SAN JOSÉ STATE UNIVERSITY

CMPE 239 - HomeWork 2

Amazon Review Classification

1) Name: Jayam Malviya
SJSU Id: 01435567

2) Rank and Accuracy at the time of writing report :
Accuracy : 53.10 %

3) Approach:

Preprocessing

- First we have cleaned out the symbols dot(.) and have also removed all the words that are less than 4 in length. And have also removed characters that are not alphabet.

- We have also cleaned our document from stop words for this we have used nltk.corpus to remove all the stop words from the document.
- We have also used Pandas Dataframe to hold our data in a two dimensional collection.

Training Data Processing

- We extract the labels of all the documents in the training data while processing them and have created a map having that label stored against the id (number/position) of that document in the training data set.
- To calculate the term frequency of the words we have used CountVectorizer of SKLearn. After calculating the term frequency of the document we calculate the Inverse Document Frequency for the document. In Order to maintain same dimension between training data and the testing data we have considered that maxFeatures allowed are 2000. In this way we both the training and test data will align in dimensions and cosine similarity will be possible.

Testing Data Processing :

- Testing data processing is also very similar to training data, here we don't have labels to extract but we still have the TfIDf calculation part for the testing data, this is required as it will enable us to calculate the cosine similarity between test & training data.

Cosine Similarity and K Nearest Neighbours:

- To calculate cosine similarity we have used scipy's spatial module. This method in takes two sparse matrix and returns the cosine similarity between them. This similarity measure is used to determine the neighbourhood of the test Data vector in the training data space.
- K Nearest Neighbour : To calculate the top closest neighbours of a vector in the train data space. We calculate the test vector data similarity with all the vectors in the training data set (this part is very costly). And then sort the vectors based on the similarity in descending order, so that the most similar ones appear before the less similar ones. Then we select the top K elements from this sorted similarity and return them as a tuple. In another subroutine we utilize these returned most similar tuples to get the majority label that can be assigned to the test vector. For this we count the number of positive and negative labels in the neighbourhood. If the positive neighbours are more than we assign positive label to the test data. Else if the negative neighbours are more we assign negative label to the test data. And if unfortunately there is a tie we assign the label randomly to the test data. The estimated time in this way for determining the label of any test data is 1 sec per document or review. This can easily take long hours if the testing data is huge like thousands in this case.