Social Networking Study on DeviantArt Artificial Network Algorithm

CmpE 492 Project

By: Ferhat Elmas
Student Id: 2006101102
Advisor: Haluk Bingöl

04.05.2011
Version 4.0

Version History

Version	Date	Explanation
1.0	30.03.2011	Algorithm Design and Specification
2.0	22.04.2011	Some errors (keyword, logic, notation) are fixed, the details of the distributions of the deviant and deviation generation and a table of some basic properties of distributions are added, algorithm is improved and details are added
3.0	27.04.2011	Parameter Analysis is added
4.0	04.05.2011	Notation is improved, appendix is added, design of the chapters is changed

Abstract

Influential users are very important in the network in terms of their ability to change the distribution of the interconnections in the network. In our case, the favorite relation of the DeviantART network is investigated. Favorite relation is a interconnection between users and resources. If a user likes a resource that he views, he adds it into his list and a favorite relation is constructed. Moreover, favorite relation is so effective to increase or decrease the value of a resource in the community. If a well known user adds a resource into his list, the value of the resource increases rapidly because a lot of users follow the well known user and imitate what the well known user does and other users also add this resource into their list, by the way this resource is entered a lot of favorite lists. In this paper, an algorithm to find the most influential user respect to favorite relation is expressed.

On the other hand, this algorithm is a result of some set of the assumptions. Therefore, an artificial network is created to test the power of the algorithm and the details of the generation and assumptions of the articial network are defined in the data generation phase.

Contents

Chapter 1

Introduction

Networks are all around us. Your circle of friends; the route you take to work; cars on the road; tiny neurons firing inside your brain; ecological systems. Understanding how elements within these networks interact has fascinated scientists for decades, with pivotal results such as 6-degrees of separation, Small World Networks, to name just a few, having application to easing road congestion, enhancing computer systems and understanding human social processes.

However, all such analysis are insufficient, yet there is a need of finding the central and/or influential nodes. This project is interested in closing the gap by presenting an algorithm to find the the important nodes. Therefore, DeviantART network is chosen because DeviantART is one of the biggest networks in the online world and there is a little research on this network.

Different algorithms can be easily developed by using different heuristics but we can't sure that the developed algorithm is the desired algorithm. Therefore, we need a network for testing.

In the first part generation of the test network is explained. Secondly, how can we find the most influential user in the network if we don't know anything about the internals of the network, this question is answered by presenting an alogrithm. Finally, test results are shown.

Presented new method provides fascinating new directions for the development of new applications and new systems on online social networks and for a better understanding of social processes such as influence, trust and information spreading.

Chapter 2

Framework

There are mainly three components to represent the DeviantART network, namely, user, resource and favorite relation. In the following pages, user will be used instead of deviant and resource will be used instead of deviation to decrease the confusion between deviant and deviation.

2.1 User

User object is defined to represent the users of the DeviantART.

There are two properties, namely, *id* and *authority*.

$$User = \{(id, authority)\}$$

id is defined to select the users uniquely and varies from 1 to the number of the users. $\mathcal{U}$ is the set of the users and U is the size of the $\mathcal{U}$.

$$|\mathcal{U}| = U$$

$$\mathcal{U} = \{u_1, u_2, \ldots, u_U\}$$

authority is to specify the influentiality of the user. *authority* is a function from $\mathcal{U}$ to $[0, 1)$ and returns a value from $[0, 1)$ for each user. The more influential user is, the closer to 1 *authority* is.

$$a : \mathcal{U} \rightarrow [0, 1)$$

2.2 Resource

Resource object is defined to represent the artworks (pictures, drawings, clips, etc.) in the DeviantART.

