Spring 2011

Users Positions in Social Networks

Jasim Qazi
San Jose State University

USER POSITIONS IN SOCIAL NETWORKS

A Project Report

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Computer Science

By

Jasim Qazi

January 2011

The Undersigned Project Committee Approves the Project Titled

USER POSITIONS IN SOCIAL NETWORKS

by

Jasim Qazi

Dr. T. Y. Lin	Department of Computer Science
Dr. Mark Stamp	Department of Computer Science
Faisal Faruqi	Nexscience Inc.

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

ABSTRACT

USER POSITIONS IN SOCIAL NETWORKS

BY Jasim Qazi

Social networks are a new phase in human interaction; using technology to connect people online, the social networks of today have become a central part of the lives of millions of people. People use social networks for sharing various information with their friends and family. This information can take the form of text, video, images, sound etc. and it is what forms the collection of data in social networks.

As social networks gain popularity and as more and more people start using social networks, it has become more important now to understand the inner structures of social networks and understand how the relationships between users are formed and what makes these relationships important.

In this paper we take a look at various simple score calculation schemes by which we can score and classify users based on their position and their actions in the social network. We also provide an application that uses the data available on Twitter to calculate how influential a user is in its social network.

LIST OF FIGURES

LIST OF TABLES

1.0 Introduction

Social networks have come a long way. They might have been used for simple messaging and group discussions until a few years back, but today, they have grown so complex that they are literally capable of tracing your life. Other than keeping a record of our personal information, likes and dislikes and friends, social networks hold opinions and thoughts that we sometimes choose to share in the form of status messages, images etc.

All this data allows us to take a closer look and understand the nature and depth of relationship between two individuals might have. Furthermore, this data gives us an insight into how people use social networks and what factors are involved in shaping the structure of the social network. Social networks can be represented as graphs and we need to figure out how the links and nodes in the graph are placed based on the relationships of the users in the network to not only understand human behavior but also find out ways that the data available can be used to get useful information.

1.1 Objective of the Paper

The goal of this paper is to find ways of how we can identify what elements are responsible for the structure of social graphs of non-traditional and commercial social networks. We believe that social importance of actors (nodes) dictate how it is positioned in a network and the kind of relationships it can have. We look at the kind of factors which can contribute to the

importance of a node:

We provide a simple score calculation scheme where we look at the different interactions nodes have with their neighbors and we present a way to classify nodes based on their importance level. This classification can be used to control privacy in the network and also find out how influence a person is in the network.

1.2 Organization of the Paper

In section 2 we present some basic information about social network, the networking graphs and some of the terms that are used to define position of nodes in the network. These terms would be used later in our explanations.

Section 3 looks at some simple scoring schemes that can be used to organize users numerically according to their importance in the network.

1.3 Graph Theory

In mathematics, a graph is a data structure that captures the notion of connection and is used to represent connections between entities or objects in a collection. Object in graphs are called vertices and the links between multiple objects are called edges. A combination of edges connected through vertices makes up a path from the start and end vertices. Paths may be finite

or infinite. Finite paths always have a start vertex, and a last vertex, called its end vertex. A cycle is a path such that the start vertex and end vertex are the same.

The links or paths in a graph may be undirected or directed. The direction of the path depends on the nature of the link between the nodes. A representation of a graph of connected objects a, b, c and d is given below:

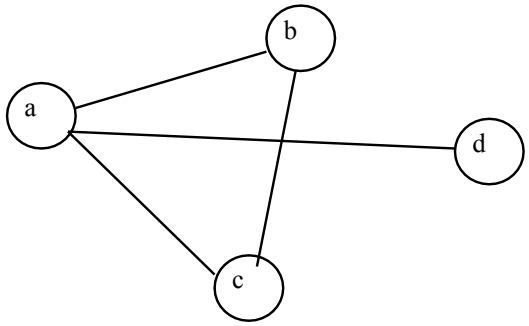

Figure 1 Graph

The above image shows four vertices (a,b,c,d) connected to each other. The lines represent the edges in the graph.

A graph can be formally defined as a pair of sets V and E together with a function f : E -> V x V . The elements of V are the vertices of the graph. The elements of E are the edges of the graph.

1.3.1 Social Network as Graph

Social networks consist of people connected to each other based on relationships. A user may have connections with multiple users in the network

and the type of relationship. In other words, a social network is a network of interconnected nodes. This network can be represented using graphs. If we try to visually conceive a social network, the actors or users in the network can be represented as nodes in a graph. The links or relationships between the actors can be shown using edges connecting the nodes. Say we have four actors or users, a, b, c and d in the network. These users have the 'is a friend' relationship. User a is connected to user b, c and d. User b is connected to a and c, c is connected to a and b, and d is connected to a only. To display this information visually, we would have nodes or vertices for each actor, and the is-a-friend relationship would be represented by drawing an edge between the vertices. For this scenario, we would have the exact arrangement as given in figure 1.

1.4 Binary Relations

In social networks, the connection between two people can be represented using binary relations. Binary Relations provide a mathematical view to organization and connection of pairs. Let X and Y be two sets. A relation R, from X to Y is a subset of the Cartesian product XxY, which gives us a collection of ordered pairs. Let x be an element of X and y be an element of Y. The notations (x, y) is an element of R and x R y (say x is in relation R to y, in graphical language x -- > y) are equivalent [26]. This means that a binary relation between two sets X and Y is a subset of XxY, which gives all the possible ordered pairs. Each of the pair represents a pair nodes with a

connecting edge, or link in between them.

Since graphs can be used to show pair-wise relationships, we can represent binary relations in terms of graphs. Graphs can have an infinite number of paths between the vertices. In order to traverse these paths, one can go into an infinite loop as there are many recursions over the same paths over and over again. If we look at all the possible number of pairs that can be formed in a network, the number is equivalent to NxN, where N is the number of users in the network. This is because each user is able to form a connection with all users, including itself.

As can be imagined, this gives us a very large set of ordered pairs, where only a subset of ordered pairs are of value and represent 'real-life' connections. In social networks, we are concerned with only the important links in the social graph, that is, we want a subset of the N^2 relations in the universal set. By important we mean links that have an intuitive meaning and represent a real world relationship between two users. The set of all the possible paths in a social network is the Universe of Discourse of our topic. Here, a path we mean a bag [31] of connected edges. A graph (binary relation) can have an infinite number of paths. In order to traverse the network, one can go into an infinite loop as there are many recursions over the same paths over and over again. So for real world path analysis, we are only concerned with a subset of this larger set.

For example, if we keep the universal set of users consistent, then we

would have two different subsets for Facebook and Twitter. The classifications and relations in these subsets depend on the identifying properties or relationships in each social network. This means that we look at the subset of ordered pairs that follow the relationships which the social network supports and its data is based on.

1.5 Neighborhood Systems

A neighborhood system is a basic concept in topological space. If *X* is a topological space and *p* is a point in *X*, a neighborhood of *p* is a set *V*, which contains an open set *U* containing *p*, [29]

$$p \in U \subseteq V.$$

Note that the neighborhood *V* need not be an open set itself. If *V* is open it is called an open neighborhood [29]. The collection of all neighborhoods of a point is called the neighborhood system at the point [29].

In this paper, we apply the concept of neighborhood system to a binary relation R, namely, at each point, we assign only one subset (that can be an empty set) [30]. A (right) neighborhood of a point p is a set containing the points that are related to p, that is, N(p) = {x | (p, x) R}. So a binary relation can be expressed by a neighborhood system; we will call it a binary neighborhood system.

Conversely, for each node p in U, a binary neighborhood system p ->

N(p) where N(p) is a subset ,called a neighborhood. The neighborhood N(p) can be interpreted as the right and left neighborhood. It can determine a binary relation

R = {(p, x) | p U, x N(p) } (or L = {(y, p) | p U, y N(p) })

These two neighborhoods can be used to define the relationship directions in binary relations. The right neighborhood gives us a collection of all the nodes which are the sources of data, and have out going links. The left neighborhood contains all the nodes where the direction of the link or edge is coming in, or all the nodes which are recipients in the relationship.

If we look at the resulting data from left and right neighborhoods, we can see that these are both equal to the indegree and outdegree of binary relations. Thus we can see that social graphs can be given a visual representation in terms of neighborhoods as well where it becomes much easier to classify users according the nature of the links that connect them.

1.5.1 Visual Representation

Binary relations and binary neighborhood system can be represented on a visual grid. This is similar to the way graphs are displayed. The universal set of all the relations in the graph can be represented as indices on the grid. Edges or connections between two vertices or nodes can be represented as a dot on the plane. The location of the dot depends on the corresponding

position of the involved vertices on the indices. The choice of axis used to show a connection for two nodes depends on the direction of the link between them. When we're talking about social networks, the link represents the relationship between the nodes or actors in the social network. The direction of the link would represent the flow of data from one user to another.

Lets take an example of set of users A, B and C which belong to a social network. If we are to display their relationship in a graph, first we would have to look at the direction of the links that they have. Lets say that A and B are connected and the direction is from A to B, meaning that data flows from user A to user B. On the graph, the whole user set is shown on the grid so that we can identify relationships between all users in our network. This is shown in figure 2.

Figure 2 User connection graph

Now, for our first case, as the direction of the connection is from A to B, we choose A on the x axis and find B on the y axis. The connection between the two would be shown by putting a dot on the position (A,B) on the graph plan (figure 1). In this arrangement, the data initiator is on the x-axis and the receiver is on the y-axis.

Similarly, if C is connected to B (C->B) then the dot representing this link is at point (C,B).

Looking at figure 2, if we are to find the Indegree of a user based on the graph, a vertical line drawn on the x axis for a particular user gives us the Indegree of that user. In other words, set of all connections that a user has where the user is the source of data is called the Indegree of that user. This is also the left neighborhood.

