**Detecting Tight Communities in Twitter**

**1. Group members:** Aparna Pavithran (axp161730)

Chandini Shankar (cxs152830)

Mahesh Paramari (mxp150830)

Manasa Rao Kondrolla (mxk150930)

**2. Dataset Details:**

This dataset consists of 'circles' (or 'lists') from Twitter. Twitter data was crawled from public sources. The dataset includes node features (profiles), circles, and ego networks.

Dataset contains .txt file with node id and follower id of twitter users.

Twitter data contains from 1,000 ego-networks, consisting of 4,869 circles and 81,362 users. The ego-networks obtained in size as 81306 nodes.

**Note:** Since we are getting clusters directly from connected components, no need of features.

**Dataset statistics**

|  |  |
| --- | --- |
| Nodes | 81306 |
| Edges | 1768149 |
| Nodes in largest WCC | 81306 (1.000) |
| Edges in largest WCC | 1768149 (1.000) |
| Nodes in largest SCC | 68413 (0.841) |
| Edges in largest SCC | 1685163 (0.953) |
| Average clustering coefficient | 0.5653 |
| Number of triangles | 13082506 |
| Fraction of closed triangles | 0.06415 |
| Diameter (longest shortest path) | 7 |
| 90-percentile effective diameter | 4.5 |

**3. Techniques planned to use:**

**Step 1:** Create the graph from the twitter dataset.

**Step 2:** Find out the connected components for the whole graph.

**Step 3:** Find out the clusters in the graph.

**Step 4:** Find the strongly connected components in the clusters.

**Step 5:** Calculate the tightness of the cluster based on a threshold k.

**4. Experimental methodology:**

Construct the graph from the twitter dataset. Find out the connected components for whole graph. Connected components will output the smallest vertex in each connected component. Find out distinct vertices given in output of connected components to obtain the isolated number of clusters in the graph.

For each isolated clusters find out the number of strongly connected components. Then in the same way as connected components calculate the strongly connected components returned by it. Iterate through edges and check more than one component attached to it or not.

Calculate the number of vertices attached to strongly connected components and then filter it to give a cluster list with a threshold value k. If the strongly connected component satisfies the threshold value then consider it else filter it out.

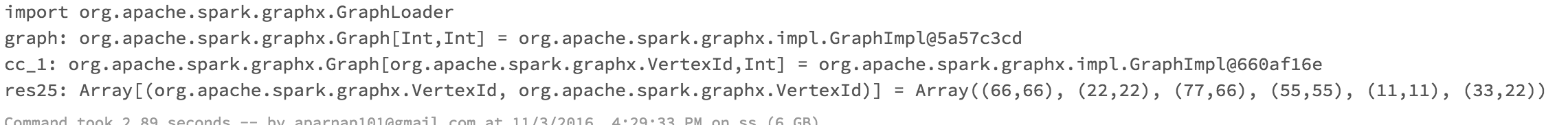
Find out the tightness in the particular cluster based on strongly connected components in it and then find the percentage of tightness for each cluster. That is a cluster has 100 percentage of tightness or density of the cluster if all of its nodes are there in the strongly connected component. Density of the cluster is the number of edges in the strongly connected component divided by the total number of edges in the cluster.

**5. Coding Language / Techniques to be used:**

Scala, Graphx

**6. Preliminary Results:**

**Note**:Tested with a sample input of data.

****

**Code:**

import org.apache.spark.graphx.GraphLoader

val graph = GraphLoader.edgeListFile(sc, "/FileStore/tables/t45uq2op1478208616424/sample.txt")

val cc\_1 = graph.stronglyConnectedComponents(2)

val scc\_1 = graph.connectedComponents()

graph.edges.count()

cc\_1.edges.count()

scc\_1.edges.count()

graph.vertices.count()

cc\_1.vertices.take(10)

scc\_1.vertices.take(10)