SocialRank: Ranking Users and Information in Online Social Networks

by

Abdulrahman I. Tarbzouni

Bachelor of Science, Massachusetts Institute of Technology (2007)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

Massachusetts Institute of Technology

June 2009

Author: ___
Department of Electrical Engineering and Computer Science
May 01, 2009

Certified by: _____
Robert H. Rines
Department of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by: _____
Arthur C. Smith
Chairman, Department Committee on Graduate Students

SOCIALRANK: RANKING USERS AND INFORMATION IN ONLINE SOCIAL NETWORKS

by

ABDULRAHMAN I. TARBZOUNI

Bachelor of Science, Massachusetts Institute of Technology (2007)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Abstract

The goal of this project is to explore the design and implementation of SocialRank. SocialRank is a personalized ranking algorithm that provides—for each user—ratings for people in his online social network. Subsequently, these ratings are used to rank incoming information received by the user from those in his social network.

We analyze the use of actions on online social networks as proxies for measuring the strength of relationships between users and introduce an action scoring mechanism that uses different factors to evaluate an action's significance.

We implement SocialRank in a generic online social network that we build as part of this research project and explore the effectiveness and usefulness of SocialRank.

Thesis Supervisor: Robert H. Rines
Title: Lecturer, Department of Electrical Engineering and Computer Science

Acknowledgements

I would like to acknowledge the following people:

My thesis advisor, Professor Robert Rines, for challenging me and being a great mentor, teacher and friend. His achievements will always remind me that the ultimate satisfaction in life comes from transforming the lives of others and making the world a better place. I am honored to have been a student of Professor Rines.

My academic advisor, President Emeritus Paul Gray, for his limitless support and dedicated guidance. I am deeply touched by his genuine care and consideration to every detail of my MIT experience. He was always there and I cannot thank him enough. His noble character is a sources of great influence to me. I am truly privileged to have had the opportunity of being advised by Professor Gray during my time at MIT.

Words cannot describe the amount of admiration and respect I have for Professor Rines and Professor Gray. I will always cherish the relationship and personal stories I've shared with them. Indeed, I will forever remain a student of them.

My wife, Jana. For her love and encouragement. For her unyielding support even in the face of impossible challenges and potential failures.
You complete me.

My dearest mother, father and siblings; Abdulmohsin, Najla and Abdulaziz. For their eternal love and support. For their unquestionable faith in what I do.

Thank you.

Contents

1. Introduction

Online social networks are becoming increasingly popular mediums of information exchange. They are lowering the cost of live communication and information sharing. People we know share updates with us through Facebook, photos through Flickr, news through Digg and broadcast messages to us through Twitter. But given how effective these systems have become as platforms of information dissemination and the ever-growing number of people adopting such platforms, the information inflow created can grow quite dramatically. Consequently, information overload is becoming a real problem in online social networks.

Information we receive through online social networks are usually displayed in chronological order without an indicator as to how important a received piece of information is. This usually leads us to linearly scan our online social network feeds to identify important items. Essentially, information is ranked based on recency, not on importance.

Take for example the case of Facebook. Facebook allows a user to connect with friends, and share blog posts, links, photos, videos. Facebook implements a news feed that lists all updates from a user's friends. An average user of Facebook has 120 friends[1]. Regardless of who the friend is, friend updates are displayed sequentially in chronological order. Consequently, the news feed becomes quickly overloaded with updates from all friends, whether their updates matter to a user or not. Facebook currently tries to solve this problem by allowing the user to

[1] Official Facebook statistics: http://www.facebook.com/press/info.php?statistics

manually "hide" specific friends from the news feed so that the user can focus on updates from more closer friends. But this process is not dynamic and requires the user to consciously filter out unwanted friend updates from his news feed.

Another example is Twitter. Twitter is a fast-growing platform for sharing information and broadcasting messages or what is referred to as "tweets" from friends. A user can sign up and follow other users to receive their tweets. The more people a user follows, the more tweets the user receives. The number of tweets a user receives can become overwhelmingly large given the amount of activity on Twitter. Tweets are also ordered chronologically which requires a linear scan to identify important tweets.

It is clear that an automated mechanism for ranking information we receive through online social networks is highly desirable. In this research paper, we present SocialRank: a personalized ranking algorithm that provides—for each user—ratings for people in his online social network. We explore a novel mechanism of using online actions made between users to infer individual ratings that reflect the importance users place on each other in the online social network. These ratings are subsequently used to rank and sort incoming information received by the user.

This paper is organized as follows: Chapter 2 provides information about background research and related work. Chapter 3 describes the design of SocialRank and analyze the use of actions in online social networks as proxies for measuring relationship strength. Chapter 4 explores the

implementation of SocialRank within an online social network that we build as part of this research project. Finally, Chapter 5 concludes with our findings and future work.