Similarly, if we take the set of all the connections where a user is the recipient of data, we get the outdegree of that user. This is represented on the graph by a horizontal line on the y-axis, covering all the points where the user is the recipient. This represents the right neighborhood

1.6 Equivalence Relations and Partitions

Binary relations are called equivalence relations if they have the following three basic relationship types:

1. A relation, R, on X is reflexive if x R x for all x in X.
2. A relation, R, on X is symmetric if x R y implies that y R x.
3. A relation, R, on X is transitive if x R y and y R z imply that x R z

[26]

The binary neighborhood system induced by an equivalence relation partitions a set into disjoint neighborhoods. In such a case, the right neighborhood is equal to the left neighborhood and is called equivalence classes. All the elements in a given equivalence class are equivalent among themselves, and no element is equivalent with any element from a different class. If we look at these properties in view of current social networks, we are able to identify groups of equivalent or identical users. These identical groups (or pairs) should have all the three properties of being symmetric, reflexive and transitive.

1.7 Real World Examples

1.7.1 Facebook

Facebook users can be connected through various relationships, however, all the connections in Facebook are bidirectional in nature. A connection in Faceobook is defined by the property of being a 'friend' f someone.

If we consider the property of being a friend a relationship, then if user A is a friend of user B, then user B is also a friend of user A. This shows that user relations have a symmetric property.

Facebook clearly does not strictly follow the transitive property as users may or may not be friends of their friends. As for reflexive, all users are their own friends as they have access to all the data that the friends have access to. It has to be stressed that the 'relationship' that we define here is the connection that people have between them.

1.7.2 Twitter

There are two types of relationships in Twitter: friends and followers

where followers receive information from their friends (or the people that they are following).

If user A is a friend of (or is following) user B (A R B -> A fr B) then user B receives data (or tweets) from user A. If user A is a follower of user B (A R B -> A fl B) then user A receives data from user B.

Neither of these properties are symmetric as none require the other one to be valid for a set of users. Also, neither of the relationships are transitive. Both of the properties are reflexive for the same reason given for Facebook.

2.0 Social Networks

In social network analysis, the observed attributes of social actors are understood in terms of patterns or relationships between units. Relational ties among actors are primary and attributes of actors are secondary [1].

Some of the types of relationship-attributes between actors in social networks are

1. Family member
2. Kinship
3. Gender
4. Age
5. Nationality
6. Hometown

7. Religion
8. Location
9. Having a ‘Friends’ relationship
10. Belonging to the same club, company, organization etc.
11. Attending the same event
12. Interested in the same items (books, movies, shows etc)

These attributes give us indicators of what factors affect how social graphs are laid out. It is seen in a Facebook study [4] that people tend to ‘follow’ or connect to people that they feel close too. The closeness between two nodes or actors can be expressed in how important one node thinks the other is. An actor x connects to actor y based on the importance of actor y in the social circle or network of actor x. Actor y, on the other hand can belong to multiple sub-networks based around identifying attributes (gender, age, race, religion etc.) and can have similar connections to nodes in these networks, and in turn have a certain importance associated with these nodes. So it would be safe to say that actor y carries an accumulated importance rating which determines its position in the social network. The more important or prestigious actor y gets, the more significant it becomes in shaping the structure of the social graph and the connections of its direct neighbors.

Significance of a node in the graph can be explained with two properties: Centrality and Prestige. Both of them consider how prominent a user is within a community [3] or within a network by summarizing structural relations among the nodes.

2.1 Centrality

Numerous methods to measure centrality have been proposed. The method used to calculate or get a quantitative analysis of centrality for a network depends on the type of flow processes that are involved in the system. Flow processes indicate the nature of the data that is begin transferred between nodes. The measure of the centrality is based on the characteristics of these processes.

For example, some measures, such as Freeman's closeness and between-ness (Freeman, 1979), count only geodesic paths, apparently assuming that whatever flows through the network moves only along the shortest possible paths. Other measures, such as flow betweenness (Freeman et al, 1991) do not assume shortest paths, but do assume proper paths in which no node is visited more than once. Still other measures, such as Bonacich's (1987; 1992) eigenvector centrality and Katz's (1953) influence, count walks, which assume that trajectories can not only be circuitous, but revisit nodes and lines multiple times along the way. Regardless of trajectory, some measures (e.g., betweenness) assume that what flows from node to node is indivisible (like a package) and must take one path or another, whereas other measures (e.g., eigenvector) assume multiple "paths" simultaneously (like information or infections).

An actor with high degree centrality maintains numerous contacts with

other network actors. A central actor occupies a structural position (network location) that serves as a source or conduit for larger volumes of information exchange and other resource transactions with other actors.

A Twitter user with many friends, or people it follows, sits in the middle of the network and has access to a lot of information from his/her friends. Such a user has a lot of connections and receives information from the people he/she follows.

2.2 Prestige

This is the measure of how many directional ties an actor has, but not many relations. Twitter users who have high number of followers have high prestige. There is a large flow of information from the user to other nodes so that adds a value to the user's position, hence a higher prestige. Users with higher prestige have the power to shape the network. Because of the higher prestige, other nodes in the network find it desirable to be 'close' to such nodes.

2.3 Indegree Centrality or Degree Prestige

Determines the importance of an actor based on how many direct neighbors it has out of the whole network. It can also be defined as how many actors vote for a particular actor, say x. x does not need to know who its

neighbors are. This scenario can be witnessed in Twitter where a person does not really care who can follow them.

$$DP(x) = n(x) / m - 1$$

Where x is an actor, n(x) is x's direct, first level neighbors, and m is the total number of actors in the social network.

In case of our Person Ranking scenario n(professor) would be the number of professors that refer to professor x, and m is the total number of professors in the system. The higher the degree prestige, the more important the person is in the whole system [2].

2.4 Proximity Prestige

This is a measure of how close nodes inside the network are to a particular node [1]. The Proximity Prestige uses the graph distance between nodes in the network. Actors are judged to be prestigious based on how close or proximate the other actors in the set of actors are to them. The problem with proximity prestige is that it does not take into account the importance of a node's neighbors. The neighbors, or nodes that are being considered in the calculation of the proximity prestige, should also be considered and their prestige should also be taken into account. If many prestigious actors choose

an actor, then that should be given more weight than if many non-prestigious actors choose an actor.

2.5 Rank Prestige

This caters to the problem with Proximity Prestige. The prestige depends not only on the distance on the graph between the different nodes, but also on the individual ranking of the nodes that are being considered in the network. “It’s not what you know but whom you know” [1]. The Prestige, or importance of a node should depend on the quantity and quality of the connections and neighbors it has in the network. The quantity is a count of the number of connections whereas the quality can be attributed to a number of factors which contribute to the image/reputation of the node in front of its neighbors.

3.0 Scoring Schemes

In this section we look at Person and Interaction score. The Person score is based on the user’s position in the network and represents the score of the user with respect to other users in the network, whereas the Interaction Score gives us a score, or numerical classification of users based on the types of interactions they have with their neighbors, and we show how this score can be used to handle privacy issues in networks.

3.1 Person Score

In this section we look at how the importance of an actor in the node can be calculated using a simple scheme. Importance or prominence of an actor can be determined by how many connections an actor has in the network, i.e. the 'degree' of the node. The degree of a node surely gives indications of its dominance in the network, but when we talk about ‘social’ networks, I believe that not only the number of connections, but also the type of connections should be taken into account. Since the node belongs to several sub-networks, the ‘importance rating’, described above, that the node accumulates from each of its network, should also be taken into account. We shall call this rating a node’s score. The total score of a person would be the sum of all the scores that the person gets from its networks, which is the score of all the nodes in those networks. So we can say that the score of node x depends on the sum of the nodes of all of its neighbors (formula 1):

$$S(x) = S(N1) + S(N2) + S(N3) + \ldots + S(Nn)$$

However, each neighbor has multiple connections and it contributes equally (?) to each of neighbor’s score. So the score that N1 contributes to x is N1/ (number of x’s neighbors). So we get (formula 2):

$$S(x) = S(N1)/C(N1) + ... + S(Nn)/C(Nn)$$

Where C(N1) is the count of all the neighbors of N1.

The structure of this formula is based off very basic scoring or ranking algorithms used to rank and index web pages in web search engines, called PageRank. The same concept of assigning a score to individual web pages is used to index web pages to show meaningful, relevant and effective search results to users using search engines.

The basic concept behind PageRank is that if page A has a link to page B, then the author of A is implicitly conferring some importance to page B [4]. Similarly, each page that links to page B is in fact adding to the overall importance of page B. Logically, this is valid; if a page is linked by or referred by a large number of pages, chances are that the content on that page is important. How much importance or score page A gives to page B depends on page A's own score. This score is dependent on the sum of all the scores of pages linking to page A.

This concept, as mentioned above, is the same as Rank Prestige that we stated. The pages are similar to the nodes in the network or graph. The hyperlinks represent the links in the graph between nodes. We call this score the Person Score.

3.2 Calculation of Person Score

As mentioned in the Person Score section, the calculation of the Person Score of a person/node depends on the score of the neighbors. The score of those nodes depends on the scores of their neighbors. This would

keep on going in a large network. When our network is represented by a graph structure, we would need to have a start point and will have to assign some fixed value to some nodes to give the network score propagation a ‘kick-start’. Once the initial values are assigned, the scores can be calculated for the neighboring nodes, and so on.

When we look at social networks, all the connections in Facebook are bi-directional. If a user is ‘friends’ with another user, the other user should also be a friend of the first user (Figure 3). What this means is that if we traverse through the social graph, we might face infinite recursions, and it might be difficult to converge the graph to a stable state. However, relatively few nodes take a much longer time to converge. Furthermore, it is observed that these slow-converging nodes are generally those nodes with a high score [5]. In Twitter, the graph structure is a little different as there is no bi-directional relationship requirement. However, there might be cases where two users are related to each other and have a bi-directional relationship (Figure 4).

