Documentation Q3

Problem Statement

In the given problem statement, we need to calculate the tf-icf score for each class in the dataset and then implement the hard-coded naive bayes model to check the accuracy of the model. Tf-icf is used for feature selection in naive bayes model. The given dataset consisted of files having various conversations and dialogues which needed to be imported and preprocessed before applying any computation. In the dataset, there are several folders but for our implementation, we required only 5 folders which will in the end be the labels for the training and testing set.

Importing and Preprocessing

The base code for both importing and preprocessing is the same as question 1. The only difference is that we have another dimension, folders/class, in our preprocessing and importing steps. The files have been imported using the standard file read operation. Each folder contains 1000 files which results in a total 5000 files in a nested dictionary of 5x1000. Every file in every folder is preprocessed using the following steps:

- First, the alphanumeric and underscore only words are filtered and they are then lowercase
- Stop words are removed
- Words are lemmatized
- We remove the underscore manually and recompute new words using the above preprocessing steps. We finally have all the alphanumeric words in each file for each folder.

TF-ICF

It is a modification of tf-idf scoring where term frequency is calculated with respect to each class and icf is the inverse class frequency which is similar to idf but only bounded to the specific class. We first calculate the unique words and bag of words for each class to make our computation easy. After that, we calculate the term frequency for each class and store it in a nested dictionary of class and words. Prior to these operations, we concatenate all the file words and create unique words for easy computation.

Term Frequency (TF): Number of occurrences of a term in all documents of a particular class
Class Frequency (CF): Number of classes in which that term occurs

ICF score is given by :

Inverse-Class Frequency : log(N / CF), where N represents the number of classes

TF-ICF:

TF-ICF(class, word) = TF(class, word)*ICF(word)

Dataset creation and Feature Selection

We split out data into a training and testing set using the inbuilt sklearn train_test_split module where we put a random parameter to shuffle our data. Using training data and tf_icf scores, we select top k features for the Naive Bayes model. Default k = 1500. We first pick each class and sort the tf-icf score in descending order and pick the first k words in the sorted order. We iterate over all the words and check if the word is there and if it is, we add it to our vocabulary list otherwise we skip them. This creates our feature list of 5Xk along with training and testing data.

Naive Bayes

There are 4 functions in the Naive Bayes class:

- **Fit** - It takes the training set, corresponding labels and features as an input. We first calculate the class probabilities using the frequency distribution of classes in the labels set and create a union of all the features and create a vocabulary. After that, we segregate the training set using the labels into 5 classes and send it to the next function, conditional along with the vocab list.

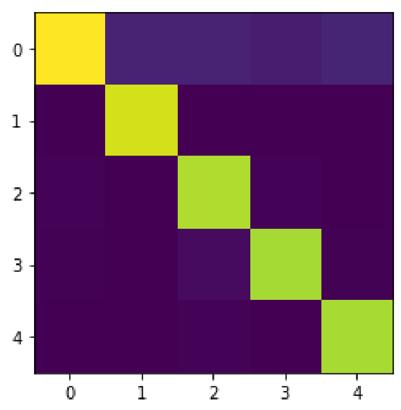
- **Conditional -** This function is used to calculate the conditional probability of each class for the given vocabulary. Here we calculate the frequency of each word which exists in the vocab and divide it by the total number of words in the vocabulary. We normalize it using the 1s scaling to prevent 0 probability cases.

- **Predict -** After fitting, we send our testing set and calculate the labels for the testing set. The conditional probability is stored in posteriors which is then finally used to predict the labels using np.argmax function.

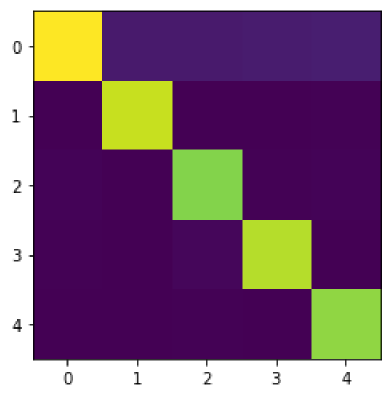
- **Accuracy -** The predicted labels from the predict function are sent to this along with the true labels and it calculates the accuracy of the prediction along with the confusion matrix using the inbuilt sklearn confusion matrix function.

Accuracy and Confusion Matrix

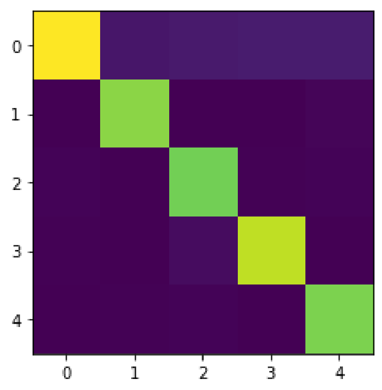
For 0.5 split
0.91



For 0.3 split
0.92



For 0.2 split
0.921


0
1
2
3
4
0
1
2
3
4