2. Background and Related Work

Ranking and trust algorithms generally follow two main approaches when rating items of interest: a global approach and a personalized approach. The global approach results in a single global rating for each item. In that case, the algorithm uses ratings shared by individual users (or entities) to compute a global rating for each item. The implementation of such algorithms could be based on collaborative filtering methods [1] or reputation algorithms. In the personalized approach, the ratings of items in the system are typically different for each user. Every user has her own ratings of the items involved.

The most well-known ranking algorithm is PageRank [2]. PageRank analyzes links between webpages to create a graph inferred from their interconnecting links. Based on that, a webpage has a high rank if the sum of the ranks of its backlinks is high. A personalized version of PageRank [3] takes user profiles and their usage patterns into account and computes a Personalized PageRank Vector that is used to compute personalized rankings of webpages for each user.

The Eigentrust algorithm [4] determines the reputation of a set of peers in P2P networks based on their interactions. Each peer determines a local trust value for every other peer based on the number of satisfactory and unsatisfactory interactions it had with that peer. The algorithm then uses those local values to computer a global reputation value for each peer in the network.

Our SocialRank algorithm is a personalized ranking algorithm that builds on some of the concepts described above. We believe that a personalized approach is more reasonable than a global approach when ranking users in online social networks. This is due to how a globally high-ranking person might not be as important in the personal context of a user when compared to others who are personally closer and more important.

3. Design and Development of SocialRank

3.1 Overview

SocialRank utilizes the user's own historical actions to construct a social graph that captures the strength of connections between the user and people in his network. Consequently, when implemented in an online social network, SocialRank rates important people in a user's social graph and ranks information received by them accordingly. The higher the rank of a person, the more important is the information shared or created by that person. Actions made between a user and people in his social network are at the center of how SocialRank computes its ratings. User A's SocialRank of user B is defined as $SR(A \rightarrow B)$ and uses the set of all actions made from user A to user B to compute that value.

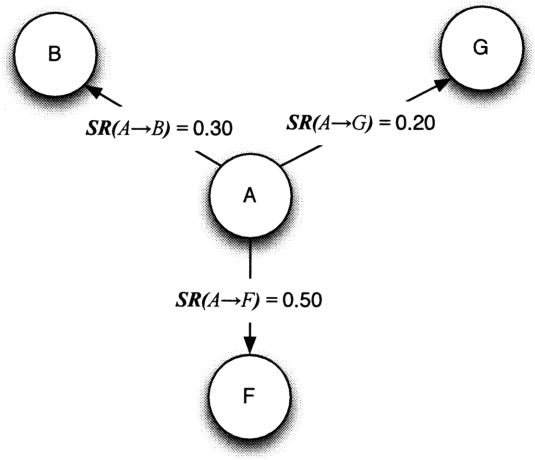


Figure 1. SocialRank Graph for user A.

3.2 Actions as Proxies of Relationship Strength

Online social networks enable users to interact with each other and share data. There are many types of online social networks. Some of them are centered around sharing news, some are built around exchanging photos, others are customized to publishing different geo-tagged data.

Depending on the theme and purpose of an online social network, there are many types of actions users could perform. We focus on actions that have an extended effect on other users or their data. For example, in a typical online social network, users could: Become friends with or follow other users, send messages to each other, comment on information shared by others, and answer questions posted by other users, etc.

We posit that any of the aforementioned online social network actions can serve as proxies for gauging the importance users place on other users in their social network. Evidently, email exchange has been used in previous social studies as a proxy for measuring strengths of relationships [5].

3.3 Determining Action Scores

The set of possible actions between two users in an online social network are many. Given a set of possible actions between two users, not all actions are equally significant. Hence, actions need to be scored. Given an action as input, the score assignment method should output a value that accurately reflect an action's significance. The action scoring mechanism also needs to be

modular so that SocialRank can be flexible enough to be applied to any type of online social network given how different online social networks have different views on how some actions are considered more significant than others.

We define $\alpha_{A \to B}$ to be an action made from user A to user B. The score of $\alpha_{A \to B}$ is computed as:

$$\text{Score}(\alpha_{A \to B}) = \delta \cdot \tau$$

where

δ = the action's directness factor

τ = the action's strength value

We will define the two factors δ and τ in the following two sections.

3.3.1 Action's Directness Factor (δ)

Given an action $\alpha_{A \to B}$, δ is a rational number between 0 and 1 that acts a scaling factor for the action's strength value (τ). The δ-value measures the level of directness in an action. Take for example two actions $\alpha 1_{A \to B}$ and $\alpha 2_{A \to B}$: $\alpha 1_{A \to B}$, is the action in which user A posts a comment on one of the articles created by user B, $\alpha 2_{A \to B}$, is the action in which user A posts a comment on the main profile page of user B. $\alpha 1_{A \to B}$ describes an action where A acted indirectly on user B — through one of B's articles—. $\alpha 2_{A \to B}$ describes an action where A acted directly on user B. The latter is more direct and hence signals more importance. This property is captured by δ when calculating the score of an action.