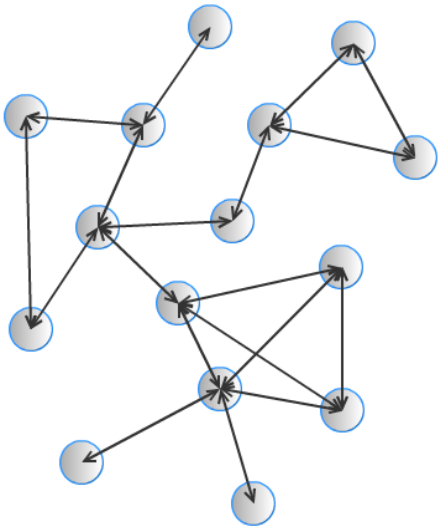

Figure 3 – Facebook Network Structure

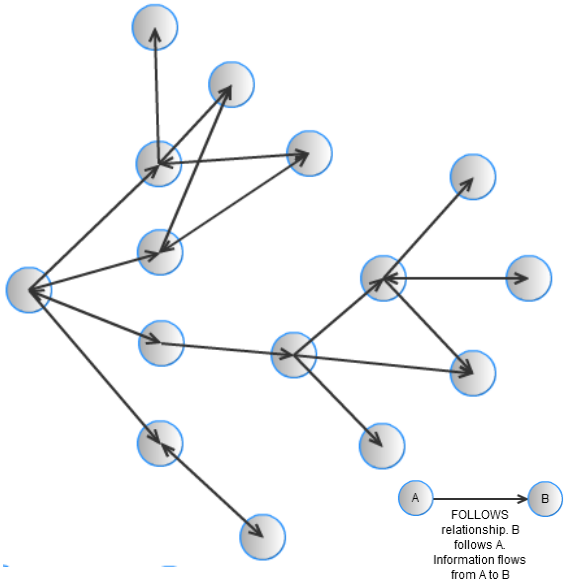

Figure 4 - Twitter Network Structure

In the next section we broaden the relationships and take into consideration the interactions that user have between them.

3.2.1 PageRank Conclusion

In the above experiment we looked at a simple way to identify important people in the network .Our approach is based on the concept of connections that people have with other people. We can see that this approach is similar to the PageRank algorithm that is used by Google's search engine to index web pages according to their importance. The importance of the web pages is dependent on the incoming and outgoing links that the pages have. From the results of the experiment we conclude

that PageRank can be efficient way to index users in the network, but it does not give us a very meaningful result. This is because we cannot compare the people in social networks to web pages in the World Wide Web. People have a number of different properties and connections types which define their position and influence in the network. In our Professors experiment, these properties can be such as different professors have contributed to multiple fields, they have different levels of importance in different sectors, or their focus area is more specialized. Also factors such as time eras of their research, technology, proven facts, importance of research on future improvements, nature of research etc. all can be used to better identify a user's importance. Taken all of this information into account can be difficult, but such information metrics certainly cannot be used in PageRank, and that is what makes the algorithm in-affective when trying to classify users.

3.3 Interaction Score

One of the problems with social networks is that it is really unclear how to handle privacy policies within networks. User may have a lot of relations in the network but that does not mean that the user would want to share all of their information with all the users in their network. This can also be applied when receiving data. A person might have someone in their network but might not want to receive and kind of messages from that user. This is similar to spam. So a solution to the privacy problem could be a possible solution to

spam, where you only receive data or messages from people that you are friendlier with.

Current social networks assume binary, symmetric relationships of equal value between all directly connected users. In reality this assumption is false as an individual has relationships of varying quality [6].

3.3.1 User Group Classification

To handle the privacy issue we require a mechanism to identify and classify people in the network so that we have a controlled flow of information. This means that if we are able to identify the important crown from the not-so-important crowd, we would be able to better handle the problem of spam information and control sensitive information more efficiently. This topic has been discussed by Felix Wu and Lerone Banks at UC Davis where they proposed a way to measure the interactions between users in social networks. If user A has user B and C as friends, the better of the two friends would be the one who interacts with user A the most. It would be safe to assume that user A would trust the user he interacts with more, more. So if say user B is a better friend, we would see more interactions between it and user A. The 'interactions' can refer to different actions on different networks. On Facebook, it could be posting on a user's Wall, tagging photos of the user or messages between the users. On Twitter it could be Direct Messages between users or the number of times a user retweets other user's tweets.

Felix Wu et al [6] also mention another step in addition to the interaction classification. They also add a questionnaire/survey as part of the their analysis which asks the social network user questions regarding what profile information would they want their particular group of friends to see. These groups are formed after the interaction intensity phase is completed and friends are grouped together based on their interaction or friendship levels.

A sample questionnaire could be:

“Please select a number representing the highest amount of profile information that you would like X to see.” [6]

And possible choices for the answer could be:

1. Thumbnail (full name, thumbnail profile photo)
2. Greater Profile(profile photo, basic information, education/work, contact information)
3. Activities (application actions (such as scores in games or gifts given), quizzes taken, your actions/interactions within the network)
4. Affiliations (Friends list, groups joined, pages added, applications added, external events attended)
5. All content(uploaded pictures, pictures tagged with user, comments, events, posted links, status updates) [6]

Such a mechanism in our opinion will not scale well and would not be appreciated by the user as

With an Interaction Score that we can use to add importance to a directed link, we are in affect adding to the quality of the link. Classification of the scores can be used to classify user in groups, or separate 'good' friends from 'bad' friends. It has to be emphasized that the approach mentioned above deals with scoring in a user's network and not whole, larger networks. It is technically difficult to gather information about user's interactions from a social network. Facebook does not make this information available to developers. One can get limited information such as the name of friends and photo tags but not information such as wall posts and private messages. In Twitter, some of the information such as retweets is available. This information can be extracted from the Twitter API but it requires one to know the id of the tweet in consideration. Some of information that can be collected in Twitter will be presented in the experiments discussions later in the report.

Each user would have an Interaction Score with its neighbor. The higher the Interaction Score, the better the friendship with the neighbor. The Interaction Score is the sum of all the interaction values that a user has with a neighbor. According to [7], users in social networks only interact with a small number of users frequently, so the privacy model will only apply to a handful of users [6]. The way we go about calculating the interaction values is to look at the number of interactions that occur in terms of wall postings, private messages, photo tagging, and assign numerical values to each.

In the current version of Facebook, some of the types of interactions that can be measured in are:

1. Tagging of friends in photos. (Tph)
2. Tagging friends in posts. (Tps)
3. Tagging friends in status updates. (Tst)
4. Private messages between friends. (PM)
5. Direct wall posts. (Wp)
6. Tagging friends at a single Place (Tpl)
7. Commenting on friend's posts(Cps)
8. Commenting on friend's photo (Cph)
9. Playing or involved in social games (G)
10. Acted upon recommendations to connect to other users (Ru)
11. Acted upon recommendations about pages or groups (Rpg)

There might be even more forms of interaction that can be measured in Facebook; these types keep on changing s Facebook evolves. Most of the information mentioned in the list, as mentioned above, cannot be measure by sources outside Facebook because the current Facebook API does not provide that much detail in the user information that it provides.

Having all the above different types, we can generate a formula to calculate the Interaction Score between two nodes. For the formula, we have to classify the importance of each of the interaction factors as compared to

each other. To achieve the best possible results, a closer inspection of all those actions will be required to determine the extent to which each might contribute to the final assessment of relationship for there would be some actions that would be more crucial and significant than others. Weights should then be assigned to each action and they should be classified as High, Medium or Low contributors.

3.3.2 Interaction Score Formula

For instance, messaging between friends or their tagging in owned albums or same spots on Places are high contributors since they indicate of a stronger bond. Messaging, private or wall, hints that the person keeps in touch with the other individual while coexistence in a picture or a location or in personal photo albums implies that they hang out together and both of these are strong indicators of a good relationship. On the other hand, actions like tagging friends in posts and status updates and commenting on their posts or photos should serve as medium contributors since they only hint that the people tagged share common interests with the individual and not necessarily someone who is close.

Other actions like responses to recommendations or playing social games should be low contributors, since they hint more of an individual's preferences than their relationship. Such actions can also be classified as dependant contributors meaning they should only be taken into account if a

relationship fares well in terms of high and medium contributors.

Having said that, we move to what our Interaction Score formula might look like. This formula would get the interaction between two nodes N1 and N2. An important factor to consider is the weight allocation for each interaction. As mentioned above, not all the interactions would carry the same weight or importance. We have chosen a scale of 0 to 100 to identify the weights of each factor. This would help compare values with other nodes and will help classify the users more efficiently.

So Interaction Score (IS) between N1 and N2 would be (formula 3):

$$IS(N1, N2) = w_1xTph + w_2xTps + w_3xTst + w_4xPM + w_5xWp + w_6xTpl + w_7xCps + w_8xCph + w_9xG + w_{10}xRu + w_{11}xRpg$$

Where the value for w_n is [0, 100].

One thing that has to be kept in perspective when assigning weights is that we can run into a scenario where there are many, say, low level interactions between two users and they contribute dominantly to the overall score of the relationship. A large number of low level actions, ending in a large IS, does not necessarily mean that the relationship is very strong between the users. We have to put in measures that limit such scenarios where the IS would give a wrong indication. A work around to this would be to add a certain scaling factor to each of our action levels. This scaling factor would be dependent on how importance, or how dominant a certain group of actions should be. If we have three levels of actions, low, medium and

high, then the scaling factor of D_L, D_M and D_H should be applied to groups of actions, and the sum of all three should be equal to 1. This would give us a better scaling mechanism to control to the IS. With the scaling factor, the IS formula becomes (formula 4)

$$IS(N1, N2) = D_L(w1xA1 + w2xA2 + w3xA3 + \ldots) + D_M(w4xA4 + w5xA5 + w6xA6 + \ldots) + D_H(w7A7 + w8A8 + w9A9 + \ldots)$$

Since we are trying to get the overall interaction degree, we should not limit the total sum of weights, i.e. there should be no upper limit to the IS achieved in a connection. After we get the IS for each friend of a user, say N1, we are in a position to classify the friends using the IS scores. The exact classification will depend on the values of the IS calculated but in general, the classification should be done depending on the levels of privacy that we have defined for the users. For example, such privacy levels can be based on the classification of best friends, good friends, colleagues, people who are not real life friends, etc etc.

4.0 The Professors Network Experiment

To demonstrate practical uses of the Person Score, we conducted experiments where we assign value to node relations and try to deduct patterns or information from the resulting data. In the first experiment, we decided we need to try out this approach on non-traditional and non-commercial social networks. We wanted to see if we could structure or organize information of different groups of people in such a way that that group resembles the structure of a social network. To start with the idea, we needed a data set of a group of people that are related to each other in some way and collectively represent a group that can be identifiable.

