# Social Simulator


## Michael Maurer and Lauren Winston

CX4230, Spring 2015, Professor Richard Vuduc


## Introduction

Why do certain posts on social media become viral or become “memes”?

**Idea:** Create an educational simulation-application that allows users to run simulations on content-spreading in their own social networks.

Allow users to analyze how different variables may affect a post’s lifetime, especially in regards to their own personal networks.

Initially wanted to analyze the comparisons in meme-spreading in

- **Twitter**
- **Facebook**
- Pinterest
- Reddit

Facebook and Twitter were deemed most feasible because of the idea of a “closed “network.”

## Model

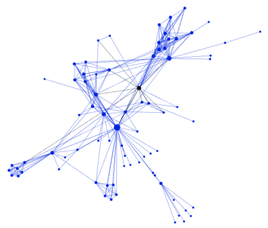
A network has three key properties:

- **Reshare Rate:** How likely a user in the network is to re-share a post that interests them
- **Unique Content:** How much unique content is posted in the network. More unique content means that a user is less likely to see a specific post in their feeds.
- **Diversity:** How closely aligned the interests the people of the network are. A higher diversity rate means that there are more diverse interests in the network and any given person in the network is less likely to be interested in a friend’s post or tweet.

Multiple people may introduce the same content to the network from an outside source.

Users may have multiple opportunities to see a post through multiple friends, and they may become interested in a post eventually, even if they were originally uninterested.

A social network can be represented as a graph with nodes and edges, where nodes are the people in the network and the edges are the connections between them.



## Implementation

We wanted our application to be available to as many people as possible so that they could learn about social networks. For this reason, we decided to create a web application.

In order be able to quickly update and easily collaborate, we decided to host our application with Google App Engine. This let us store all of our code in a git repository which was easily pushed online. We thought this would save us time in dealing with server issues, but as it turned out, Google App Engine had plenty of issues of its own.

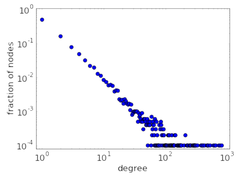
We decided to make the server in Python’s Flask, which is a simple web framework that will easily return html pages from the repository based on the url.

For display and graphing, we used sigma.js, a javascript library that allowed us to display and manipulate graphs as well as add callbacks to clicking on the visualization.

For the landing page, we used bootstrap to get a decent landing page and from-scratch html and javascript for the other pages.

## Power Law

One concept we wanted to investigate is the idea of scale-free networks in regard to everyday social networks. A scale free network is one where the degree distribution follows the power law. As we learned about in class, this would mean that the number of nodes with a certain amount of connections decreases exponentially (usually with an exponent of -(2-3)) as the number of connections increases. We believe that this behavior would be exhibited in the social networks you see in real life. Celebrities like Katy Perry have 69 million followers, but everyday people (which make up the majority of nodes) regularly have around 200. We plan to generate a graph as a next step, and because of the believed scale-free property think it will look something like:




## Limitations

- Twitter is represented with undirected edges, instead of directed edges
- Twitter rate-limits the amount of requests you can make, so the graph is not a full representation
- Facebook removed the ability to get a list of friends from their API, so we had to use a simple example network of characters from Les Miserables
- No element of “time” to the model
- No weighting of connections: people may be more likely to look at, and therefore be interested in, posts by best friends than casual acquaintances
- A user interested in a post may decide to re-share it after seeing another post on it
- Users might re-share their own content
- No interaction with other posts “fighting” for attention

## Life Cycle of Content

User posts original content to social network.

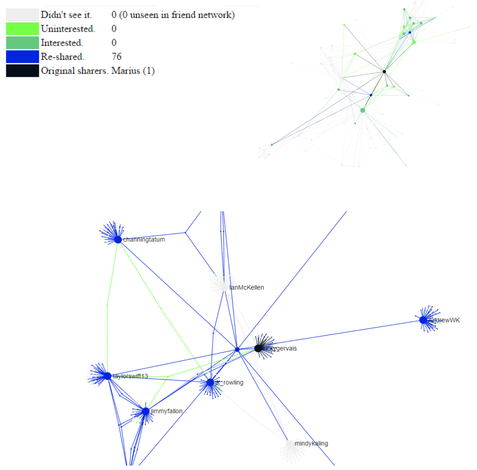
Friends log into social network. Whether or not they see the post depends on the “**Uniqueness**” parameter.

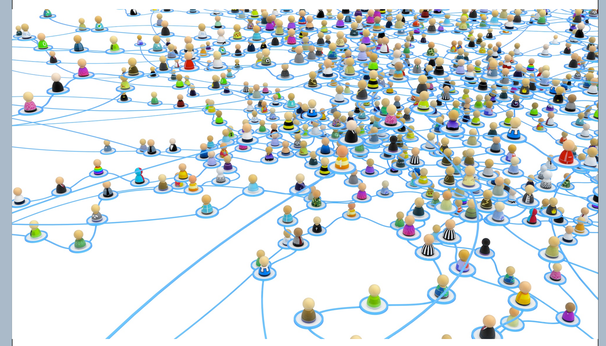
If they see the post, they will be either interested in it or not, randomly determined based on the “**Interested**” parameter.

If interested, the “**Re-Share Probability**” is used as the threshold for whether or not they will re-share it

If re-shared, the process repeats with all of that person’s connections.

Because a node may have more than one connection in the network, a node may be revisited by the the content. If they originally didn’t see the post or were uninterested, they have another opportunity to see it or to become interested in it, as more interest is generated the more a user sees a post.



## Results / Final thoughts

- The simulation exports data from multiple trials to a CSV file that users are able to download and analyze on their own
- What we created was a good framework for further investigating spread of information in social network
- It helps people analyze what friends are the biggest cause of information spread in their own social network
- We found that the biggest factor that determined the final network spread was the probability of reshares. In the real world, this widely varies based on many variables and relationships, which is one thing we hope to improve in future iterations of our model.
- When the content originates from more than one source in a given network, it greatly increases the speed of the spread.

| | | | | | |
|---|---|---|---|---|---|
| Network Size: | 77 | | | | |
| Start Node(s): | Gavroche | | | | |
| Diversity: | 0.5 | | | | |
| Uniqueness: | 0.5 | | | | |
| Reshare Rate: | 0.5 | | | | |
| | | | | | |
| | Current | Average (with 10 trials) | | | |
| Reshare Count: | 5 | 10 | | | |
| Interest Count: | 6 | 9.3 | | | |
| Uninterest Count: | 5 | 9.4 | | | |
| Unseen Count in Visited Nodes: | 14 | 11.7 | | | |
| Total Unseen Count: | 60 | 47.3 | | | |
| Number of Iterations: | 3 | 5.1 | | | |
| | | | | | |
| Most Significant Person: | Gavroche | 6 | | | |
| Significant Person List: | Javert | 0 | Bossuet | 0 | |

## What’s Next

- Directed graph for Twitter
- Animations for visualizations
- Analyze differences between amount of sharing in the different social networks
- Power law analysis
- Including other factors in the probability that people will share
- Betweenness and Closeness centrality analysis
- Caching user results in order to get full Twitter graph
- Making the information easier to understand for beginners
- Add a parameter to represent how interesting a post is

## Try It Out!

**social-simulator.com/appspot**