There are four properties, namely, *id*, *quality*, *favorite list*, *time list*.

$$Resource = \{(id, quality, favorite\ list, time\ list)\}$$

id is defined to select the resources uniquely and varies from 1 to the number of the resources. $\mathcal{R}$ is the set of the resources and R is the size of the $\mathcal{R}$.

$$|\mathcal{R}| = R$$

$$\mathcal{R} = \{r_1, r_2, \ldots, r_R\}$$

quality is to specify the art-criticique of the artwork. *quality* is a function from $\mathcal{R}$ to $[0, 1)$ and returns a value from $[0, 1)$ for each resource. The higher quality resource is, the closer to 1 *quality* is.

$$q : \mathcal{R} \rightarrow [0, 1)$$

favorite list holds the ids of the users that added this resource to their favorite list.

$|fl(r_i)| = k \Longrightarrow fl(r_i) = \{u_1, u_2, \ldots, u_k\}$ where $r_i \in \mathcal{R}$ and $\{u_1, u_2, .., u_k\} \subseteq \mathcal{U}$

D is the length of the days that have passed from the start of the DeviantART to present.

$$|\mathcal{D}| = D$$

$$\mathcal{D} = \{d_1, d_2, \ldots, d_D\}$$

Corollary to *favorite list, time list* holds the days when these users added the resource. If a user added the resource before than others, his index will be smaller than others.

$|tl(r_i)| = k \Longrightarrow tl(r_i) = \{d_1, d_2, .., d_k\}$ where $r_i \in \mathcal{R}$ and $d_1 \leq d_2 \cdots \leq d_{k-1} \leq d_k \leq D\}$

$|flr_i| = |tl(r_i)|$ must hold since they are respective lists of each other

2.3 Network

Network object is defined to represent favorite relation between users and resources in the DeviantART. Favorite relation maps users onto resources so each user chooses a subset of the resources set.

$$Favorite = \{user, resource, day\}$$

$$\mathcal{F} = \{f(u_i, r_j, d_k) | u_i \in \mathcal{U}, r_j \in \mathcal{R}, d_k \in \mathcal{D}\}$$

We can imagine favorite relation ($\mathcal{F}$) as a bipartite matching where users and resources are nodes and adding into favorite list is the edge because there are two partitions composed of user ($\mathcal{U}$) and resource ($\mathcal{R}$) sets and there will be a link between partitions, not inside the partitions since users can add resources but can't follow the other users since we are just interested in favorite relation or resources can't be linked with other resources . Therefore, there will be an edge f_{ijk} only if u_i adds the r_j into his list on the day d_k.

Chapter 3

Data Generation

3.1 Distributions of the Generation

We definitely need a probability distribution to generate *authority* values of the users and *quality* values of the resources. Since a lot of users should have low *authority* values and a lot of resources should have low *quality* values, we need sharply decreasing probability distributions. Therefore, we chose two distributions: $2 - 2x$ and the family of $\frac{1}{x^n}$ where $n \geq 1$ and $x \in [0, 1)$. The first distribution is a real probability distribution but second one isn't since

$$\int_{-\infty}^{\infty} (2 - 2x)\, dx = 1$$

$$\int_{-\infty}^{\infty} \frac{1}{x^n}\, dx \neq 1$$

Therefore, there is a check while generating random stream with these distributions. If generated random number isn't in $[0, 1)$, repeat the procedure to get a number in the range.

Inverse transform method is used to generate random variates from these distributions.

Random variate calculation of the $(2-2x)$ distribution:

$$\begin{aligned}\int_{-\infty}^{x}(2-2x)\,dx = 2x-x^2 &= y \text{ where } y \in [0,1) \text{ is a random number}\\ x^2-2x+1 &= 1-y\\ (x-1)^2 &= 1-y\\ x-1 &= -\sqrt{(1-y)}\\ x &= 1-\sqrt{(1-y)}\end{aligned}$$

Since y is randomly distributed, $1-y$ is also randomly distributed so we can use y instead of $1-y$ and generator formula becomes

$$1-\sqrt{y} \text{ where } y \text{ is a random number}$$

Random variate calculation of the family of the $(\frac{1}{x^n})$ distribution:

$$\begin{aligned}\int_{-\infty}^{x}(\frac{1}{x})\,dx = \ln x &= y \text{ where } y \in [0,1) \text{ is a random number and } n=1\\ x &= e^y\end{aligned}$$

However, by this formula we can calculate numbers which are outside of the range

$$x \notin [0,1)$$

since the distribution isn't a probability distribution. We can check the generated values and take the values that are in the range but this is also so costly.

