Gossips, Gossips

Spreading Rumors in Social Networks

# Abstract

# Background

All humans are active agents in rumor spreading. While we generally think of rumors as high-school gossip or schoolyard trash talk, rumors can have a large impact. In late 2013, rumors regarding how China would react to the rise of Bitcoin greatly affected the price of the currency days before any official statements were made. During the 2008 presidential campaign, rumors about Senator Obama’s place of birth greatly affected the legitimacy of his candidacy. In March 2014, actor Wayne Wright had to publicly declare that he was in-fact not dead after rumors declaring otherwise spread through social networks. However, the spread of rumors is not always harmful. For example, the concept of viral marketing uses the quick spread of rumors through social media to promote goods and services by various companies.

Since the internet boom, the spread of rumors has greatly accelerated. With social products such as Facebook and Twitter, spreading a rumor from a friend halfway around the world is as easy as clicking a “share” button or “retweeting” a status update. Due to their well-defined user relationships, these products allow us to easily analyze the nature of such phenomena in ways we could not before.

# Model 1 Equations

We will begin with an SIR model for Rumor Spreading. Consider a population consisting of N individuals which are subdivided into ignorants (I), spreaders (S), and stiflers (R).

Assumptions:

* The rumor spreads by direct contact of the spreaders with others in the population.
* The population size is constant during the lifetime of a rumor.
* Each person comes into contact with a percentage of the population k.
* Whenever a spreader contacts an ignorant, the ignorant becomes a spreader at a rate .
* When a spreader contacts another spreader or a stifler the initiating spreader becomes a stifler at a rate .

We may simplify the above model by reducing it to two equations since we are assuming the population is constant. We may make the substitution , where is a constant.

In this model, the only steady state is clearly along the line , and it is a stable steady state. This is because Ignorants depend on the presence of Spreaders for their population to change, and if there are no Spreaders, no additional Stiflers may be created.

GRAPH

# Model 2 Equations

In Model 1, we assumed that spreaders would become stiflers only if they were themselves stifled by another stifler or spreader. However, spreaders may also spontaneously decide to become stiflers for a variety of reasons. For example, the spreader may realize the rumor isn’t as exciting as it used to be, or the spreader may decide the rumor is harmful to a specific individual or group of people and feel guilty about continuing to spread it. To account for this, we may introduce a parameter to represent the rate at which a spreader may spontaneously decide to become a stifler.

Reducing this to a model of two variables as we did before, we attain the following equations:

Once again, the only steady state occurs along the line S = 0, and it is a stable steady state.

GRAPH

# Agent-Based Model

# Input Data

Our Data came from SNAP (Stanford Network Analysis Project). The data consists of Facebook friends lists. It is split up into various graphs that represented different connected components of the collected data. Each graph has an ID, which represents the node whose friend list was used to generate the graph. For a given ID, the graph is formed by connecting the node with the given ID to all nodes in its friends list, and recursing on each of those friends. Statistics of the various graphs are shown below.

|  |  |  |
| --- | --- | --- |
| **Graph** | **Size** | **Average Friends Per Node** |
| 0 | 334 | 17.08 |
| 107 | 1035 | 53.69 |
| 348 | 225 | 30.36 |
| 414 | 161 | 24.41 |
| 686 | 169 | 21.59 |
| 698 | 62 | 10.68 |
| 1684 | 787 | 37.64 |
| 1912 | 748 | 82.28 |
| 3437 | 535 | 19.99 |
| 3980 | 53 | 7.47 |