To assign a δ-value to an action, we argue that there are three levels of directness when classifying actions in online social networks:

1) Direct Actions

A direct action from user A to user B is defined as any kind of activity on the online social network where user A's action is directly targeted at user B. Examples include user A sending a message to user B, user A posting a personal note on the main profile page of user B, or user A marking user B as a "favorite", etc.

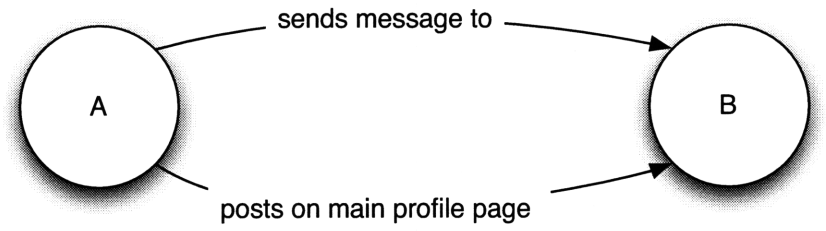


Figure 2. Two direct actions from user A to user B.

2) Indirect Actions

An indirect action from user A to user B is defined as any kind of activity on the online social network where user A's action is not directly targeted at user B but rather on an object owned by user B . Examples include user A commenting on a post by user B, user A marking an article published by user B as a favorite, or user A answering one of the questions posted by user B, etc.

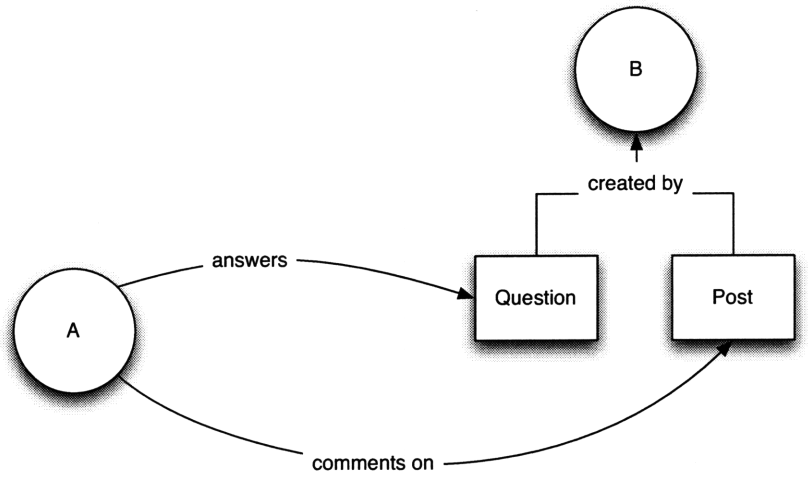


Figure 3. Two indirect actions from user A to user B.

3) Mutual Actions

Given three users A, B and C, a mutual action is defined as a pair of actions where both users A and B act mutually, either directly or indirectly, on user C. The pair of actions that constitute a mutual action should be of the same type and have the same target. Examples include user A and user B both sending a direct message to user C, or user A and user B both commenting on the same post created by user C.

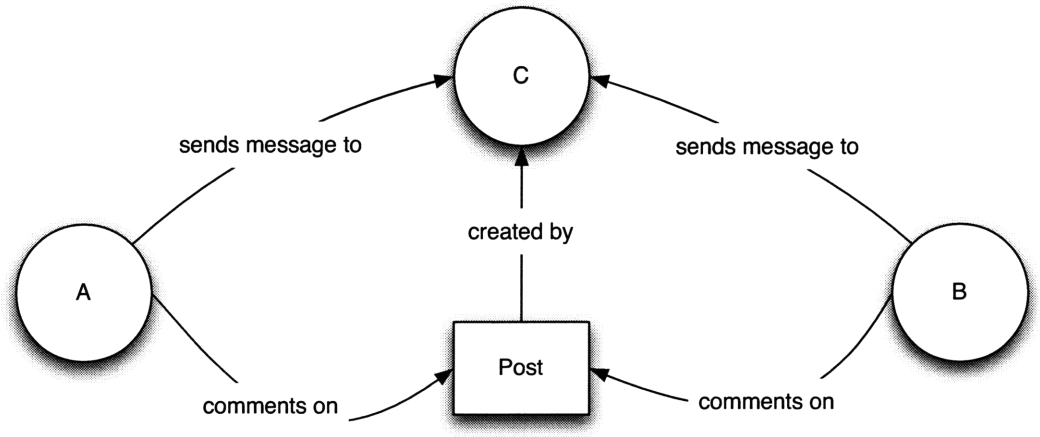


Figure 3. Two mutual actions between user A and user B.