Therefore, we thought about using minus logarithm of the random numbers to simulate the distribution and by dividing with n, whole family can be represented.

Generator formula becomes

$$\frac{-\log y}{n} \text{ where } y \text{ is a random number and } n \geq 1$$

Moreover, we yet need to check each generated variate since this can also generate variates outside the range $[0, 1)$ but the probability of this is much smaller than the previous generator.

While n is increasing, the mean of the distribution decreases to zero which means that a lot of users have small *authority* values and have a minor effect on the network.

$$\lim_{n \to +\infty} \frac{-\log y}{n} = 0$$

3.2 Generation of the Users

id property is incremented by one in each user generation. The value of the *authority* property is generated by the chosen distribution function.

$$u_i = \{(i, a_i) | 1 \leq i \leq U, a_i \in [0, 1)\}$$

3.3 Generation of the Resources

id property is incremented by one in each resource generation. The value of the *quality* property is generated by the chosen distribution function.

$$r_i = \{(i, q_i) | 1 \leq i \leq R, q_i \in [0, 1)\}$$

3.4 Matching of Deviants and Deviations

We have users and resources

$$\mathcal{U} = \{u_i | 1 \leq i \leq U\}$$
$$\mathcal{R} = \{r_i | 1 \leq i \leq R\}$$

By matching these two partitions we get favorite relation.

$$\mathcal{F} = \{f_{ijk} : (u_i, r_j, d_k) | u_i \in \mathcal{U}, r_j \in \mathcal{R}, d_k \in \mathcal{D}\}$$

Matching algorithm:

For $d = 1$ to D (3.1)

$t_{max} \leftarrow \mathcal{N}(\mu, \sigma^2)$ (3.2)

For $t = 1$ to t_{max} (3.3)

$u_i \leftarrow \text{Uniform}[1, U]$
$r_j \leftarrow \text{Uniform}[1, R]$ (3.4)

if $f(u_i, r_j, d_k) \notin \mathcal{F}$ where $\exists d_k \in \mathcal{D}$ (3.5)

$$\begin{aligned} fl(r_j) &= fl(r_j) \cup u_i \\ tl(r_j) &= tl(r_j) \cup d \\ f(u_i, r_j, d) &\in \mathcal{F} \text{ with } \mathcal{P}(q(r_j)) \end{aligned} \quad (3.6)$$

$$\begin{aligned} fl(r_j) &= fl(r_j) \\ tl(r_j) &= tl(r_j) \\ f(u_i, r_j, d) &\notin \mathcal{F} \text{ with } 1 - \mathcal{P}(q(r_j)) \end{aligned} \quad (3.7)$$

$$q(r_j) = \max\{q(r_j, a(u_i))\} \text{ if } f(u_i, r_j, d) \in \mathcal{F} \quad (3.8)$$

end

end

end

There are *three* parameters used in the data generation, namely, *simulation length* (D), α and β. *simulation length* is self descriptive and is used to specify the number of the days to generate matching data.

