

Learning Resource Center

BOOK NUM.:

This book was issued is overdue due on the date stamped below. If the book is kept over due, a fine will be charged as per the library rules.

[illegible]

Opinion Mining and Sentiment Analysis: User Categorization in Social Networks

SUBMITTED BY:

Siddharth Srivastava (081203)

Devashish Gupta (081207)

Abhinav Raj (081258)

SUPERVISED BY:

Mr. Suman Saha



MAY 2012



Submitted in partial fulfilment of the degree of

**BACHELOR OF TECHNOLOGY
IN**

COMPUTER SCIENCE AND ENGINEERING

**JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY,
WAKNAGHAT**

TABLE OF CONTENTS

Chapter No.	Topics	Page No.
	Certificate from the supervisor	3
	Acknowledgement	4
	Summary	5
	List of Figures	6
Chapter-1	Introduction	7-9
	1.1 User Categorization examples	8
	1.2 Objectives	9
	1.3 Project Vision	9
Chapter-2	Categorization: Theory and application	10-21
	2.1 Clustering	10
	2.1.1 Goals	11
	2.1.2 Possible Applications	11
	2.1.3 Requirements	11
	2.1.4 Applications in Social Networking	
	2.2 Classification	12
	2.3 Algorithms	12
	2.3.1 K means algorithm	12
	2.4 Weka: tool for clustering	14
	2.4.1 Arff format	15
	2.4.2 Test data for clustering	16
	2.4.3 Clustering results	17
Chapter-3	Analysis of large scale social network graph	22
	3.1 Graph patterns in large scale social networks	22
	3.1.1 Scale free networks	22
	3.1.2 Small world effect	24
	3.1.3 Community Structures	25
	3.2 Challenges	25

3.3 PEGASUS	26
3.3.1 How the tool works	26
3.3.2 Improvements on Pegasus result	30
3.4 Implementations on data	30
3.4.1 YOUTUBE results	31
3.4.2 TWITTER results	38
Chapter-4	
Future Scope	42
References	48
Appendices A	49

CERTIFICATE

This is to certify that the work titled "**Opinion Mining and Sentiment Analysis: User Categorization in Social Networks**" submitted by **Siddharth Srivastava, Devashish Gupta** and **Abhinav Raj** in partial fulfilment for the award of degree of B. Tech of Jaypee University of Information Technology, Waknaghat has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor:

Suman Saha

Name of Supervisor:

SUMAN SAHA

Designation:

Sr. Lecturer

Date:

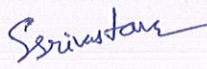
29/5/12

ACKNOWLEDGEMENT

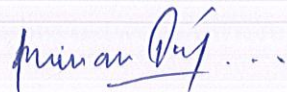
Apart from the efforts, the success of any project depends largely on the encouragement and guidelines of many others. Therefore we take the opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

We would also like to show our appreciation to our project guide Mr. Suman Saha. Without his able guidance, tremendous support and continuous motivation the project work would not be carried out satisfactory. His kind behaviour and motivation provided us the required courage to complete our project.

Special thanks to our project panel because it was their regular concern and appreciation that made this project carried out easily and satisfactorily.

Siddharth Srivastava (081203) 

Devashish Gupta (081207) 

Abhinav Raj (081258) 

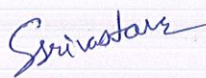
Date: 29/05/12

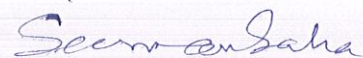
SUMMARY

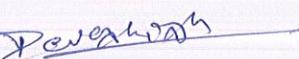
User categorization in social networks is a recent development in the field of graph mining. It uses traditional topics of clustering and classification along with several new recognition techniques to determine important conclusions and predictions about social networks. In a time when social networking sites like Facebook and Twitter are popular not only as a tool for connecting with friends but also provide enormous potential for business opportunities, spreading awareness etc.

This project implements categorization technique of clustering on tabular data using graph mining library Weka and tests for accuracy of clustering. After this we observe large scale social network graphs which comprise of billions of node and mine for connected components and node degrees. We identify the important actors in the social network. To address the problem of scalability we use PEGASUS tool which is build over Hadoop platform and is a very efficient tool for mining graphs for large sizes.

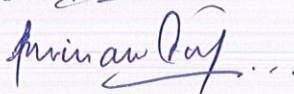
In the end we suggest further improvements like building GUI and, implementing pagerank, and differentiating between different linktypes.

Siddharth Srivastava 


Mr. Suman Saha

Devashish Gupta 

Date:

Abhinav Raj 

Date:

LIST OF FIGURES

Sr. No.	Name	Page No.
1	Fig 1: Clustering example	4
2	Fig 2: Snapshot of Weka	8
3	Fig 3: Normal distribution	13
4	Fig 4: Power law distribution in social networks	16
5	Fig 5: Small world effect	17
6	Fig 6: In degree plot of youtube group-edges.csv	24
7	Fig 7: Out degree plot of youtube group-edges.csv	25
8	Fig 8: Inout degree plot of youtube edges.csv	28
9	Fig 9: In degree plot of youtube edges.csv	28
10	Fig 10: out degree plot of youtube edges.csv	29
11	Fig 11: connected component plot of youtube edges.csv	30
12	Fig 12: In degree plot of Twitter edges.csv	31
13	Fig 13.: Out degree plot of Twitter edges.csv	32
14	Fig 14: Inout degree plot of Twitter edges.csv	33
15	Fig 15: Connected component plot of Twitter edges.csv	34
16	Fig 16: Node traversal for pagerank	35
17	Fig 17: Pagerank plot	38
18	Fig 18: Cliques, clans and clubs	39

CHAPTER 1

INTRODUCTION

Use Categorization in Social Network includes the analysis of social structures, social position, role analysis, and many others feature of a social network. Usually, the relationship between people and groups, e.g., friends, followers, subscribers, etc. are presented as a graph. In a network graph nodes represent individuals and edges represent the relation between them. Usually this type of data collection confines traditional social network analysis to a limited scale.

With the prosperity of Internet and Web, many social networking and social media sites are emerging, and people can easily connect to each other in the cyber space. The prosperity of Web and social media brings about many diverse social networks of unprecedented scales, which present new challenges for more effective graph-mining techniques.

This also facilitates social network analysis to a much larger scale comprising millions of actors or even more in a network; e.g. include email communication networks, instant messenger networks, mobile call networks, friends networks. Other forms of complex network, like co-authorship or citation networks, biological networks, metabolic pathways, genetic regulatory networks, food web and neural networks are also examined and demonstrate similar patterns. These large scale networks of various entities yield patterns which are normally not seen in small networks. In addition, they also pose challenges for computation as well as new tasks and problems for the social network analysis. We present some graph patterns that are generally seen in large-scale social networks.

Social network analysis involves a variety of tasks. The major ones are:

- Centrality analysis aims to identify the “most important” actors in a social network. Centrality is a measure to calibrate the “importance” of an actor. This helps to understand the social influence and power in a network.
- Community detection. Actors in a social network form groups. This task identifies these communities through the study of network structures and topology.
- Position or Role analysis identifies the role associated with different actors during network interaction. Who serves as the bridge between different groups?
- Network modelling attempts to simulate the real-world network via simple mechanisms such that the patterns presented in large-scale complex networks can be captured.

- Information diffusion studies how the information propagates in a network. Information diffusion also facilitates the understanding the cultural dynamics, and infection blocking.
- Network classification and outlier detection. Some actors are labelled with certain information. For instance, in a network with some terrorists identified, is it possible to identify other people who are likely to be terrorists by leveraging the social network information.
- Viral marketing and link prediction. The modelling of the information diffusion process, in conjunction with centrality analysis and communities, can help achieve more cost-effective viral marketing. That is, only a small set of users are selected for marketing. Hopefully, their adoption can influence other members in the network, so the benefit is maximized.

Normally, a social network is represented as a graph. How to mine the patterns in the graph for the above tasks becomes a hot topic thanks to the availability of enormous social network data. In this chapter, we attempt to present some recent trends of large social networks and discuss graph mining applications for social network analysis. In particular, we discuss graph mining applications to community detection, a basic task in social network analysis to extract meaningful social which also serves as basis for some other related social network analysis tasks. Representative approaches for community detection are summarized. Interesting emerging problems and challenges are also presented for future exploration.

For convenience, we define some notations used throughout this report. A network is normally represented as a graph $G(V, E)$, where V denotes the vertexes (equivalently nodes or actors) and E denotes edges (ties or connections). The connections are represented via adjacency matrix A , where $A_{ij} = 1$ means $(v_i, v_j) \in E$, while $A_{ij} = 0$ means $(v_i, v_j) \notin E$. The degree of node v_i is d_i . If the edges between nodes are directed, the in-degree and out-degree are denoted as d^- and d^+ respectively. Number of vertexes and edges of a network are $|V| = n$, and $|E| = m$. The shortest path between a pair of nodes v_i and v_j is called geodesic, and the geodesic distance between the two is denoted as $d(i, j)$. $G_s(V_s, E_s)$ represent a sub graph in G . The neighbours of a node v are denoted as $N(v)$. In a directed graph, the neighbours connecting to and from one node v are denoted as $N^-(v)$ and $N^+(v)$, respectively. Unless specified explicitly, we assume a network is un-weighted and undirected.

1.1 User Categorization: examples

- Twitter: Predicting whom a person is likely to follow
- Facebook: Providing friend suggestions.
- IMDb: Providing movies suggestions

- Youtube: Suggesting videos which the user may find interesting.

1.2 Objectives

- Observe the patterns and structure of social networks.
- Study how groups and communities are formed in social networks.
- Learn the techniques of clustering and classification to divide entities into relevant groups accurately and also to assign correct group to a new node.
- To learn how large scale social networking graphs which have size in GBs can be mined.
- To find connected components in a social network so that future links can be predicted.
- To find important actors in a social network by the use of node degrees and pagerank.
- To learn about graph mining libraries like Weka and PEGASUS and make their use in our project.

1.3 Project Vision

- Having better knowledge about the mechanism of social networks.
- Finding groups of customers with similar behavior, can be extended to the field of marketing and entertainment.
- Being able to provide accurate suggestions like movie suggestions, friends suggestions etc.
- Finding hidden communities which are not explicit.

CHAPTER 2

CATEGORIZATION: THEORY AND APPLICATIONS

2.1 Clustering

- A *cluster* is a collection of objects which are “similar” between them and are “dissimilar” to the objects belonging to other clusters.
- Therefore *Clustering* is the process of categorizing objects into groups whose members are similar in some way.

Clustering can be considered the most important *unsupervised learning* problem; so, as every other problem of this kind, it deals with finding a *structure* in a collection of unlabeled data. Another definition of clustering could be “the process of organizing objects into groups whose members are similar in some way”.