Our initial search for research papers for this report led me to many research topics and papers related to social networks. When going through some of the research papers and books, we would find ourselves going through the papers/articles/books that were mentioned as references in those papers/books. If the information was relevant to our needs, one would actually go through the referenced papers and might go through some of their references as well. This behavior of mine indicated to me that the papers form

a sort network where the papers themselves are nodes in the network and that the references are actually ways for these papers to link to each other. If we talk about these papers in terms of the authors, we deduce that the references are in fact ways for authors to validate their work and give credit to the work of the references. So if we look at it, a reference from user A to user B would mean that user A owes a part of its importance to user B. This looks really similar to the concept behind Rank Prestige. The collection of these papers referencing each other indicates a structure similar to a social network. This is what made us decide that we should focus on such networks or collections.

During the search, we accidentally stumbled upon an article listing the top 50 research papers in ACM in the field of programming language design during the 20 years from 1979 to 1999 [8]. This list has titles of the research papers and the authors of each paper. The fact that these 50 papers are the most important ones in the field would mean that the authors of these papers must be some of the most important or influential people in the computing field. So now what we have in the list are actually the important people of a larger network. Their ‘importance’ is important as it allows us to have a classification in the network and provides a good starting point, or initial values, to populate the network, as discussed in previous sections.

To get started what we did was to gather all the information about the papers involved and get all the authors mentioned as references. The source of all the information was the ACM (Association for Computing Machinery)

website which contains the papers that we had in our list as well as all the references cited by these papers. The information, including the paper titles and the titles and reference information is presented on the web and can be extracted using a web crawler. The availability of this information made the data population easier.

4.1 Implementation Details

To start the process of score allocation, we needed the following information:

1. List of all the important papers and the authors.
2. List of all the referenced papers and referenced authors of the important papers.

As mentioned the list of all important papers was available from [8]. To search for all the papers' details on the ACM site, we had to provide the paper name as the search parameter to the site and it gave the hyperlink to the queried paper's page. The page contained the References in an html tag with the 'references' keyword. That tag contained the list of references(titles and names). Some of the listed items were references to the listed article, so they could also be accessed on the ACM site.

The crawler that we made was provided with an initial_list.html file that contained the list of the important papers and the links to their web pages on ACM's website, received from the search feature. Generating this html file

involved collecting these links manually and putting them in an html format so that it could be crawled easily by the crawler. The crawler was then tasked to go through each link, get the references list, parse the names in the list, follow the links for each reference (if exists) and then get the author information for the referenced paper. This information is stored in a MySQL table called connections. At the end of this step, the data in the table contains directional links between all the nodes in the network.

To calculate the scores, initial score was allotted to each initial, important paper. This initial Person Score was set at 100, for no particular reason. This score would be divided equally amongst the authors of the paper. So if a paper had three contributing authors, a score of 33.33 would be allotted to each individual author. Any authors that are connected to these authors would be allotted a score based on formula 2.

The next step is to use the connections/link information and the initial scores to and calculate the scores for each node based on the score of its neighbor. At the start of the process, the state of the table is such that only the initial/important nodes have assigned scores. All other users have a score of zero. So, naturally, the way to start the score allocation would be to visit each initial node and assign scores to its neighbors, based on its own score.

The process is that we loop through all the professor nodes that we have in the database, and for each node we are looking up the list of neighbors from *connections* table along with the *Count* of the number of

incoming links. The outgoing links represents the 'referred by' relationship between nodes. We are only going through a single direction on a link at a time, but since we are going through all the professors, we are making sure that all the bidirectional links will be covered by the loops. It has to be stressed here that the outgoing or incoming nature of the link represents the flow of data with respect to the node. If node A references node B, the flow of data is from node B to node A, so node A would have an incoming link from node B.

The algorithm is defined below:

```
Foreach (professor in database.professor AS node)
{
        Neighbors = array(set of professor_ids from database.connections where
                                referred_by = node.id);

        Count = count(Neighbors);

        Foreach(neighbor in Neighbors AS neighbor_node)
        {
                neighbor_node.score += node.score/Count;
        }
}
```

At the end of the above code execution, all of the professors in our database have allotted scores that is based on the cumulative scores of the neighbors. Some of the initial nodes that were manually assigned scores had bidirectional links with other nodes. This means that although they were contributing some score to their neighbors, due to the recursive nature of traversal in the graph, some other connected nodes would be contributing part of their scores to the initial nodes too. With our approach, we are only

traversing through the graph once only. Because our data set is small, having multiple recursions would not have much effect on the results.

4.2 Results

The resulting data is a set of professors, their scores, the number of nodes the professor refers to and the number of nodes that refer the professor.

If we sort the result by the accumulated score, the Person Score, in descending order, we get the top results in table 1.

Professor Name	Person Score	Refers	Referred By	Group
George Radin	144.98	0	7	0
Thomas Johnsson	116.098	24	1	12
David W. Wall	111.111	9	1	17
Monica S. Lam	111.019	0	9	20
Janet Fabri	107.143	14	2	3
Ken Kennedy	101.704	62	17	8
Gregory Chaitin	100	1	0	7
Susan L. Graham	87.1979	4	9	5
Fred Chow	84.8246	17	3	9
Marc Auslander	75.3968	9	5	4
Martin Hopkins	75.3968	9	5	4
Jack W. Davidson	69.2561	40	2	10
Christopher W. Fraser	64.3998	40	2	10
Michael G. Burke	60.2752	23	6	13
Keith Daniel Cooper	56.9556	50	9	14
Thomas Pennello	53.7936	0	2	2
Frank DeRemer	53.5704	13	2	2
Steven S. Muchnick	53.5181	9	1	15

Table 1- Professors Experiment result 1

The Group field in the dataset represents the initial paper from the list at [8]. A group of zero means that the author was not part of the initial important papers list. So the professor on the top, George Radin gets referred by seven papers and does not refer anyone (in our network) in their own paper. All the accumulated score is from other outgoing links.

It should be noted that George Radin is a Fellow at the IBM Corporation and worked at the Advanced Computer Utilization Department, where he worked on advanced compiler technology and PL/I.

If we look at the result by order of the outgoing and incoming links (the refers and referred by fields), the results are shown in table 2 and 3. Table 2 shows us the nodes which have been most influential in shaping the structure

of the network. These nodes have the highest number or referrals, meaning they have the most incoming connections in the network. The direction indicating the flow of data. If a node A 'refers' node B, then it is in fact using the data *from* node B.

Professor Name	Person Score	Refers	Referred By	Group
Ken Kennedy	101.704	62	17	8
Thomas Reps	7.9087	57	7	0
Susan Horwitz	46.013	57	7	19
David Binkley	38.8815	57	7	19
Jonathan Rees	18.5214	51	6	16
Richard Kelsey	16.6667	51	0	16
James Philbin	16.6667	51	0	16
Paul Hudak	21.9537	51	7	16
Norman Adams	18.5693	51	6	16
David Kranz	20.1848	51	1	16

Table 2 – Results by order of Referrals given

Professor Name	Person Score	Refers	Referred By	Group
Ken Kennedy	101.704	62	17	8
John L. Hennessy	29.2018	0	15	0
Steven O. Hobbs	19.7027	0	13	0
Charles M. Geschke	19.7027	0	13	0
William Allan Wulf	19.7027	0	13	0
Charles B. Weinstock	19.7027	0	13	0
Richard K. Johnsson	19.7027	0	13	0
Alfred V. Aho	15.7595	0	10	0
Jeffrey D. Ullman	17.0164	0	10	0
Andrew W. Appel	7.79019	0	9	0

Table 3 – Results by order of Referrals received

Table 3 shows us the most referred authors in the data set. This information does not represent any significant result in terms of the network structure, but these authors can be considered as important figures in the

industry as they have written papers which have helped on influenced the most people in the list.

Now if we look at the data sets above with the analogy of web pages, table 1 represents the pages with the highest Rank. These pages would appear most in search engines. This is because although the page is linked to by relatively fewer other pages (George Radin has 7 versus Ken Kennedy has 15 referrers), the quality of the referrals for George Radin is greater, because his paper(s) has been referred by more prominent scientists in the field. The structure that has evolved from the above data set shows resemblance to a social network where we can measure the centrality and the prestige of the nodes based on the number of outgoing and incoming links. Users who have a large number of outgoing links, or are referred by others, have a high Prestige value as they are the data providers in the network and have the power to influence the network (table 3). Users who are data accumulators, and sit in the network such that they can gather data from incoming links have a dominant Centrality measure (table 2). In our next experiment, we apply the same model to Twitter and see if we can arrive at more intuitive results.

5.0 Twitter

Twitter® is one of the largest and popular social networking and microblogging sites today [9]. It has about more than 190 million users worldwide, as of June 2010 [10]. Twitter enables its users to send and read messages called *tweets*. Tweets are text-based posts of up to 140 characters displayed on the user's profile page [11]. Tweets are publicly visible by default; however senders can restrict message delivery to their friends only.

Users may subscribe to other users' tweets—this is known as *following* and subscribers are known as *followers* [12].

Tweets are made up of 140 characters and can have special tags or identifiers to indicate other users, replies and *topics* or *keywords*. The characters 'RT' at the beginning of a tweet represent that this is a *retweet,* which is a tweet by another user, forwarded to you by someone you follow, often used to spread news or share valuable findings on Twitter [12]. Retweets can be used to find out how far a certain tweet has spread in the network.

A *hashtag* (represented by the # symbol) before a word marks that word as a *keyword* in a tweet. A hashtag is similar to other web tags- it helps add tweets to a category [13]. Hashtags are used to categorize or organize tweets according to a keyword. These can represent topics, events, companies etc. and are an easy way to search for tweets on the system.

Lists are use to organize people you follow into groups, or lists. A person can only add users he/she follows to a list. People are also able to subscribe to lists. This notifies them whenever a new member has been added to the list by the creator. Subscribers can be used to find out how important a list is.

Direct Message is a way for users to send personal messages to people who follow them. One cannot receive direct messages from users who they do not follow. This is a way in Twitter to limit spam messages.

The flow of information (in the form of tweets) in Twitter is from a user to their followers. If an account is public, the tweet is broadcast to all the followers, and is also available in search results for the entire Twitter user base. Right now, Twitter does not have any way to identify user group permissions, where you can specify permissions according to user groups. This is the same problem that led to the Interaction Score that we discussed. We will discuss the IS in Twitter's context in a later topic.

