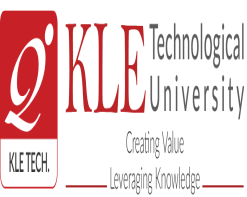
KLE Society's

KLE Technological University



**Social Network Analysis**

**Project Report**

On

*submitted in partial fulfillment of the requirement for the degree of*

**Bachelor of Engineering**

**In**

**Computer Science and Engineering**

Submitted By

Suyog Satish Bhat 01FE17BCS225

Ganesh Hulyal 01FE18BCS430

Tanaya N Nagekar 01FE17BCS228

Saketh V.K 01FE17BCS237

**Faculty In-charge:**

Prof. Lalita

SCHOOL OF COMPUTER SCIENCE & ENGINEERING

# Problem Statement:

To analyse and represent twitter data using social network analysis tool-Gephi.

# Goals

1. To learn how to use Gephi
2. To represent data using Gephi

# Specifications

## Gephi Installation

## Gephi Project Setup

# Comparison

# Milestones

## Twitter Data Fetch

## Data Import and Graph Representation

# 

# Gephi Project Setup

**To import dataset from a csv file:**

* Start with the “*File*” tab and click “*New Workspace*” .
* In the “*Data Lab*” click on “*nodes*” and then on “*Import Spreadsheet*” and make sure that “*Nodes Table*” is clicked on the popup window (and the file is a csv file).
* Import the file.
* Next, do the same for the “*edges*” and be sure that the “*Edges table*” is clicked in the popup window.
* Go back to the overview window to draw the figure. Click the arrow at the bottom right to see the various sizing options.
* To color nodes: click on the “*Nodes*” tab in the left, and click on the “*Partition*” button above it. “*Refresh*” it with the green and then pick a category. You can reset the colors by clicking on the color boxes. Then hit “*Apply*”.
* You can rule various “*Layout*” algorithms and hit “*Stop*” - by playing with the “*Optimal* *Distance*” and “*Repulsion*” etc. settings, you can get the picture to fit better.
* You can “*size*” nodes by left clicking size and then resetting the number and then double-clicking the button again.

# Comparison

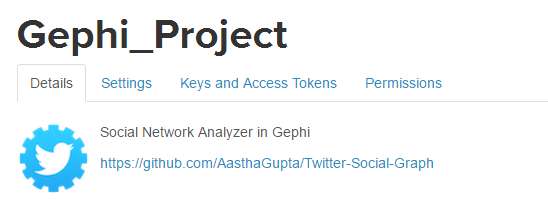
Comparison with NodeXL

* Gephi can handle large network very easily. Whereas NodeXL is good for small network.
* Gephi is free and open source whereas NodeXL is not.
* NodeXL supports fetching dataset from different social media platform like twitter or Facebook.
* Gephi is a standalone software application while NodeXL comes as a microsoft excel plugin.

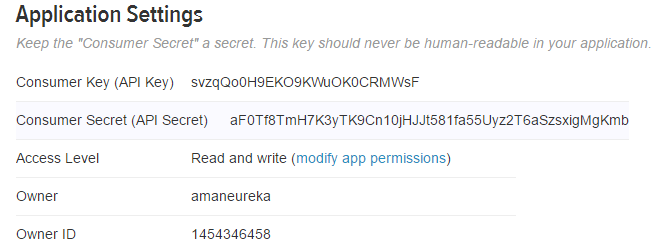
# Twitter Data Fetch Twitter REST API The REST APIs provides programmatic access to read and write Twitter Data. The REST API identifies Twitter applications and users using [OAuth](https://dev.twitter.com/oauth); responses are available in JSON.

**Creating a New Application**

* Visit Twitter developers dashboard <https://apps.twitter.com/app/new>
* Fill appropriate details in form.
* Accept Agreement and Click on “*Create your Twitter Application*”.



* Click “*Keys and Access Tokens*” Tab.
* Generate “*access token*”.



**Fetching Data**

We are using tweepy python module to connect with Twitter OAuth.

