**Detecting Tight Communities in Twitter**

**By**:

Aparna Pavithran (axp161730)

Chandni Shankar (cxs152830)

Mahesh Paramati (mxp150830)

Manasa Rao Kondrolla (mxk150930)

**INDEX**

[1 Introduction and Problem Description](#_8hrdzlbzvjv8) [3](#_8hrdzlbzvjv8)

[2 Dataset Description](#_8hrdzlbzvjv8) [3](#_8hrdzlbzvjv8)

[3 Related Work](#_c6ds2begzatr) [4](#_c6ds2begzatr)

[4 Proposed Solution](#_p7ddaa8o04tq) [4](#_p7ddaa8o04tq)

[5 Experiment and Results](#_bjcgeuky44p2) [6](#_bjcgeuky44p2)

[6 Future Work](#_vf17aon89ns6) [8](#_vf17aon89ns6)

[7 Conclusions](#_xo0i1iqacwnw) [8](#_xo0i1iqacwnw)

[8 References](#_wb75uhyosbo4)9

**1 Introduction and Problem Description**

Social network has increasingly become a commodity and a necessity for people around the world on one hand and on the other, catering to customer specific interests have become a business. This gives rise to a need to understand the people’s activities, their associations and their interest groups which would further help businesses like Pinterest, Twitter, Google, Facebook to customize the web pages specific to each individual’s liking. Detecting tight communities have become important to understand each person’s engagements and suggest helpful customizations that would interest the user and increase involvement in using the app.

This project aims at detecting tight communities in Twitter social network. The Twitter data contains 1000 ego-networks consisting of 4869 circles and 81362 users. The people in the Twitter social network are represented as nodes and the connection between two people are represented as edges in the graph that represents the entire Twitter data.

# **2 Dataset Description**

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.

|  |  |
| --- | --- |
| 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 |

**Figure 1: Dataset description**

Twitter data is obtained from 1,000 ego-networks, consisting of 4,869 circles and 81,362 users. The ego-networks have 81306 nodes and the directed edges indicate person a follows person b. The set of circles for each node is all the groups the user is a part of and each circle is a set of nodes indicating all the users of that group and their connection. These circles were obtained by using the features mentioned in the feature name file.

We used the file that contained all the edges i.e. all the connections between every user in the Twitter data to construct a graph of the social network with each user as a node and the connection of that user with other users as an edge in the graph.

# **3 Related Work**

Currently, we can identify circles in twitter manually for each user. This is not advisable because it’s not automatic, time consuming and error pruning. If we find out the circles manually for each user, it is practically impossible to find out a strongly connected circle or communities in the network manually.

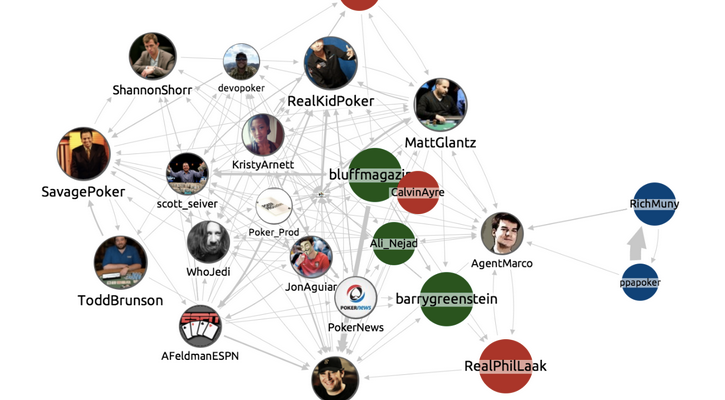
Twitter data is little more complex as it is not working in the pattern of Facebook or Google+ data. In Twitter when user1 follows another user2, it is a directed edge from user1 to user2. Hence in order to find out a strongly coupled circle in Twitter network, we have to consider a directed graph of users. This makes the graph more complex.