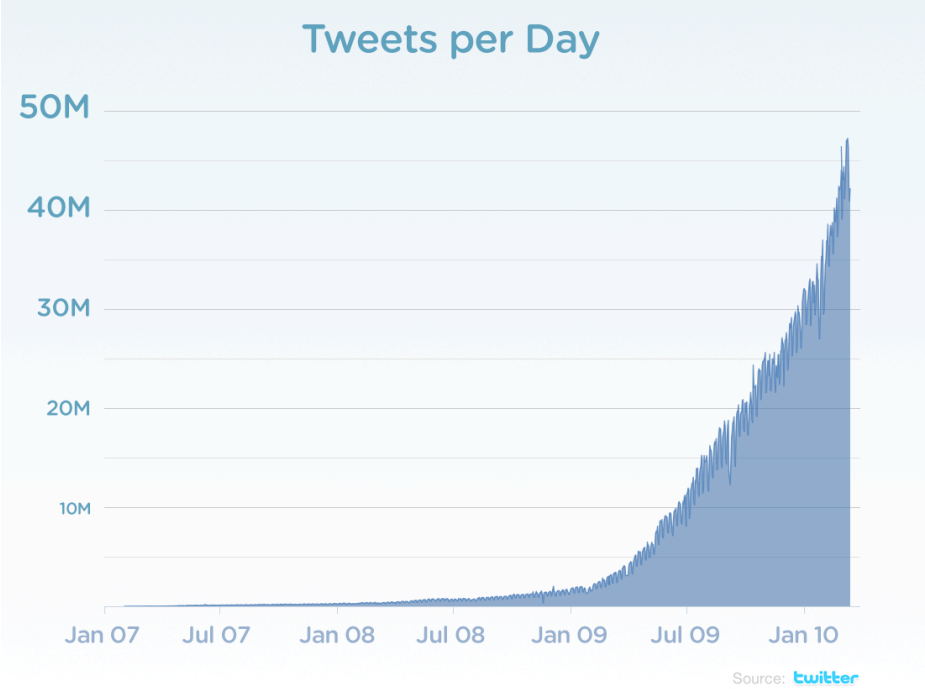

Figure 5 Tweets per day, as of February 2010

5.1 Person Score in Twitter

As before, to calculate the Person Score, we go to each user and check all the outgoing links from that user. In Twitter-sense, this means that we have to check and get a cumulative score from all the users that follow the particular user in question. This is because the outgoing link represents the data flow from the user, and data can only flow to your followers, or you can receive tweets from people you follow.

To calculate the score for the whole network, we need starting nodes from where the score calculation could start. This is the same procedure as used in the professors' score calculation. The starting data in this case is the list of top Twitter users. The users are ranked according to how many followers they have. This list is available from Twitaholic [14]. The top 30 people on the list were used. Information for each user was extracted using the Twitter API, which has methods to get various data about Twitter users. Some of the information requires authentication and approval from the users but some information as the number of followers, number of following, and the names of people being followed is available without any authentication. However, Twitter uses a throttling mechanism where they only provide 100 records per page, and only 150 pages per hour. This is limiting as it does not allow us to calculate the scores dynamically. All the data has to be downloaded into our database before we can start making connections. We started by getting a list of all the people *followed by* the top 30 Twitter users.

The top 30 users, with their Twitter user names are given in table 4.

Name	Twitter user name

Lady Gaga	ladygaga
Britney Spears	britneyspears
Ashton Kutcher	aplusk
Ellen DeGeneres	TheEllenShow
Kim Kardashian	KimKardashian
Taylor Swift	taylorswift13
Oprah Winfrey	Oprah
Ashley Tisdale	ashleytisdale
Ryn Seacrest	RyanSeacrest
Justin Biener	justinbieber
Katy Perry	katyperry
Twitter Inc.	twitter
Shakira	shakira
50Cent	50cent
CNN	cnnbrk
Justin Timberlake	jtimberlake
Marian Carey	MariahCarey
Selena Gomex	selenagomez
Shaquille O'neal	THE_REAL_SHAQ
Cold Play	coldplay
Twitter Inc.	twitter_es
Demi Moore	mrskutcher
Paris Hilton	ParisHilton
iamdiddy	iamdiddy
Jimmy Fallon	jimmyfallon
Chelsea Handler	chelseahandler
NYTimes	nytimes
Alicia Keys	aliciakeys
Perez Hilton	PerezHilton

Table 4 Twitter users

Once the data of all the users followed by our top 30 users was collected, all the connections between the users were recorded. For each of the top user, we have a separate field in the database that is used to store that user's connection information with each row, in the form of a 0 or 1 value. This arrangement allows us to see which of the top users a particular row

is connected to. We also have a field that holds a count of the number of connection that row has. The schema for this database table is given in index 1.

Another field in the database is used to hold the calculated, cumulative Person Score for each of the extracted user. These are the user's which are followed by the top users. The score calculated the same way as in equation 2:

$$S(x) = S(N1)/C(N1) + ... + S(Nn)/C(Nn)$$

For user x, we calculate the score by summing up the score contributions from each of its neighbors (the top users). C represents the total users followed by a node. So, as before, all the top users are contributing an equal amount of their initial scores to their neighbors. The initial score of each top user is set to 1,000,000. The same score was allotted to each top user so that the resulting score could be comparable to all users. The results from the data, sorted by the number of followers and the calculated scores are given in table 5 and 6.

Twitter user name	friends_of	followers	following
Oprah	14	4572161	20
PerezHilton	14	2632122	323
TheEllenShow	14	5468667	49500
justinbieber	13	6097870	90282
RyanSeacrest	12	3658444	246
aplusk	11	6057947	610
jimmyfallon	11	2969820	2585
BieberFanClubs	10	181176	10
iamdiddy	10	3055563	307
taylorswift13	10	4634118	48
EvaLongoria	9	948407	206
Sn00ki	9	886563	74
jtimberlake	9	3480287	19
KimKardashian	9	5382806	104
chegra	9	1381	78
kingsthings	9	1731382	199
rustyrockets	9	1587675	48
THE_REAL_SHAQ	9	3343668	621
ladygaga	9	7126588	146467
BarackObama	9	5925191	709816
ConanOBrien	8	1892740	1
NICKIMINAJ	8	1697215	917
kinggayle	8	162529	52
RevRunWisdom	8	1592871	0
katyperry	8	4628603	62
ev	8	1264928	1294
kanyewest	7	1640709	0
rihanna	7	2091538	21
RedHourBen	7	1580391	97
thekaue	7	2984	325

Table 5 - Results, ordered by followed_by

Note: The followed_by field holds the count of the number of connections between the row in question and the top users. So the first user ‘Oprah’ is connected to 14 of the 30 top users.

Twitter user name	Person Score	friends_of	followers	following
TheEllenShow	470047	14	5468667	49500
jimmyfallon	332737	11	2969820	2585
maryjblige	310170	7	1236149	68
THE_REAL_SHAQ	306873	9	3343668	621
aplusk	304482	11	6057947	610
kingsthings	295012	9	1731382	199
QueenRania	273678	5	1389738	67
JennyMcCarthy	271137	5	206442	279
kinggayle	270664	8	162529	52
ev	263404	8	1264928	1294
mrskutcher	255685	7	3186261	192
PerezHilton	249424	14	2632122	323
iamwill	236518	5	561532	42
Nate_Berkus	232442	3	48698	48
RealHughJackman	232442	3	581584	36
DrOz	229570	3	404097	38
LisaErspamer	229570	3	13282	155
GStephanopoulos	229570	3	1660471	586
TheOprahShow	229227	3	86670	56098
SheriSalata	229217	2	13400	180
OprahWinfreyNet	229106	1	3797	24
justinbieber	198546	13	6097870	90282
selenagomez	189050	7	3313836	212
Oprah	178528	14	4572161	20
katyperry	162482	8	4628603	62
jtimberlake	157033	9	3480287	19
TheRealJordin	148537	5	684144	332
nickjonas	136876	5	1823950	81

Table 6 - Results, ordered by Person Score

Table 5 gives us the users in terms of the number of connections that users have with neighbors. According to our data set, the user Oprah is the most connected user in the network with 14 connections. This user has the highest prestige in the network, as it has the most outgoing connections in our data set.

Table 6 represents those users who are followed by influential people. The scores are accumulated from the scores of the followers. TheEllenShow is on the top of the list here. Although it also has 14 outgoing links, the set of connections that it has contribute a larger score to it than to Oprah, which appears in the bottom section of the list. This means that the set of connections of TheEllenShow is more influential. And since TheEllenShow sends out data to these users, and influences them, it has the highest score and is the most influential user in the data set. It has to be pointed out that TheEllenShow is also among the top 30 users list.

One factor that has an effect on the result in table 6 is the initial score that was allotted to the top users. We have right now used 1,000,000 as the initial score for each of the top users. This can be a problem because the top users are classified as such because of the number of followers that they have. The top user, ladygaga, has about 7,466,457 followers whereas the last user, PerezHilton, has about 2,718,357 followers. This is a big deviation and allotting both these users the same score to start with is not fair.

