ASSIGNMENT 6

Assignment 6 is described in the next few pages. Below are the instructions for this assignment:

- There are three parts to this assignment – Parts I is worth 15 points, part II is worth 35 points, and part III is worth 10 points.

- For parts I and II, you can use any of the following languages – Java, Python, C#, Ruby. You can use external libraries for loading and parsing the data, but you cannot use any machine learning libraries or packages. Please mention what libraries you used and how to compile your code in the README file.

- For part III, you are free to use any language, external package or library. You are free to use OpenCV or any other image processing packages in R or Python.

- Requirements of what to turn in are clearly mentioned in each part. **Keep them in separate folders titled PartI, PartII, and Part III.**

- The TA should be able to run your code from command line as explained in each of the parts.

- Please ask questions only through Piazza, so others with the same doubts can benefit.

Part I (15 points)

In the first part of this assignment, you have to implement the k-means algorithm using **Euclidean distance** on a dataset with two attributes. The dataset is available for download at: http://www.utdallas.edu/~axn112530/cs6375/unsupervised/test_data.txt

The algorithm was discussed in class and you can write code in any language, **but it should be runnable from the command line**. The parameter to the program should be the number of clusters (k), input file location, and output file location. The TA should be able to run your code as follows:

k-means <numberOfClusters> <input-file-name> <output-file-name>

Initialization:

Based on the input parameter(k), you should randomly select k points as means (centroids).

Termination Condition:

The usual termination condition in k-means is when the centroids no longer more. In this assignment, you should also limit your update step to a maximum of 25 iterations.

Input:

The input file will be specified by the parameter <input-file-name>.

Validation:

The usual method of evaluating the goodness of clustering will be used. It is the Sum of Squared Error (SSE) function, defined as:

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(m_i, x)$$

where m_i is the centroid of the i^{th} cluster.

You should write a method/function to compute the SSE of clustering i.e. it should be separate from main method. This value should be written in the output file.

Output:

Your code should also output to a file called specified by the parameter <output-file-name> and should be in the format:

<cluster-id> <List of points ids separated by comma>

For example,
1 2, 4, 7, 10

At the end of the above file, you should output the value of SSE.

How many times to run your code:

You should run your code at least 5 times with different values of k and include the details of SSE in your report.

What to Turn In for Part I :
(1) The source code in any of the following languages – Java, Python, C#, Ruby
(2) README file indicating how to build and compile your code.
(3) A report file containing details of the 5 runs and their SSE values.

Part II (35 points)

Tweets Clustering using k-means

Twitter provides a service for posting short messages. In practice, many of the tweets are very similar to each other and can be clustered together. By clustering similar tweets together, we can generate a more concise and organized representation of the raw tweets, which will be very useful for many Twitter-based applications (e.g., truth discovery, trend analysis, search ranking, etc.)

In this assignment, you will learn how to cluster tweets by utilizing Jaccard Distance metric and K-means clustering algorithm.

Objectives:

- Compute the similarity between tweets using the Jaccard Distance metric.
- Cluster tweets using the K-means clustering algorithm.

Introduction to Jaccard Distance:
The Jaccard distance, which measures dissimilarity between two sample sets (A and B). It is defined as the difference of the sizes of the union and the intersection of two sets divided by the size of the union of the sets.

$$Dist(A,B) = 1 - \frac{|A \bigcap B|}{|A \bigcup B|} = \frac{|A \bigcup B| - |A \bigcap B|}{|A \bigcup B|}$$

For example, consider the following tweets:

Tweet A: the long march

Tweet B: ides of march

$|A \cap B| = 1$ and $|A \cup B| = 5$, therefore the distance is $1 - (1/5)$

In this assignment, a tweet can be considered as an unordered set of words such as {a,b,c}. By "unordered", we mean that {a,b,c}={b,a,c}={a,c,b}=...
A Jaccard Distance Dist(A, B) between tweet A and B has the following properties:

- It is small if tweet A and B are similar.
- It is large if they are not similar.
- It is 0 if they are the same.
- It is 1 if they are completely different (i.e., no overlapping words).