* pip install tweepy

**Python Code**

import time

import tweepy

from sets import Set

import os.path

limit\_fetch = 100

seed\_profile = ''

consumer\_key = ''

consumer\_secret = ''

access\_token = ''

access\_token\_secret = ''

auth = tweepy.OAuthHandler(consumer\_key, consumer\_secret)

auth.set\_access\_token(access\_token, access\_token\_secret)

api = tweepy.API(auth)

counter = 0

profiles = [seed\_profile]

completed\_list = Set()

while counter < limit\_fetch and counter < len(profiles):

current\_user = profiles[counter]

output\_file = 'Datasets/{}.dataset'.format(current\_user)

completed\_list.add(current\_user)

followers = tweepy.Cursor(api.followers, screen\_name=current\_user).items()

print "fetching followers of @{}".format(current\_user)

if not os.path.exists(output\_file):

with open(output\_file, "a") as file:

while True:

try:

user = next(followers)

except tweepy.TweepError:

time.sleep(60\*15)

user = next(followers)

except StopIteration:

break

file.write("{}\n".format(user.screen\_name))

if (user.screen\_name not in completed\_list):

profiles.append(user.screen\_name)

counter = counter + 1

**Getting Data Ready for Import**

Creating data file tha can be imported in Gephi.

**Python Code**

import os

import glob

from sets import Set

def create\_node\_and\_edge\_list(inputfiles):

nodes = Set()

edges = []

for filename in glob.glob(inputfiles):

node\_a = os.path.splitext(os.path.basename(filename))[0]

nodes.add(node\_a)

if len(nodes) > 3000:

break

with open(filename, 'r') as file:

for line in file:

node\_b = line[:-1]

nodes.add(node\_b)

edges.append([node\_a, node\_b])

print 'dumping nodes...' + str(len(nodes))

with open('Datasets/nodes.txt', 'w') as file:

file.write('Id;Label\n')

id=1

for node in nodes:

file.write('{};{}\n'.format(id, node))

id=id+1

print 'dumping edges...' + str(len(edges))

with open('Datasets/edges.txt', 'w') as file:

file.write('Source;Target\n')

for edge in edges:

file.write('{};{}\n'.format(edge[0], edge[1]))

create\_node\_and\_edge\_list("Datasets/\*.dataset”)

**Calculating Edge Weight Matrix**

Calculating normalised weights for the edges using their followers count.

**Python Code**

import os

import glob

def create\_weight\_list(inputfiles):

output\_file=open("Datasets/weights.txt","w")

output\_file.write('Source;Target;Weight\n')

followers = {}

for filename in glob.glob(inputfiles):

user = os.path.splitext(os.path.basename(filename))[0]

count=0

with open(filename) as f:

count = sum(1 for line in f)

followers[user]=count

with open("Datasets/edges.txt") as edges:

for edge in edges:

user1=edge.split(";")[0]

user2=edge.split(";")[1][:-1]

if user1 not in followers :

continue

f1=followers.get(user1, 0)

f2=followers.get(user2, 0)

delta=abs(f1-f2)

w=100.00/(1+delta)

w=w\*(1+min(f1,f2))

output\_file.write('{};{}\n'.format(edge[:-1], w))

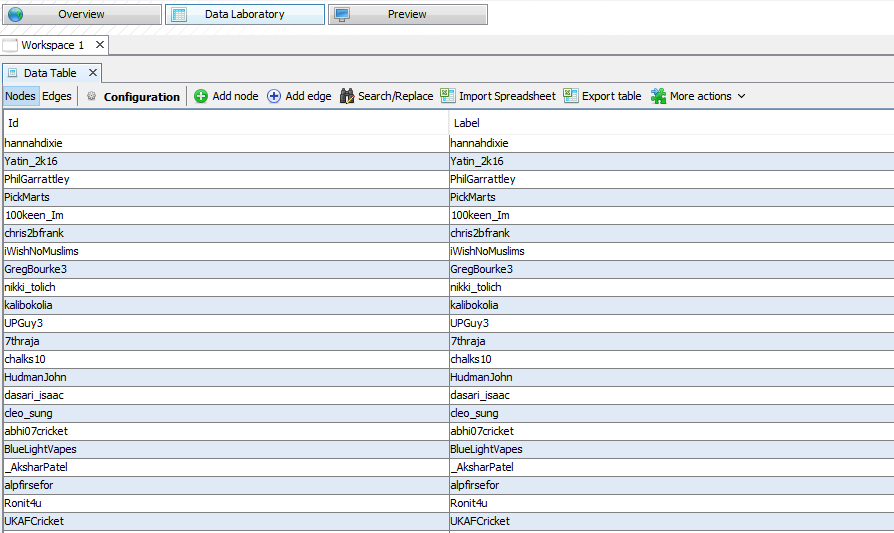
create\_weight\_list("Datasets/\*.dataset")