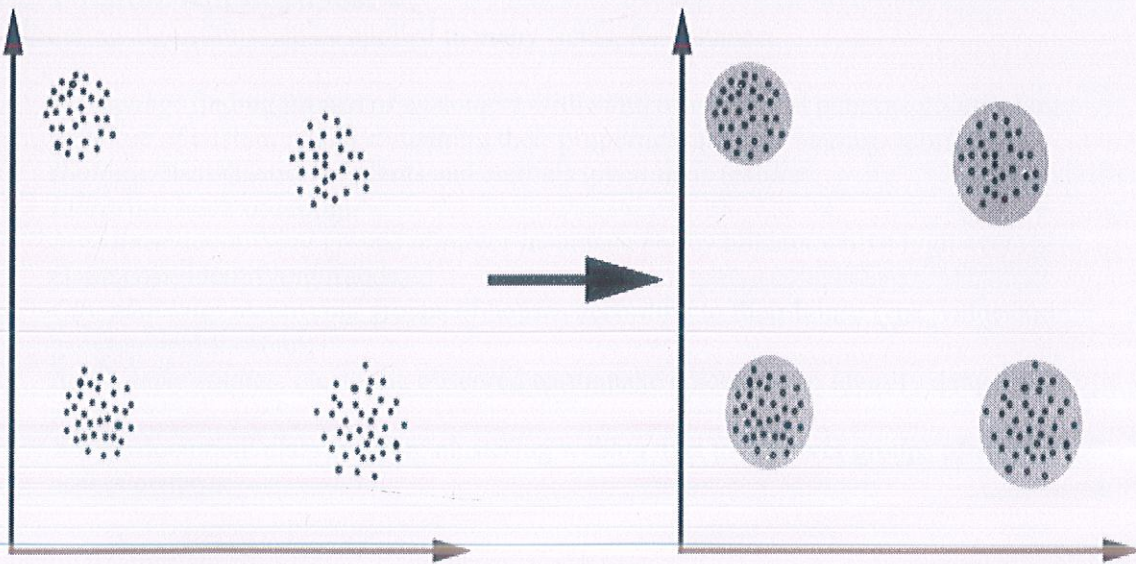


Figure 1: Clustering example

In this case we easily identify the 4 clusters into which the data can be divided; the similarity criterion is *distance*: two or more objects belong to the same cluster if they are “close” according to a given distance (in this case geometrical distance). This is called *distance-based clustering*.

2.1.1 Goals:

The goal of clustering is to determine the intrinsic grouping in a set of unlabeled data. What constitutes a good clustering? It can be shown that there is no absolute “best” criterion which would be independent of the final aim of the clustering. Consequently, it is the user which must supply this criterion, in such a way that the result of the clustering will suit their needs. For instance, we could be interested in finding representatives for homogeneous groups (*data reduction*), in finding “natural clusters” and describe their unknown properties (“*natural*” *data types*), in finding useful and suitable groupings (“*useful*” *data classes*) or in finding unusual data objects (*outlier detection*).

2.1.2 Possible Applications:

clustering algorithms can be applied in many fields, for instance:

- *Marketing*: finding groups of customers with similar behavioral patterns given a large database of customer data containing their properties and past buying records;
- *Biology*: classification of plants and animals given their features;
- *Libraries*: book ordering;
- *Insurance*: identifying groups of motor insurance policy holders with a high average claim cost; identifying frauds;
- *City-planning*: identifying groups of houses according to their house type, value and geographical location;
- *Earthquake studies*: clustering observed earthquake epicenters to identify dangerous zones;
- *WWW*: document classification; clustering weblog data to discover groups of similar access patterns.

2.1.3 Requirements:

The main requirements that a clustering algorithm should satisfy are:

- scalability;

- dealing with different types of attributes;
- discovering clusters with arbitrary shape;
- minimal requirements for domain knowledge to determine input parameters;
- ability to deal with noise and outliers;
- insensitivity to order of input records;
- high dimensionality;
- Interpretability and usability.

2.1.4 Application in social networking:

If we have data in tabular format where attributes of all items are given then we can cluster them into as many groups as we want. Some groups can be familiar like people belonging to same college, school, and workplace. We can also discover new groups which are not explicit but implicit because of hobbies and behaviour of people. There is a high probability that the people falling in same clusters can become friends in future if they are not already friends provided that we define the parameters for clustering properly. This can be useful in providing friend suggestions, movie suggestions, video suggestions, community suggestions etc.

2.2 Classification

- Classification is a data mining function that assigns items in a collection to target categories or classes.
- The goal of classification is to accurately predict the target class for each case in the data.
- For example, a classification model could be used to identify loan applicants as low, medium, or high credit risks.
- Goal: previously unseen records should be assigned a class as accurately as possible.

2.3 Algorithms

We are using k-means algorithm for clustering the data.

2.3.1 K-means Algorithm

```
public class SimpleKMeans extends RandomizableClusterer
implements NumberOfClustersRequestable, WeightedInstancesHandler
```

Cluster data using the k means algorithm.

Valid options are:

- N <num>: number of clusters. (default 2).
- V: Display std. deviations for centroids.
- M: Replace missing values with mean/mode.
- S <num>: Random number seed. (default 10)
- A <classname and options>:
 Distance function to be used for instance comparison
 (default weka.core.EuclidianDistance)
- I <num>: Maximum number of iterations.
- O: Preserve order of instances.

The basic methods used from this class are:

buildClusterer():

```
public void buildClusterer(Instances data)
                        throws java.lang.Exception
```

Generates a clusterer. Has to initialize all fields of the clusterer that are not being set via options.

numberOfClusters():

```
public int numberOfClusters()
                        throws java.lang.Exception
```

Returns the number of clusters.

setNumClusters():

```
public void setNumClusters(int n)
                        throws java.lang.Exception
```

Set the number of clusters to generate.

setOptions():

```
public void setOptions(java.lang.String[] options)
                        throws java.lang.Exception
```

Parses a given list of options.

Valid options are:

- N <num>: number of clusters. (default 2).
- V: Display std. deviations for centroids.
- M: Replace missing values with mean/mode.
- S <num>: Random number seed. (default 10)
- A <classname and options>:
 Distance function to be used for instance
 comparison
 (default weka.core.EuclidianDistance)
- I <num>: Maximum number of iterations.
- O: Preserve order of instances.

getClusterSizes():

```
public int[] getClusterSizes()
```

Gets the number of instances in each cluster

2.4 Weka: tool used for clustering

We have used Weka library to for clustering data. Weka is used for tabular data.

- Weka (Waikato Environment for Knowledge Analysis) is a popular suite of machine learning software written in Java, developed at the University of Waikato, New Zealand.
- Weka supports several standard data mining tasks, more specifically, data preprocessing, clustering, classification, regression, visualization, and feature selection.

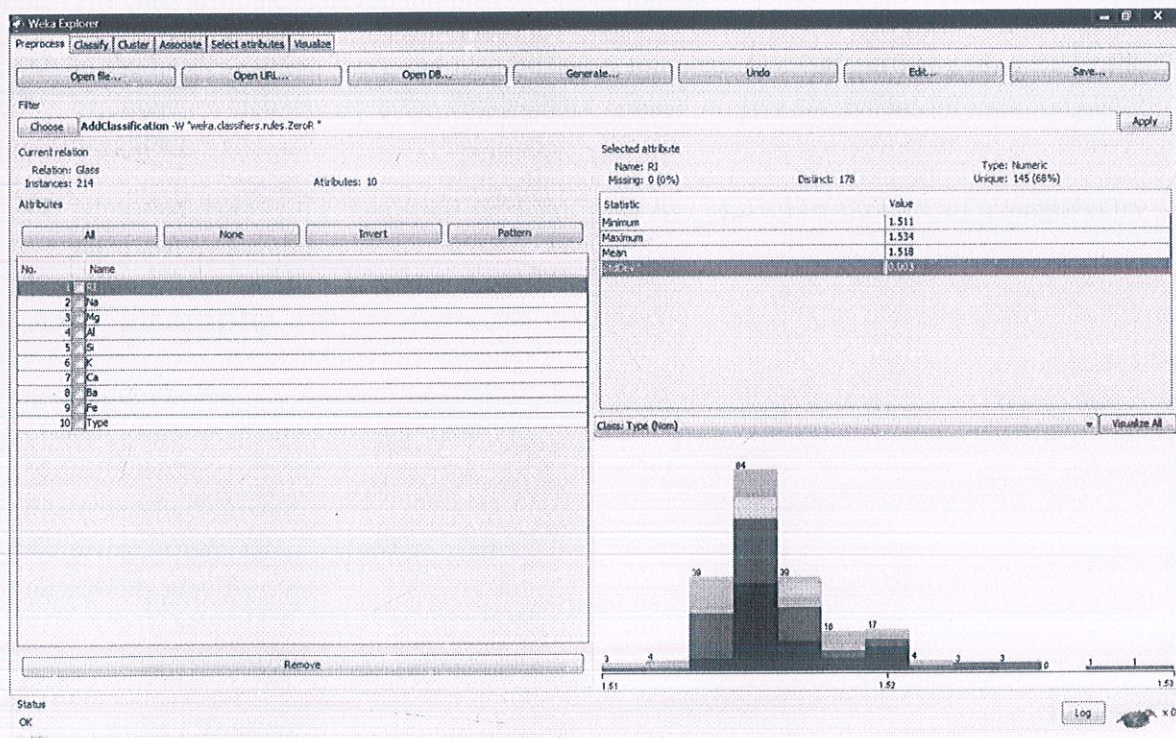


FIGURE2. Snapshot of Weka

2.4.1 Arff format

- Attribute Relationship File Format (ARFF) is the text format file used by Weka to store data in a database.
- The ARFF file contains two sections: the header and the data section.
- The first line of the header tells us the relation name. Then there is the list of the attributes (@attribute...).
- Each attribute is associated with a unique name and a type.
- The latter describes the kind of data contained in the variable and what values it can have.
- The variables types are: numeric, nominal, string and date.
- The class attribute is by default the last one of the list.
- In the header section there can also be some comment lines, identified with a '%' at the beginning, which can describe the database content or give the reader information about the author.
- After that there is the data itself (@data), each line stores the attribute of a single entry separated by a comma.

Arff Example

```
@relationweather
@attributeoutlook{sunny,overcast,rainy}
@attributetemperaturreal
@attributehumidityreal
@attributewindy{TRUE,FALSE}
@attribute play {yes, no}
```

```
@data
sunny,85,85,FALSE,no
sunny,80,90,TRUE,no
overcast,83,86,FALSE,yes
rainy,70,96,FALSE,yes
rainy,68,80,FALSE,yes
```


rainy,65,70,TRUE,no
overcast,64,65,TRUE,yes

2.4.2 Test data for clustering

For testing we have used a data that closely resembles social networking data. The data is in tabular format. There are various attributes for each item and after applying k-means algorithm for clustering we are able to arrange the items into clusters. Because we already know the original clusters so we can also check the accuracy of the clustering algorithm.