We call a *transaction* that a user looks a resource and whether s/he adds it into her/his *favorite* list and denoted by t. α is used to calculate the mean (μ) of the normal distribution which is used to generate the number of daily transactions (t_{max}). β is used to scale the mean (μ) which is calculated by using α, to calculate the variance (σ^2) of the normal distribution and the calculated μ and σ^2 are used in the equation (3.2). The details of calculation of each as follows:

$$\mu = \alpha \times U \times R \tag{3.9}$$

$$\sigma^2 = \beta * \mu \tag{3.10}$$

Here are the step by step procedure explained:

- Calculate the mean (μ) of the normal distribution which is used to generate the number of daily transactions by the equation (3.8).
- Calculate the variance (σ^2) of the normal distribution which is used to generate the number of daily transactions by the equation (3.9).
- Create a pseudo variate generator for the normal distribution with parameters μ and σ^2.
- Set the simulation day to one
- Repeat as much as *simulation length* (D) - equation (3.1)
  - Generate a number for daily transactions (t_{max}) - equation(3.2)
  - Repeat as much as t_{max} - equation (3.3)
    - Randomly choose a user u_i - equation(3.4)
    - Randomly choose a resource r_j - equation(3.4)
    - Check the *favorite list* ($fl(r_j)$) of the resource r_j for the chosen user (u_i) - equation (3.5)
    - If chosen user u_i has already added the chosen resource r_j into her/his list, there is no action and continue for new transaction t. In other words, the favorite list of the resource ($fl(r_j)$) already contains the randomly chosen deviant - equation(3.5) is false

* Else equation (3.5) is true and user u_i hasn't added the chosen resource r_j yet, so give a chance to deviant to add the deviation into his list. User u_i adds the resource r_j into his list with a propability proportional to its quality ($q(r_j)$) - equation (3.6) and (3.7)
* If chosen user u_i adds the chosen resource into his list by equation (3.6), update the *quality* of the resource (r_j) - equation (3.8). If influential users add the resource into their list, the *quality* of the resource ($q(r_j)$) increases and so the probability increases so that other users can add the resource r_j when they meet the resource r_j

Chapter 4

Analysis of the Generated Data

4.1 Algorithm

We have explained how to generate data, now we will explain the how to analyze the data. Analysis phase doesn't know the internal dynamics of the generated data but since we know true values, we can easily check their correctness and the power of the algorithm.

Firstly, create a *map* because at the end of anaysis this data structure will be filled. This data structure holds a flag for each user-resource pair and this number specifies whether this user is the most influential user for the resource. Therefore, we can define this data strucure a function from users and resources to $\{0, 1\}$ since a user is the most influential user or not.

$$h : \mathcal{U} \times \mathcal{R} \to \{0, 1\}$$

$$\mathcal{H} = \{h(u_i, r_j) | u_i \in \mathcal{U}, r_j \in \mathcal{R}\}$$

$$hsum(u_i) = \sum_{j=1}^{R} h(u_i, r_j)$$

$$vsum(u_i) = \sum_{j=1}^{R} \sum_{k=1}^{D} f(u_i, r_j, d_k)$$

The bigger $hsum(u_i)$ is, the more influential user u_i may be. After scaling with the size *favorite list* of resources and the number of resources that users has added their list, the distribution of $h(u_i)$s is correlated with the distribution that is used to generate the users.

Frequency list of a resource is the distribution of the numbers of the users that have added this resource into their list. Therefore, frequency list can be represented as a function from the days of the data to the number of users since at most all users can add the resource in a day.

$$freq : [1, D] \rightarrow [0, U]$$

$$freq(r_j, k) = \sum_{i=1}^{U} f(u_i, r_j, d_k) \text{ where } 1 \leq k \leq D$$

$$freqList(r_i) = \{freq(r_i, k) | 1 \leq k \leq D\}$$

Moreover, we have two more parameters, namely, *derivation window* and *setup window*. The sizes of these windows are effected by the length of the data and daily transactions. Therefore, these window sizes must be chosen while taking consideration the length (D) and the average of the daily transactions (μ). Since we can calculate μ and D in real data, we can easily agree on the window sizes.

$$\begin{aligned} dw &\leftarrow \textit{derivation window} \\ sw &\leftarrow \textit{setup window} \end{aligned}$$