Another approach for the initial score that we can have is to use the number of followers as a factor in calculating the initial score. Intuitively, users with a large user base of followers should have a higher score. However, this does not seem like the case if we compare the results from table 5 and table 6, where it is evident, as discussed before, those users with the same number of followers as not equal. But just out of curiosity, we recalculated the results based on the number of followers that each user has. The new initial score

was the number of followers divided by the count of the incoming connections. In other words, this was the ratio of the followers to the following. The results with the new scores are given in table 7:

Twitter user name	Person Score	friends_of	followers	following
TheEllenShow	693497	14	5468667	49500
jimmyfallon	542887	11	2969820	2585
Oprah	480944	14	4572161	20
aplusk	470789	11	6057947	610
THE_REAL_SHAQ	456196	9	3343668	621
PerezHilton	408602	14	2632122	323
kingsthings	406385	9	1731382	199
maryjblige	392115	7	1236149	68
QueenRania	391755	6	1389738	67
kinggayle	354294	8	162529	52
JennyMcCarthy	328618	5	206442	279
ev	294527	8	1264928	1294
justinbieber	289792	13	6097870	90282
taylorswift13	271354	10	4634118	48
mrskutcher	266996	7	3186261	192
iamwill	257994	5	561532	42
GStephanopoulos	255344	4	1660471	586
Nate_Berkus	245906	3	48698	48
RealHughJackman	245906	3	581584	36
DrOz	238209	3	404097	38
LisaErspamer	238209	3	13282	155
TheOprahShow	237203	3	86670	56098
SheriSalata	237137	2	13400	180
OprahWinfreyNet	237023	1	3797	24
NICKIMINAJ	234415	8	1697215	917
lilyroseallen	222524	4	2485303	115
ciara	218947	5	724223	67
carmeloanthony	216198	3	481507	237
RyanSeacrest	212744	12	3658444	246
twitter	207570	5	3963906	355

Table 7 - Results based on ratio of followers/following

The new results are similar to the older results, with the first two users

being the same in both lists. The user Oprah has moved up the list because it has large ratio of followers to following, as it only has only 20 people it is following.

5.2 Results

The results so far show that TheEllenShow is the most important user in our dataset. And because it has outward connections to about half of the top users, any tweet by TheEllenShow has a greater *chance* of reaching a wider audience because if the combined user base of the followers is quite large. (The combined number of followers of top followers for TheEllenShow is 53,150,135 but this includes redundant users). This is an indication that TheEllenShow is also the most influential user in the group. In this analysis, we are only considering the connections that a user has to get its importance. A problem with this approach is that we are not really looking at the interactions of the user with its followers to see what kind of influence the user has. A numerical score can tell us which user has a higher prestige, but the importance of the user also should depend on level of interactions the user has with the followers, not only on its position in the network. We introduce a new concept of Influence Reach to better represent the user's prestige.

6.0 Influence Reach

From the results of the above experiment, it is clear that a simple scoring scheme is not enough to explain or represent a user's importance in a social network. To add more meaning to the score, we need to introduce a new concept called Influence Reach in the network. Influence reach is used to define how far a node's data or influence can spread in the network. The influence is directly related to how important an actor is. The more important the user, the higher influence he/she would have in the network. The influence that a person has in a social network depends on what actions are available to the person in that particular social network. This concept is similar to the Interaction Score discussed earlier. We apply it to Twitter and use the interactions given in the previous topics to present a new method to determine the importance of a person's position in a social network. In this method we would look at the live interactions that the user has with its followers, and calculate the Interaction Score for the user.

The Twitter-specific interactions that we can measure are:

1. Retweets
2. Mention in tweets
3. Inclusion in lists
4. Direct messages

Apart from these interactions, the number of friends and followers

and the subscribers of the lists a user is a part of can be measured and will be used in the calculation of a person's Influence Reach. Next we are going to take an overview of our application that lets us gather all data discussed above.

6.1 Implementation Details

To get the information from the Twitter database, we have to use Twitter's developers' API [15]. This API provides various methods for someone wanting to implement a Twitter client. These methods get us data such as names of followers, the direct messages etc. of a user.

6.1.1 Authentication

Twitter uses the open authentication standard OAuth for authentication [16]. OAuth provides a method for clients to access server resources on behalf of a resource owner (such as a different client or an end-user). It also provides a process for end-users to authorize third-party access to their server resources without sharing their credentials [17]. This allows a Twitter user to log into our application and give the application the required permissions to get the user's data from Twitter.

The above authentication process is actually a limitation as it does not allow us to get all the required information from Twitter, for all users in the

network. We can get the data of users who give us permission.

Due to this limitation, for the non-authenticated users, we can only get the following information:

1. Number of friends
2. Number of followers
3. Lists included in
4. Number of tweets

For authenticated users, additional data about retweets and direct messages is also available.

6.1.2 Application Flow

The following diagram gives us the flow of the application:

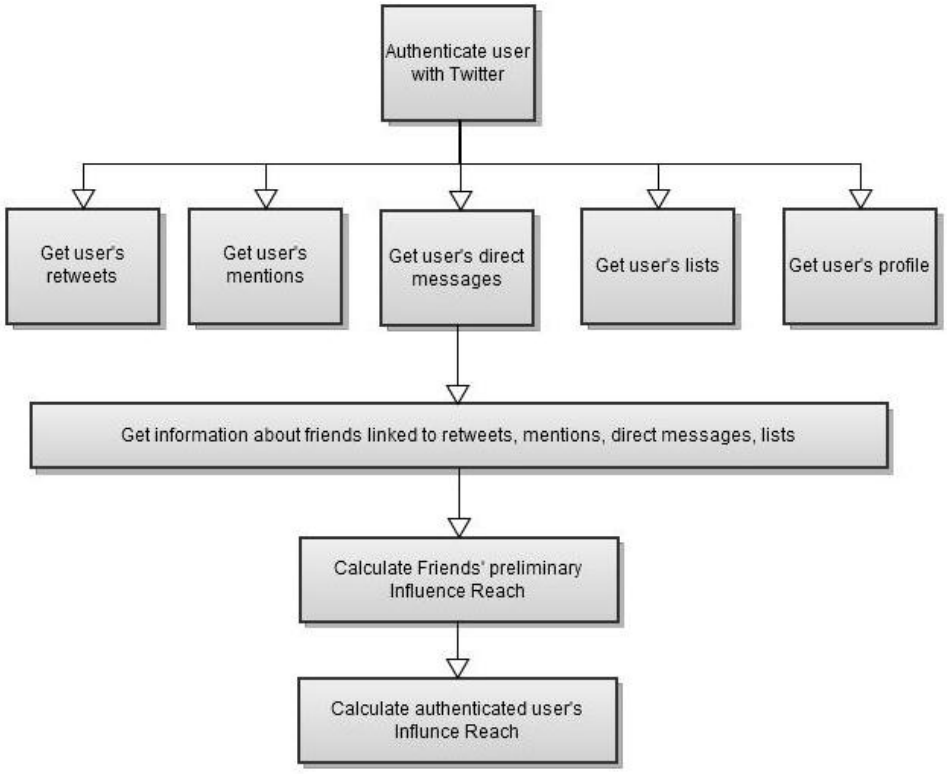

Figure 6 - Application Flow

The following is a more detailed explanation:

1. After the user is authenticated by Twitter, the application receives an access token which it uses to act on behalf of the user when communicating with Twitter. The token is made up of the oauth token and a token secret. These both are always sent as part of the request

to the server. Along with these, a consumer key and consumer secret are also part of the request. These are generated by Twitter when an application is registered with them.

2. Once the authentication process is complete, we are able to make requests to the API to get the data. We call the following methods:

    a. direct_messages

    http://api.twitter.com/version/direct_messages.format

    Returns the 20 most recent direct messages sent to the authenticating user [18].

    b. statuses/mentions

    http://dev.twitter.com/doc/get/statuses/mentions

    Returns the 20 most recent mentions [19].

    c. statuses/retweets_of_me

    http://api.twitter.com/version/statuses/retweets_of_me.format

    Returns the 20 most recent tweets of the authenticated user that have been retweeted by others [20].

    d. statuses/:id/retweeted_by

    http://api.twitter.com/version/statuses/:id/retweeted_by.format

    Show user objects of up to 100 members who retweeted the status [21].

    e. user/lists/memberships

*http://api.twitter.com/version/:user/lists/memberships.format*List

the lists the specified user has been added to [22].

f. users/show

http://api.twitter.com/version/users/show.format

Returns extended information of a given user [23].

3. All the methods defined above are called using AJAX on our main panel. This allows us to make multiple asynchronous calls so that the panel doesn't hang while the data is being loaded. The resulting data is formatted and then displayed on the panel to the user. The panel with the retrieved data has the following layout:

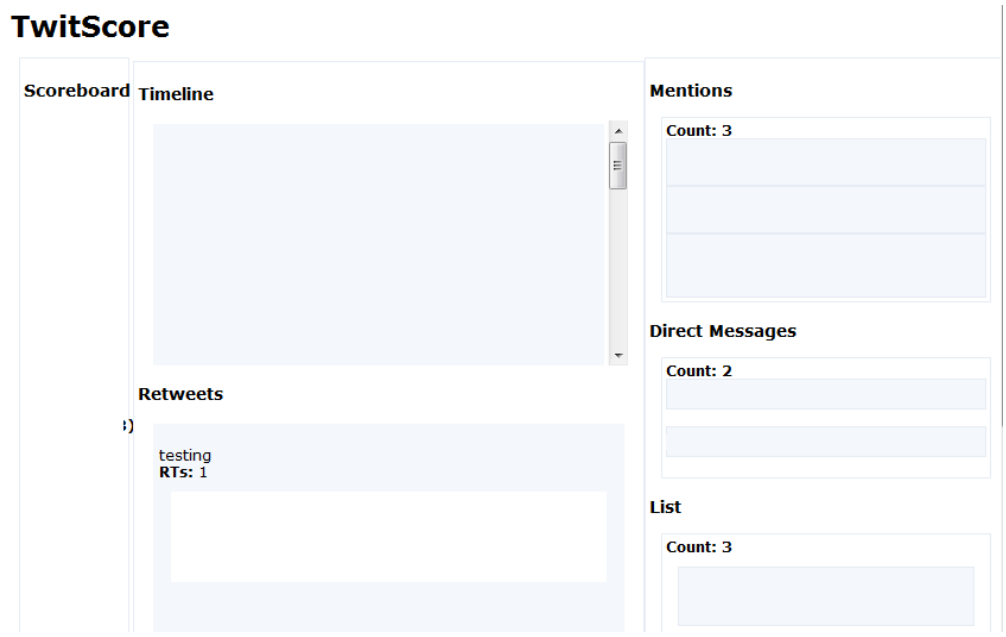

Figure 7 - Panel Layout

4. All the non-authenticated users (friends) which are part of the authenticated user's data (mentions, lists, direct messages, retweets) are stored in the database and their followers, friends, lists and status information is retrieved. Note that this is the only useful information that we can get for non-authenticated users. This data is store in the database against the authenticated user. Also the interaction type that connects the authenticated user with the non-authenticated user is stored. The following web service is used to get a user's data:

   users/show
   http://api.twitter.com/version/users/show.format

5. The preliminary score for the friends is calculated based on the data available. As with the Interaction Score, each data variable is assigned an importance. This importance is given as D_H, D_M, D_L. H represents High, M Medium and L for Low. The values of these three can be adjusted. We have set them at

   $$D_H = 0.50$$

   $$D_M = 0.30$$

   $$D_L = 0.20$$

   The values of these depend on how much importance we want to assign to the different data variable used in calculating the score. The data variables for the non-authenticated users we have are:

   a. The number of followers – a person with a larger fanbase is likely to be more influential. This has high importance (D_H)

b. The ratio of followers to friends - If a person has more followers than they are following, they're probably a good person to at least consider following. If they are following more than they have more followers, the opposite may be true [24][25]. This is given high importance (D_H).

c. The number of lists the user is on – lists are important because it makes it easier for other users to find users of interest to follow. The more lists a person is a part of, the more likely he/ she is to appear in search results and suggestion lists. This is given medium importance (D_M).

d. The number of statuses or tweets the user has in the system. A higher number represents that the user has been active, but it can also mean that the person has sent out a lot of spam messages. The real importance of these tweets would be available if we know the number of retweets of the tweets. This information is not available so we cannot assume anything here. Due to this, this is given low importance (D_L).