Sources:

Stefan Aeberhard, email: stefan@coral.cs.jcu.edu.au

-- These data are the results of a chemical analysis of Wines grown in the same region in Italy but derived from three different cultivators.

--The analysis determined the quantities of 13 constituents found in each of the three types of wines .

-- The attributes are

- 1) Alcohol
- 2) Malic acid
- 3) Ash
- 4) Alcalinity of ash
- 5) Magnesium
- 6) Total phenols
- 7) Flavanoids
- 8) Nonflavanoid phenols
- 9) Proanthocyanins
- 10) Color intensity
- 11) Hue
- 12) OD280/OD315 of diluted wines
- 13) Proline

--Class Distribution: Number of Instances

Class 1 59

Class 2 71

Class 3 48

2.4.3 Clustering result after applying k-means algorithm:

Simple kMeans:

Number of iterations: 8

Within cluster sum of squared errors: 48.970291155139165

Missing values globally replaced with mean/mode

Cluster centroids:

Attribute	Cluster#			
	Full Data (178)	0 (60)	1 (55)	2 (63)
a	13.0006	13.7193	13.0998	12.2295
b	2.3363	1.964	3.1609	1.9711
c	2.3665	2.4565	2.4075	2.2451
d	19.4949	17.2783	21.0436	20.254
e	99.7416	107.8667	98.6545	92.9524
f	2.2951	2.8455	1.6898	2.2994
g	2.0293	2.9748	0.8478	2.1602
h	0.3619	0.2887	0.4578	0.3478
i	1.5909	1.9273	1.1336	1.6697
j	5.0581	5.4627	6.9365	3.0329
k	0.9574	1.0718	0.7168	1.0586
l	2.6117	3.1573	1.7093	2.8798
m	746.8933	1117.8167	624.8545	500.1746

Clustered Instances

```
0 60 (34%)
1 55 (31%)
2 63 (35%)
```

Class attribute: class

Classes to Clusters:

```
0 1 2 <-- assigned to cluster
58 0 1 | 1
2 7 62 | 2
0 48 0 | 3
Cluster 0 <-- 1
Cluster 1 <-- 3
```


Cluster 2 <-- 2

Incorrectly clustered instances: 10.0 5.618 %

Accuracy: 94.382 %

Make Density Based Clusters:

Wrapped clusterer:

kMeans

Number of iterations: 8

Within cluster sum of squared errors: 48.970291155139165

Missing values globally replaced with mean/mode

Cluster centroids:

Attribute	Full Data (178)	0 (60)	1 (55)	2 (63)
a	13.0006	13.7193	13.0998	12.2295
b	2.3363	1.964	3.1609	1.9711
c	2.3665	2.4565	2.4075	2.2451
d	19.4949	17.2783	21.0436	20.254
e	99.7416	107.8667	98.6545	92.9524
f	2.2951	2.8455	1.6898	2.2994
g	2.0293	2.9748	0.8478	2.1602
h	0.3619	0.2887	0.4578	0.3478
i	1.5909	1.9273	1.1336	1.6697
j	5.0581	5.4627	6.9365	3.0329
k	0.9574	1.0718	0.7168	1.0586
l	2.6117	3.1573	1.7093	2.8798
m	746.8933	1117.8167	624.8545	500.1746

Fitted estimators (with ML estimates of variance):

Cluster: 0 Prior probabilities: 0.337

Attribute: a

Normal Distribution: Mean = 13.7193 StdDev = 0.488

Attribute: b

Normal Distribution: Mean = 1.964 StdDev = 0.6304

Attribute: c

Normal Distribution: Mean = 2.4565 StdDev = 0.2255

Attribute: d

Normal Distribution: Mean = 17.2783 StdDev = 3.0118

Attribute: e

Normal Distribution: Mean = 107.8667 StdDev = 13.17

Attribute: f

Normal Distribution: Mean = 2.8455 StdDev = 0.3403

Attribute: g

Normal Distribution: Mean = 2.9748 StdDev = 0.399

Attribute: h

Normal Distribution: Mean = 0.2887 StdDev = 0.0696

Attribute: i

Normal Distribution: Mean = 1.9273 StdDev = 0.4408

Attribute: j

Normal Distribution: Mean = 5.4627 StdDev = 1.2953

Attribute: k

Normal Distribution: Mean = 1.0718 StdDev = 0.115

Attribute: l

Normal Distribution: Mean = 3.1573 StdDev = 0.3598

Attribute: m

Normal Distribution: Mean = 1117.8167 StdDev = 212.3306

Cluster: 1 Prior probability: 0.3094

Attribute: a

Normal Distribution: Mean = 13.0998 StdDev = 0.5248

Attribute: b

Normal Distribution: Mean = 3.1609 StdDev = 1.1686

Attribute: c

Normal Distribution: Mean = 2.4075 StdDev = 0.1938

Attribute: d

Normal Distribution: Mean = 21.0436 StdDev = 2.4652

Attribute: e

Normal Distribution: Mean = 98.6545 StdDev = 10.9813

Attribute: f

Normal Distribution: Mean = 1.6898 StdDev = 0.3665

Attribute: g

Normal Distribution: Mean = 0.8478 StdDev = 0.3291

Attribute: h

Normal Distribution: Mean = 0.4578 StdDev = 0.1221

Attribute: i

Normal Distribution: Mean = 1.1336 StdDev = 0.4231

Attribute: j

Normal Distribution: Mean = 6.9365 StdDev = 2.477

Attribute: k

Normal Distribution: Mean = 0.7168 StdDev = 0.1494

Attribute: l

Normal Distribution: Mean = 1.7093 StdDev = 0.2687

Attribute: m

Normal Distribution: Mean = 624.8545 StdDev = 122.5548

Cluster: 2 Prior probabilities: 0.3536

Attribute: a

Normal Distribution: Mean = 12.2295 StdDev = 0.5378

Attribute: b

Normal Distribution: Mean = 1.9711 StdDev = 1.0376

Attribute: c

Normal Distribution: Mean = 2.2451 StdDev = 0.3261

Attribute: d

Normal Distribution: Mean = 20.254 StdDev = 3.1686

Attribute: e

Normal Distribution: Mean = 92.9524 StdDev = 13.8798

Attribute: f

Normal Distribution: Mean = 2.2994 StdDev = 0.5143

Attribute: g

Normal Distribution: Mean = 2.1602 StdDev = 0.6854

Attribute: h

Normal Distribution: Mean = 0.3478 StdDev = 0.112

Attribute: i

Normal Distribution: Mean = 1.6697 StdDev = 0.5316

Attribute: j

Normal Distribution: Mean = 3.0329 StdDev = 0.8984

Attribute: k

Normal Distribution: Mean = 1.0586 StdDev = 0.2037

Attribute: l

Normal Distribution: Mean = 2.8798 StdDev = 0.4044

Attribute: m

Normal Distribution: Mean = 500.1746 StdDev = 132.878

Clustered Instances:

0	59 (33%)
1	54 (30%)
2	65 (37%)

Class attribute: class

Classes to Clusters:

0 1 2 <-- assigned to cluster

57 0 2 | 1

2 6 63 | 2

0 48 0 | 3

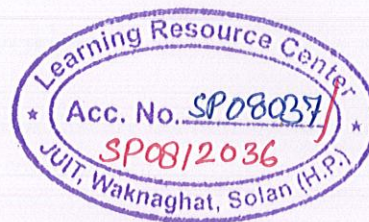
Cluster 0 <-- 1

Cluster 1 <-- 3

Cluster 2 <-- 2

Incorrectly clustered instances: 10.0 5.618 %

Accuracy: 94.382 %



Conclusion: we can say that above clustering techniques can be used to identify clusters social networking data in tabular format with decent accuracy.

CHAPTER 3

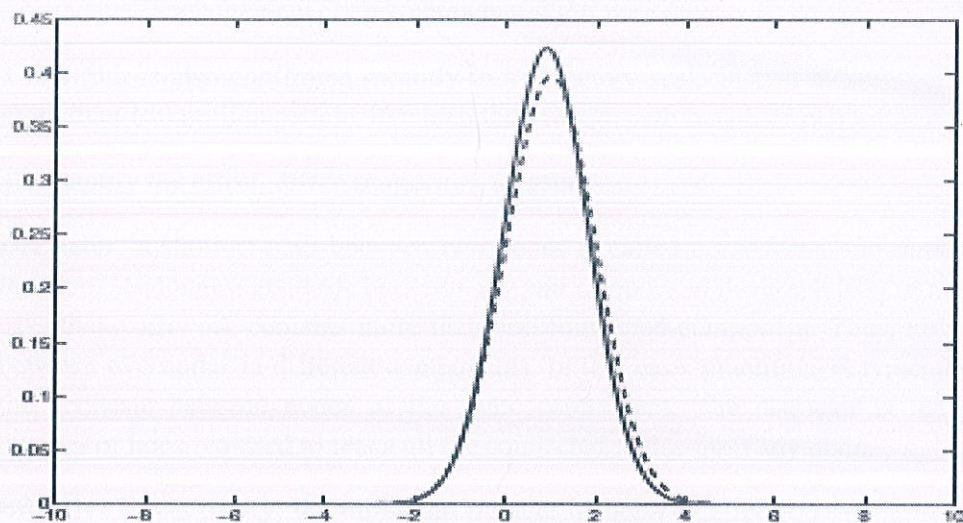
ANALYSIS OF LARGE SCALE SOCIAL NETWORK GRAPHS:

3.1 Graph Patterns in Large-Scale Networks:

- Most large-scale networks share some common patterns that are not noticeable in small networks. Among all the patterns, the most well-known characteristics are:
 - scale-free distribution
 - small world effect
 - strong community structure.

3.1.1 Scale Free Network

- Many statistics in real-world have a typical “scale”, a value around which the sample measurements are centred.
- For instance, the height of all the people in the United States, the speed of vehicles on a highway, etc.
- But the node degrees in real-world large scale social networks often follow a power law distribution (a.k.a. Zipfian distribution, Pareto distribution).
- While the normal distribution has a “centre”, the power law distribution is highly skewed.
- For normal distribution, it is extremely rare for an event to occur that are several deviations away from the mean.
- On the contrary, power law distribution allows the tail to be much longer.
- That is, it is common that some nodes in a social network have extremely high degrees while the majority have few connections.
- The reason is that the decay of the tail for a power law distribution is polynomial.



