CSCI 455: Project 3

Brayden Faulkner and Christina Hinton

April 21, 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 using the classify function. 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.

3 5-Folds Cross Validation

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.

4 Modding the classifier

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.

5 Classify function

The classify function is used to classify documents based on which centroid they are closest to. It takes a document object named doc and a list of group objects named groupList as parameters. It then loops through each object in groupList and determines it distance from doc using Euclidean distance, and stores each's distance in a list called distances. Then the program loops back through each item in distance in order to see which is the shortest distance from the doc.

6 Document class

The Document class is used to store necessary information about documents in the dataset. It has three members name, scores, and className. Name is a string that is used to store the name of each document. Scores is a list of floats that stores the tf-idf score of each term we are using to classify the dataset in the document. Lastly the className members is a string that stores the name of which class/group the document is in.

7 Group class

The Group class is used to store/act over the classes/groups that the documents in our dataset are divided into. It is made up of five members and three methods, including the constructor.

The docList member is a list of Document objects that is used to store each in document in a given class. The centroid member is a list of floats that is used to store the centroid of a given class. The count member is an int that stores the number of documents in a given class. The totalScores members is an array of floats that is used to store the sum of the scores from each document in a class in order to make calculating the centroid easier. Lastly the className member is a string that stores the name of a given class.

The first method is the constructor, which takes the className as the parameter. Next is the addItem method which is used to add new documents, which it takes as a parameter, to the class. First the method appends the actual document object to the docList object. Then the count int is increased by one. Then it loops through the totalScore list and add the new documents score to it. The last method is the getCentroid method, which is used to calculate the centroid of each class. It loops through each item in centroid and sets it equal to the corresponding item in the totalScores list divided by the total number of elements in the group.

8 Results

We used two different forms of accuracy testing. The first test case puts every tenth file in the test set. The second test case is the 5-fold cross validation mentioned earlier in the report. The average accuracy for both these was around 50 percent. Only using the top 500 terms had little to no affect on the accuracy, while only using the bottom 500 terms greatly reduced the accuracy. The classifier seemed to be more likely to put things into the student, department, and course classes.

9 Conclusion

We had originally expected greater accuracy from our classifier. Considering its simplicity the classifier actually worked quite well, but considering that around forty-five percent of documents belong to the other class, it would almost be as accurate to just assume that all documents are a part of the is class.

References