6. The preliminary score is calculated using the following formula:

   D_H x followers + D_H x ratio + D_M x lists_count + D_L x statuses_count

7. Once the score of all the friends in calculated, the IR score of the authenticated user is calculated:

   a. The Retweets are taken into consideration. For each follower who

retweeted a tweet, the ratio of the user's tweet to the total tweets is calculated. This is multiplied by the total score of that user. The total for all the followers gives us the accumulated score for retweets. Ideally, for a follower, we would like to use the ratio of the total retweets to the user's retweets, but this information is not available.

b. Mentions are also calculated. These are calculated the same way as retweets, because mentions are also tweets; a mention-tweet would have the name of the user being mentioned, just like a retweet. The difference is that in tweet with a mention, the tweeter wants to draw attention to the person being mentioned, meaning that the name of the mentioned user is broadcast to all the followers. Twitter has a special menu to show all the user's mentions. This gets a higher importance than retweets.

c. As mentioned in 5(c), lists are an important feature for users to get discovered and recognized. Lists are made up of members and subscribers. The creator can add members to a list and the public can subscribe to the list. Importance of the list can be attributed to the ratio of subscribers to members and the score of the individual subscribers.

d. For the messages, we get the number of messages and the scores of the senders, for each message. We get the cumulative score and scale it by $D_{H.}$

8. Once we have all the scores of the friends, the Influence Reach is calculated for the authenticated user. This is the sum of the user's pre score and scaled lists, mentions, retweets and direct messages score of the friends. The scaling is done using the importance factors. The following formula is used:

   pre_score + (DL x list_score + DM x rt_score + DH(dm_score + mention_score))

A few things have to be pointed out here. The scoring code only takes the pre-score of the friends dynamically from Twitter. If a friend logs into the app, a new score would be calculated for them and this new score would be used for further for calculation of other user's scores. We might run into cycles when we have bidirectional relationships between users, but this has been manually limited in the implementation.

6.2 Results

The result obtained from the above application is the pre score of the authenticated user based on only the number of friends, number of followers, the number of lists and the number of tweets in the system, plus the contribution of score from other users in the form of interactions: retweets, mentions, direct messages and list inclusion. The obtained result for our test user is given in Table 8.

PRE-SCORE	45
RETWEETS	RT: 1.48% razzman: 111.35248 calyps: 0 anasimtiaz: 94.017 Pyronyx: 7.99992 **Total: 213.3694**
MENTIONS	calyps: 75.9168 anasimtiaz: 136.752 **Total: 212.6688**
LISTS	List Count: 2 User Count: 2 **Total:31**
DIRECT MESSAGES	Msg Count: 2 **Total:8**
Influence Reach	226

Table 8 - Application Results

The table shows that for this particular user, the score (pre-score) was 45 when the user was considered independently, but the actual score because of the different interactions with the neighbors is 226.

The final score is the one that can be used to compare this user with other users in the network. A higher score means that the user is more influential in the network and that it holds a good position as far as its ability to reach other people is concerned. An example of where such information can be used is for social marketing: A company wishes to tweet about its products and uses famous Twitter users to deliver the tweets. Using the IR information, one can compare users in terms of their interactions with other users. A higher IR would ensure that the user's message has a higher chance of spreading in the network.

7.0 Future Direction: Rough Set Analysis

If we look at the data we've collected, we can see that the data represents the interactions that users have which each other. For each pair of user, we know the level or degree of interaction. If we look at the interactions as links between nodes of a graph, the resulting database that we have will be the relational representation of the network. Being of relational nature, this data can be used for data mining and analysis.

As Collins (1988, page 413) tells us, "Social life is relational; it's only because, say, blacks and whites occupy particular kinds of patterns in networks in relation to each other that 'race' becomes an important variable".

The data is organized in such a way that the relationships (or

interactions in Twitter) are used to link group of (two) users together. This group represents a set which is based around the interactions. All similar groups can be found using the information of the attributes and these are called equivalence classes of the group.

This means that if the dataset is X, then the equivalence class of an element *a* in X is the subset of all elements in X which are equivalent to *a*. We can find subsets of similar users and similar relationships based on the data that is available to us.

References

[1] Social network analysis: methods and applications By Stanley Wasserman, Katherine Faust, page 8

[2] Alexander C.N., ”A method for processing sociometric data”, Sociometry 26

[3] User Position Measures in Social Networks

[4] Efficient computation of PageRank TH Haveliwala – 1999

[5] Adaptive methods for the computation of PageRank. Sepandar Kamvar, Taher Haveliwala, and Gene Golub

[6] All Friends are NOT Created Equal: An Interaction Intensity based Approach to Privacy in Online Social Networks - Lerone Banks, Shyhtsun Felix Wu Department of Computer Science UC DAVIS

[7] User interactions in social networks and their implications. Christo Wilson, Bryce Boe, Alessandra Sala, Krishna P.N. Puttaswamy, Ben Y. Zhao. EuroSys '09: Proceedings of the 4th ACM European conference on Computer systems

[8] 20 Years of the ACM SIGPLAN Conference on Programming Language Design and Implementation (1979 - 1999) http://www.cs.rutgers.edu/tmp/webarchives/ZmdgIBGXEwUXUpAXaXxQ/

[9] http://socialmediatoday.com/soravjain/195917/40-most-popular-social-networking-sites-world

[10] http://techcrunch.com/2010/06/08/twitter-190-million-users/

[11] http://webtrends.about.com/od/glossary/g/what-is-a-tweet.htm

[12] http://support.twitter.com/groups/31-twitter-basics/topics/104-welcome-to-twitter-support/articles/166337-the-twitter-glossary#f

[13] http://support.twitter.com/articles/49309-what-are-hashtags-symbols

[14] http://twitaholic.com/

[15] http://dev.twitter.com/

[16] http://dev.twitter.com/pages/auth#at-twitter

[17] http://tools.ietf.org/html/rfc5849

[18] http://dev.twitter.com/doc/get/direct_messages

[19] http://dev.twitter.com/doc/get/statuses/mentions

[20] http://dev.twitter.com/doc/get/statuses/retweets_of_me

[21] http://dev.twitter.com/doc/get/statuses/:id/retweeted_by

[22] http://dev.twitter.com/doc/get/:user/lists/memberships

[23] http://dev.twitter.com/doc/get/users/show

[24] http://techcrunch.com/2009/08/26/twitters-golden-ratio-that-no-one-likes-to-talk-about/

[25] http://arcware.net/the-twitter-following-followers-tff-ratio/

[26] http://www.iscid.org/encyclopedia/Binary_Relation

[27] GRANULAR COMPUTING: From Rough Sets and Neighborhood Systems to Information Granulation and Computing with Words T. Y. Lin

[28] T. Y. Lin Neighborhood systems and relational databases. CSC '88 Proceedings of the 1988 ACM sixteenth annual conference on Computer science

[29] http://en.wikipedia.org/wiki/Neighbourhood_(mathematics)

[30] T. Y. Lin, "**Granular Computing on Binary Relations I: Data Mining and Neighborhood Systems.**" *In: Rough Sets In Knowledge Discovery*, A. Skowron and L. Polkowski (eds), Physica-Verlag, 1998, 107-121

[31] Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer D. Widom. Database Systems: The Complete Book. Prentice Hall. 2nd Ed

INDEX 1

1. Database Table Schema

   a. Professors

```
CREATE TABLE  `professor` (
  `prof_id` int(11) NOT NULL auto_increment,
  `prof_name` varchar(45) default NULL,
  `prof_score` float default NULL,
  `group` int(10) unsigned NOT NULL,
  `refers` int(10) unsigned NOT NULL,
  `referred_by` int(10) unsigned NOT NULL,
  `score` float NOT NULL,
  PRIMARY KEY  (`prof_id`)
)

CREATE TABLE  `connections` (
  `prof_id` int(11) default NULL,
  `ref_to_id` int(11) default NULL
)
```

