HW1: Movie Review Classification

Details:

Name: Aniket Pandey
Miner2 Username: 414
GMU Id: apandey7
Rank: 18
Public Score: 0.80

Steps to run the code:

1. Download MovieReviewClassification.ipynb file
2. Open JupyterLab on Anaconda Navigator or Google Colab Notebook
3. On the menu bar click File -> Open -> MovieReviewClassification.ipynb
4. Keep the test.dat and train.dat file in the same directory
5. One by one run each cell in the notebook

Introduction:

The K-Nearest Neighbor classification algorithm assumes that similar things exist in proximity. In other words, similar things are near to each other. Using this approach, we will write our own kNN classification algorithm in Python and predict the sentiments based on the review.

Solution:

K-Nearest Neighbor is implemented and then tested on 25000 reviews to predict their sentiment, The positive reviews are marked with +1 and negative with -1. The k-NN model was first trained using 25000 training data which consisted of Reviews and Sentiments already known to us.

Process/Approach:

1. Import necessary packages and libraries. Following libraries are used in the classifier code:
   - Numpy for working mathematical operations on arrays
   - BeautifulSoup for cleaning text
   - TfidfVectorizer for Term Frequency - Inverse Document Frequency
   - RE for working on Regular Expression
   - Various libraries of NLTK for preprocessing data

- Matplotlib for plotting graphs
- 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 cleaning methods:
   - Remove HTML Scripts – BeautifulSoup is used to parse the review and remove all the associated HTML tags in the review.
   - Remove Brackets and Special Characters – Using regular expressing, 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.

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. 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()

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

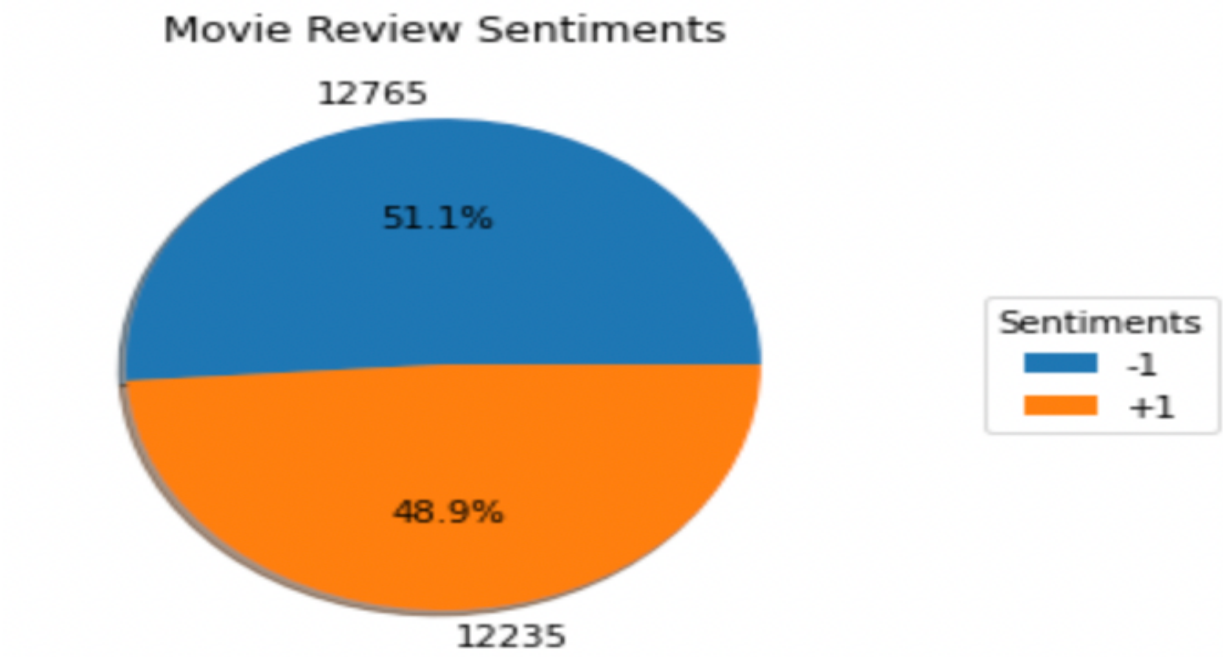
11. Get the value of k by taking sqrt(no of review) i.e. sqrt(25000) = 158 (Approx.)
    The value of k should not be too high as this will increase overfitting and the result will be biased.

12. 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.

13. 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 25000 test reviews, 48.9% i.e., 12235 reviews are Positive [+1] and 51.1% review i.e., 12765 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 review.