The classification of actions based on the three levels aforementioned allows us to have three distinct δ-values: δ_{direct}, $\delta_{indirect}$, δ_{mutual}. Generally, in a typical online social network where direct actions carry more significance than indirect or mutual ones, it is fair to assume that $\delta_{direct} > \delta_{indirect} > \delta_{mutual}$. We could program the algorithm with preset values to the three δ-values (for example, $\delta_{direct} = 1$, $\delta_{indirect} = 0.5$, $\delta_{mutual} = 0.3$) or give the user more control in determining the δ-values appropriate for his own usage pattern.

There are interesting cases to consider where that general case ($\delta_{direct} > \delta_{indirect} > \delta_{mutual}$) might not be the best scenario. One example is social networks where users are interested in discovering other users who share their same behavior and interest in specific information. In that case, boosting δ_{mutual} to a higher value relative to the other δ-values would allow the ranking algorithm to present better ranked results to the users.

3.3.2 Action's Strength Value (τ)

Indeed, classifying actions as direct, indirect or mutual already gives us one level of granularity in determining scores for actions. But we still need to evaluate a numeric value that reflects the strength of the action itself. This could be done using different heuristics. Time duration of an action is one heuristic; the more time an action requires, the higher its strength value. Another would be the size of an action represented by the number of characters involved.

We follow a more general approach of classifying actions based on their distinct types (for example: commenting, posting, answering actions, etc.) and assigning a strength value τ to that general type that gets inherited by all individual actions of that type.

Again, we could program the algorithm with preset τ-values to different action types (for example, $\tau_{comment} = 5$, $\tau_{post} = 8$, $\tau_{answer} = 6$) or give the user more control in determining the τ-values appropriate for his own usage pattern.

To summarize:

- Given a direct action $\alpha_{A \to B}$ with a strength value τ, its score is calculated as: $\delta_{direct} \cdot \tau$
- Given an indirect action $\alpha_{A \to B}$ with a strength value τ its score is calculated as: $\delta_{indirect} \cdot \tau$
- Given a pair of mutual actions $\alpha_{A \to B}$ and $\alpha_{C \to B}$ (whether they are a pair of indirect actions or a pair of direct actions) with equal values of τ, the score for the pair of mutual actions is calculated as: $\delta_{mutual} \cdot \tau$

3.3 Calculating SocialRank Values

We model the online social network as a weighted directed graph $G = (V,E)$. Where V, the set of nodes, represent users and E, the set of edges, represents some relationship between users. We define the weight function for every edge connecting two nodes A and B to be SR(A→B).

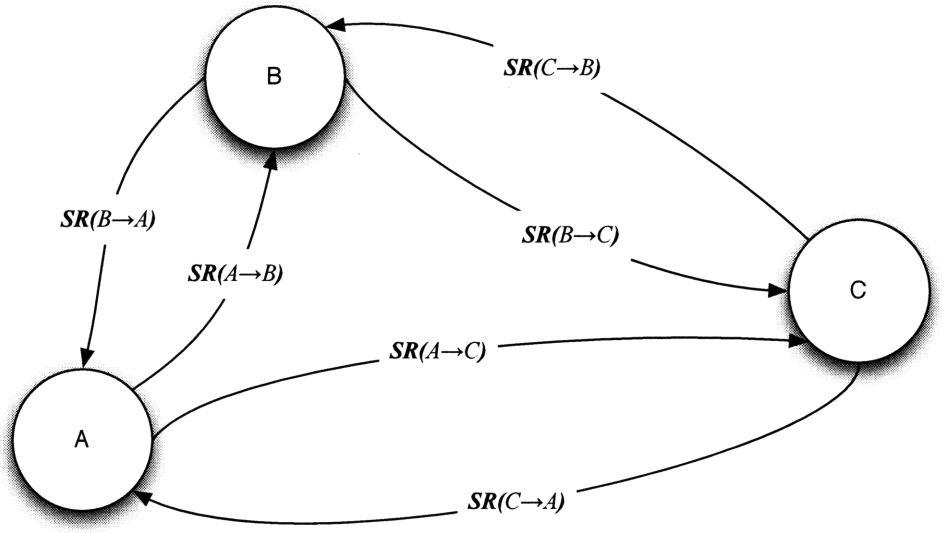


Figure 4. SocialRank Graph for all users.