(a) Normal Distribution

Figure3. normal distribution

Power Law Distribution

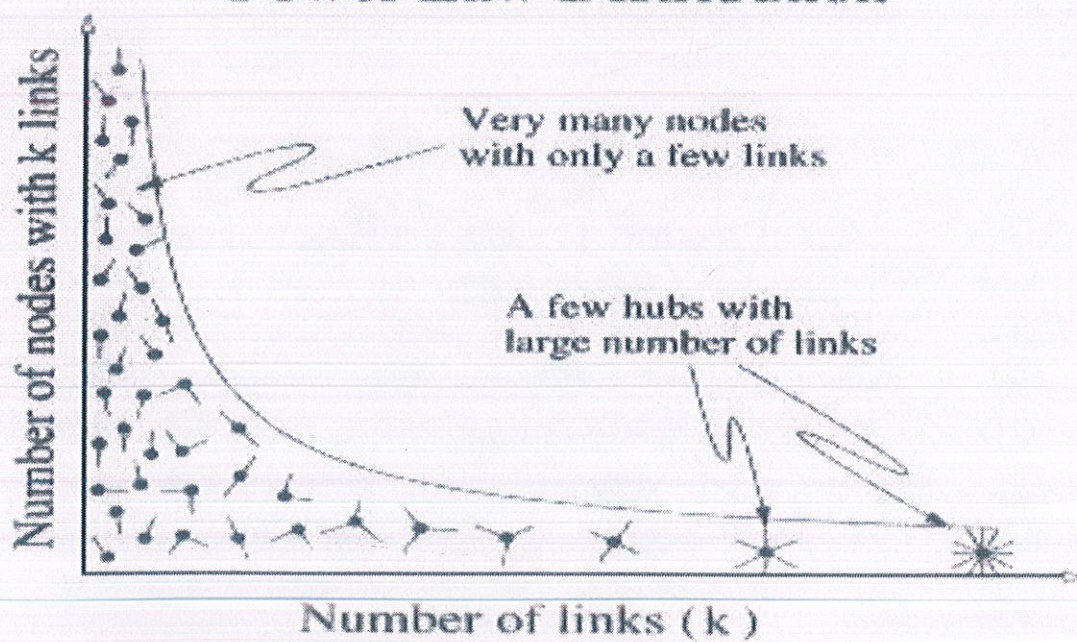


Figure4. power law distribution in social networks.

3.1.2 Small World Effect

- Two actors in a huge network are actually not too far away.
- This result is also confirmed recently in a planetary-scale instant messaging network of more than 180 million people, in which the average path length of two messengers is 6.6
- To quantify the effect, different network measures are used:
- **Diameter:** a shortest path between two nodes is called a *geodesic*, and *diameter* is the length of the longest geodesic between any pair of nodes in the graph [61]. It might be the case that a network contains more than one connected component. Thus, no path exists between two nodes in different components. In this case, practitioners typically examine the geodesic between nodes of the same component. The diameter is the minimum number of hops required to reach all the connected nodes from any node.
- **Effective Eccentricity:** the minimum number of hops required to reach at least 90% of all connected pairs of nodes in the network. This measure removes the effect of outliers that are connected through a long path.
- **Characteristic Path Length:** the median of the means of the shortest path lengths connecting each node to all other nodes (excluding unreachable ones). This measure focuses on the average distance between pairs rather than the maximum one as the diameter.

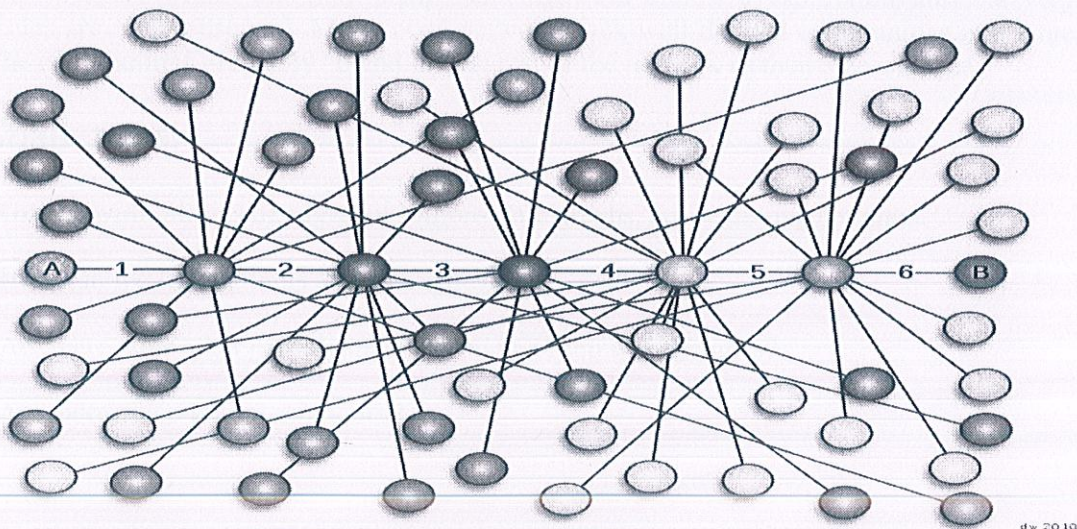


Figure 5. small world effect

3.1.3 Community Structures

Social networks demonstrate a strong community effect. That is, a group of people tend to interact with each other more than those outside the group. To measure the community effect, one related concept is *transitivity*. In a simple form, friends of a friend are likely to be friends as well. *Clustering coefficient* is proposed specifically to measure the transitivity, the probability of connections between one vertex's neighbouring friends.

Clustering Coefficient: Suppose a node v_i has d_i neighbors, and there are k_i edges among these neighbors, then the clustering coefficient Graph Mining Applications to Social Network Analysis is

$$C_i = \begin{cases} \frac{k_i}{d_i \times (d_i - 1) / 2} & d_i > 1 \\ 0 & d_i = 0 \text{ or } 1 \end{cases}$$

While clustering coefficient and transitivity concentrate on microscopic view of community effect, communities of macroscopic view also demonstrate intriguing patterns. In real-world networks, a giant component tends to form with the remaining being singletons and minor communities. Even within the giant component, tight but almost trivial communities at very small scales are often observed. Most social networks lack well-defined communities in a large scale. The communities gradually “blend in” the rest of the network as their size expands.

3.2 Challenges:

- Graphs with billions of edges, or billion-scale graphs, are becoming common.
- How can we find patterns and anomalies?
- Are there nodes that participate in too many or too few triangles?
- Are there close-knit near-cliques?

3.3 PEGASUS

There are a huge number of graph mining algorithms, computing communities, finding important nodes (e.g., Page Rank), computing the number of triangles computing the diameter , topic detection, attack detection ,with too-many-to-list alternatives for each of the above tasks. Most of the previous algorithms do not scale, at least directly, to several millions and billions of nodes and edges. For connected components, there are several algorithms, using Breadth-First Search, Depth-First-Search, “propagation” or “contraction” . These works rely on a shared memory model which limits their ability to handle large, disk-resident graphs.

MAPREDUCE is a programming framework for processing huge amounts of unstructured data in a massively parallel way. It has two major advantages: (a) the programmer is oblivious of the details of the data distribution, replication, load balancing etc. and furthermore (b) the programming concept is familiar, i.e., the concept of functional programming. Briefly, the programmer needs to provide only two functions, a *map* and a *reduce*. The typical framework is as follows : (a) the *map* stage sequentially passes over the input file and outputs (key, value) pairs; (b) the *shuffling* stage groups of all values by key, (c) the *reduce* stage processes the values with the same key and outputs the final result.

HADOOP is the open source implementation of MapReduce, a very promising tool for large scale graph mining applications

Due to its power, simplicity and the fact that building a small cluster is relatively cheap, HADOOP is a very promising tool for large scale graph mining applications.

PEGASUS, an open source Peta Graph Mining library which performs typical graph mining tasks such as computing the diameter of the graph, computing the radius of each node and finding the connected components

As the size of graphs reaches several Giga-, Tera- or Peta-bytes, the necessity for such a library grows too. It is implemented on the top of the **HADOOP** platform

3.3.1 How the Tool Works

PEGASUS requires Hadoop to be installed in the system. The typical usage of PEGASUS is to run it on a gateway server of the Hadoop cluster. Alternatively, you can install Hadoop on a single machine and run PEGASUS on the single machine.

PEGASUS requires additional four programs to run. They are Java, Apache Ant, Python, and Gnuplot. The purposes of these programs are:

- Java is required since Hadoop runs on top of Java.
- Apache Ant is needed to rebuild the code after you modify it.
- Python and Gnuplot is needed to generate plots in the interactive command line UI `pegasus.sh`.

The social networking data in graphical format is added to pegasus. Then various attributes of the graph like degrees of nodes, connected component etc. can be mined by using the available functions. The results are stored in a file. The plot of the results can also be generated.

Sample graph

PEGASUS works on graphs with TAB-separated plain text format. Each line contains the source and destination node id of an edge. The node id starts from 0. For example, here is an example graph which is included in the installation file. It has 16 nodes.

```
0      1
1      2
1      3
3      4
3      6
5      6
6      7
6      8
6      9
10     11
10     12
10     13
10     14
10     15
```

Commands available in PEGASUS

Command	Description
add [file or directory] [graph_name] HDFS	upload a local graph file or directory to HDFS
del [graph_name]	delete a graph
list	list graphs
compute ['deg' or 'pagerank' or 'rwr' or 'radius' or 'cc'] [graph_name]	run an algorithm on a graph

plot ['deg' or 'pagerank' or 'rwr' or 'radius' or 'cc' or 'corr'] [graph_name]	generate plots
help	show this screen
demo	show demo
exit	exit PEGASUS

Computing degree distribution

To run Degree Distribution, you need to do the two things:

- copy the graph edge file to a HDFS directory, say dd_edge
- execute ./run_dd.sh

The syntax of run_dd.sh is:

./run_dd.sh [in or out or inout] [#_of_reducers] [HDFS edge_file_path]

-, where:

[in or out or inout]:	type of degree to compute.
[#_of_reducers]:	number of reducers to use in hadoop.

- The number of reducers to use depends on the setting of the hadoop cluster.
- The rule of thumb is to use (number_of_machine * 2) as the number of reducers.

[HDFS edge_file_path]: HDFS directory where edge file is located

Ex: ./run_dd.sh inout 16 dd_edge.

The outputs of Degree Distribution are saved in the following HDFS directories:
dd_node_deg:

- Each line contains the degree of each node in the format of (nodeid TAB degree_of_the_node).
- For example, the line "1 3" means that the degree of node 1 is 3.
- Each line contains a degree and the number of nodes with the degree.
- For example, the line "1 12" means that 12 nodes have degree 1.

Computing connected components

To run Connected Component-plain, you need to do the two things:

- copy the graph edge file to the HDFS directory `cc_edge`
- execute `./run_ccmpt.sh`

The syntax of `run_ccmpt.sh` is:

`./run_ccmpt.sh [#_of_nodes] [#_of_reducers] [HDFS edge_file_path]`

where

`[#_of_nodes]`: number of nodes in the graph

`[#_of_reducers]`: number of reducers to use in hadoop.

- The number of reducers to use depends on the setting of the hadoop cluster.
- The rule of thumb is to use $(\text{number_of_machine} * 2)$ as the number of reducers.

