**SJSU CMPE 239 Data Mining**

**Homework 2 - Report**

**Submitted By: Niveditha Bhandary [010820550]**

**Spring 2017**

**Rank & Accuracy score at the time of writing the report: 6, 79.94**

**Steps Followed:**

First I started with a general framework for the KNN classifier assuming K = 3. Later

added evaluation mode to find the best K. Also added several preprocessing steps

such as filterLen, stemmers, K-mer (with K=2 and K=3).

For Preprocessing the train and test files, I followed the following steps:

1. Open train.dat file in read mode and read the lines.

2. Create list variable train\_labels with all the labels (+1 and -1) from train.dat file

3. Remove special characters and convert all words to lowercase

4. Use filterLen function to filter out words that have less than 4 letters except for

the word “bad”

5. Remove suffixes (and in some cases prefixes) in order to find the root word or

stem of a given word using Potter2 stemmer.

6. K-mer implementation with K=2 and K=3: Every document is passed through

grouper function which groups 2 simultaneous and 3 simultaneous words and

adds them to the original list of words.

7. Repeat the steps 1-6 (except step 2) for test.dat file.

8. Preprocessed train and test reviews are obtained in 2 different lists. Python

extend function was used to combine these two lists.

For building the csr\_matrix and to normalize it following steps were followed:

9. The build\_matrix function was used to transform the list of lists of words

obtained in Step 8 into a sparse matrix

10. csr\_idf function was used to decrease the importance of popular words

11. csr\_l2normalize function was then used to normalize the matrix to simplify

cosine similarity calculation

After step 11, The first half of the csr\_matrix had normalized train vectors and second

half had normalized test vectors.

For finding cosine similarity between test vectors and each of the train vectors, initially

I used numpy.dot function. This implementation took a lot of time (hours) to run and

sometimes didn’t give results. I learnt that numpy is a dense linear algebra library. It

converts the sparse matrix to dense matrix before computing the dot product. Hence

the computation took a lot of time. Then I used cosine\_similarity function from

sklearn.metrics.pairwise to get pairwise cosine similarity using the sparsity of the

matrix. This computation took relatively less time. (under 10 minutes)

For K nearest neighbor classifier following steps were followed:

12. Pairwise similarities are converted to list of lists with similarity of each test

vector with each of the train vectors.

13. Each of the list within the list of lists was zipped with train\_label information

obtained from Step 2 to get each similarity and corresponding label in tuple

format.

14. Then the list of lists is sorted in descending order and top k similarities are

chosen. The corresponding labels are added, if the result is zero, then a random

integer between (-1,2) is chosen until +1 or -1 is obtained. If a positive result is

obtained, then a label ‘+1’ is assigned, else a label ‘-1’ is assigned.

15. The assigned labels are then written to output file.

For finding the Best k value I defined 2 modes of operation (K values in range [0,20]

were considered):