Here is the reference for more details about Jaccard Distance:
http://en.wikipedia.org/wiki/Jaccard_index

Hint: Note that the tweets do not have the numerical coordinates in Euclidean space, you might want to think of a sensible way to compute the "centroid" of a tweet cluster. *This could be the tweet having minimum distance to all of the other tweets in a cluster.*

Exercise:

Implement the tweet clustering function using the Jaccard Distance metric and K-means clustering algorithm to cluster redundant/repeated tweets into the same cluster. You are free to use any language or package, but it should be clearly mentioned in your report.

Note that while the K-means algorithm is proved to converge, the algorithm is sensitive to the k initial selected cluster centroids (i.e., seeds) and the clustering result is not necessarily optimal on a random selection of seeds. In this assignment, we provide you with a list of K initial centroids that have been tested to generate good results.

Inputs to your K-means Algorithm:

(1) The number of clusters K (default to K=25).

(2) A real world dataset sampled from Twitter during the Boston Marathon Bombing event in April 2013 that contains 251 tweets. The tweet dataset is in JSON format and can be downloaded from http://www.utdallas.edu/~axn112530/cs6375/unsupervised/Tweets.json

(3) The list of initial centroids can be downloaded from:
http://www.utdallas.edu/~axn112530/cs6375/unsupervised/InitialSeeds.txt

Note that each element in this list is the tweet ID (i.e., the id field in JSON format) of the tweet in the dataset.

How to run your code:

The TA should be able to run your code as follows:

tweets-k-means <numberOfClusters> <initialSeedsFile> <TweetsDataFile> <outputFile>

<numberOfClusters> is the number of Clusters and
<initialSeedsFile> is a text file containing value of the initial seed data.
<TweetsDataFile> is the data file containing Tweets in JSON format
<outputFile> is the output file explained below.

The default value for <numberOfClusters> should be 25 and for the <initialSeeds> to be the file provided to you containing number of seeds.

Output:

Your code should also output to a file called tweets-k-means-output.txt. It should contain following:

<cluster-id> <List of tweet ids separated by comma>

For example,
1 323906397735641088, 900906397735641088, ..

It should also contain the SSE value.

Validation:

The same method as described in part I will be used. The distance metric will be Jaccard.

What to Turn In for Part II :
(1) A result file that contains the clustering results. Each line represents a cluster. It is in the form of *cluster_id: a list of tweet IDs that belongs to this cluster*
(2) The source code in any of the following languages - Java, Python, C#, Ruby. Be sure to include a README file indicating how to build and compile your code.
(3) A report file that contains details on how to compile your code, and any other relevant details.
* Idea and data courtesy of Dong Wang, University of Notre Dame *

Part III (10 points)

In this part, you will use unsupervised learning to reduce the number of color points in an image. This application is also referred to as *image segmentation* or *color quantization*. You are free to use either k-means or EM algorithm for the unsupervised learning.

Below are some resources for learning more about image segmentation:

- http://www.ics.uci.edu/~dramanan/teaching/ics273a_winter08/projects/avim_report.pdf
- http://docs.opencv.org/3.1.0/d1/d5c/tutorial_py_kmeans_opencv.html#gsc.tab=0
- http://www.pyimagesearch.com/2014/07/07/color-quantization-opencv-using-k-means-clustering/
- http://cs.gmu.edu/~kosecka/cs682/lect-segmentation-part1.pdf
- https://courses.cs.washington.edu/courses/cse576/book/ch10.pdf (If you have lots of time)

You are free to use any language or package. Some of the popular ones are:
- OpenCV : (Python or Java)
- R package ggplot2 and jpeg

Input:

You are given a set of 5 images that you can download from:

http://www.utdallas.edu/~axn112530/cs6375/unsupervised/images

You can select any 3 out of these and run the image segmentation algorithm on them.

Parameters:

It is up to you to select the best value of the number of clusters and any other parameters for the algorithm.

Output:

The output should be a set of clustered images. You should create a folder called "clusteredImages" and place the images there.

How to run:

The TA should be able to run your code from the command line. Please include instructions in the report file.

Report File:

Please include a report file indicating which language/tools/packages you used, which parameters you used, and how to run your code.

What to Turn In for Part III:

(1) Source code in any language of your choice. Be sure to include a README file indicating how to build and compile your code.
(2) 3 Clustered images as output (in a separate folder)
(3) Report containing details mentioned above.