`[HDFS edge_file_path]`: HDFS directory where edge file is located

Ex: `./run_ccmpt.sh 16 3 cc_edge`

The output is saved in the following HDFS directory:

`concmpt_curbm`:

- Each line contains the component id of each node in the format of (nodeid TAB "msf" component_id_of_the_node).
- For example, the line "2 msf1" means that the component id of node 2 is 1. The component id is the minimum node id of it.

`Concmpt_distr`:

- Each line contains the size of connected component and frequency of such components
- For example "10 7" means that there are 7 connected components of size 10 each.

`concmpt_summaryout`: The distribution of connected components.

- The first column is the component id.
- The second column is the number of nodes in the component.

3.3.2 Improvement on the results:

The results given by pegasus leave a lot of scope for improvement. For example in degree distribution Pegasus does not tell us which degree has the highest degree?

0	1
1	3
2	1
3	3
4	1
5	1
6	5
7	1
8	1
9	1
10	5
11	1
12	1
13	1
14	1
15	1

In the above result the node with highest degree can be observed but if the nodes are in millions we will need a program to do this.

Also in case of connected component it does not tell us which the largest connected component is.

While we aim to mine such information, we have developed a JAVA code which performs this task for us. It takes the Pegasus results file as input and returns with the id of the largest group or the node with highest degree, as required. Code is given in Appendix A.

3.4 Implementation on data

DATASET: TWITTER

Number of Nodes: 11,316,811

Number of Edges: 85,331,846

2 files are included:

nodes.csv: it's the file of all the users. This file works as a dictionary of all the users in this dataset. It's useful for fast reference. It contains all the node ids used in the dataset.

edges.csv: this is the friendship/followership network among the bloggers. The friends/followers are represented using edges. Edges are directed.

DATASET: YOUTUBE

Number of users : 1,138,499

Number of friendship pairs: 2,990,443

Number of groups: 47

4 files are included:

1. nodes.csv: it's the file of all the users. This file works as a dictionary of all the users in this data set. It's useful for fast reference. It contains all the node ids used in the dataset.
2. groups.csv: it's the file of all the groups. It contains all the group ids used in the dataset.
3. edges.csv: this is the friendship network among the users. The user's friends are represented using edges.
4. group-edges.csv: the user-group membership. In each line, the first entry represents user, and the 2nd entry is the group index.

3.4.1 RESULTS: YOUTUBE

group-edges.csv: size 460kb

Objective: to identify most popular node i.e. the node participating in max. No. Of groups and the largest group i.e. the group with most members.

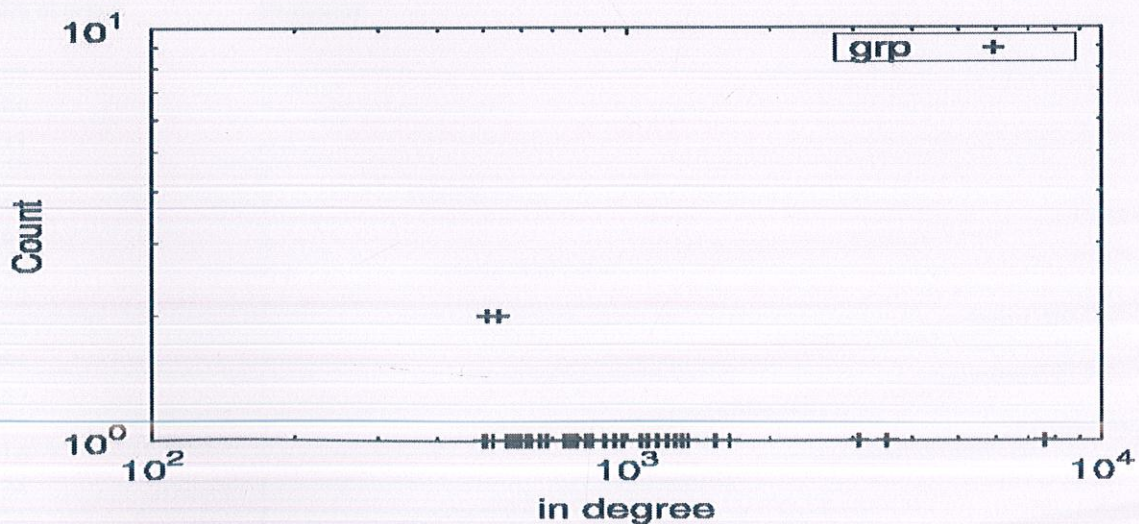


Fig6. In degree shows the no. of members in a group.

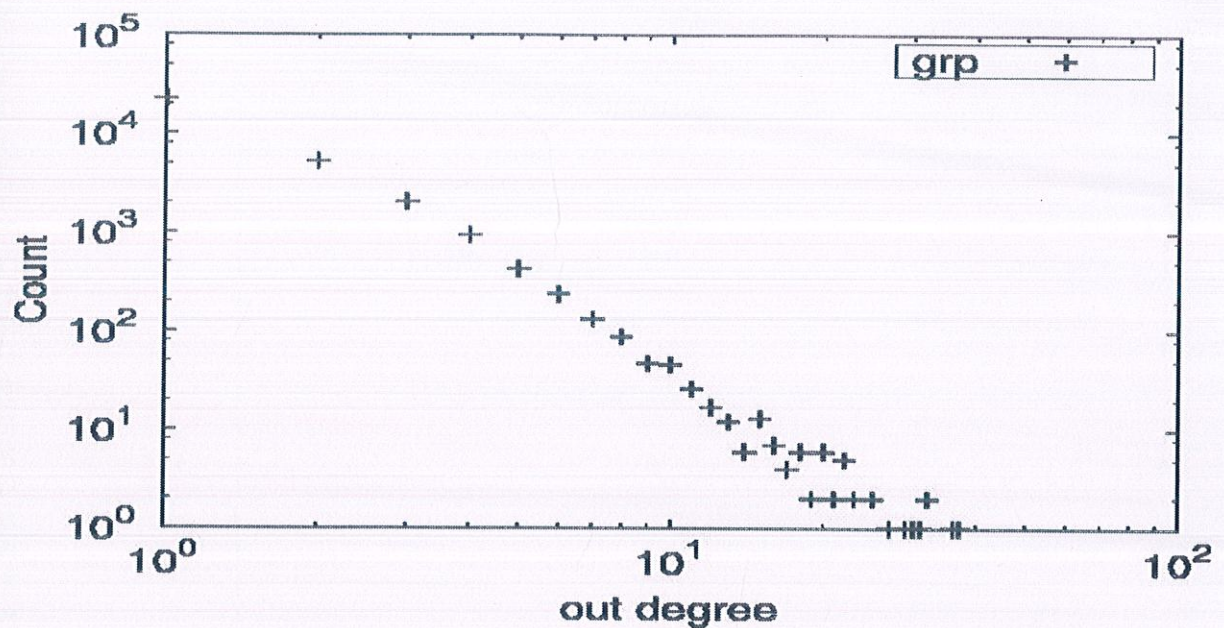


Fig7.Out degree shows the no. of groups a person is participating in.

From fig6, we learn that there are 3 distinct large groups of size between 3000 to 10000 with the largest in the region of 5000-10000. Before that the plot is fairly continuous i.e. the values are close to each other.. And the smallest groups are not smaller than size approx. 500. We also observe that two sizes have count >1 but both are towards lower end. When we actually see the results file, we find that:

Size of group	frequency
502	1
509	1
511	2
525	1
540	2
560	1
567	1
572	1
583	1
584	1
589	1
598	1
616	1
624	1
626	1
629	1
658	1
681	1

742	1
763	1
779	1
788	1
790	1
823	1
830	1
832	1
843	1
898	1
944	1
982	1
1071	1
1074	1
1104	1
1107	1
1108	1
1160	1
1218	1
1269	1
1310	1
1348	1
1539	1
1645	1
3083	1
3533	1
7583	1

Size of smallest group- 502

Size of largest group- 7583

Size of 2nd largest group-3533

Size of 3rd largest group-3083

2 group each with sizes 511 and 540.

The group with id 7 has highest degree of 7583.

From fig7, we observe that an extremely large no. of people belong to only one group. And the person which is the most popular i.e. belongs to most no. of groups has an out degree between 35 to 40. There are very few people who participate in more than 10 groups and even rarer are the ones who participate in more than 20 groups. From results file we observe:

No. of groups a person belongs to	frequency
1	22374
2	5261
3	2033
4	942
5	426
6	237
7	132
8	87
9	47
10	46
11	26
12	17
13	12
14	6
15	13
16	7
17	4
18	6
19	2
20	6
21	2
22	5
23	2
25	2
27	1
29	1
30	1
31	1
32	2
36	1
37	1

22374 belong to only one group.

The user which belongs to most no. of groups is a member of 37 groups and has group id 10553.

edges.csv: size 36.9 mb

Objective: to find connected components in the social networking graphs, which will help in giving grind suggestions because the people in connected to each other in some way even if not directly. They may become friends in future. Also to identify the most popular nodes which have high in degree and out degree? In degree and out degree:

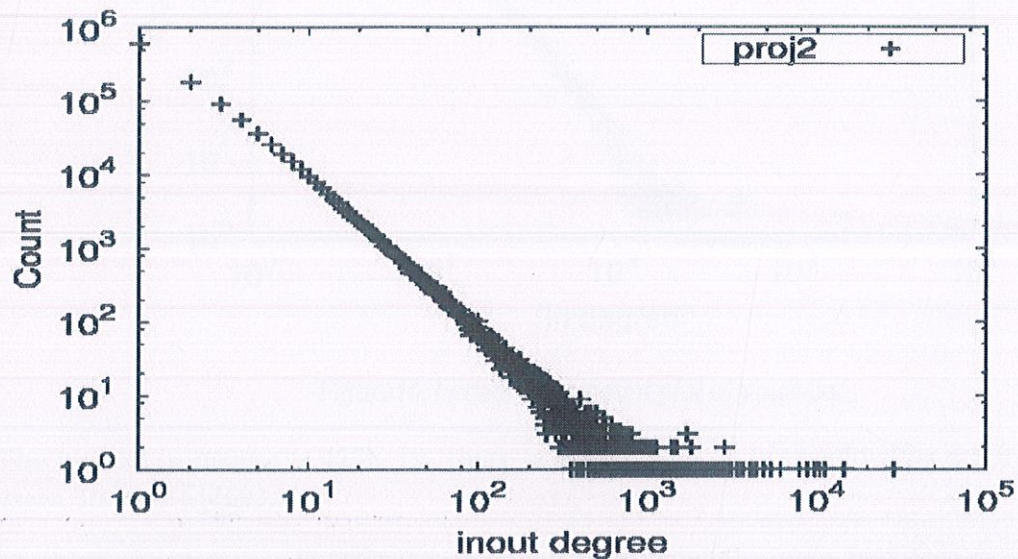


Figure8. inout degree vs count plot of youtube.

