CSCI 455: Project 3

Brayden Faulkne and Christina Hintonr

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.

3 Conclusion

Earlier versions of this program took as long as two-three hours to run. This was because a lot of the code was done using classes and contained many functions that involved reading over the same list multiple times, greatly increasing the run time of the program. After modifying the base code using functions and libraries I found, namely the NLTK library. We also discovered that operating over a list of all the words, rather than just the words in each file, at once saved a significant amount of time.

4 Results

Text files containing all the, the 200 least used, and the 200 most used, words can be found elsewhere in the directory. There were 51123 unique words were identified after stop word removal and stemming. The 200 most common words consist mainly of used in computer science, academia, and everyday talk, such as days of the week. Some of these words were stemmed in strange manners, as is expected of a Porter stemmer, but other than that it seems all the words were processed properly. The least used words consist of typos, names, and terms that have no general meaning, such as class numbers or room names. There are some words that we believe that may have been the result of an error in processing, but we were unable to identify or recreate the circumstances that caused them, so this might not be the case.

References