GE 103 – Introduction to Computing and Data Structures

Dr. Abhinav Dhall

Submission deadline – 18/04/2018

Assignment 4

INSTRUCTIONS TO SUBMIT CODE

1.Save the questions in separate C files with the names YourRollNumber_Q1.c YourRollNumber _Q2.c eg: 2010csb1001_Q1.c

If the filename nomenclature is not followed, the respective program will not be evaluated.

2. On Moodle upload the material as a single zip file.
3. Create Readme file which contains instructions how to run the program.
4. Report which elaborates the logic of code for the questions.
5. No plagiarism
6. Marks distribution for the questions is for: 3 marks for the correct execution of the program, 1 mark for the logic and 1 mark for documentation, code indentation and nomenclature.
7. No extension will be granted.

For completing the assignment and all the functions extra marks = 5 will be awarded.

The task is to write a program, which can predict if a SMS is spam or not.

SMS is a series of strings.

The data is available at http://www.esp.uem.es/jmgomez/smsspamcorpus/SMSSpamCorpus01.zip

Download the zip file and have a thorough look at the readme.

The file "English.txt" should be used in the experiment.

There are 1002 legitimate messages and a total of 82 spam messages. Each line is a SMS.

Keep the first 50% of both types for Train and and the rest for Test.

Functions to Implement (Note the name of the functions has to be the same as follows)-

ReturnDistance(Char, Char*);* // Here parameters are the two words in which the distance needs to be calculated. Read about Levenshtein Distance and implement it (1 mark)

*ComputeKMeans(Char *[], Number of modes);* // Parameters are the strings and the number of cluster centres. The function returns the cluster strings, let us call them the representative strings. The function calls *ReturnDistance ()* to compute the distance/similarity between two words/strings. Read about KMeans algorithm. Display the representative strings on the screen.

ComputeHistogram(char, char*);* //First parameter are the words of a SMS. Second parameters are the words we computed in *ComputeKMeans().* The function ComputeHistogram() returns an integer array, which is a histogram. Each index in the array contains the number of times the word representative strings occur in the SMS.

ComputeL1distance(int [], int []); // The function compares the arrays created using *ComputeHistogram()* and return the absolute difference.

In the main process the distance of a Test SMS with all the Train SMS's histogram array. Chose the smallest difference and assign the label of that Train SMS to the Test SMS. Perform this for all the SMS.

In the end display the number of Test SMS, which were correctly identified as spam or not based on their original label.