The algorithm to find the most influential user:

Foreach $r_j \in \mathcal{R}$

$p(r_j) = 0, inc(r_j) = 0$

for $k = 1$ to $D - dw$

$p(r_j) = k$ if $\begin{cases} p(r_j) \leq \sum_{i=k}^{k+dw} freq(r_j, i) \\ freq(r_j, p(r_j)) - freq(r_j, p(r_j) - 1) \leq freq(r_j, k) - freq(r_j, k-1) \end{cases}$

for $k = p(r_j - sw)$ to $p(r_j)$

$\mathcal{H} = \mathcal{H} \cup \{h(fl(r_j)[m], r_j) = 1\}$ where $tl(r_j)[m] = p(r_j), 0 \leq m \leq |fl(r_j)|$

endfor

endforeach

$maxList = \max\{fl(r_j) | r_j \in \mathcal{R}\}$

for $i = 1$ to U

for $j = 1$ to R

$h(u_i, r_j) = \frac{h(u_i, r_j) \times fl(r_j)}{maxList}$

endfor

endfor

$u_{influential} = max\{\frac{hsum(u_i)}{vsum(u_i)} | 1 \leq i \leq U\}$

4.2 Parameter Analysis

- **User Distribution Analysis**

$$U = 1,000; R = 10,000; D = 1,000; dw = 15; sw = 15; \alpha = 0.005; \beta = 2; 12 \ Resource \ Distribution; 100 \ Runs$$

n	μ_{real}	$\mu_{estimated}$	Δ
0	0.9713	0.8285	0.1428
2	0.9966	0.6147	0.3819
4	0.9876	0.9061	0.0814
6	0.9503	0.9474	0.0029
8	0.8540	0.8491	0.0049
10	0.7374	0.7363	0.0010

- **Resource Distribution Analysis**

$$U = 1,000; R = 10,000; D = 1,000; dw = 15; sw = 15; \alpha = 0.005; \beta = 2; 12 \ User \ Distribution; 500 \ Runs$$

n	μ_{real}	$\mu_{estimated}$	Δ
0	0.6213	0.6205	0.0008
2	0.6301	0.6288	0.0013
4	0.6303	0.6300	0.0003
6	0.6161	0.6155	0.0006
8	0.6310	0.6308	0.0002
10	0.6308	0.6300	0.0008

- **Derivation and Setup Window Analysis**

$$U = 1,000; R = 10,000; D = 1,000; \alpha = 0.005; \beta = 2; 10 \ Resource \ and \ User \ Distribution, 100 \ Runs$$

$dw - sw$	μ_{real}	$\mu_{estimated}$	Δ
1	0.7274	0.6483	0.0791
2	0.7326	0.7204	0.122
3	0.7405	0.7324	0.0081
4	0.7236	0.7210	0.0025
6	0.7201	0.7172	0.0029
8	0.7216	0.7154	0.0061

- **Simulation Length Analysis**

$U = 1,000; R = 10,000; dw = 10; sw = 10; \alpha = 0.005; \beta = 2; 8$ *Resource and User Distribution*; 100 *Runs*

D	μ_{real}	$\mu_{estimated}$	Δ
100	0.8665	0.8640	0.0025
200	0.8588	0.8575	0.0014
300	0.8529	0.8518	0.0011
400	0.8395	0.8303	0.0092
600	0.8465	0.8454	0.0011
800	0.8623	0.8607	0.0016

- **Number of Resources and Daily Transactions Analysis**

$U = 1,000; D = 500; dw = 10; sw = 10; \alpha = 0.005; \beta = 2; 8$ *Resource and User Distribution*; 100 *Runs*

R	μ_{real}	$\mu_{estimated}$	Δ
300	0.8623	0.5774	0.2848
625	0.8489	0.7309	0.1181
1250	0.8672	0.8287	0.0385
2500	0.8470	0.8356	0.0114
5000	0.8547	0.8464	0.0083