# Algorithm

We chose to implement our algorithm in Python. From a high level, our model iterates over all possible combinations of our parameters, and runs a “model object” for each combination. A model object is defined by a set of parameters and a graph with nodes and edges. Upon creation, the model randomly chooses which nodes will be the initial spreaders in accordance with the provided value for the number of initial spreaders. It then partitions the nodes into three distinct sets of ignorants, spreaders, and stiflers. Initially, all non-spreaders are ignorants. We then define a function run, which does all the work for a single time step - we chose each time step to represent one hour. Specifically, run will increment the time value, and for each spreader already in the spreader set, it will determine which nodes to spread to, whether it is stifled by an interaction, and whether it spontaneously is stifled. For spreading and stifling, each spreader considers each friend in a group of “contacted friends” which is random subset of a node’s friends of size determined by the contact fraction parameter. Each model will call run exactly 48 times to represent a two day lifespan of a rumor, and reports the counts of ignorants, spreaders, and stiflers at each time step. Because of the element of randomness, we can choose to run the model several times for a given set of parameters and average the results. For our initial training data, we chose not to do this; however, for the actual simulations that we analyzed, we averaged each model over 10 iterations. (Our training data and actual simulations are discussed later.) Finally, we plot the percentage of each subpopulation over time, and we display the non-ignorant population (spreaders and stiflers who have become savvy to the rumor) over time as well.

# Meaning of the Parameters

The following are descriptions of parameters that are used to initialize a model object in our Python script:

* graph: This is a collection of nodes and edges that represent a social network.
* spreadChance: This is the chance that an interaction between a spreader and an ignorant will result in the ignorant becoming a spreader. This value is highly dependent on the actual rumor in addition to the behaviors of the users in the network; therefore, it is difficult to accurately assign this value for a given rumor.
* stifleChance: This is the chance that an interaction between a spreader and a non-ignorant will result in the spreader becoming a stifler. This value is also highly dependent on the actual rumor rather than just the users in the network. For example, rumors that are known to be false by some stiflers will have a high stifleChance.
* numSpreaders: This is the initial number of spreaders in the network.
* contactFraction: This represents the average fraction of friends that any node has an interaction with in a given hour. This average value transformed into a specific fraction by multiplying it by a factor that depends on the time of day. Certain times of day, Facebook users are either more or less active. The data used for this transformation was acquired from Mashable.
* spontaneousStifleChance: This is the chance that a spreader will spontaneously become a stifler in a given hour. The reciprocal of this value is the expected duration of a node’s spreader lifetime if the spreader has no interactions with other nodes. In Model 1, this value was just zero, so spreaders would remain spreaders forever if they had no interactions.

It is worth noting that spreadChance, stifleChance, and spontaneousStifleChance are dependent upon a specific rumor, but contactFraction is an attribute of a given network. Obviously in reality, each of the users in a given network won’t have the same contactFraction, but we chose to model it this way for simplicity. Also numSpreaders is neither an attribute of the rumor nor of the network; it is a result of external conditions resulting in rumor introduction in the network.

# Results of Training Graph

We started with our training graph, graph 0. This graph had 334 nodes and an average of 17 friends per node. This graph was conveniently around the median for both number of nodes and average friends per node. For each of our 5 parameters, we ran the simulation with 3 different values per parameter, leading to a total of 243 simulations. From there, we pruned our results. First, we eliminated all stagnant graphs – graphs in which the populations did not change, which was generally caused by extreme values in our parameters. We then grouped various simulations together and sampled them, noting their interesting features and recording their parameter values. The parameters we considered for these simulations are shown below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Parameter** | **Spread Chance** | **Stifle Chance** | **Initial Number of Spreaders** | **Contact Fraction** | **Spontaneous Stifle Chance** |
| **Values** | 0.1, 0.5, 1 | 0.01, 0.1, 0.5 | 1, 5, 25 | 0.01, 0.1, 1.0 | 0, 0.1, 0.5 |

Below is a chart of specific observations for selected graphs. These findings aided in our decision on which parameter values to consider in our actual simulations.

CHART OF OBSERVATIONS

From here, we were able to reduce the parameters for our actual simulations, shown below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Parameter** | **Spread Chance** | **Stifle Chance** | **Initial Number of Spreaders** | **Contact Fraction** | **Spontaneous Stifle Chance** |
| **Values** | 0.10, 0.50 | 0.001, 0.10 | 1, 25 | 0.05, 0.50 | 0, 0.20 |

# Graphs Considered in Actual Simulations

We wanted to choose graphs with different structure, so we decided we would pick a graph with a small number of nodes (SHD -- small, high density), a graph with a large number of nodes and a large number of friends per node (LHD -- large, high density), and a graph with a large number of nodes and a small number of friends per node (LLD -- large, low density). This led us to choose Graph 698 for SHD (62 nodes, 10.68 friends per node), Graph 3437 for LLD (535 nodes, 19.99 friends per node), and Graph 1912 for LHD (748 nodes, 82.28 friends per node).