Tightly connected component in a directed graph of data is defined as when user1 follows user2 and user2 follows user1, user1 and user2 does not follow any other users. Then the circle including user1 and user2 is a strongly connected component. If we find out a strongly connected component, it will be helpful to find out terrorist group or any other unusual activists group in any social network. As described earlier, finding a strongly connected component manually in any social media with millions of dataset is nearly impossible.

# **4 Proposed Solution**

We are proposing a solution for the above problem by using GraphX and Scala. Given a dataset of millions of users, we can find out how many connected components are there in the given dataset. Also find all the tightly connected components in the dataset. All these tasks can be done automatically. Manually it might take hours and hours to do the same thing. But in our proposed solution, it might take few minutes to complete the entire complex process.

Dataset needs to be in the above-described format. It is allowed to load dataset in .txt format. Basically this file contains edges of the graph. That is user1 user2 in the dataset means user1 follows user2. Millions of users data is taken in this fashion and represented as a graph. From the graph, find out connected components and strongly connected components.



**Figure 2: Example twitter dataset**

**Methodology:**

Construct the graph from the twitter dataset. Find out the connected components for whole graph. By finding out the connected components in the graph, we can identify the separate and isolated clusters of users in the 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.

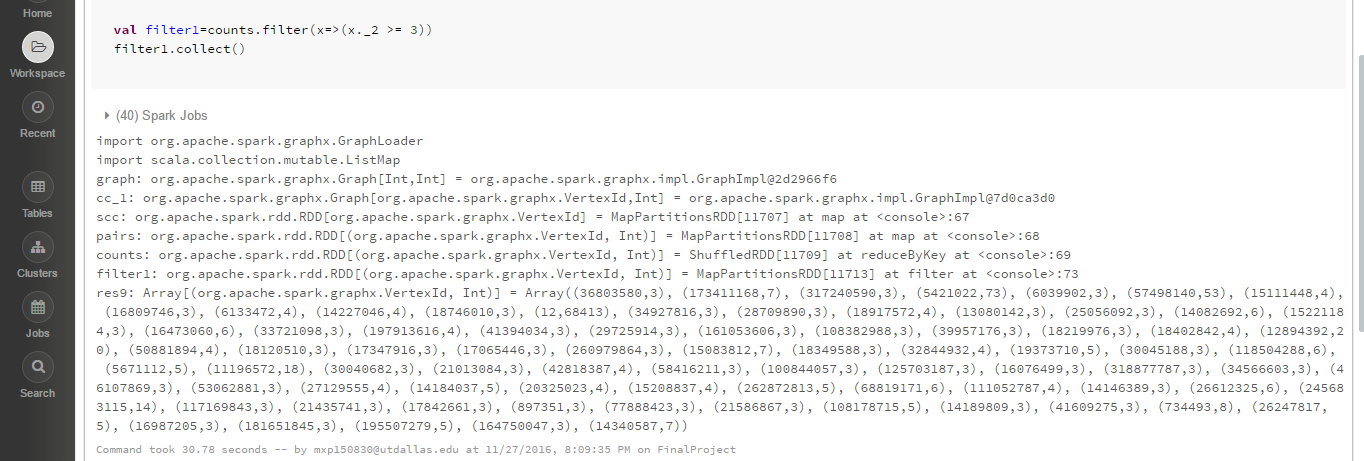
For each isolated cluster find out the number of strongly connected components. Then in the same way as connected components calculate the strongly connected components returned by it. Iterate through edges and check more than one component attached to it or not.

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. We have given threshold value as 3 for our experiment.

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.