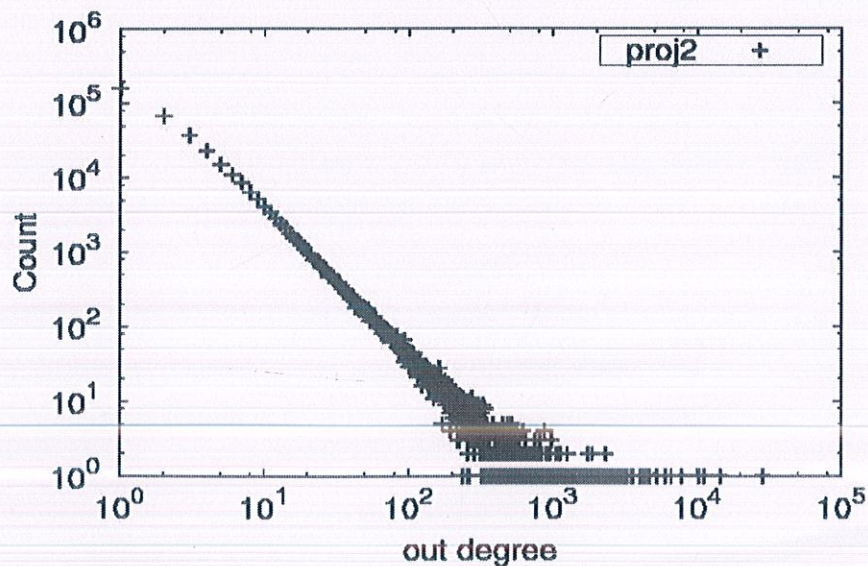


Figure 9. out degree vs count plot of youtube.

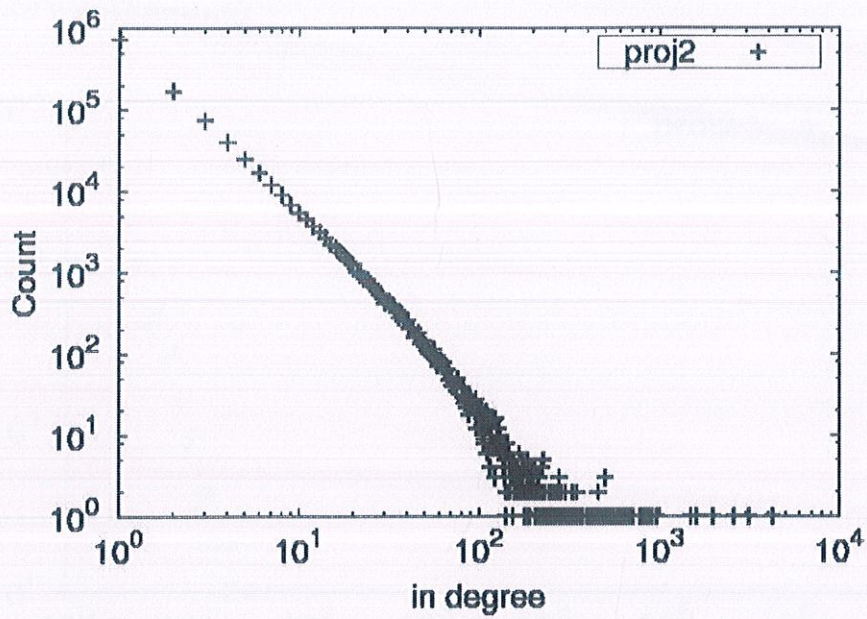


Figure10. in degree vs count plot of youtube.

The highest in degree is 4256. i.e. there is a user who has been added as a friend by 4256 users. Its id is 644603.

The highest out degree is 28576 i.e. there is a user who has added 28576 users as his friend. Its id is 1072.

Connected components

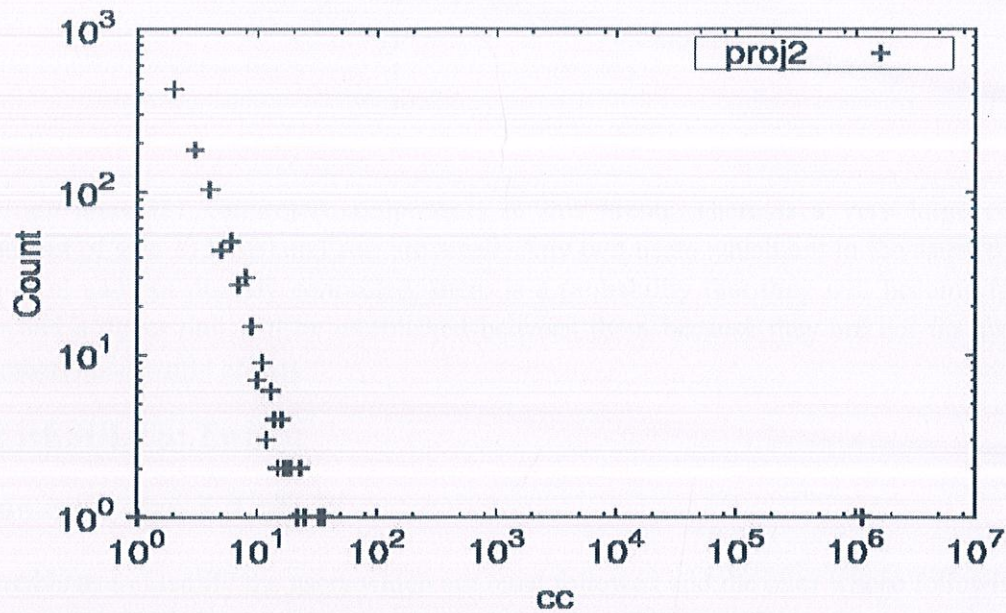


Figure11. Size of connected component vs count plot of youtube

We observe that there is a single very large connected component and lot of small connected components. From results file:

Size of connected component	frequency
1	1
2	430
3	182
4	104
5	44
6	50
7	27
8	30
9	15
10	7
11	9
12	3
13	6
14	4
15	2
16	4
17	2
18	2

19	2
22	1
23	2
25	1
33	1
36	1
1134890	1

There are total 931 connected components in this graph. There is a very large connected component of size 1134890 and rest are small. Any two users which are in the same connected component and not directly connected, there is a probability that they will become friends in future and a direct link will be established between them because they are not far away from each other (small world effect).

3.4.2 RESULTS: Twitter

edges.csv: size 1.1.gb

Objective: to identify the users which are most followed and the ones whose following count is the highest. And also to observe why it does not make much sense to identify connected components in this type of graph.

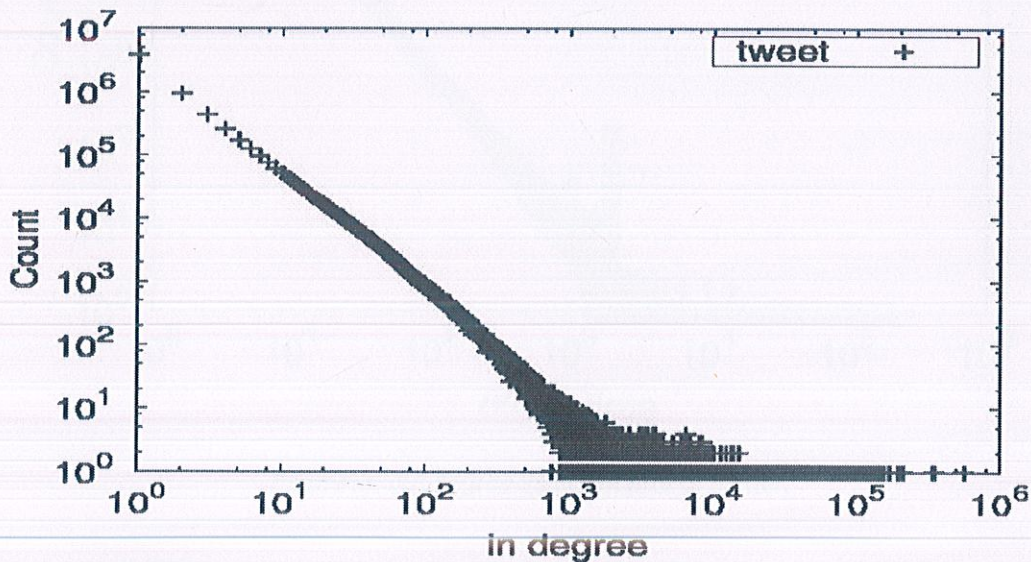


Figure12. in degree vs count plot of twitter.

In degree shows how many users follow a particular user. We observe that a large no. of users (between 1000000 and 10000000) have in degree 1. Then the pattern follows a linear curve till

the in degree of 1000 where we can say that we start seeing popular users. The count of such popular users remains steady till about in degree 10000. After that the in degree drops really low and we start observing celebrities but the plot is still continuous. After the in degree of 100000, we see users which are exceptionally popular worldwide and are very few in no. We observe that the user which is most followed has in degree close to 5 lacs. From the results file we observe:

Most popular user is followed by 564512 users.

2nd most popular user is followed by 350885 users.

37, 76,805 users have in degree 1.

Id 5994113 has the maximum in degree 564512.

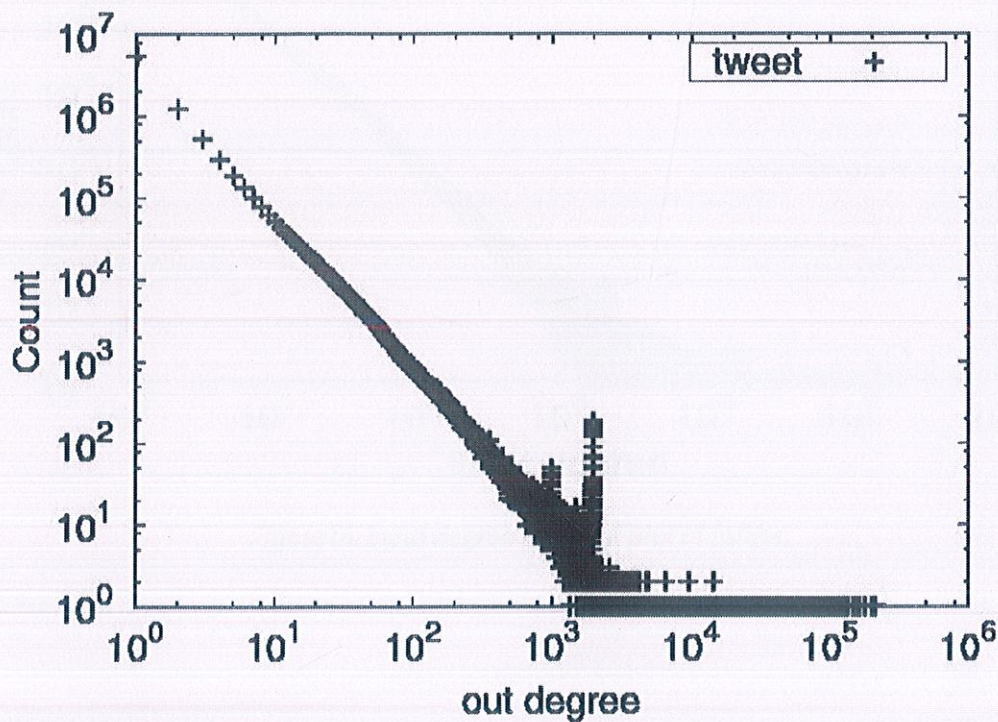


Figure13. out degree vs count plot of twitter.