# 

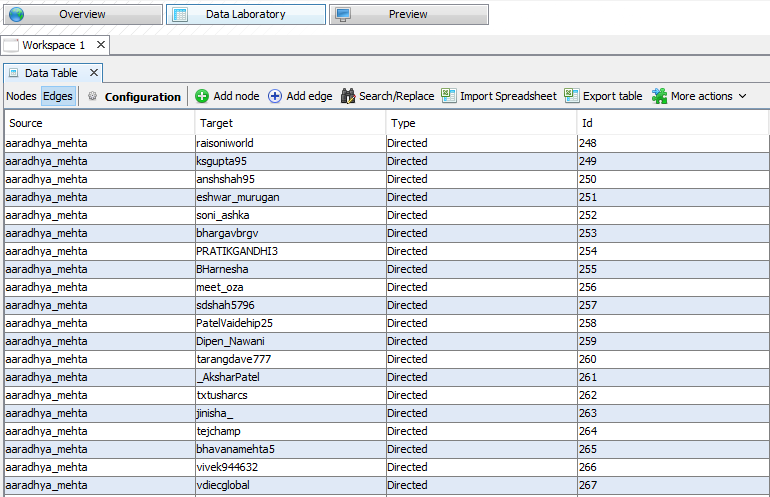
# 

# Data Import and Graph Representation

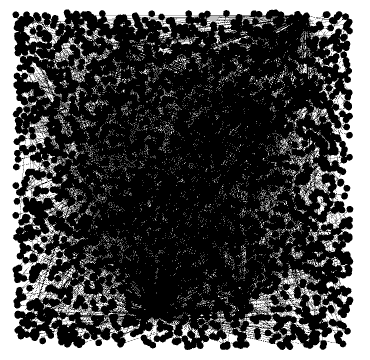
* Import “*Node*” data file generated by **fetch.py**



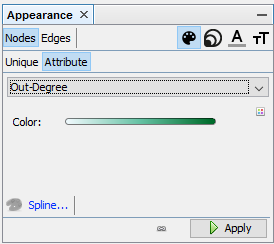
* Import “*weight*” data file generated by **weights.py**



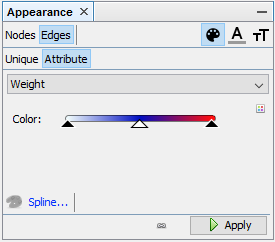
* Click on “*Overview*” Tab.



* In “*Appearance*” Tab. Select “*Nodes*” and change node color on the basis of “*Out-Degree*” Attribute.



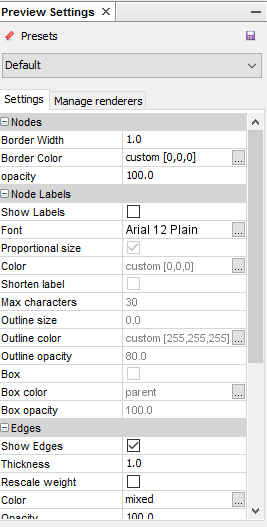
* Click “*Apply*”
* In “Appearance” Tab. Select “*Edges*” and change edges on the basis of “*weight*” Attribute.