When using action scores to calculate SR(A→B), we need to normalize action scores by dividing them by the total sum of all action scores made by user A to all other users. User A's SocialRank value of user B is defined as:

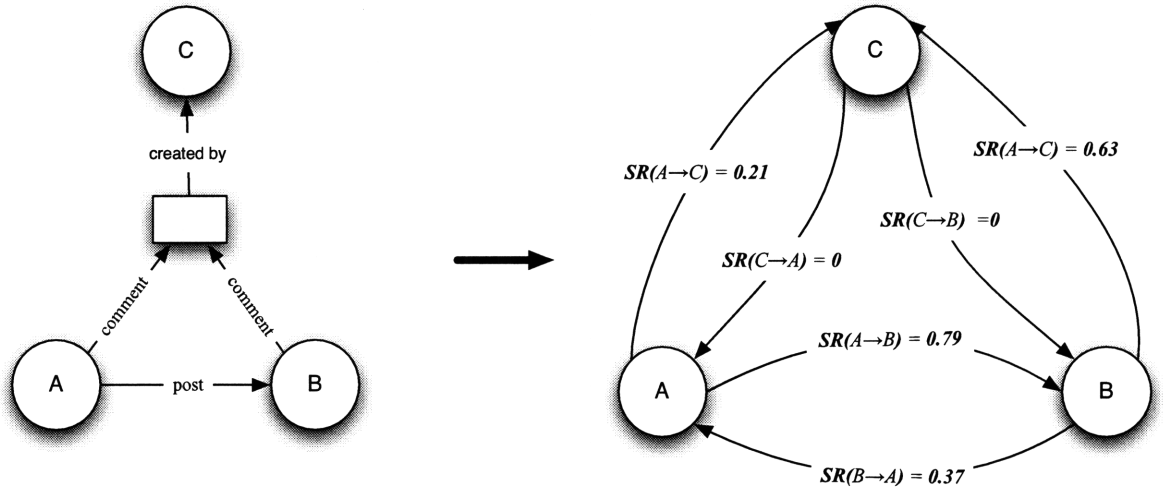
$$SR(A \rightarrow B) = \sum Score(\alpha_{A \rightarrow B}) / \sum Score(\alpha_A)$$

where

$\sum Score(\alpha_{A \rightarrow B})$ = Sum of scores of all actions from user A to user B

$\sum Score(\alpha_A)$ = Sum of scores of all actions from user A to all other users

The SocialRank values are cached until they are updated every preset time interval. The figure below shows an example of how SocialRank determines the personalized ranks for three users based on their actions.



$$SR(A \rightarrow C) = 5 * 0.5 / [\,(5 * 0.5) + (8 * 1) + (5 * 0.3)\,] = \mathbf{0.21}$$

$$SR(A \rightarrow B) = [\,(8 * 1) + (5 * 0.3)\,] / [\,(5 * 0.5) + (8 * 1) + (5 * 0.3)\,] = \mathbf{0.79}$$

$$SR(B \rightarrow C) = 5 * 0.5 / [\,(5 * 0.5) + (5 * 0.3)\,] = \mathbf{0.63}$$

$$SR(B \rightarrow A) = 5 * 0.3 / [\,(5 * 0.5) + (5 * 0.3)\,] = \mathbf{0.37}$$

action type	τ-value
post	8
comment	5

τ-values

action directness	δ-value
δ direct	1
δ indirect	0.5
δ mutual	0.3

Figure 5. An example showing how the set of direct, indirect and mutual actions in the graph on the left were used to compute SocialRank values SR(A→C) and SR(A→B) in the graph on the right.

4. Implementation of an Online Social Network with SocialRank

4.1 Building an Online Social Network

In this chapter we build a fully functional online social network that uses SocialRank. The implementation of the online social network is meant as a proof of concept of the design of the SocialRank algorithm. The platform includes a suite of typical features found in social networking websites. Users can register for accounts, create profile pages, upload personal avatars, follow friends, create blog posts and forum posts, comment on posted items, ask questions, and answer questions, etc. Moreover, it allows different user-generated data to be tagged and rated. The home page—for every user—lists the latest updates from people in his social network in a newsfeed style.

4.2 System Architecture and Specifications

The online social network is a client-server web application programmed using Ruby on Rails[2] . Ruby on Rails is an open source MVC-based web application framework. The online social network is hosted on a server using Mongrel[3] as a web server, and SQLite[4] as a database server.

---

[2] Ruby on Rails Framework: http://rubyonrails.org

[3] Mongrel: http://mongrel.rubyforge.org/

[4] SQLite: http://www.sqlite.org/

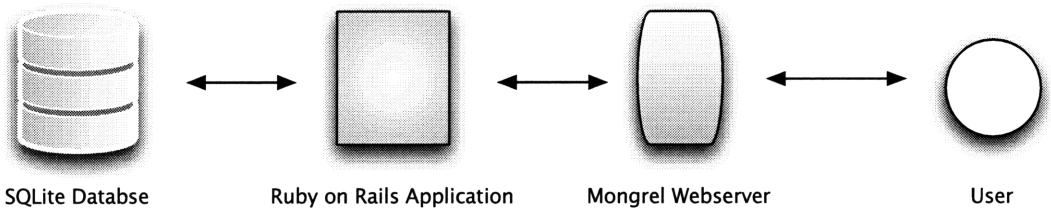


Figure 6. System architecture for the online social network.

