REPORT : IRHW3

Rajashree Rao Polsani

UIN : 223001584

This folder contains two python files

1. tweetcollector.py
   This file generates 32 files one for each query containing 50 tweets each. If a new query has to be entered then one can change the query term manually in the api search function. (The file names are same as that of the query and are in part1tweets folder)

2. cluster.py
   This file generates clusters by using k means algorithms (while running this code 32 files are generated one for each query-the file names are numbers). The program may take upto 5-6 min to run (the best clustering results are presented after reseeding several times). Finally a graph is presented between RSS and k values.

   The results are as follows

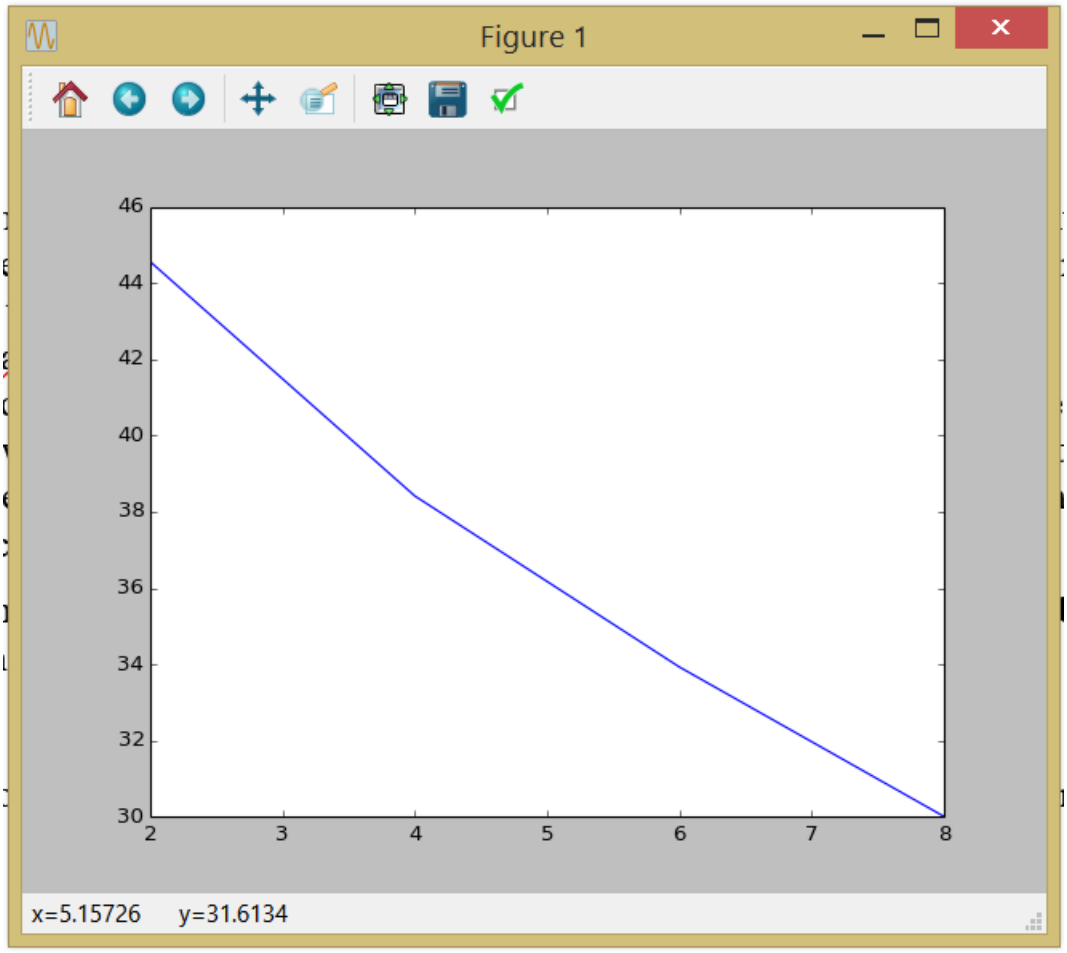
2clusters
RSS value = 44.5587483499

4clusters
RSS value = 38.4170460961
purity = 0.78125
this dictionary includes no.of of documents in each cluster and how they are distributed within the original classes
{0: [3, 8, 2, 1], 1: [1, 0, 6, 0], 2: [0, 0, 0, 7], 3: [4, 0, 0, 0]}

6clusters
RSS value = 33.9324279733

8clusters
RSS value = 30.0141003572

Graph between RSS and number of clusters(k)




For k-means clustering I have used cosine similarity for finding out which centroid does the document vector belong to. This is due to the fact that sometimes even though the documents are similar (ie the vectors are in the same direction) the euclidean distance between their endpoints is high. And also purity values are good when cosine similarity is considered (all the points are being put into the same cluster when k-means based on Euclidean model was implemented). New centroids are generated by taking the average of normalized unit vectors of the document. While calculating RSS unit vectors of each document are considered.

The spherical k-means (k-means when we use cosine similarity) almost resembles the k-means using Euclidean distance.

$(a-b)^2 = a^2 + b^2 - 2.a.b = 2*(1-\cos(a,b))$

The above formula explains the similarity between spherical and actual k-means.