   b. Twitter user database

```
CREATE TABLE  `friends` (
  `twitter_id` bigint(20) unsigned NOT NULL default '0',
  `screen_name` varchar(45) NOT NULL,
  `score` float NOT NULL default '0',
  `klout_score` float NOT NULL default '0',
  `friends` int(10) unsigned NOT NULL default '0',
  `followers` int(10) unsigned NOT NULL default '0',
  `friends_of` int(10) unsigned NOT NULL default '0',
  `ladygaga` smallint(5) unsigned NOT NULL default '0',
  `britneyspears` smallint(5) unsigned NOT NULL default '0',
  `aplusk` smallint(5) unsigned NOT NULL default '0',
  `lancearmstrong` smallint(5) unsigned NOT NULL default '0',
  `TheEllenShow` smallint(5) unsigned NOT NULL default '0',
  `KimKardashian` smallint(5) unsigned NOT NULL default '0',
  `taylorswift13` smallint(5) unsigned NOT NULL default '0',
  `Oprah` smallint(5) unsigned NOT NULL default '0',
  `ashleytisdale` smallint(5) unsigned NOT NULL default '0',
```

```
  `RyanSeacrest` smallint(5) unsigned NOT NULL default '0',
  `justinbieber` smallint(5) unsigned NOT NULL default '0',
  `katyperry` smallint(5) unsigned NOT NULL default '0',
  `twitter` smallint(5) unsigned NOT NULL default '0',
  `shakira` smallint(5) unsigned NOT NULL default '0',
  `50cent` smallint(5) unsigned NOT NULL default '0',
  `cnnbrk` smallint(5) unsigned NOT NULL default '0',
  `jtimberlake` smallint(5) unsigned NOT NULL default '0',
  `MariahCarey` smallint(5) unsigned NOT NULL default '0',
  `selenagomez` smallint(5) unsigned NOT NULL default '0',
  `THE_REAL_SHAQ` smallint(5) unsigned NOT NULL default '0',
  `coldplay` smallint(5) unsigned NOT NULL default '0',
  `twitter_es` smallint(5) unsigned NOT NULL default '0',
  `mrskutcher` smallint(5) unsigned NOT NULL default '0',
  `ParisHilton` smallint(5) unsigned NOT NULL default '0',
  `iamdiddy` smallint(5) unsigned NOT NULL default '0',
  `jimmyfallon` smallint(5) unsigned NOT NULL default '0',
  `chelseahandler` smallint(5) unsigned NOT NULL default '0',
  `nytimes` smallint(5) unsigned NOT NULL default '0',
  `aliciakeys` smallint(5) unsigned NOT NULL default '0',
  `PerezHilton` smallint(5) unsigned NOT NULL default '0',
  PRIMARY KEY  USING BTREE (`twitter_id`),
  KEY `Index_friends_of` (`friends_of`),
  KEY `Index_score` (`score`)
)
```

c. Twitter application

```
CREATE TABLE  `twitter`.`direct_messages` (
  `dm_id` bigint(20) unsigned NOT NULL default '0',
  `user_id` bigint(20) unsigned NOT NULL default '0',
  `sender_id` bigint(20) unsigned NOT NULL default '0',
  `text` text NOT NULL,
  `recipient_screen_name` varchar(45) NOT NULL,
  `sender_screen_name` varchar(45) NOT NULL,
  PRIMARY KEY  USING BTREE (`dm_id`,`user_id`)
)

CREATE TABLE  `twitter`.`list_members` (
  `id` bigint(20) unsigned NOT NULL auto_increment,
  `list_id` bigint(20) unsigned NOT NULL,
  `screen_name` varchar(45) NOT NULL,
  `friends` int(10) unsigned NOT NULL,
  `followers` int(10) unsigned NOT NULL,
```

```
  `lists` int(10) unsigned NOT NULL,
  `statuses` int(10) unsigned NOT NULL,
  `ratio` float NOT NULL,
  `score` float NOT NULL,
  PRIMARY KEY  (`id`)
)

CREATE TABLE  `twitter`.`lists` (
  `list_id` bigint(20) unsigned NOT NULL default '0',
  `name` varchar(45) NOT NULL,
  `slug` varchar(45) NOT NULL,
  `list_owner_id` bigint(20) unsigned NOT NULL default '0',
  `list_owner_name` varchar(45) NOT NULL,
  `user_id` bigint(20) unsigned NOT NULL default '0',
  `member_count` int(10) unsigned NOT NULL,
  `subscriber_count` int(10) unsigned NOT NULL,
  PRIMARY KEY  USING BTREE (`list_id`,`user_id`)
)

CREATE TABLE  `twitter`.`mentions` (
  `tweet_id` bigint(20) unsigned NOT NULL auto_increment,
  `user_id` bigint(20) unsigned NOT NULL,
  `tweeted_by_id` bigint(20) unsigned NOT NULL,
  `text` text NOT NULL,
  PRIMARY KEY  USING BTREE (`tweet_id`,`user_id`)
)

CREATE TABLE  `twitter`.`rt` (
  `id` int(10) unsigned NOT NULL auto_increment,
  `tweeter_id` bigint(20) unsigned NOT NULL default '0',
  `user_id` bigint(20) unsigned NOT NULL,
  `name` varchar(45) NOT NULL,
  `text` text NOT NULL,
  `hash` varchar(32) NOT NULL default ' ',
  PRIMARY KEY  USING BTREE (`id`,`user_id`)
)

CREATE TABLE  `twitter`.`user_scores` (
  `id` bigint(20) unsigned NOT NULL auto_increment,
  `screen_name` varchar(45) NOT NULL,
  `user_id` bigint(20) unsigned NOT NULL default '0',
  `friends` int(10) unsigned NOT NULL,
  `followers` int(10) unsigned NOT NULL,
  `lists` int(10) unsigned NOT NULL,
  `statuses` int(10) unsigned NOT NULL,
```

```
  `ratio` float NOT NULL,
  `score` int(10) unsigned NOT NULL,
  `mention` int(10) unsigned NOT NULL,
  `dm` int(10) unsigned NOT NULL,
  `list` int(10) unsigned NOT NULL,
  `rt` int(10) unsigned NOT NULL,
  `main` int(10) unsigned NOT NULL default '0',
  PRIMARY KEY  USING BTREE (`id`,`screen_name`,`user_id`)
)

CREATE TABLE  `twitter`.`users` (
  `id` bigint(20) unsigned NOT NULL auto_increment,
  `name` varchar(45) NOT NULL,
  `score` int(10) unsigned NOT NULL,
  PRIMARY KEY  (`id`)
)
```

2. Code

a. Pre score calculation

```php
/*
*    params:
*    user_id - the id of the authenticated user
*    name - the screen name of the authenticated user
*    type - the type of interaction
*    count - the the sequuence of the interaction
*/
function calculate_user_score($user_id, $name, $type, $count)
{
    global $conn;

    $user = json_decode(@file_get_contents("http://api.twitter.com/1/users/show.json?screen_name=$name"));

    if(empty($user))
        {echo("Cannot calculate score - Requests overloaded<br/>");return 0;}

    $ratio = get_user_ratio($user->followers_count,$user->friends_count);
    $score = ceil(DH * $user->followers_count + DH * $ratio + DM * $user->listed_count + DL * $user->statuses_count);

    $sql = "INSERT INTO user_scores(id, user_id, screen_name, friends, followers, lists, statuses, score, ratio, $type)
        VALUES({$user->id}, $user_id, '{$name}', {$user->friends_count}, {$user->followers_count}, {$user->listed_count},
        {$user->statuses_count}, $score, $ratio, 1) ON DUPLICATE KEY UPDATE $type = $count";

    mysql_query($sql, $conn) or die(mysql_error());

    $sql = "select id, score from users where id=".$user->id;
    $res = mysql_query($sql, $conn) or die(mysql_error());

    if(mysql_num_rows($res) > 0)
    {
        $row = mysql_fetch_array($res);
        $sql = "update user_scores set score={$row['score']}, main = 1 where id=".$user->id;
        mysql_query($sql, $conn) or die(mysql_error());
    }

    return $score;
```

b. Direct Messages score calculation

```php
/*DIRECT MESSAGES*/
echo "<br/><br/><b>DIRECT MESSAGES</b>";
$sql = "select sender_id from direct_messages where user_id = $user_id";
$res = mysql_query($sql, $conn) or die(mysql_error());

$dm_count = 0;
while($row = mysql_fetch_array($res))
{
    $dm_count++;
    $sql = "select score from user_scores where id=".$row['sender_id'];
    $res2 = mysql_query($sql, $conn) or die(mysql_error());
    $row2 = mysql_fetch_array($res2);

    $total_dm_score += (DH*$row2['score']);
}

echo "<br/><b>Msg Count:</b> $dm_count";
echo "<br/>Total:$total_dm_score";

$new_score = $pre_score + (ceil(DL*$total_list_score + DM*$total_rt + DH*$total_dm_score
    + DH*$total_mention ) );
echo "<br/><br/><b>IR: $new_score</b>";
```

c. Mentions score calculation

```php
/*MENTIONS*/
echo "<br/><br/><b>MENTIONS</b>";

$sql = "select distinct u.friends, u.screen_name, u.statuses, u.mention, u.score
    FROM mentions m join user_scores u on
    (u.id = m.tweeted_by_id) where u.user_id = $user_id";
$res = mysql_query($sql, $conn) or die(mysql_error());

while($row = mysql_fetch_array($res))
{
    $score = round($row['mention']/$row['statuses'], 5)*$row['friends']*$row['score'];
    echo "<br/><b>".$row['screen_name']."</b>: ".$score;

    $total_mention += $score;
}

echo "<br/>Total: $total_mention";
```

d. Lists score calculation

```
119     /*LISTS*/
120     echo "<br/><br/><b>LISTS</b>";
121     $sql = "select list_id, list_owner_name, slug from lists where user_id = $user_id";
122     $res = mysql_query($sql, $conn) or die(mysql_error());
123
124     $list_count = 0;
125     while($row = mysql_fetch_array($res))
126     {
127         $users = json_decode(file_get_contents("http://api.twitter.com/1/
128             {$row['list_owner_name']}/{$row['slug']}/subscribers.json"));
129
130         $list_count++;
131         $user_count = 0;
132         if($row['member_count'] == 0 || $row['subscriber_count'] == 0)
133             $ratio = DL;
134         else
135             $ratio = $row['subscriber_count'] / $row['member_count'];
136
137         foreach($users->users as $user)
138         {
139             $user_count++;
140             $ratio = get_user_ratio($user->followers_count,$user->friends_count);
141             $score = ceil($ratio*(DH * $user->followers_count + DH * $ratio + DM * $user->listed_count
142                 + DL * $user->statuses_count));
143
144             $sql = "INSERT INTO list_members (list_id, screen_name, friends, followers, lists, statuses,
145                 ratio, score) VALUES(
146                 {$row['list_id']}, '{$user->screen_name}', {$user->friends_count}, {$user->followers_count},
147                 {$user->listed_count},
148                 {$user->statuses_count}, $ratio, $score)";
149             mysql_query($sql, $conn) or die(mysql_error());
150
151             $total_list_score += $score;
152         }
```