Social Networking Study on DeviantArt
Artificial Network Algorithm

CmpE 492 Project
By: Ferhat Elmas
Student Id: 2006101102
Advisor: Haluk Bingöl

22.04.2011
Version 2.0

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

Abstract

Effective users are very important in the network because they can easily affect the distribution of the network via some relations such as favorites. Favorite relation is a powerful tool to increase or decrease the value of a deviation. If a well known user adds the deviation into his list, the value of the deviation increases rapidly. In this paper, the algorithm to find the most effective 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.

Contents

Chapter 1

Introduction

An algorithm which can determine the most effective user with 100% guarantee is the main purpose of the Social Analysis on DeviantART project.

Different algorithms can be easily developed but we can't sure that developed algorithm is the desired algorithm. Therefore, a network is needed whose values are well established (known).

Moreover, we need data structures (framework) to generate a network, namely, deviant, deviation and network. Firstly, framework is explained. Secondly, network generation is detailed and finally, the algorithm to find the most effective user and its related parameters are explained.

Chapter 2

Framework

There are three objects to represent users (deviant), artworks (deviation) and relations with each other (network).

2.1 Deviant

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

There are two properties, namely, *id* and *authority. id* is defined to select the users uniquely and varies between 0 and the number of the users - 1. *authority* is here to specify the effectiveness of the user. *authority* varies between 0 and 1. The more effective user is, the closer to 1 *authority* is.

2.2 Deviation

Deviation object is defined to represent the artworks in the DeviantART.

There are four properties, namely, *id*, *quality*, *favorite list*, *time list. id* is defined to select the artworks uniquely and varies between 0 and the number of the artworks - 1. *quality* is a number to specify the quality (art-criticque) of the artwork. The higher quality artwork is, the closer to 1 *quality* is. *favorite list* holds the ids of the users that added this artwork to their favorite list and corollary, *time list* holds the days when these users added the artwork. If a user added before than others, his index will be smaller than others.

2.3 Network

Network object is defined to represent relations in the DeviantART.

There are six properties, namely, *deviants*, *deviations*, *deviantCount*, *deviationCount*, *authorityID* and *random*. *deviantCount* and *deviationCount* are the numbers of the deviants and deviations, respectively. *deviants* and *deviations* hold the generated deviants and deviations, respectively. *authorityID* is the id property of the most effective user in the network. Finally, *random* is the pseudo random number generator which is used in generation of the properties and matching of the users and artworks.

Chapter 3

Network Generation

3.1 Generation of the Deviants

id property is incremented by one in each deviant generation. *authority* property comes from one probability distribution and this distribution is chosen by the given parameter. There are distributions in the form of $\frac{1}{x^n}$ or $2x - 2$. If zero is entered to select distribution, the authority of the deviants are generated with $2x - 2$ so pseudo number for *authority* is generated via this calculation: $1 - \sqrt{R}$ where R is a random number berween 0 and 1. Otherwise, the authority of the deviants are generated with $\frac{1}{x^n}$. For example, when 1 is entered, distribution becomes $\frac{1}{x}$ and when 2 is entered, distribution becomes $\frac{1}{x^2}$ and it is calculated via this calculation: $-\log R/n$ where R is a random number berween 0 and 1. While input is increasing, the mean of the distribution decreases to zero which means that a lot of users have small *authority* and have a minor effect on the network.

3.2 Generation of the Deviations

id property is incremented by one in each deviation generation. *quality* property comes from one probability distribution and this distribution is chosen by the given parameter. There are distributions in the form of $\frac{1}{x^n}$ or $2x - 2$. If zero is entered to select distribution, the quality of the deviations are generated with $2x - 2$ so pseudo number for *quality* is generated via this calculation: $1 - \sqrt{R}$ where R is a random number berween 0 and 1. Otherwise, deviations are generated with $\frac{1}{x^n}$. For example, when 1 is entered, distribution becomes $\frac{1}{x}$ and when 2 is entered, distribution becomes $\frac{1}{x^2}$ and it is calculated via this calculation: $-\log R/n$ where R is a random number berween 0 and 1. While input is increasing, the mean of the distribution decreases to zero which means that a lot of artworks have small *quality* and really good artworks will be minority in the network. In this sense, deviation generation is very similar to deviant generation, the only possible difference is their chosen distributions because deviants and deviations can be generated separately from different distributions.

Distributons of Deviant and Deviation Generation

These values is just runs of the distribution *10000* times

Distribution Select	**Mean**	**Variance**
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

3.3 Matching of Deviants and Deviations

There are *three* parameters used in this step, namely, *simulation length*, α and β. *simulation length* is self descriptive and is used to specity the number of the days for which simulation runs. α is used to calculate the mean of the normal distribution which is used to generate the number of daily transactions. β is used to scale the mean which is calculated by using α to calculate the variance of the normal distribution. The calculation of the mean and variance as follows:

Mean of the Daily Operations:

$$\mu = \alpha * \pi(deviants) * \pi(deviations)$$

Variance of the Daily Operations:

$$\sigma^2 = \beta * \mu$$

π is used as the number-of operator.