# Results of specific graphs

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Graph #** | **SHD Time** | **LLD Time** | **LHD Time** | **SHD Yield %** | **LLD Yield %** | **LHD Yield %** | **SHD Peak SP %** | **LLD Peak SP %** | **LHD Peak SP %** |
| 1 | 9 | 8 | 6 | 92 | 86 | 97 | 72 | 74 | 96 |
| 2 | 10 | 6 | 6 | 24 | 72 | 95 | 17 | 42 | 64 |
| 3 | 12 | 9 | 6 | 98 | 99 | 100 | 84 | 86 | 80 |
| 4 | 14 | 8 | 6 | 79 | 95 | 98 | 45 | 60 | 66 |
| 5 | 12 | 9 | 6 | 88 | 95 | 98 | 83 | 74 | 75 |
| 6 | 8 | 8 | 4 | 72 | 74 | 94 | 41 | 40 | 63 |
| 7 | 6 | 10 | 4 | 98 | 99 | 100 | 89 | 85 | 78 |
| 8 | 6 | 6 | 5 | 93 | 93 | 98 | 56 | 53 | 64 |
| 9 | 2 | 6 | 3 | 9 | 56 | 85 | 10 | 24 | 42 |
| 10 | 1 | 6 | 3 | 13 | 54 | 86 | 10 | 19 | 42 |
| 11 | 6 | 8 | 4 | 42 | 68 | 91 | 12 | 23 | 43 |
| 12 | 8 | 6 | 5 | 27 | 66 | 89 | 14 | 22 | 42 |
| 13 | 1 | 4 | 6 | 53 | 52 | 87 | 47 | 25 | 38 |
| 14 | 2 | 5 | 4 | 52 | 52 | 87 | 40 | 22 | 38 |
| 15 | 3 | 4 | 4 | 81 | 74 | 89 | 40 | 23 | 38 |
| 16 | 3 | 4 | 3 | 85 | 72 | 90 | 40 | 24 | 39 |
| 17 | 6 | 2 | 3 | 98 | 97 | 99 | 89 | 93 | 92 |
| 18 | 6 | 2 | 2 | 79 | 93 | 100 | 64 | 76 | 84 |
| 19 | 3 | 3 | 2 | 98 | 99 | 100 | 97 | 96 | 95 |
| 20 | 3 | 2 | 2 | 98 | 100 | 100 | 82 | 85 | 85 |
| 21 | 6 | 3 | 2 | 98 | 97 | 98 | 96 | 93 | 91 |
| 22 | 3 | 3 | 2 | 95 | 99 | 98 | 73 | 78 | 76 |
| 23 | 3 | 2 | 2 | 98 | 100 | 100 | 97 | 96 | 92 |
| 24 | 2 | 2 | 2 | 98 | 100 | 100 | 83 | 82 | 77 |
| 25 | 2 | 3 | 2 | 74 | 90 | 98 | 67 | 66 | 58 |
| 26 | 3 | 3 | 2 | 69 | 88 | 99 | 52 | 60 | 55 |
| 27 | 3 | 3 | 2 | 98 | 99 | 100 | 57 | 59 | 52 |
| 28 | 3 | 3 | 2 | 95 | 99 | 100 | 53 | 56 | 53 |
| 29 | 2 | 3 | 2 | 85 | 88 | 100 | 75 | 62 | 75 |
| 30 | 1 | 2 | 2 | 85 | 90 | 97 | 67 | 63 | 76 |
| 31 | 3 | 2 | 2 | 96 | 98 | 100 | 67 | 40 | 75 |
| 32 | 2 | 2 | 2 | 96 | 98 | 99 | 65 | 45 | 75 |

# Conclusion