Out degree shows the no. Of people a user follows. From graph, we can say that close to 10000000 users have out degree 1. Then the slope follows a linearly downward slope till around out degree 1000, where we see an aberration. Then we observe a straight line with count =1 till

we observe the user with highest out degree which is close to 2 lacks. From results file we observe that:

- The highest out degree is 2, 14,381.
- 54, 59,717 users have out degree 1.
- The aberration comes close to out degree 2000 where we observe count between 150 and 200.
- Id 3493 has the maximum out degree 214381.

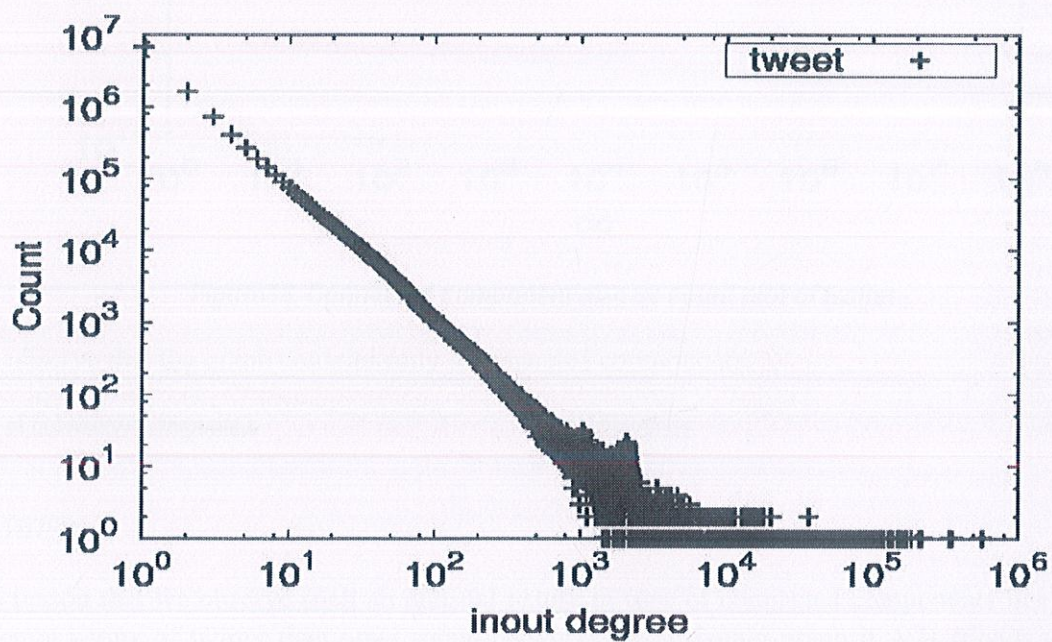


Figure14. inout degree vs count plot of twitter.

Connected components

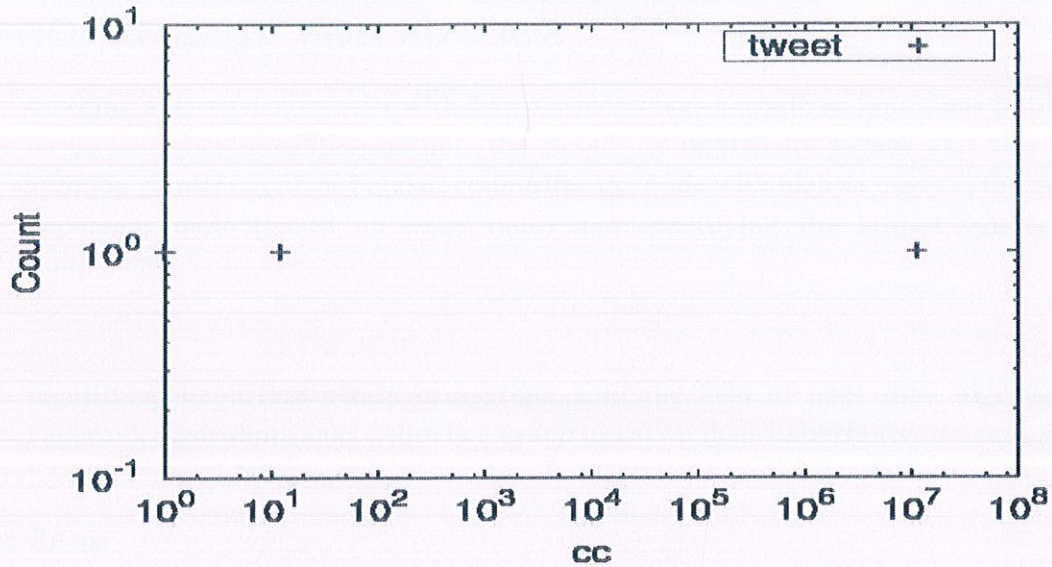


Figure15. Connected component size vs count plot of twitter

We observe that the graph contains only 3 connected components.

Size of connected component	frequency
1	1
12	1
11316799	1

The reason that we observe such an unusual nature in case of connected components lies in the different nature of twitter than other social networks. In any other graph if A is friends with B and C is also friends with B then A and C will be considered very close to each other and there will be a strong possibility that they will become friends in future. On the other hand in Twitter if A follows a famous personality B and C also follows the personality B then they will get connected even if the possibility of either following the other is extremely low. In other words in twitter the links are made very easily and almost all users end up being connected in one way or the other in a very large connected component. To predict which user will follow which in future we will have to identify closely connected cliques in the large connected component.

CHAPTER 4

IMPROVEMENTS FOR FUTURE

1. Building a user friendly GUI with JAVA which takes a graph as input, has buttons for various functionalities, outputs the Results computed by pegasus and also the addition results based on Pegasus results like the node with highest degree, the most important node (based on page rank) and identifying the largest connected components.
2. Identifying important actors in a graph with the help of page rank algorithm. Pagerank algorithm ranks nodes in a graph based on their importance.

Page Rank

For link analysis we assign to every node in the web graph a numerical score between 0 and 1, known as its *Page Rank*. The Page Rank of a node will depend on the link structure of the web graph. Given a query, a web search engine computes a composite score for each web page that combines hundreds of features such as cosine similarity and term proximity, together with the Page Rank score. This composite score is used to provide a ranked list of results for the query.

Consider a random surfer who begins at a web page (a node of the web graph) and executes a random walk on the Web as follows. At each time step, the surfer proceeds from his current page A to a randomly chosen web page that A hyperlinks to. Figure 3 shows the surfer at a node A, out of which there are three hyperlinks to nodes B, C and D; the surfer proceeds at the next time step to one of these three nodes, with equal probabilities $1/3$.

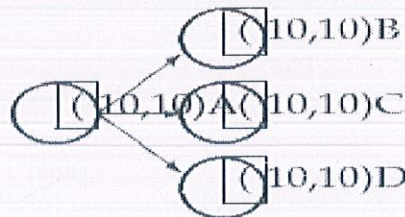


Figure 16: The random surfer at node A proceeds with probability $1/3$ to each of B, C and D.

As the surfer proceeds in this random walk from node to node, he visits some nodes more often than others; intuitively, these are nodes with many links coming in from other frequently visited nodes. The idea behind Page Rank is that pages visited more often in this walk are more important.

What if the current location of the surfer, the node A, has no out-links? To address this we introduce an additional operation for our random surfer: the *teleport* operation. In the teleport

operation the surfer jumps from a node to any other node in the web graph. This could happen because he types an address into the URL bar of his browser. The destination of a teleport operation is modelled as being chosen uniformly at random from all web pages. In other words, if N is the total number of nodes in the web graph, the teleport operation takes the surfer to each node with probability $1/N$. The surfer would also teleport to his present position with probability $1/N$.

In assigning a Page Rank score to each node of the web graph, we use the teleport operation in two ways:

- (1) When at a node with no out-links, the surfer invokes the teleport operation.
- (2) At any node that has outgoing links, the surfer invokes the teleport operation with probability $0 < \alpha < 1$ and the standard random walk (follow an out-link chosen uniformly at random as in Figure 3) with probability $1 - \alpha$, where α is a fixed parameter chosen in advance. Typically, α might be 0.1.

When the surfer follows this combined process (random walk plus teleport) he visits each node v of the web graph a fixed fraction of the time $f(v)$ that depends on

1. the structure of the web graph and
2. the value of α . We call this value $f(v)$ the Page Rank of v .

Finding Pagerank in Pegasus

- copy the graph edge file to the HDFS directory `pr_edge`
- execute `./run_prblk.sh`

The range of node id is from 0 to `number_of_nodes_in_graph - 1`.

The syntax of `run_prblk.sh` is:

```
./run_prblk.sh [#_of_nodes] [#_of_reducers] [HDFS edge path] [makesym or nosym]
[block width]
```

where

`[#_of_nodes]`: number of nodes in the graph
`[#_of_reducers]`: number of reducers to use in hadoop.

- The number of reducers to use depends on the setting of the hadoop cluster.
- The rule of thumb is to use $(\text{number_of_machine} * 2)$ as the number of reducers.

`[HDFS edge_file_path]`: HDFS directory where edge file is located
`[makesym or nosym]`: makesym-duplicate reverse edges, nosym-use original edge file

- When the input graph is directed and you want to calculate directed PageRank, and then use 'nosym' in the 4th parameter.
- When the input graph is directed and you want to calculate undirected PageRank, and then use 'makesym' in the 4th parameter.
- When the input graph is undirected, use 'nosym' in the 4th parameter.

`[block_width]`: block width, usually set to 16.

Ex: `./run_prblk.sh 16 3 pr_edge makesym 2`

The output of Page Rank is saved in the following HDFS directory:

pr_vector :

- Each line contains the Page Rank of each node in the format of (nodeid TAB "v"PageRank_of_the_node).
- For example, the line "1 v0.10231778333763829" means that the Page Rank of node 1 is 0.10231778333763829.

pr_minmax: The minimum and the maximum Page Rank.

- The minimum Page Rank is the second column of the line that starts with "0".
- The maximum Page Rank is the second column of the line that starts with "1".

pr_distr:

The histogram of Page Rank, it divides the range of (min_PageRank, max_PageRank) into 1000 bins and shows the number of nodes which have Page Ranks that belong to such bins.

