CSCI 455: Project 3

Brayden Faulkner and Christina Hinton

April 20, 2019

1 Introduction

This program reads in a file containing the frequency scores, names, and class of a set of documents and finds the centroid of each class. Then the program can use the centroids to classify other docs in the data set based on which centroid they are closest to.

2 Overview

The program first reads in the documents names, scores, and classes and stores them in a list of custom document objects call allDocs. The class and file names are stored as strings, while the scores are converted to a list of floats using a list comprehension. Then a new document object is created and appended to the allDocs list. Next the program takes in the name of the file containing the test cases. The test documents are read in using the file.read() method and then split into a list call testDocs using the string.split() method to split at every space in the string.

Next the program begins to separate each file that is not in the test case into their appropriate classes. It does this using a for loop that starts at 0 and ends when the index i equals the length of the array allDocs. Then it loops through the testDocs array and compares the name of the current file to each in the testDocs array to see if it is in the test case. If it is not in the test case it then loops through the list groupList, which is an array of Group objects that store the methods and members associated with classes, to check if one exist with the same class name as the current document. If one with the same class name does exist, it is added to that Group object using the addItem method. If a Group with the same class name does not exist, a new group is created and added to the groupList list.

Next the program loops through groupList and calls the getCentroid method, which calculates the centroid of each class, on each Group object. Then the program begins to classify all the test docs. Upon classifying the documents, the the program calculates the accuracy by comparing the program's classification to the documents' actual classifications, and then dividing the sum of correct classifications by the length of the document list.

For the 5-fold cross validation, the document list is randomized in order to ensure each

fold will contain a member of each class. For 5 iterations, the program selects a set of documents that is a fifth of the total size of the document list. This set becomes the test set for an iteration, with the rest being the training set. It then follows the same steps as above, calculating the accuracy of each iteration. Once the loop is complete, the program calculates the average accuracy.

To determine if more prominent words attributed to greater accuracy, we created a few different files containing the TF-IDF vectors for each document. This included files using the top 1000 words, top 500 words, and bottom 1000 words. We determined that using more prominent words was more useful, as they consistently gave us an accuracy of about 50 percent, where using less prominent words lowered the accuracy dramatically.

3 Conclusion

4 Results

References