In each daily operations, an deviant and a deviation are randomly chosen. Then, *favorite list* of the deviation is controlled whether chosen deviant is in the list. If the deviant has already added the deviation into his list, the deviant is in the list so there is no need to take action and transaction is finished. Otherwise, the deviant can add the deviation and it is according to the *quality* of the deviation. One random number is generated and it is compared to *quality* of the deviation. If the generated random number is smaller than the *quality*, the deviant will add the deviation into his list to satisfy the basic addition probability. The deviation is added and *favorite list* and *time list* are updated. Moreover, if the *authority* of the deviant is bigger than the *quality*, then *quality* of the deviation is increased to the *authority* of the deviant.

Simulation is started in the first day and in each day a number is generated from $N(\mu, \sigma^2)$ and daily operations are performed as much as this number and then day is incremented and another day begins. When simulation length is hit, the algorithm stops.

Here are the step by step procedure:

- Calculate the μ of the normal distribution which is used to generate the number of daily transactions via $\alpha * \pi(deviants) * \pi(deviations)$

- Calculate the σ of the normal distribution which is used to generate the number of daily transactions via $\sigma^2 = \beta * \mu$
- Create a pseudo number generator from the normal distribution with parameters μ and σ
- Set the simulation day to one
- Repeat as much as *simulation length*
  - Generate a number for daily transactions
  - Repeat as much as the *generated number*
    - Randomly choose a deviation
    - Randomly choose a deviant
    - Check the *favorite list* of the deviation for the chosen deviant
    - If chosen deviant has already added the chosen deviation into his list, there is no action and continue for new transaction. In other words, the favorite list of the deviation already contains the randomly chosen deviant
    - Else deviant hasn't added the chosen deviation yet, so give a chance to deviant to add the deviation into his list. This chance is proportional to the *quality* of the deviation and is independent from the *authority* of the deviant. In other words, deviant make the deviation his favorite with the probability of its *quality*
    - If chosen deviant adds the chosen deviation into his list according to above negotiation, update the *quality* of the deviation with the max of *quality of the deviation* and *authority of the deviant*. If effective users add the deviation into their list, the *quality* of the deviation increases and so the probablity increases so that other users can add the deviation when they meet the deviation

Chapter 4

Network Resolution - Algorithm

If we use the Network generation algorithm, we will have an artificial network which is very similar to DeviantART network but we definitely know its parameters so when we run the Network resolution algorithm on this network, if the algorithm can find the actual parameters or at least with 90% probability, we have a powerful algorithm.

- Create *Global hashmap*, this data structure holds the number of the deviations for that the deviant is the most effective user. Therefore, the bigger this value is, the more effective this deviant is. The distribution of values is correlated with the distribution that is used to generate the deviants
- Calculate the maximum size of the favorite list of the deviations
- Repeat for each deviation,
  - Get the *favorite list* and *time list* of the deviation
  - Calculate the *frequency vector* of the deviation, the length of this vector is equal to the length of the simulation.

    $\textit{Frequency vector}[i] = \sharp$ of the transactions that the deviation is marked as favorite in the day i
  - Sum the successful transactions of the consecutive days (indices of *frequency vector*) in a sliding window fashion. There is an important parameter here, namely *derivation window size.* This parameter specifies the number of consecutive days (size of the sliding window).

  - Pick the day where maximum increase has happened (summation is maximum), there can be multiple days so from this step, the vector of peak days is found.
  - Since these maximums are summation of the consecutive days, we need the actual day in which peak has happened. Therefore, go back as much as the size of the *derivation window* and get the maximum of the partition. This is maximum of the smoothed maximums. However, there can be multiples so to eliminate the multiplicity check difference between the number of the favorites in the actual day and previous day. Pick the day(s) that has the biggest difference. Found day can be first day which prevents us to check the day before. In this situation, we neglect the first day because if lots of users have added the deviation into their favorite lists in the first day that deviation is uploaded, this implies deviation is intrinsically attractive and good, there is no effect from users.
  - Repeat for each day of the found peak vector
    * For each peak day, mark the deviants that added the deviation k days ago as effective user, and update *global hashmap.* Moreover, k is the *setup window size* and it is supplied into the algorithm. However, while marking, there is a weight since all favorite lists are not in the same size and weight is calculated via division of the size of the *favorite list* of the deviation to max of the sizes of the *favorite list* of the deviations. As a result, weights are in the range of [0, 1]
  - Get the deviant(s) that has the biggest marking number from *global hashmap.*

- *Derivative window size* and *Setup window size* are calculated according to the α and *simulation length* parameters. $\alpha * \pi(deviations)$ approximately specifies the number of transactions that each deviant can do each day. *Simulation length is related* because if *simulation length* is long, we should extend the size of the windows or decrease, otherwise.

- We can calculate α of the real network and also we know the length of the data. Therefore, this algorithm can be easily extended to DeviantART.

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

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

- **TADA: Toolkit for Analysis of deviantART**
  Bart Buter, Nick Dijkshoorn, Davide Modolo, Quang Nguyen, Sander van Noort, Bart van de Poel

- **Community Structure in social and biological networks**
  M. Girvan and E.J Newman

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

- **http://www.dmusic.com**

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