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[1]. 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 $r(t) = R(t)/N$.

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

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 The Network

how did we get the network?
how did we get the coordinates with gephi?

5.2 Homogeneous SIR-model

Urs

5.3 Agent-based model

5.3.1 Initial condition

At the beginning of each simulation, all agents are ignorant but one, which is a spreader. This agent was determined pseudo-randomly.

5.3.2 Determine the meetings

(See `talkstep.m` for details.)

In order to determine who meets who, the program goes through the vector of agents (1:N) randomly (line 14-16). With a probability corresponding to the activity of that agent, he may meet somebody (line 19). The person he meets is determined randomly and must be one he knows and one which is not already meeting another agent in this time step (line 30). In order to be able to implement that agents with more friends meet more people and to keep the simple data structure first a random person is chosen out of all the persons in the network and after that, the program checks whether they know. Like that, lot of "finding attempts" land on pairs which are not connected, that's why more than one attempt is performed each round (line 22-39). `attempt` is basically just a scaling factor. Tests show that the number of meetings is still proportional to the product `activity * number of friends`.

5.4 Status changes in meetings

6 Simulation Results and Discussion

most important question!

7 Summary and Outlook

blub

8 Appendix

Code

../../code/talkstep.m

```
%script thats let people meet, eventually meet and talk, updates status

%%
%Determine who meets who


%vector which stores if someone's already in a meeting with who
%0: free, k: already meeting person k, i:not meeting anybody
meeting=zeros(1,N);

%random vector, so it doesnt start at node 1 always
choose=randperm(N);

for k=1:N

    i=choose(k);

    if(meeting(i)==0)%check if i already meets somebody
        if(rand>(person(i).activity))
            meeting(i)=i; %doesnt meet anybody
        else
            who=0; %who: meeting partner of i
            attempt=1; %dont check forever, maybe no partner available
            while(who==0);
                who=round(rand*N+0.5); %random partner

                attempt=attempt+1;

                %check if who is not meeting, and who and i know each other
                if(meeting(who)==0 && connect(i,who))
                    meeting(i)=who;
                    meeting(who)=i;

                    nummeetings(i)=nummeetings(i)+1;
                    nummeetings(who)=nummeetings(who)+1;
                else
                    if(attempt<50) %change this!
                        who=0;
```

```
                    else who=1; %just not 0
                    end
                end
            end
        end
    end

    %some can be 0, not cool for further programming
    if(meeting(i)==0)
        meeting(i)=i;
    end
end




%% change status for each meeting

check=zeros(1,N); %vector to check if status was updated, otherwise
%multiple updates in one round are possible



for i=1:N

    p1=i;                       %person1
    p2=meeting(i);              %person2


    %check if status wasnt updated yet and if i is actually meeting sombody
    if (check(p1)==0 && check(p2)==0 && p1~=p2)

        %if both are ignorant or both are stiflers nothing happens...

        %if one is ignorant and the other is a spreader, both become
        %spreaders
        if((status(p1)+status(p2))==1)
            %make p1 the one whos already infected
            if status(p2)==1
                dummy=p2;
                p2=p1;
                p1=dummy;
            end

            %make pinform a function of p1 and p2
            pinform=common(p1,p2)/maxcommon;

            if(rand<pinform) %probability that they talk about this info

                infections(p1)=infections(p1)+1;
```

```
                    status(p2)=1;

                    SaveMeeting;
                end


        %if both are spreaders, one becomes a stifler
        elseif(status(p1)==1 && status(p2)==1)
            if(rand<pforget) %probability that they forget
                if(rand<0.5)%only one of them forgets(randomly chosen)
                    status(p1)=2;
                else
                    status(p2)=2;
                end
            end

        %if one is a stifler and one a spreader, both become stiflers
        elseif((status(p1)+status(p2))==3)
            if(rand<pforget)  %only by a certain probability
                status(p1)=2;
                status(p2)=2;
            end
        end
    end
    %remember that you updated the status
        check(p1)=1;
        check(p2)=1;

end
```

../../code/parameters.m

```
%pmeet=0.8; %probability, that a person wants to meet another person
            %used in "talkstep" to determine meeting-vector

%pinform=0.9; %probability that a spreader meeting a ignorant tells him
            %as "lambda" in (Alain Barrat)

pforget=0.2; %p, that if a spreader meets a spreader, one of them becomes a
            %stifler. or if a stifler meets a spreader, both become
            %stiflers
```

References

[1] A. Barrat, M. Barthlemy, A. Vespignani. Dynamical Processes on Complex Networks. Chapter 10.