4.3 Data Model for the Online Social Network

Rails uses ActiveRecord[5] to implement Object Relational Mapping (ORM). ORM is a programming technique that maps objects in object-oriented programming languages to database records in an SQL table. Hence, the schema of each SQL table in the database maps to a class model.

Provided in the table below is a brief description of each of the data models used in the online social network and the names of the tables that hold SQL records of that model (refer to appendix A for the full listing of SQL tables and the schema used to build the online social network's database):

Class Model	SQL Table	Description
User	Users	Represents a user with different attributes such as name, login, hashed password and total sum of scores of actions made by the user.

[5] ActiveRecord: http://ar.rubyonrails.org/

Class Model	SQL Table	Description
Action	Actions	Represents an action. Attributes include a reference to the user who the action was made from, a reference to the user who the action was made to (directly, indirectly or mutually) and a score for the action.
SocialRanks	SocialRanks	Represents the collection of SocialRank values. Attributes include a reference to the ranked user, a reference to the ranking user, and the value of the ranking.
Post	Posts	Represents a post with different attributes such as title, body, and a reference to the user who created the post.
Comment	Comments	Represents a comment with different attributes such as body, a reference to the user who created the comment, and a reference to the parent object of the comment.
Question	Questions	Represents a question with different attributes such as title, body, a reference to the user who created the question and a reference to the parent object of the question.
Answer	Answers	Represents an answer with different attributes such as body, a reference to the user who created the answer and a reference to the parent question-object of the answer.
Friendship	Friendships	Represents a friendship connection between two users with attributes such as references to the user, the friend, and their friendship status.
Category	Categories	Represents a category which a question, answer or post can be listed under.
Rating	Ratings	Represents a rating with different attributes such as value, and a reference to the parent object which the rating is applied to.
Tag	Tags	Represents a tag which a given user could apply to any post, comment, question, or answer.

Table 1. Class model and SQL tables used in building the online social network.

4.4 Integrating SocialRank

The actions table described above holds records of all the actions created on the online social network. Below is the full table schema for the actions table:

Field Name	Field Type	Description
id	integer	action's unique identifier [Primary key]
from_user_id	integer	reference to user who is the source of the action [Foreign key]
to_user_id	integer	reference to user who is the target of the action. [Foreign key]
object_id	integer	Reference to the object that was created/modified by this action. [Foreign key]
object_type	string	the type of object that was created/modified by this action.
object_parent_id	integer	Reference to the parent of the object that was created/ modified by this action
object_parent_type	string	the type of the parent of the object that was created/modified by this action
score	float	the value of the action's score
hash	string	a string that concatenates: to_user_id + object_type + object_parent_id + object_parent_type. Used to quickly find mutual actions when scanning the table.
created_at	datetime	date and time of when the action was made
description	string	text description of the action

Table 2. Actions table scheme.

Every time an action is made, a new record is added to the actions table. We show below a partial sample view of the actions table after some activity on the online social network—highlighting

only the important parts of the records:

From	To	object_id	object_ type	object_ parent_ id	Object_ parent_ type	Score	Hash	Description
1	3	20	Post	3	User	8	3 , Post, 3, User	"Jana posted a message to Abdul"
1	5	27	Comment	11	Post	4	5, Comment, 11, Post	"Jana commented on Bill's post"
2	4	33	Answer	72	Question	7	4, Answer, 72, Question	"Steve answered Tom's question"

Table 3. Sample partial view of the actions table

The table above shows three actions. The first one captures the direct action where Jana (user_id: 1) sent the message (message_id: 20) to Abdul (user_id: 3). The action was given a score of an 8 using the mechanisms described in the last chapter. The action's hash is a string that concatenates together the action's target (3), its object type (Post), its object parent id (3) and its object parent type (User); resulting in "3, Post, 3, User". This string helps when scanning the actions table to find mutual actions between users.

The second action captures the indirect action where Jana (user_id: 1) created the comment (comment_id: 27) on post (post_id: 11) by Bill (user_id: 5). The action was given a score of a 4. Information for the third action is inferred similarly.

