Name: Arkil Thakkar

Student_Id : 013825292

F1 -Score: 0.3820 Rank :40

Data Pre-processing:

1. Using Pandas Dataframe loaded training data from 'train.dat' file in a 'train_data' and testing data from 'test.dat' file in a 'test_data' data frame.
2. Gave header columns names to data read from data files.
3. Extracted Class Label from train_docs and formed a list of class.
4. Splitting extracted_train_data and extracted_test_data in to list as docs and test__docs respectively.
5. Filter the list on the basis of length, so list having less than 3 are filtered out.
6. Using stopwords for English from NLTK library removed stopped words like an, about, between etc.
7. Applied Lemmatization and Stemming to text using WordNetLemmatizer and LancasterStemmer from NLTK library to form lemm_docs and stemm_docs
8. In 'build_matrix' method build a CSR matrix having indices, values and pointers
9. Using 'build_matrix' built CSR matrix from training data in 'docs' which will return CSR matrix 'mat' and a vocabulary dictionary 'idx', which represents features (dimensions, columns) of our CSR matrix – 'train_mat'.
10. Used same vocabulary dictionary 'idx' to build CSR matrix for test data in 'test_data' list which will return CSR matrix - 'test_mat'. By this way, we are going to use the same features to build test CSR matrix as we used for train CSR mat
11. Applied inverse document frequency to CSR matrix to reduce importance of words' to generate mat2 and test_mat2
12. To create non sparse coefficient used L2 normalization on CSR matrix and generated mat3 and test_mat3'.

Model training and testing:

1. Used Cosine Similarity as a parameter to find K nearest Neighbors.
2. Build cosine_similarity taking dot product of 2 matrices
3. Compared Cosine Similarity of test row with all instances of training data to find K nearest neighbors of test instance.
4. From K neighbors obtained, used an epsilon value to filter out the neighbors having similarity less than epsilon
5. From obtained K-X {X – number of eliminated neighbors}, used majority class voting to obtain the class label for the given test_instance
6. Tried different values of K and Epsilon to get optimal F1 Score.
7. Predicted the class labels and wrote the output in 'format_final.dat' file

.

Parameter Tuning:

1. Different combinations of K and Epsilon.
2. Tried Porter Stemming, Lancaster Stemming and Lemmatization for preprocessing
3. Tried changing min_len to filter the words that are less than min_len characters.