EOF with keeping the header as None. Thus, this gave a file which gave all the review scores with the review on the same lines separated by a tab for the Training set. After which by creating a dictionary, a new dataframe was made which separated the “Scores” and the “Reviews” into different columns. The test was simply stored in a dataframe.

To check the train data homogeneity, “value.counts()” was used which gave a number of positive and negative reviews. This is a bar graph representation, using Matplot.pyplot.

This indicates that the data is in good shape and totally balanced. This is a good type of data while considering a majority vote type model. The Scores of the reviews were then encoded



in ‘1s’ and ‘0s’ using the Scikit-learns’ Label Encoder and the inverse was taken right after, representing “+1 as 1” and “-1 as 0”.

Now, the reviews were to be cleaned. The four main libraries used were:

1. PorterStemmer() - To stem the words to the root word
2. WordNetLemmatizer() - To lemmatize the stemmed words
3. stopwords.words('english') - To remove the common words
4. Regular Expression - To remove all extra expressions, punctuations.

A function named “cleaned_rev_joined_stem()” was created after clubbing various different methods to preprocess text. The reviews were initially made lower case as it is the best way to combine the same features together as our machine is not case sensitive. As mentioned earlier this data contains HTML tags namely the “<br\><br\>” tag which occurred in this pattern and were removed. The “re” library was made use of to remove the punctuation marks and extra expressions in the reviews. Then, the stop words were removed and the stemming of the words was done. Stemming was followed by Lemmatization as it gavebetter classification of the features.
These reviews were then vectorized using the TF-IDF vectorizer to fit_transform the Train dataset. This gives a sparse matrix of these dimensions <25000x50432> which means that there are 25000 rows and 50432 columns i.e Features in the Training dataset. Thus the Test data was then Transformed only using the same TF-IDF vectorizer. Thus the Test was transformed w.r.t the fit of the Training Dataset. They had the same dimensions.

Classification of Data:

The data was classified using the KNN i.e. K-Nearest-Neighbours Method.

To do this, there were 3 major steps:

1. Finding the Cosine Similarities between the Test and Train vectorized sparse matrices.
2. Sorting the neighbours and selecting the ‘k’ nearest.
3. Predicting the output

The cosine_similarity() function was imported from Sklearn directly. Thus, the similarity of the Test data was taken w.r.t to the Train data. Here, the input form was the vectorized sparse matrices. Cosine similarity, or the cosine kernel, computes similarity as the normalized dot product of X and Y:

$$K(X, Y) = \langle X, Y \rangle / (||X||*||Y||)$$

Where, X = Test Data
Y = Train Data

After we get the similarity values, it looks something like this:

```
In [45]: cs

Out[45]: array([[0.02456665, 0.04573063, 0.0396872 , ..., 0.03987518, 0.0541265 ,
                 0.04621835],
                [0.02218302, 0.0406358 , 0.1031707 , ..., 0.03802199, 0.0495488 ,
                 0.02724528],
                [0.02598966, 0.01422231, 0.04292663, ..., 0.00684698, 0.05870772,
                 0.07401918],
                ...,
                [0.01531705, 0.02497788, 0.04454334, ..., 0.02766161, 0.0723699 ,
                 0.0198918 ],
                [0.0445442 , 0.01788178, 0.05424373, ..., 0.03570718, 0.04258252,
                 0.03287905],
                [0.02215335, 0.01695519, 0.02252441, ..., 0.02338865, 0.03734186,
                 0.01798841]])
```

The KNN function is defined which just sorts these values and takes in to account the values of 'k'. Here, Numpys' argsort() is used to sort the values in a descending order. 1 being the **maximum similarity** and 0 being the **lowest similarity.**
The "predict()" function is then defined which has two count variables initially set to '0' and takes into account the values from 'y_train' which are the ratings of the reviews. The count variables are incremented depending on whether the neighbour is negative or positive. When there is a negative neighbour, the negativeReviewsCount increases by 1 and similarly for positiveReviewsCount depending on the positive review. Thus, these counts are then compared and the higher count is the output prediction for that review. Finally, the value of k is set as per the cross validation step which is '143' for this model. Here a list named test_sentiments is created which stores all the values of the predictions. The predicted values are in the form "+1" for a positively predicted review and "-1" for a negatively predicted review. This file is then converted and saved as a '.dat' file.

Methodology:

The training data was like any other real world data which was full of unwanted materials and needed preprocessing. I saved it as a csv directly which helped me read the data easily as compared to when I used read_table as I had previously saved it as a '.txt' file. I converted the labels from "+1" to 1 and "-1" to 0 as a machine readable form. Removing stop words, stemming and then lemmatizing definitely helped the model to perform faster and better. I got an accuracy as well as time boost in the cross validation part with these parameters to preprocess the data. The removal of <br\><br\> tags separately was important despite regular expression being used. If that wouldn't have been done, the "br" was being considered as a separate feature which didn't make anysense to the review. Also, there were these tags present on almost all reviews which could not be good for the vectorization part.
I made use of TF-IDF as it gave a frequency of features with respect to the entire document as shown by the name in precise fractional value. This increased the efficiency of the model.
I performed cross validation by incrementing the value of k randomly ranging from k=3 to k=201. Here I took a 80-20 split with the train as 80% and the test as 20%. I chose random state as 1 for uniform split of the data. The k value decided for this model was '143' which gave the highest accuracy during k-fold method which was '0.827'.

References:

https://scikit-learn.org/
pandas - Python Data Analysis Library (pydata.org)
Natural Language Toolkit — NLTK 3.6.2 documentation