ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Lecture with Computer Exercises:
Modelling and Simulating Social Systems with MATLAB

Project Report

Simulation of Information Spreading in a Facebook Network

Urs Lustenberger , Nino Wili , Patrick Zöchbauer

Zurich
December 2013

Agreement for free-download

We hereby agree to make our source code for this project freely available for download from the web pages of the SOMS chair. Furthermore, we assure that all source code is written by ourselves and is not violating any copyright restrictions.

Urs Lustenberger Nino Wili Patric Zöchbauer

Contents

1 Abstract

2 Individual contributions

3 Introduction and Motivations

Everyone is on facebook
Commercials are personalized, the flow of info is interesting for companies
Are there "more important" persons in typical facebook network?

important questions (does the inhomogeneity of the real network influence the "total" evolution? are there "influentials"?)

4 Description of the Model

The process of information spreading has many similarities with epidemiology, so the well known SIR-model was adapted[**?**]. The "susceptibles", are not aware of the information and are called "ignorants" within this work. "Infected" individuals know about the information and are willing to share it with other people, in other words they spread it and are therefore called "spreaders". The adaptation of the epidemiological term "recovered" is not that straightforward because it is not entirely clear, what that means in this context but they can be looked at persons that are aware of the information but don't want to tell it others. They are called "stiflers".

In the SIR-model as well as in the Agent-based model, the following set of "reaction equations" was used(I: Ignorant, S: Spreader, R: Stifler):

$$\begin{cases} I + S \xrightarrow{\lambda} 2S & (1) \\ S + R \xrightarrow{\alpha} 2R & (2) \\ S + S \xrightarrow{\alpha} S + R & (3) \end{cases}$$

The main difference to the standard SIR-model is that spreaders don't become stiflers spontaneously but this this change is only induced by meeting an other spreader or stifler. Also it can be shown that for $\frac{\lambda}{\alpha} > 0$, i.e. for any positive rate, the number of stiflers is non-zero for large times. That means that there is no epidemic threshold as in the standard SIR-model.

Flow Chart of overall steps

Figure 1: Steps performed in each timestep.

4.1 Homogeneous SIR model

In the mathematical treatment of the model in a homogeneous system, the set of differential equations is expressed in terms of the densities $i(t) = I(t)/N$, $s(t) = S(t)/N$ and $r(t) = R(t)/N$.

$$\begin{cases} \dfrac{\mathrm{d}i(t)}{\mathrm{d}t} = -\lambda \cdot s(t)i(t) & (4) \\ \dfrac{\mathrm{d}s(t)}{\mathrm{d}t} = \lambda \cdot s(t)i(t) - \alpha \cdot s(t)[s(t) + r(t)] & (5) \\ \dfrac{\mathrm{d}r(t)}{\mathrm{d}t} = \alpha \cdot s(t)[s(t) + r(t)] & (6) \end{cases}$$

4.2 Agent-based model

In the inhomogeneous, agent-based model, the individuals are connected in a certain manner (facebook friends in this particular work). Only connected agents are able to meet. If a meeting occurs, transitions are induced at a certain probability depending on the relation between the agents (details are discussed later). Additional to the different degree of the agents, they also have a different activity, i.e. different probability to meet somebody. In total, the number of meetings in a certain time is proportional to the product of the degree (number of friends) and the activity. This has to be verified in the implementation of the model.

To convert the reaction equations into an agent-based model, time was discretized and in each time step a series of two steps is performed shown in Figure 1. First, the agents randomly meet another agent they know or nobody. Only two-agent meetings are possible. In the second step, the status of the agents change corresponding to the situation. There are six possible combinations of ignorants, spreaders and stiflers. Three of them correspond to the "reactions" 1-3, the other ones (I+I, I+R and R+R) have no other influence than "occupying" the agents. The probability λ was chosen to be dependent on the number of common friends. For simplicity, α was a constant.

5 Implementation

5.1 How did we get the network

To get the data for our simulation, we used the so-called Facebook Graph API. What does Graph API mean? A Facebook Grpah API is a programming tool designed to support better access to conventions on the Facebook social media platform.[1]

Unfortunately, there is no implementation for Matlab, so we programmed the data-fechting in python.

5.2 Coding

For the coding in python we used the facebook sdk[2].

5.2.1 Get access

First, one needs to get access to the social graph. Therefore, you need to enter an access token[3].

```
1 token = raw_input('Enter your Access Token: ')
2 anonym = input('Do you want to anonymize your Data? Yes (1) or No (0)? ')
3 graph = facebook.GraphAPI(token)
```

5.2.2 Get graph

Having enterd a valid token, we can now access all information this token provide us. For simplicty, we'll only discuss the non-anonymized case in details. [For more details on the anonymization, look at subsection anonymization]

```
1 profile = graph.get_object("me")
2 friends = graph.get_connections("me", "friends")
```

In our case, we're interested in data about our own profile, as well as all available information we get from our friends. The object friends is nothing else but an array that contains all id's of our friends. For each id there is another array, named data, that contains all (available) information about this specific friend.

```
1 friend_list = [friend['id'] for friend in friends['data']]
```

[1] http://www.techopedia.com/definition/28984/facebook-graph-api
[2] https://github.com/pythonforfacebook/facebook-sdk
[3] https://developers.facebook.com/docs/facebook-login/access-tokens/

With this simple for-loop, one can now get a list of all friends. Since we're not primarly interesseted in our friends, but the connection in between them, we have to access at least the mutualfriends.

```
1 mutualfriends = graph.get_connections(tfriend, "mutualfriends")
```

The object mutualfriends provides us a list with all mutualfriends of "me" and friend "tfriend" (which is nothing else but one of the id's we have in the saved in "friend_list"

Repeat this proceedure for all friends, and we'll get exatly what we're looking for - a graph of all connection between our friends.

5.2.3 Get activity

How did we define the parameter activity? We simply determine for each friend how many posts he makes on average per day. This gives us an indicator, how active a certain person is on facebook.

Again, this is done fairly quickly (using graph api).

```
1 statuses = graph.get_connections(tfriend, "statuses", fields="updated_time",
    limit="100")
```

This objecct provides us, all information we need (a list of all posts incl. a time stamp when it's been posted) to calculate the above mentioned quotient.

5.2.4 Get coordinates

urs will tell you!

5.3 Anonymization

The anonymization is acutally only a pseudo-anonymization. First all id's get arranged in order of size. Secondly, we create a list from 1 to n (number of friends). We then define a bijection between the two sets, such that the i-th element of the first set, maps to the i-th element of the second set.

Eventhough, this is not really anonymizing our data. It is more then sufficient for our purpose.

5.4 Homogeneous SIR-model

Urs