Example: finding page rank of a test graph:

0	1
1	2
1	3
3	4
3	6
5	6
6	7
6	8
6	9
10	11
10	12
10	13
10	14
10	15

PR_DISTR:

1	4
4	1
9	2
16	5
418	1
463	1
893	1
1000	1

PR_MINMAX:

0	0.0368153863548244
1	0.18009901942165274

PR_VECTOR:

0	v0.03801584360515127
1	v0.10303820070044137
2	v0.03801584360515127
3	v0.09660774794874713
4	v0.03730713379139067
5	v0.0368153863548244
6	v0.16475368492982068
7	v0.0368153863548244
8	v0.0368153863548244
9	v0.0368153863548244
10	v0.18009901942165274
11	v0.038980196115669455
12	v0.038980196115669455
13	v0.038980196115669455
14	v0.038980196115669455
15	v0.038980196115669455

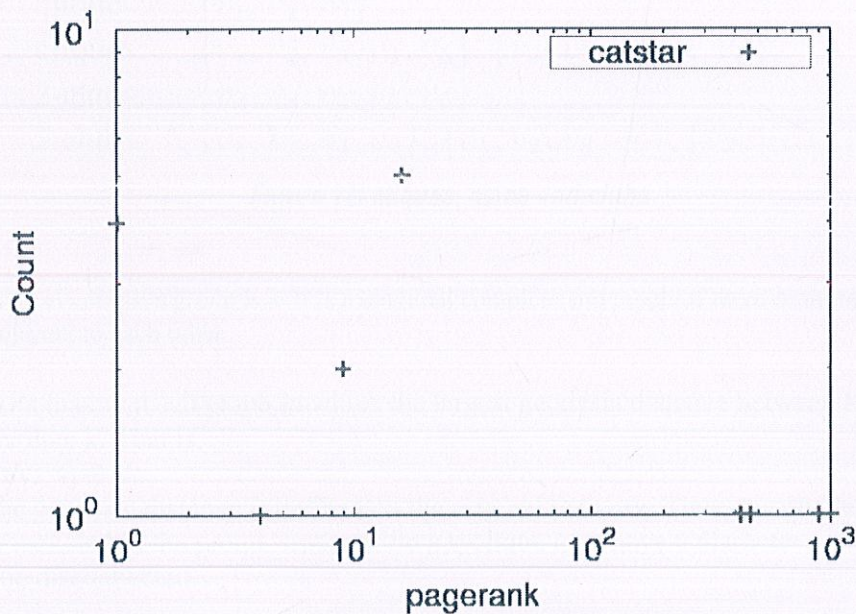
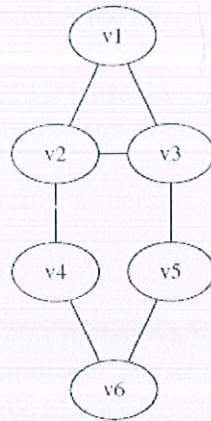


Figure17. page rank plot

2. Finding closely connected components in a connected component and predicting which links will be formed in future accurately.

We have seen that sometimes the size of connected components is very large and it is very difficult to say whether two nodes will connect in future or not. If they are far off from one another and the distance between them is quite large then the possibility that there will be a node

between them in future is low even if they are in the same connected component. So, to predict whether an edge will be formed we should first identify cliques, near cliques and triangles.



- cliques: $\{v_1, v_2, v_3\}$
 2-cliques: $\{v_1, v_2, v_3, v_4, v_5\}, \{v_2, v_3, v_4, v_5, v_6\}$
 2-clans: $\{v_2, v_3, v_4, v_5, v_6\}$
 2-clubs: $\{v_1, v_2, v_3, v_4\}, \{v_1, v_2, v_3, v_5\}, \{v_2, v_3, v_4, v_5, v_6\}$

Figure 18: cliques, clans and clubs

clique An ideal cohesive group is a. It is a maximal complete sub graph of three or more nodes all of which are adjacent to each other.

k-clique is a maximal sub graph in which the largest geodesic distance between any two nodes is no greater than k. That is,
 $d(i, j) \leq k \forall v_i, v_j \in V_s$

Note that the geodesic distance is defined on the original network. Thus, the geodesic is not necessarily included in the group structure. So a k-clique may have a diameter greater than k or even become disconnected.

k-clan is a k-clique in which the geodesic distance $d(i, j)$ between all nodes in the sub graph is no greater than k for all paths within the sub-graph. A k-clan must be a k-clique, but it is not so vice versa. For instance, $\{v_1, v_2, v_3, v_4, v_5\}$ in Figure is a 2-clique, but not 2-clanas the geodesic distance of v4 and v5 is 2 in the original network, but 3 in the sub graph.

k-club restricts the geodesic distance within the group to be no greater than k. It is a maximal substructure of diameter k. All k-clans are k-cliques, and k-clubs are normally contained within k-cliques. These substructures are useful in the study of information diffusion and influence propagation.

Finding all these tight communities is easy if the size of graph is small. But when we talk about Social networking graphs the size of graph is in GBs so the existing methods do not scale well. We address this problem with the using HEIGEN algorithm developed by U Kang Brendan Meeder Christos Faloutsos of Carnegie Mellon University, School of Computer Science which is designed to be accurate, efficient, and able to run on the highly scalable MAPREDUCE (HADOOP) environment. This enables HEIGEN to handle matrices more than 1000* larger than those which can be analyzed by existing algorithms.

3. Making use of link types for better understanding of patterns of friendship and community structures.

Social Networks are platforms that allow people to publish details about themselves and to connect to other members of the network through links. Recently, the population of such online social networks has increased significantly. For instance, Facebook now has over 150 million users. Facebook is only one example of a social network that is for general connectivity. Other instances of social networks are LinkedIn, Last.fm, orkut, aNobii, and the list continues.

These networks allow users to list details about themselves, but also allow them to not specify details about themselves. However, these hidden details can be important in the administration of a social network. Most of these sites are free to the end user and support advertising. If we assume that advertisers want to reach the people most likely to be interested in their products, then identifying those individuals becomes a priority for maintaining much-needed advertisers. However, by just using specifically defined information, the site may be missing a large number of potentially interested users. Taking the specified knowledge from some users and using it to infer unspecified data may allow the site to extend its target audience for particular advertisements.

The implications of classification in social network data extend far beyond the simple case of targeted advertising. For instance, such ideas could be used for addressing classification problems in terrorist networks. By using the link structure and link types among nodes in a social network with known terrorist nodes, we can attempt to classify unknown nodes as terrorist or non-terrorist. In such a classification process, the type of the link shared among nodes could be really critical in determining the final success of the classifier. For example, assume that there exist two different individuals Ram and Rob that are linked to some known terrorist, Mack. Furthermore, assume that Ram works at the same place as Mack, but they are not friends. (i.e., Ram and Mack have a relationship. The type of relationship is "coworker"). In addition assume that Mack talks frequently on the phone with Rob and they are friends (i.e., Rob is linked to Mack and the link type is "friend"). Given such a social network, to our knowledge, all the existing classification methods just use the fact that individuals are linked and they do not use the information hidden in the link types. In some cases, one link type like friendship can be more important than other link types for increasing the accuracy of the classification process. Therefore, social network classification techniques that consider link types could be very useful for many classification tasks.

REFERENCES:

❖ -Spectral Analysis for Billion-Scale Graphs: Discoveries and Implementation

U Kang, Brendan Meeder, Christos Faloutsos Carnegie Mellon University, School of Computer Science

{ukang,bmeeder,christos}@cs.cmu.edu

❖ PEGASUS: A Peta-Scale Graph Mining System - Implementation and Observations

U Kang *SCS, Carnegie Mellon University* ukang@cs.cmu.edu

Charalampos E. Tsourakakis *SCS, Carnegie Mellon University* ctsourak@cs.cmu.edu

Christos Faloutsos *SCS, Carnegie Mellon University* christos@cs.cmu.edu

❖ PEGASUS: Mining Peta-Scale Graphs

U Kang, Charalampos E. Tsourakakis, and Christos Faloutsos

School of Computer Science, Carnegie Mellon University, Pittsburgh PA, USA

❖ PEGASUS User's Guide

❖ GRAPH MINING APPLICATIONS TO SOCIAL NETWORK ANALYSIS

Lei Tang and Huan Liu

Computer Science & Engineering

Arizona State University

L.Tang@asu.edu, Huan.Liu@asu.edu

❖ Social Network Classification Incorporating Link Type Values

Raymond Heatherly Jonsson School of Engineering and Computer Science

The University of Texas at Dallas Email: rdh061000@utdallas.edu

Murat Kantarcioglu Jonsson School of Engineering and Computer Science

The University of Texas at Dallas Email: muratk@utdallas.edu

Bhavani Thuraisingham Jonsson School of Engineering and Computer Science

The University of Texas at Dallas Email: bxk043000@utdallas.edu

❖ Opinion mining and sentiment analysis

Bo Pang *Yahoo! Research, 701 First Ave. Sunnyvale, CA 94089, U.S.A.,* bopang@yahoo-inc.com

Lillian Lee *Computer Science Department, Cornell University, Ithaca, NY 14853, U.S.A.,* llee@cs.cornell.edu

APPENDIX A: JAVA PROGRAM FOR IDENTIFYING THE NODES WITH HIGHEST DEGREE AND THE LARGEST CONNECTED COMPONENT.

```
import java.awt.BorderLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import javax.swing.JButton;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;

public class Calc extends JFrame implements ActionListener{
    JTextField t;
    public Calc(){
        super("Calculate Degree");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(400,300,400,200);
        t=new JTextField(20);
        t.setEditable(false);
        JButton browse=new JButton("Browse");
        JButton submit=new JButton("Submit");
        browse.addActionListener(this);
        submit.addActionListener(this);
        JPanel p=new JPanel();
        p.add(t);
        p.add(browse);
        this.add(p, BorderLayout.CENTER);
        this.add(submit, BorderLayout.SOUTH);
        setVisible(true);
    }
    public void actionPerformed(ActionEvent e) {
        if(e.getActionCommand().equals("Browse")){
            JFileChooser chooser=new JFileChooser();
            chooser.showOpenDialog(this);
            t.setText(chooser.getSelectedFile().getAbsolutePath());
        }
        if(e.getActionCommand().equals("Submit")){
            int max=0,id=0;
            try{
                BufferedReader cin=new BufferedReader(new
FileReader(t.getText()));
                String s=cin.readLine();
                String[] in=s.split("\\t");
```



```

max=Integer.parseInt(in[1]);id=1;
int temp;
while(true){
    s=cin.readLine();
    if(s==null) break;
    if(s.equals("") || s.length()==0) break;
    in=s.split("\t");
    System.out.println(s);
    temp=Integer.parseInt(in[1]);
    if(max<temp){ max=temp;id=Integer.parseInt(in[0]); }
}
}catch(IOException ioe){ioe.printStackTrace();}
JOptionPane.showMessageDialog(this, "Id "+id+" has the
maximum degree "+max,
    "Degree",JOptionPane.INFORMATION_MESSAGE);
}
}
public static void main(String[] args){
    new Calc();
}
}

```