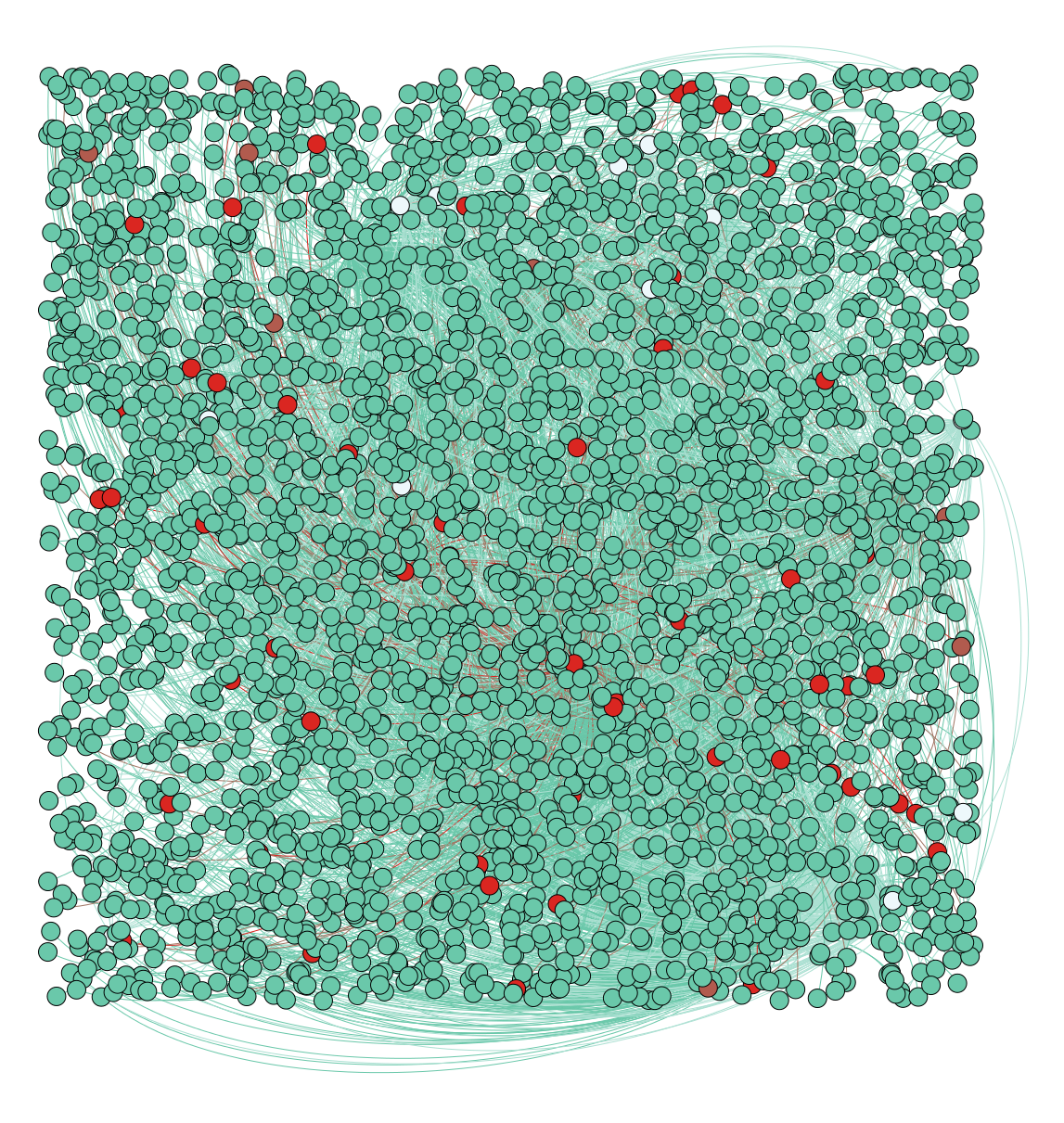
* Click “*Apply*”
* Click on “*Preview*” Tab.