Given the table of actions, we run the SocialRank procedure to determine the SocialRank values for users in the online social network. The SocialRank procedure takes as input the table of actions. A Ruby on Rails-like pseudocode for the SocialRank procedure is provided below:

```
user_scores_sums = Hash.new
user_friends_scores_sums = Hash.new
For each user in Users
        user_scores_sums[user] = 0.0
        user_friends_scores_sums[user] = Hash.new
        Let userActions = the set of all actions made by user

        For each friend in {Users - user}
                users_friends_scores_sums[user][friend] = 0.0
                direct_and_indirect_actions_sum = 0.0
                mutual_actions_sum = 0.0

                Let direct_and_indirect_actions_sum = the sum of scores of direct and indirect actions
                from user to friend

                Let user_friends_scores_sums[user][friend] += direct_and_indirect_actions_sum

                Let friendActions be the set of all actions made by friend
                For each user-action in userActions
                        Unless user-action.hash has already been seen before
                                Let friend-action = the first action where
                                friend-action.hash = = user-action.hash

                                Let mutual_action_score = the score of the pair of mutual
                                actions friend-action and user-action

                                Let mutual_actions_sum += mutual_action_score
                        End
                End
                user_friends_scores_sums[user][friend]  += mutual_actions_sum
                user_scores_sums[user] += user_friends_scores_sums[user][friend]
        End
End

For each pair of users (user1, user2)
SocialRank(user1,user2) = user_friends_scores_sums[user1][user2] / user_scores_sums[user1]
End
```

Figure 7. Pseudocode for SocialRank procedure

After running the procedure, the SocialRank table gets populated with SocialRank values for all pairs of users on the online social network. A sample partial view of what the SocialRank table looks like after running the SocialRank procedure is included below:

From	To	SocialRank
Jana	Abdul	0.4
Jana	Bill	0.2
Bill	Jana	0.7

Table 4. Sample partial view of the SocialRank table

5. Conclusion

5.1 Summary of Results

After implementing the online social network discussed in Chapter 4, we deployed two versions of it in parallel: the first one used SocialRank to rank friend updates in the user's newsfeed, the second one did not use SocialRank. We had two groups of users separately using each of the two deployments. Group 1 tested the deployment that used SocialRank, while group 2 tested the deployment that did not use it. The experiment ran for 4 weeks and each of the two groups had 5 users. The δ-values to determine actions scores were $\delta_{direct} = 1$, $\delta_{indirect} = 0.5$, $\delta_{mutual} = 0.3$. The τ-values where $\tau_{friendship} = 9$, $\tau_{post} = 8$, $\tau_{answer} = 6$, $\tau_{comment} = 4$. The SocialRank procedure ran once every 6 hours to update values in the SocialRank table.

The lack of a large user base did not enable SocialRank to have enough critical mass to be fully effective. But we did notice two interesting patterns:

1) On average, users in group 1 spent less time on the newsfeed than users in group 2.
2) On average, users in group 1 had a higher click-through rate on the newsfeed items than users in group 2.

In spite of the small number of users involved in the experiment, we believe that these findings support the idea that SocialRank could indeed be useful in ranking information in online social networks. It does so by potentially lowering the amount of time a user spends on finding personally relevant and important information. It also shows that when users are presented with

more important information first on a list—in this case, important friend updates appearing first on the newsfeed—users are more likely to click on that information; signaling that users' attention span and click-potential is limited to the first items on a list of information.

5.2 Future Work

A fully scalable ranking mechanism for online social networks is a complex problem and much remains to be done. We plan on improving the efficiency of the ranking procedure and allowing it to scale for larger numbers of users. Moreover, another area of improvement is dynamically determining when to rerun the SocialRank procedure to update rank values. We are also working on a hybrid global-personalized ranking algorithm that takes into account the global ratings determined by a global variant of SocialRank and factors those ratings in when determining personalized SocialRank values for users.

Lastly, we are working on implementing SocialRank as a webservice that aggregates all friend updates from different online social networks. The webservice would rank updates from friends on all online social networks—even when the same friend has different screen names on different platforms—and display ranked information in one unified interface. Given the enormous amounts of social interaction data available in social networking websites, we believe SocialRank's approach of using actions as proxies of relationship strength is a step in the right direction but only touches the surface of what is possible. Ranking algorithms for online social networks is an uncharted path and a very rich field of experimentation and research ideas. SocialRank is only the beginning of more interesting work in that field.

A. Database Schema for Online Social Network

Provided below is a full listing of the SQL tables used in building the online social network along with their detailed database schema :

Users

Field Name	Field Type	Description
id	integer	user's unique identifier. [Primary key]
login	string	user's unique login handler
first_name	string	user's first name
last_name	string	user's last name
email	string	user's email address
city	string	user's city
country	string	user's country
gender	string	user's gender
kind	string	kind of user class (admin, power user, etc.)
preferences	string	A hash structure with different user preferences
locale	string	user's language choice
birthday	date	user's birthdate
crypted_password	string	user's password encrypted using SHA1
salt	string	user's password salt

Field Name	Field Type	Description
activation_code	string	randomly generated activation code sent to user's email upon signup to confirm user's details
activated_at	datetime	date of when current user was activated
posts_count	integer	cached count of posts made by user
action_scores_sum	integer	total sum of scores for all the user's actions
created_at	datetime	date and time when user's record was created
updated_at	datetime	date and time when user's record was last updated