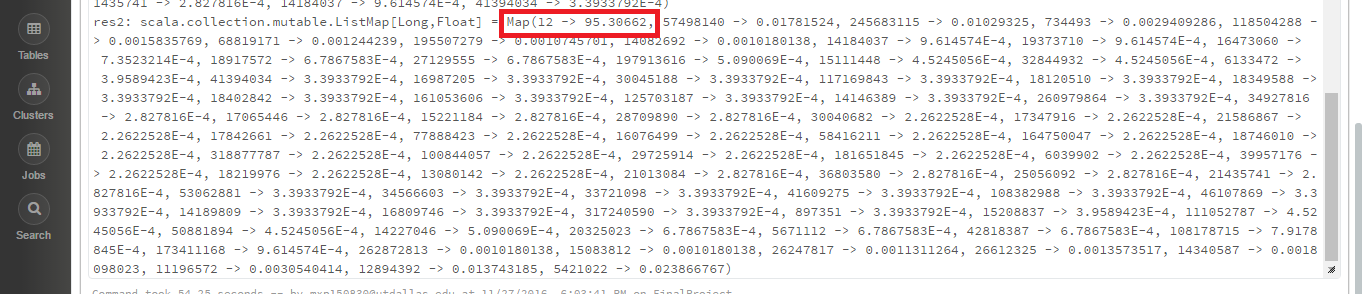
# **5 Experiment and Results**

The experiment complies with the methodology proposed. It is carried out using the Databricks framework wherein the data is first loaded. The total number of vertices in the graph is 81306 and the total number of edges in the graph is 1768149. Finding out the connected components in the graph follows this. The number of clusters in the graph is found using the previously computed connected components which is equal to “1”. Hence there is one connected component in the graph. We have then found the number of strongly connected components which is equal to “12248”. The number of strongly connected components with a threshold of k=3 thus obtained is “80” which are represented below.



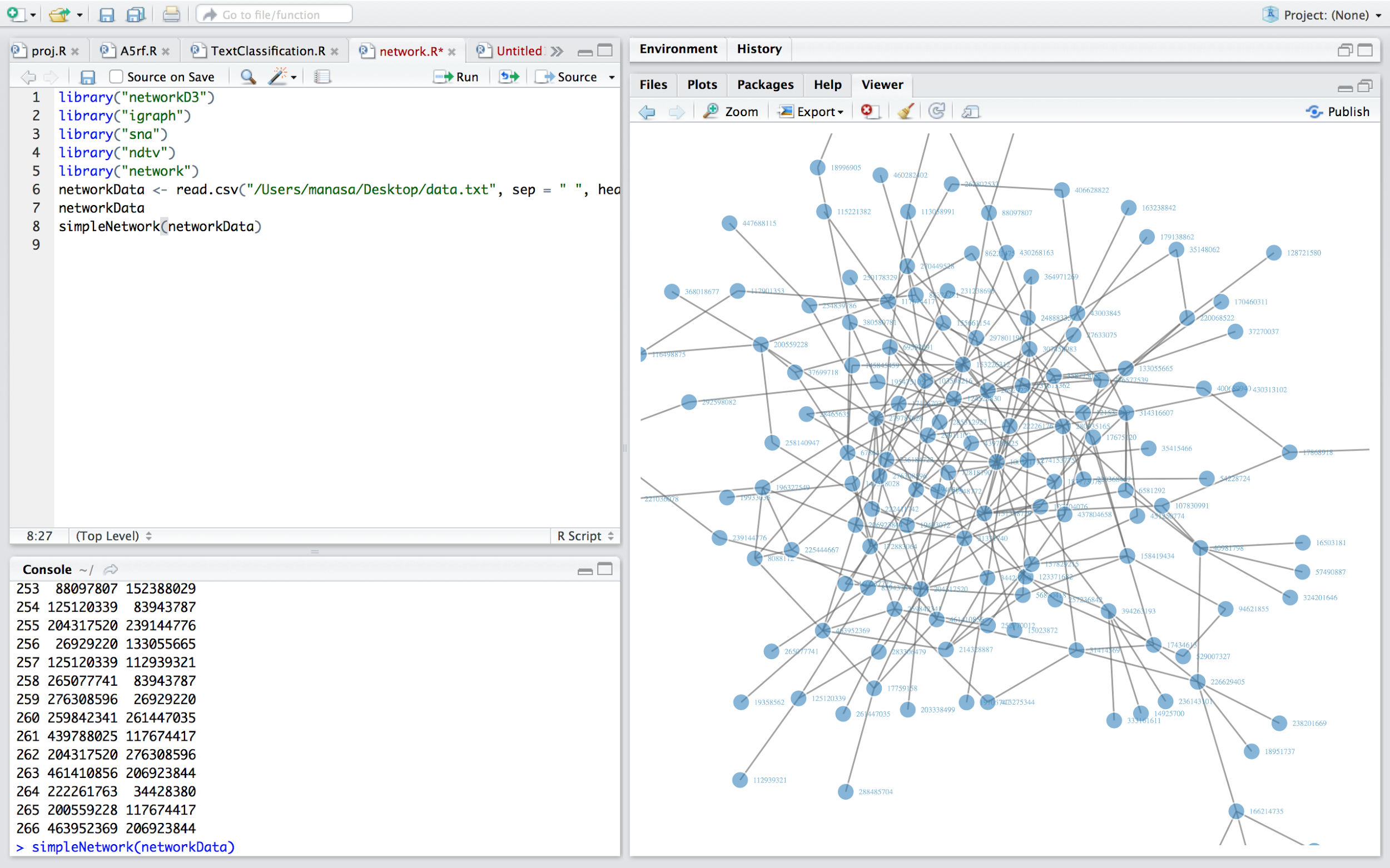
**Figure 3: Strongly connected components in Twitter Dataset**

