Sentiment Analysis of movie reviews

In this program, I analyzed the sentiment of movies if it is positive or negative by processing around fifty thousand reviews by using machine learning algorithms. I used the movie reviews as my raw data. The data was divided into training data and test data in two different text files. At first, before using any machine learning algorithm to process the data, the data was pre processed by first reading the file into a list. The raw text was pretty messy for to be used so the data was cleaned up by removing unnecessary texts and symbols like '
', '.','?', etc. These things does not influence the sentiment of the movie reviews, so they were removed. Similarly, the stop words like 'a', 'the', 'of', 'from', 'at', 'I' etc. were also removed as they do not contribute in the sentiment of the reviews.

Then, I converted each review to a numeric representation so that it makes sense to our machine learning algorithm which is called vectorization. It was done by creating a one very large matrix with one column for every unique words in the corpus. And, instead of the just simply noting whether a word appears in the review or not, the number of times a given word appears was also included. This helps increase the predictive power of the sentiment classifier.

The data was normalized too as it helps to convert all of the different forms of a word into one. Stemming approach was used to normalize the data. Among several stemming algorithms implementations, porter stemmer was implemented.

Moreover, n-gram technique was implemented in building the best model. N-gram helps to add more predictive power to our model by adding the word sequence instead of just one word. N-gram technique helps to capture some of the negative sentiments that would have been missed if we just take one word while analyzing the data. In my model, I used ngram range as (1,3) so I considered 3-word sequences in addition to single words which helped to increase the accuracy of the model.

Finally, after transforming our data set into a format suitable for modeling, I built a classifier. I used Neural Network Multiple layer perceptron classifier which was imported from from the sci-kit learn library to build my model. I tried to build the model using SVM and logistic regression but I did not get the accuracy as good as I got it from the neural network. The target/labels we use will be the same for the training and testing since both datasets are structured the same where the first 12.5k are positive and the last 12.5k are negative.

Final Result:

Hence, to build the final model that provided me with the highest accuracy, I preprocessed the data by removing useless words and symbols. A small set of stop words were removed. The data was normalized by using the stemming approach. A n-gram range (1,3) was used. And, while vectorizing the words, we considered the number of times the given word appears instead of just representing each review as a binary vector(0s and 1s).

The pre processed data was then trained using Neural network Multiple Layer Perceptron Classifier. The parameters of the classifier like 'activation' was set to 'relu' which boosted my accuracy than using other activation attributes like 'logistic'. The 'solver' was set to 'adam' which is also the default solver and it works relatively good in pretty large datasets like used in this program. The 'hidden_layer_size' was set to (10,). The default hidden layer size is (100,) which can take comparatively a lot of more time to run. I tried multiple number of hidden layer sizes and ended up with this which takes fair enough amount of time to run and provides good accuracy percentage. Similarly, the 'random_state' was set to 1. And, the number of max iterations was reduced to 3 whose default value is 200 which might take enormous amount of time to run. The training data was fit into this model and the accuracy of the test data was calculated. The final accuracy percentage was determined to be around 91.3 %.(0.91296)

Note: I have also built another model by varying some features such as (binary=True) and (hidden layer sizes=(11,). Just, in case, the model I described above shows different accuracy value as I got slightly different value sometimes while running. In, this second model, I got accuracy around 91.21%(0.91216) .