Posts

Field Name	Field Type	Description
id	integer	post's unique identifier [Primary key]
title	string	post's title
body	text	post's body
postable_id	integer	reference to post's parent object [Foreign key]
postable_type	string	type of post's parent object
category_id	integer	reference to post's category [Foreign key]
comments_count	integer	cached count of post's comments
user_id	integer	reference to user object which created this post [Foreign key]
created_at	datetime	date and time post was created
updated_at	datetime	date and time post was last updated

Comments

Field Name	Field Type	Description
id	integer	comment's unique identifier [Primary key]
body	text	comment's body
commentable_id	integer	reference to comment's parent object [Foreign key]
commentable_type	string	type of comment's parent object
user_id	integer	reference to user that created comment [Foreign key]
created_at	datetime	date and time of when comment was created
updated_at	datetime	date and time of when comment was last updated

Friendships

Field Name	Field Type	Description
id	integer	friendship's unique identifier [Primary key]
user_id	integer	reference to the befriending user [Foreign key]
friend_id	integer	reference to the befriended user [Foreign key]
status	string	status of friendship connection
created_at	datetime	date and time friendship connection was created
updated_at	datetime	date and time friendship connection was last updated

Questions

Field Name	Field Type	Description
id	integer	question's unique identifier. [Primary key]
title	string	question's title
body	text	question's body
questionable_id	integer	reference to question's parent object [Foreign key]
questionable_type	string	type of question's parent object
category_id	integer	reference to category [Foreign key]
answers_count	integer	cached count of question's answers
user_id	integer	reference to user that created this question [Foreign key]
created_at	datetime	date and time question was created
updated_at	datetime	date and time question was last updated

Answers

Field Name	Field Type	Description
id	integer	answer's unique identifier [Primary key]
body	text	answer's body
question_id	integer	reference to answer's question [Foreign key]
user_id	integer	reference to user that created answer [Foreign key]
created_at	datetime	date and time of when answer was created
updated_at	datetime	date and time of when answer was last updated

Categories

Field Name	Field Type	Description
id	integer	category's unique identifier. [Primary key]

Field Name	Field Type	Description
name	string	category's name
parent_id	integer	reference to parent category, if any [Foreign key]
created_at	datetime	date and time of when category was created
updated_at	datetime	date and time of when category was last updated

Actions

Field Name	Field Type	Description
id	integer	action's unique identifier [Primary key]
from_user_id	integer	reference to user who is the source of the action. [Foreign key]
to_user_id	integer	reference to user who is the target of the action. [Foreign key]
object_id	integer	reference to the object that was created/modified by this action. [Foreign key]
object_type	string	the type of object that was created/modified by this action.
object_parent_id	integer	Reference to the parent of the object that was created/modified by this action
object_parent_type	string	the type of the parent of the object that was created/modified by this action
score	float	the value of the action's score
hash	string	a string that concatenates: to_user_id + object_type + object_parent_id + object_parent_type. Used to quickly find mutual actions when scanning the table.
created_at	datetime	date and time of when the action was made
description	string	text description of the action

Ratings

Field Name	Field Type	Description
id	integer	rating's unique identifier [Primary key]
user_id	integer	reference to the user creating the rating [Foreign key]
value	float	value of the rating
rating_type	string	type of the rating
ratable_id	integer	reference to the object being rated [Foreign key]
ratable_type	string	type of object being rated
created_at	datetime	date and time of when the rating was created
updated_at	datetime	date and time of when the rating was updated

SocialRanks

Field Name	Field Type	Description
id	integer	ranking's unique identifier [Primary key]
from_user_id	integer	reference to the user who is the source of the ranking [Foreign key]
to_user_id	integer	reference to the user who is the target of the ranking [Foreign key]
socialRank	float	value of the rank

Tags

Field Name	Field Type	Description
id	integer	tag's unique identifier [Primary key]
name	string	tag's name

Taggable

Field Name	Field Type	Description
id	integer	unique identifier [Primary key]
tag_id	integer	reference to tag [Foreign key]
taggable_id	integer	reference to the object being tagged [Foreign key]
taggable_type	string	type of the object being tagged

6. References

[1] P. Resnick and H.R. Varian. Recommender Systems. Communications of the ACM, 40(3):56–58, 1997.

[2] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford University, 1998

[3] G. Jeh and J. Widom. Scaling personalized web search. In Proc. of the 12th Intl. WWW Conference, 2003.

[4] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In Proc. 12th International World Wide Web Conference, Budapest, Hungary, May 2003.

[5] J. Tyler, D. Wilkinson, and B. Huberman. Email as spectroscopy: automated discovery of community structure within organizations. In M. Huysman, E. Wenger, and V. Wulf, editors, Communities and Technologies, pages 81–96, Deventer, the Netherlands, 2003. Kluwer.