The sub graphs of these 80 strongly connected components were constructed. The tightness for each cluster is calculated and the results are computed in descending order of the tightness. It is observed that the tightness of the largest cluster is 95.30%. The results are as follows.



**Figure 4: Tightness percentage final result**

We have also created a visualization of the entire graph using RStudio. The dataset has two columns. The D3 network graph is constructed by using a function called “simpleNetwork” which is included in the package “networkD3”. The first two columns of the dataset are the names of the linked units. A part of the graph is shown below.



**Figure 5: Dataset Graph representation in RStudio**

# **6 Future Work**

Though our system provides an easier solution to find the degree of tightness in a community, we cannot be sure that this is a centralized solution to find tight communities for all online social networks. We intend to find a more general approach that would target all networks and provide an easier way of amending the graph instead of reconstructing the graph when any changes occur to the network.

**7 Conclusions**

We have presented a system to identify tight communities in a social network. Using our solution, we can find completely tight communities i.e. isolated communities and degrees of tightness in a well-connected community. Our robust solution doesn't require complex and time consuming efforts of clustering to identify the community the user is a part of to evaluate the tightness of the community. On the contrary, our approach is adaptive and requires only the connection between users, which is used by the model to find the strongly connected components in the graph. Finally, the accuracy and efficiency of our approach was evaluated against the Balanced Error Rates of stochastic models in the original paper.

# 

# **8 References**

1. <https://arxiv.org/abs/1210.3266>
2. <http://www.cise.ufl.edu/~mythai/files/socialcom11.pdf>
3. <http://spark.apache.org/docs/latest/quick-start.html>
4. <http://spark.apache.org/graphx/>
5. <http://i.stanford.edu/~julian/pdfs/nips2012.pdf>
6. J. McAuley and J. Leskovec. Learning to Discover Social Circles in Ego Networks. NIPS, 2012.
7. T. Yoshida. Toward finding hidden communities based on user profiles. In ICDM Workshops, 2010.
8. M. Newman. Modularity and community structure in networks. PNAS, 2006.
9. A. Mislove, B. Viswanath, K. Gummadi, and P. Druschel. You are who you know: Inferring user profiles in online social networks. In WSDM, 2010
10. P. Krivitsky, M. Handcock, A. Raftery, and P. Hoff. Representing degree distributions, clustering, and homophily in social networks with latent cluster random effects models. Social Networks, 2009.