ASSIGNMENT-05

README

How to Run the Code?

1. Extract the files of 20_newsgroups.
2. Must have NLTK.
3. Open Jupyter Notebook and make sure that files are available in the same folder as jupyter notebook and then start running the code from the beginning.

Preprocessing Steps:

Normalization:

• converting all text to the same case (upper or lower).
• removing punctuations

Stop Words:

- We may omit very common words such as the, a, to, of, be from unigram inverted index not from positional index.

Stemming:

- Used porter stemmer to stem the words. It is faster than lemmatization and does a good enough job of stemming related words to the same stem.

Tokenization:

- Cut character sequence into word tokens.

Num2Words:

- Convert number to words.

Single Character:

- Remove a character of length 1.

Removal of Header.

Methodology:

Question-01:

Naive Bayes:

TF-IDF:

1. *Loaded documents from the folder 20_newsgroup.*
2. *Read files and split the data into train and test data using train_test_split using* **train_test_split** *function which takes all files ,corresponding class name and split ratio as input.*
3. *Preprocess the data using* **preprocess_data** *function and made the dictionary as* **train_X_dic** *containing train data only where it contains class name as key of outer dictionary and term as key of inner dictionary and storing term frequency as values.*
4. *Made another dictionary named* **total_words** *containg the vocabulary of a class where key is class name and value is total words in that class.*

5. *Dictionaries idf and tf-idf contains their respective values of terms which has nested dictionaries where outer dictionary key is the class name and inner dictionary key is a term.*
6. *Then tfidf is sorted in reverse order and some percentange of **k** is taken into consideration and those values of tfidf is updated into the train_X_dic.*
7. *Created another dictionary named* **unique_words** *that stores the unique words of entire corpus.*
8. *Prior stored in* **prior** *dictionary.*
9. *Then conditional probability of every word is computed after feature selection and stored in* **conditional_prob** *dictionary.*
10. *Test data is read from the corpus and for each term in document, if it is present in train data, it's conditional probability is fetched from the conditional_prob and prior dictionary.*
11. *On the basis of probability, a class with maximum value is predicted and added into a list named* **predicted**.
12. *At last, accuracy and confusion matrix is calculated between actual values and observed values via* **accuracy_metric** *and* ***confusion*** *function which takes test_Y and predicted lables as input.*
13. *User need to enter split ratio and percentage of k at run time.*

Mutual Information:

1. Same procedure from step 1 to step 4.
2. Created a nested dictionary class_terms containg outer key as class name, inner key as document id and value as another dictionary where term is a key and value as it's occurence.
3. MI nested dictionary is created which has outer key as class label, inner key as term and value has another dictionary having for 4 keys for each term i.e N11, N10, N01 and N00.
4. Created a I dictionary where mutual information computed values are stored in it. It is also a nested dictionary where outer dictionary key is a class name an inner dictionar key is a term.
5. Then I is sorted in reverse order and some percentage of **k** is taken into consideration and those values are updated into the train_X_dic.
6. Then Same procedure is followed from step 7 to step 13.

Question 02

KNN:

Mutual Information:

1. *Loaded documents from folder 20_newsgroup.*
2. *Read files and split the data into train and test data using train_test_split.*
3. *Preprocess the data and made the dictionary as* **train_X_dic** *containing train data only.*

4. *Made another dictionary named* **class_terms** *containg the*
5. *Made a dictionary for computing MI i.e mututal information storing each class data viz. class name, term and 4 parameters N11,N10,N01,N00 respectively.*
6. *A dictionary named* **I** *stores the value after computation of MI, then that dictionary is sorted in reverse order and top K values are selected according to the given percentage.*
7. *Those selected terms are updated in* **train_X_dic.**
8. *Created a document vector dictionary as* **doc_vector** *containing class labels, document id and mutual information value of terms in a document in respective clases.*
9. *Created another dictionary for* **mapping_doc** *purpose, having document_id as key and class_labels as values.*
10. *Then test documents are read and for all documents, document vector is generated.*
11. *Distance is computed between train and test docouments using* **compute_distance** *function and those distances are updated in* **dist_class** *dictionary having document id as key and computed distance as value.*
12. *Then top five maximum values are determined from the* **dist_class** *by weighing the votes of the* ***k*** *nearest neighbours on the basis of the computed_distance. And then we assign document to the class with highest score.*
13. *Those predicted values for the document and then they are stored in a list named* **predicted_1** *for 1KNN* **predicted_3** *for 3KNN and* **predicted_5** *for 5KNN.*
14. *At last accuracy and confusion matrix is calculated in* **accuracy_metric** *,* **confusion** *function which takes actual labels and predicted labels as input.*

TF-IDF:

1. Here, for computing tf_idf same procedure is followed as in Naive Bayes from step 1 to step 6, only difference is that in this tf_idf is of vocab length.
2. Again as mentioned above in KNN section for mutual information, same steps are followed after performing feature selection using tf_idf from step 8 to 14.