Social Networking Study on DeviantArt Artificial Network Algorithm

CmpE 492 Project

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

30.03.2011
Version 1.0

Version History

Version	Date	Explanation
1.0	30.03.2011	Algorithm Design and Specification

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 users that added this artwork to their favorite list and corollary, *time list* holds the times 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 the deviants and deviations, respectively. *authorityID* is the id property of the most effective user. 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 distribution in the form of $\frac{1}{x^n}$ or $2x - 2$.

3.2 Generation of the Deviations

id property is incremented by one in each deviation generation. *authority* property comes from one probability distribution and this distribution is chosen by the given parameter. There are distribution in the form of $\frac{1}{x^n}$ or $2x - 2$. In this sense, deviation is very similar to deviant generation, the only possible difference is chosen distributions.

3.3 Matching of Deviants and Deviations

There are mainly two parameters for this step, α and β. α is used to calculate the total number of transactions in the simulation and β is used to calculate the number of the daily transactions. Their calculations are as follows:

$$Total\ Operations = \alpha * \pi(deviants) * \pi(deviations)$$

$$Daily\ Operations\ \mu = \alpha * \beta * \pi(deviants) * \pi(deviations)$$

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 zeroth day and when daily operations are done, day is incremented by one. When all transactions are processed, the simulation length is set to this day variable.

Here are the step by step procedure:

- Calculate Total Operations by using the number of deviants and deviations, and given α paramater
- Calculate Daily Operations by using the number of deviants and deviations, and given α and β paramaters
- Set the simulation day to the zero
- Generate daily operations from $N(\mu, 2\mu)$ where μ is the calculated daily operations number in step 2
- Pick a deviant randomly

- Pick a deviation randomly
- Check the *favorite list* of the deviation for chosen deviant
- If the list contains the deviant, done
- Unless the list contains the deviant,
  - Generate a random number
  - if the random number is smaller than the *quality* of the deviation, addition is successful, update connections
  - if addition is successful and the *authority* of the deviant is bigger than *quality* of the deviation, set *quality* to the *authority*
- Decrement daily operations by one, if daily operations are done, increase the day else return to pick randomly a deviant step
- Decrement all transactions by daily operations, unless all transactions are processed, return to the daily operation generation step else network generation is done

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, for each deviant
- 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
- 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.
    Frequency vector$[i]$ = ♯ of the transactions that the deviation is marked as favorite in the day i
- Sum the successful transactions of the consecutive days(actually, 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, there can be multiple days so from this step, the vector of peak days is got.
- 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.
- Get the deviant that has the biggest marking number from *global hashmap*.

  Derivative window size and *Setup window size* are calculated according to the α and β parameters. α is just a scaling factor to limit simulation execution time and β is just another scaling factor to at the average generate daily transaction number. In the artificial network, we know the α and β so we could generate some size and we can do on the real network because we already know the total favorites and can calculate the average transactions by dividing total transaction into crawled data length.

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**