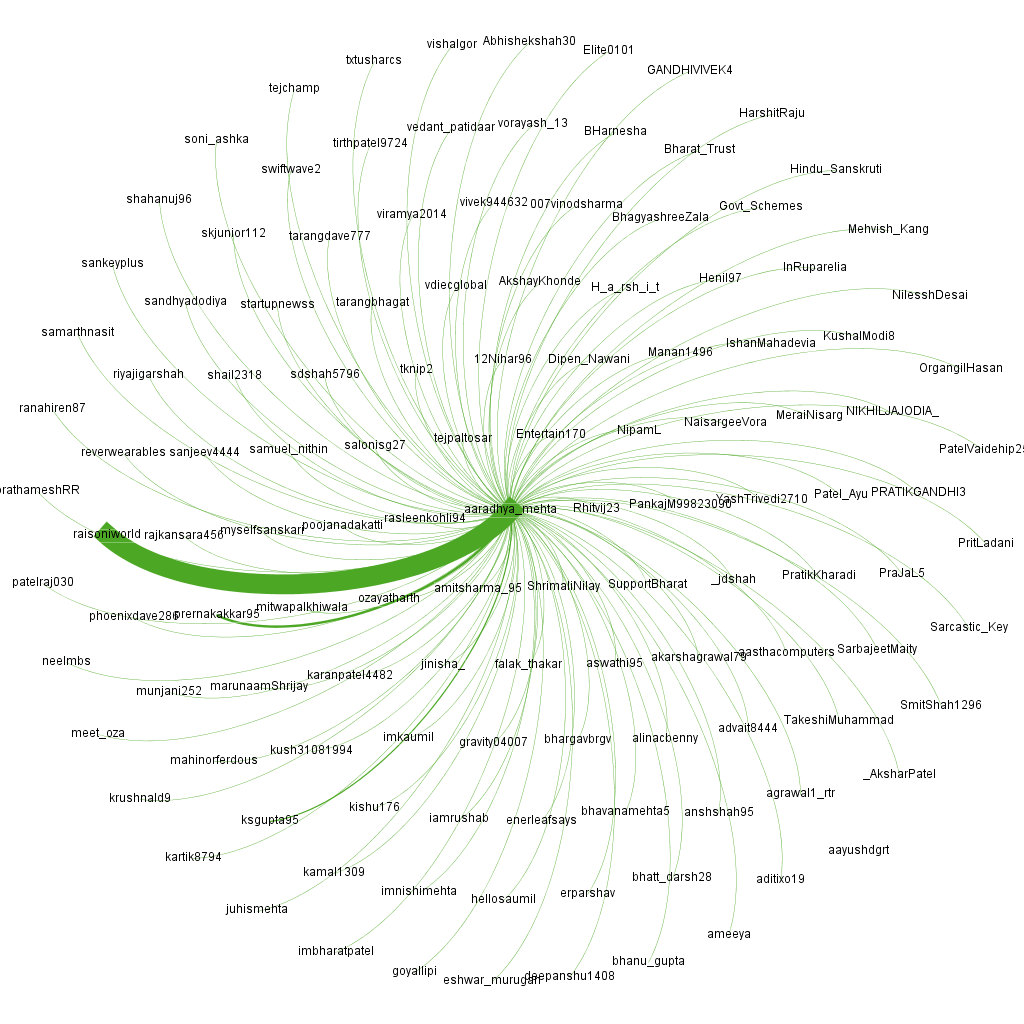
* In “*Preview Settings*” Select “*Default*”



* Click “*Refresh*”



* “*Red*” Nodes represent profiles having followers count more than threshold.
* “*Edges*” represent similarity between two profiles on the basis of social popularity index.
* In “*Preview Settings*” Select “*Tag Cloud*”
* Click “*Refresh*”



# 

# The [degree](https://en.wikipedia.org/wiki/Degree_(graph_theory)) of a node in a network (sometimes referred to incorrectly as the [connectivity](https://en.wikipedia.org/wiki/Connectivity_(graph_theory))) is the number of connections or edges the node has to other nodes. If a network is [directed](https://en.wikipedia.org/wiki/Directed_graph), meaning that edges point in one direction from one node to another node, then nodes have two different degrees, the in-degree, which is the number of incoming edges, and the out-degree, which is the number of outgoing edges.

# Average Degree: 1.025

# 

# 

**Network Interpretation: directed**

**Number of iterations: 100**

**Sum change: 0.0881009119407231**

In [graph theory](https://en.wikipedia.org/wiki/Graph_theory), **eigenvector centrality** (also called **eigen centrality** or **prestige score**) is a measure of the influence of a [node](https://en.wikipedia.org/wiki/Node_(networking)) in a [network](https://en.wikipedia.org/wiki/Network_(mathematics)). Relative scores are assigned to all nodes in the network based on the concept that connections to high-scoring nodes contribute more to the score of the node in question than equal connections to low-scoring nodes. A high eigenvector score means that a node is connected to many nodes who themselves have high scores.

**References**

<https://en.wikipedia.org/wiki/Degree_distribution>

<https://towardsdatascience.com/building-a-network-graph-from-twitter-data-a5e7b8672e3>