HW1: Movie Reviews Classification

Details:

Name: Siddhanth Kalyanpur
Miner2 Username: mb13
GMU Id: skalyanp
Rank: 116
Public Score: 0.81

Steps to run the code:

1. Download MovieReviews.ipynb file.
2. Open Jupyter Lab(or IDE of choice) on Anaconda Navigator.
3. On the menu bar click File -> Open -> MovieReviewsClassification.ipynb.
4. Keep the test.dat and train.dat file in the same directory as the above file.
5. Run each cell sequentially.

Introduction:

K Nearest Neighbour is a simple algorithm that stores all the available cases and classifies the new data or case based on a similarity measure. It is mostly used to classifies a data point based on how its neighbours are classified.

Solution:

K-Nearest Neighbor is implemented and then tested on 15000 reviews to predict their sentiment, The positive reviews are marked with +1 and negative with -1. The k-NN model was first trained using 15000 training data which consisted of Reviews and Sentiments provided to us.

Approach:

1. **Import libraries**. Following are used in the classifier code:
   - Numpy for working mathematical operations on arrays.
   - BeautifulSoup for cleaning text.
   - Islice used for slicing list and Tuple.
   - Sci-kit learn feature extraction for Count vectorization.
   - TfidfVectorizer for Term Frequency - Inverse Document Frequency.
   - RE for working on Regular Expression.
   - Natural Language Toolkit for Language processing on reviews.
   - Matplotlib to plot the predictions.
   - Counter for working on hashable objects.

2. **Read** the training and test data using FileReader.

3. **Split** the training file into two files: Reviews and Sentiments.

4. **Remove Noise** from the data using below methods:
   - Remove HTML Scripts – BeautifulSoup library for pulling out of HTML and XML features.
   - Remove Brackets and Special Characters – Using Regex, all the special characters and brackets present in review is removed.

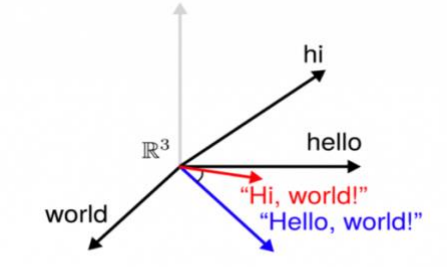
5. **Preprocess** the data using following steps:
   - Remove stop words – Use the stopwords module of NLTK library to get a list of stop words and then remove the stop words from the review
   - Apply Stemming – Use LancasterStemmer for Stemming the words in review. It is fast and better than PorterStemmer.
   - CountVectorizer used to convert the train reviews into vector. The parameters min_df and max_df discards the minimum and maximum frequency of the words which do not contribute any value to the data. ngram_range is used to keep track of the sequence of words that needs to be searched , ex : ngram_range=(1,3) can capture “very good movie”.

6. The review is converted to a sparse matrix of word representation using TfidVectorizer.

7. The sparse matrix is then used to find the inverse document frequency, it reduces the priority of frequently occurring words.

8. Next it is **normalized** to get the cosine similarity.

9. To compute the **cosine similarity**, you need the word count of the words in each document. The CountVectorizer or the TfidfVectorizer from scikit learn lets us compute this. The output of this comes as a sparse_matrix . The word vectors will help us group similar words like shown below :



Soft Cosine Measure

10. The training data is fit_transform() so that we can scale the training data and also learn the scaling parameters of that data and test data is transform() using the respective mean and variance calculated in fit_transform()

11. Calculate the cosine similarity by taking dot product of the test matrix and transposed train matrix.

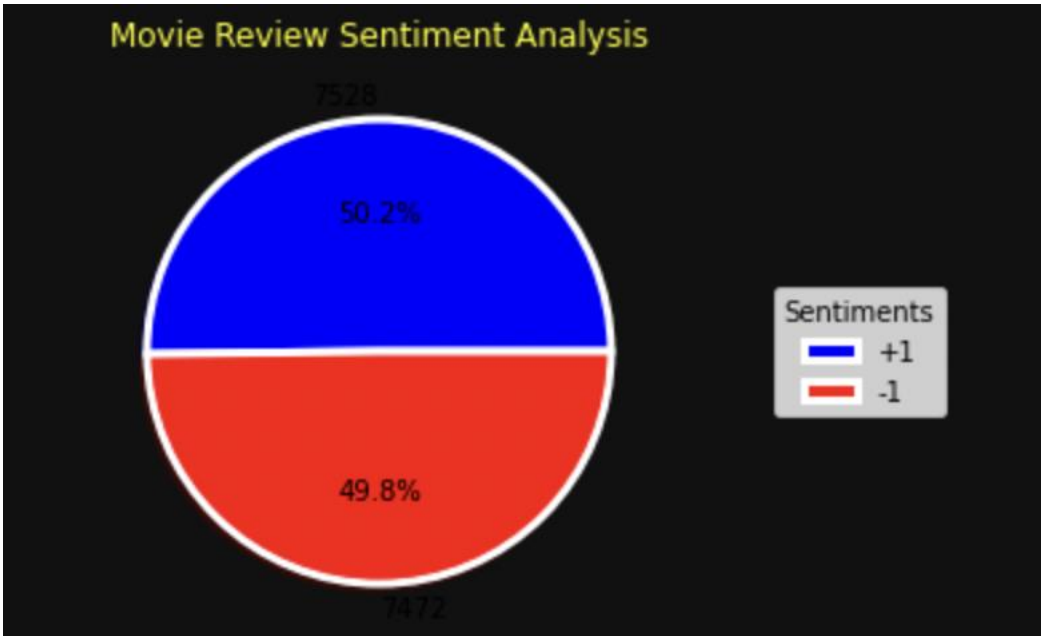
12. The value of k should not be too high as this will increase overfitting and the result will be biased. It can't be too low as noise could have higher influence on the result. My implementation of k was based on majority vote of number of positive and negatives for the particular k value and also based on accuracy for the particular k value.

13. K-Nearest Neighbor algorithm is applied as below:
    - Cosine Similarity matrix is taken, and each row column is arranged in increasing order of similarity.
    - Match the corresponding review with sentiment
    - Put together the positive and negative neighbors.
    - If the count of positive neighbors is more than negative neighbors, then the review is classified as positive sentiment else negative sentiment.

14. Write the predicted output in a file using FileWriter.

Output/Result:

The sentiments predicted for Testing Data [Review] by k-NN model is as below:



According to the above prediction, out of 15000 test reviews, 50.2% i.e., 7530 reviews are Positive [+1] and 49.8% review i.e., 7470 are Negative [-1]

Improvement:

1. Instead of using PorterStemmer() for stemming the words, LancasterStemmer() is used which minimizes the runtime while preprocessing the data.
2. For word Tokenization ToktokTokenizer() is used which speeds up the token creation and is faster than word_tokenize()
3. Instead of using for loop, dot product of the matrix is done. This is considerably faster and reduces a lot of runtimes.

Conclusion:

Successfully implemented K Nearest Neighbor classification algorithm using Python. The model can predict sentiments either positive or negative based on the reviews.