Name: Vishweshkumar Patel

F1 – Score: 0.7736

Approach:

Data Pre-processing:

1. load training data from ‘train.dat’ file in a ‘df’ data frame and testing data from ‘test.dat’ file in a ‘tdf’ data frame.

2. Convert ‘df’ data frame to two separate lists named ‘docs’ and ‘cls’. ‘docs’ is a list of strings, where each string represents a document and ‘cls’ is a list of strings, where each string represents a class to corresponding document in ‘docs’ list. Convert ‘tdf’ data frame to a list named ‘test_docs’, made of string where each string represents a document.

3. Convert string value of ‘cls’ list to integer value.

4. Convert ‘docs’ and ‘test_docs’ to list of lists, where each inner list contains words from a single document.

5. Remove Stop words of ‘English’ language from each inner list that represents a document.

6. Filter out terms like ‘a’, ’an’, ’the’, etc. that are too short, by providing minimum length to ‘filterLen()’ function.

7. Convert each word in a list of lists to lower case and remove white space characters, special characters and all unnecessary characters and just keep alphanumeric characters in both ‘docs’ and ‘test_docs’.

8. Build CSR matrix for training data in ‘docs’. That will return CSR matrix ‘train_mat’ and a vocabulary dictionary ‘idx’, which represents features (dimensions, columns) of our CSR matrix – ‘train_mat’.

9. Use same vocabulary dictionary ‘idx’ to build CSR matrix for test data in ‘test_data’ list. That will return CSR matrix - ‘test_mat’. Thus, we will be using the same features to build test CSR matrix as we used for train CSR matrix.

10. Apply inverse document frequency technique to reduce importance of most popular words contained in CSR matrices 'train_mat' and 'test_mat' and generate 'train_mat1' and 'test_mat1'.

11. Apply l2 normalization to both of the CSR matrices 'train_mat1' and 'test_mat1' and generate 'train_mat2' and 'test_mat2'.

Model training and testing:

1. Used 'SGDClassifier' from 'sklearn.linear_model'.

2. Used 'hinge' - loss function, 'l2' - penalty, 1e-3 – alpha, 42 – random_state, 1000 – max_iter and None – tol ---- tuning parameters to create a classifier model for 'SFDClassifier'.

3. Created a model by using 'fit()' function of the classifier and provided 'train_mat2' CSR matrix and 'cls' list as input parameters to the classifier.

4. Once model is trained then use 'predict()' function by passing 'test_mat2' CSR matrix as input to the function to predict(classify) test data. Store classified data in a list. Then, Store that predicted data to a file.

Methodology for choosing approach and parameters:

Data Pre-processing:

- Removed Stop words of 'English' language. I removed stop words because stop words are those common words that carry less importance/meaning as a feature in CSR matrix compared to regular dictionary words.

- Removed white space, special characters and other unnecessary characters present in word in a document. Lower cased each word, so each unique word comes in account for only one time as a feature (dimension) in CSR matrix.

- Filtered document to retain words with length '4' or greater to remove some of the most common words.

- Converted document to CSR matrix, which is memory efficient representation of a sparse document vector. With CSR matrix, we can easily and efficiently do computations and train the model for large documents that may contain millions of words.

- Model will have only those features(dimensions) that are provided in training data, so to retain same dimensions I have used same dictionary for test data to generate CSR matrix.

- Documents may contain words that are more frequent in each of the documents, so if we use their frequency directly in CSR matrix then model will be trained in biased manner. To avoid that scenario, I have used Inverse document frequency technique, where idf = log(N/df), where N = total number of documents in collection, df = document frequency.

- To make distribution of values fall in a particular range, I have normalized CSR Matrix by l2 Norm, that is nothing but Euclidean distance measure.

Model Selection, Training and Testing:

- I have tried two different models, namely 'Naïve Bayes - MultinomialNB' and 'Stochastic Gradient Descent Classifier'.

- Naïve Bayes operates on principal of conditional probability and tries to classify a data tuple based on conditional dependence on features of the model. It selects the classifier with the highest confidence as the predicted classifier for a data tuple.

- As the name suggests Stochastic Gradient Descent classifier depends on stochastic process with gradient descent approach. In gradient descent approach we try to find optimized weights for a model in such a way that for a given training data set, function can predict labels correctly and approximate as many as labels same as accurate labels. We try to optimize weights in each iteration and we stop when we have similar weights and accuracy when compared to previous iteration. To avoid overfitting in SGD, we use regularization technique.

- I have used sklearn's built in algorithms so, the internal workings and all things were taken care of by sklearn. I just need to set hyper parameters for SGDClassifier model. I used 'hinge' - loss function, 'l2' – 'penalty', 1e-3 – 'alpha', 42 – 'random_state', 1000 – 'max_iter' and None – 'tol' ---- tuning parameters to create a classifier model for 'SGDClassifier'.