- **Daily Transactions Analysis**

$U = 1,000; R = 2,000; D = 500; dw = 10; sw = 10; \beta = 2; 8$ *Resource and User Distribution*; 100 *Runs*

α (t_{max})	μ_{real}	$\mu_{estimated}$	Δ
0.001 (2)	0.8627	0.8053	0.0574
0.002 (4)	0.8603	0.8482	0.0122
0.0025 (5)	0.8519	0.8444	0.0075
0.005 (10)	0.8555	0.8351	0.0204
0.01 (20)	0.8438	0.8016	0.0422
0.02 (40)	0.8645	0.7401	0.1244
0.025 (50)	0.8569	0.6786	0.1783
0.05 (100)	0.8671	0.4177	0.4494

- **Summary:**
  - User Distribution is the most important parameter in the simulation. When the mean of the authority values of the most effective users goes down, algorithm gives better results.
  - When User Distribution is selected carefully, actually sufficiently small, artwork distribution doesn't matter.
  - The sizes of the derivation and setup windows behave in a bell curve so these window sizes should be chosen by the analysis and should be sufficiently large, not so much because then results worsen.
  - The longer simulation length is, the better results are found. Therefore, simulation length must be long as much as possible.
  - The number of daily transactions is also important and also follows a bell curve and there is an optimum value to be determined.

Chapter 5

Conclusions and Future Work

This algorithm sufficiently gives good results when the parameters are chosen with caution. However, there are some difficulties in choosing parameters and distributions that are used to generate the deviants and deviations.

Furthermore, network resolution algorithm works well on the networks whose internal dynamics run as in the network generator algorithm generated networks and we aren't sure DeviantART works as well.

We will specifiy the procedure of the selection of the parameters and then run the algorithm on the DeviantART. We will follow the users that are found as the most effective user by the algorithm to test the results.

Chapter 6

Appendix

Distributons of User and Resource Generation

These values are just runs of the distribution *10,000* times

n	μ	σ^2
0	0.3327	0.0562
1	0.4178	0.0789
2	0.3459	0.0694
3	0.2791	0.0558
4	0.2300	0.0425
5	0.1903	0.0316
6	0.1647	0.0267
7	0.1415	0.0198
8	0.1262	0.0158
9	0.1097	0.0115
10	0.1006	0.0102
11	0.0903	0.0080
12	0.0825	0.0069
13	0.0765	0.0059
14	0.0722	0.0052
15	0.0663	0.0043
16	0.0622	0.0039
17	0.0581	0.0034
18	0.0560	0.0032
19	0.0520	0.0027
20	0.0502	0.0025

Chapter 7

References

- **DeviantART**

  http://www.deviantart.com

- **DeviantART in Spotlight: A Network of Arts and Artists**

  Almila Akdag Salah, Albert Ali Salah, Bart Buter, Nick Dijkshoorn, Davide Modolo, Quang Nguyen, Sander van Noort, Bart van de Poel VKS/KNAW, Cruiquisweg 31, 1019 AT, Amsterdam, the Netherlands. Informatics Insitute, University of Amsterdam, 1098 XG, Amsterdam, the Netherlands.

- **TADA: Toolkit for Analysis of deviantART**

  Bart Buter, Nick Dijkshoorn, Davide Modolo, Quang Nguyen, Sander van Noort, Bart van de Poel

  http://code.google.com/p/ppis-deviantart/

- **Community Structure in social and biological networks**

  M. Girvan and E.J Newman, 10.1073/pnas.122653799 PNAS June 11, 2002 vol. 99 no. 12 7821-7826

  http://www.pnas.org/content/99/12/7821.full.pdf

- **Wikipedia**

  http://en.wikipedia.org/wiki/DeviantArt

- **Wayback Machine**

  http://web.archive.org/web/*/http://www.deviantart.com