Capstone Project
Machine Learning Engineer Nanodegree
Hope Atina
August 21, 2016

Simulating a suggested social evolution of a Slack Community

I. Definition

Project Overview

Building an online community is one of the major challenges in the evolution of civilization nowadays[i]. New online forums, chat-rooms, and social networks are springing up every day like mushrooms. Management of these online communities involves many challenges. Management of a community is an inherently human action ranging in scope from the 3-person family to an international conglomerate. Current attempts to manage online communities, focus only on understanding hierarchy on a routine basis. This project aims to manage the growth of a simulated "Slack" community by the use of reinforcement learning. Using this technique to determine the best internal matches will significantly increase group engagement.

Problem Statement

The problem here is framed from the perspective of Slack community managers, commonly called Slack mods (short for moderators), or Slack administrators. At any point in time a Slack mod has a plethora of options for moderating and growing his or her group. The Slack mod can:

(1) Create new content by engaging in the public message stream,
(2) Engage with a single community member via a direct message or reply
(3) Reprimand existing negative activity
(4) Invite new members to the group

Navigating these options to maximize the value provided within the group is the primary concern of the Slack moderator. The current standard approach to managing an online community consists of having multiple moderators as the group grows[ii]. This has been an effective model in various forms of social media including Reddit where there are assigned mods, Facebook where moderation is done by members of the community, and similarly with other social networks like Twitter. One drawback of this approach is the lack of a quantitative understanding of the state of the community in order to maximize time utilization. This project only addresses the second option (2); the challenge of engaging with a single member via a direct message or reply. Interacting with an individual member allows a Slack mod to provide specific advice, information, and guidance to a member. However, as groups grow, creating custom connections for many individuals becomes extremely taxing and time consuming. When considering the volume of information within a chat, it becomes difficult for a moderator to serve as the sole source of guidance for individuals. These problems present an opportunity to automate the connection of individuals within a community.

In this project, I propose a virtual agent, Percey, which works with Slack mods to understand the state of the community and provide connections on an individualized basis. In order to complete the job of an administrator or human, the agent (Percey) must complete 2 jobs with higher speed, accuracy, and precision than a human:

(1) Understand Quantitatively the State of the Community
(2) Suggest connections within the community

1. Understand Quantitatively the State of the Community

Percey would obtain rudimentary statistics on the Slack group like the number of messages per user, keywords, and aggregate statistics. However this alone will not provide a similar representation of the group to that of a human. Percey models the concepts and entities used within the group along with their relationships in order for it to provide the best contextual connections. For the sake of scope, this part of the project was first modeled using data from BioBreaks, a biotech focused community. A demo web application was built to visualize the underlying network within the community. It can be viewed at percey.herokuapp.com/demo. Network statistics were drawn from this community. The statistics can be compared to those of characterized networks in literature.

2. Suggest connections within the community

Percey determines the best possible connections within a community at any point in the network's development. In order to do so Percey will simulate the growth and development of the community and suggest user connections daily for the BioBreaks community.

The following five assumptions are made to model the interactions within the group:

- A power law distribution of user types within the group[iii].
- A constant growth rate of the community
- Members do not transition between user types upon connection

- Mentioning another user is considered as a relationship between two users
- Individual users have a propensity to connect when prompted with a probability. In a normal chat scenario the outcome of a suggested connection would be dependent on the participation of the individuals. We model that probability as the average of the individual user's per match archetype probabilities.

In suggesting connections Percey takes into account the network state, individual user states, along with "tags" that reflect the content of messages in the chat stream. Based on the collected information, Percey runs a Markovian Decision Process (MDP) and Reinforcement learning to predict the best connections for the community on a daily basis.

Term Definitions

The important terms are described below:

Simulation - The imitative representation of the functioning of one system or process by means of the functioning of another [iv]
Suggestion - the process by which one thought leads to another especially through association of ideas [v]
Social Evolution - the gradual development of society and social forms, institutions, etc., (usually through a series of peaceful stages)[vi]
Slack - a free messaging app for teams or groups
Slack community - a group of people brought together under a common theme utilizing Slack as their communication tool
Match - a suggested successful relationship from Percey
Connection - the relationship between 2 entities, user or tag, in Slack

Metrics

In order to evaluate the model, the metrics will be measured for users, the network, and model output.

A. Users

Users will be scored based on a formula I derived:

$$UserScore = \frac{m}{n(m)} + \frac{ce}{n(ce)} + \frac{b}{n(b)} + \frac{cl}{n(cl)} + \frac{e}{n(e)}$$

where:
n(x) = the average “x” across all the users in the graph
{x= ”m”, ”ce”, ”b”, ”cl”, or ”e” }
m = the number of msgs per day a user sends
ce = the centrality of the user

b = betweeness of the user (how central is the user)
cl = closeness of the user (cumulative distance)
e = eigenvector of the user (influence)

This score will be useful when determining new connections for the graph. It represents the importance of an individual within the network. Although these stats are used independently in previous research[vii], this combination and normalization of user statistics will allow for simple matching.

Each User will also have an associated weighted Tag property which reflects their interests based on messaging. Each Tag will have a weight which when multiplied by their messages per day reflects their relative significance for a given tag. In other words we can determine the most active User for a particular tag based on this calculation. This tag property is not used in determining the UserScore. It is only used to match Users within the community. Matching is discussed further in *Analysis>Take and Evaluate Action*

B. The Network

The network will be measured based on individual statistics including:

Radius - The distance (number of relationship lengths) from the central node or user to the furthest node
Diameter - the longest distance across the graph
Centrality - indicates whether users within the network have a different number of relations. A network-degree-centrality of 0 means that all Users have the same number of relations, while a network-degree centrality of 1 indicates a star network (all Users are connected to one User).
Average clustering coefficient - indicates how well neighboring Users are connected to each other[viii]
Node connectivity - The minimum number of nodes that must be removed to disconnect the graph or render it trivial.
Node count – The number of nodes in the graph
Components – The number of individual components or clusters within the graph
Average path length – The average shortest path between a node and all other nodes in the network

These statistics were chosen based on findings in literature, common approaches to characterizing graphs, socilab [ix], a LinkedIn network visualization tool, and algorithm availability in networkX.[x]

Network Types

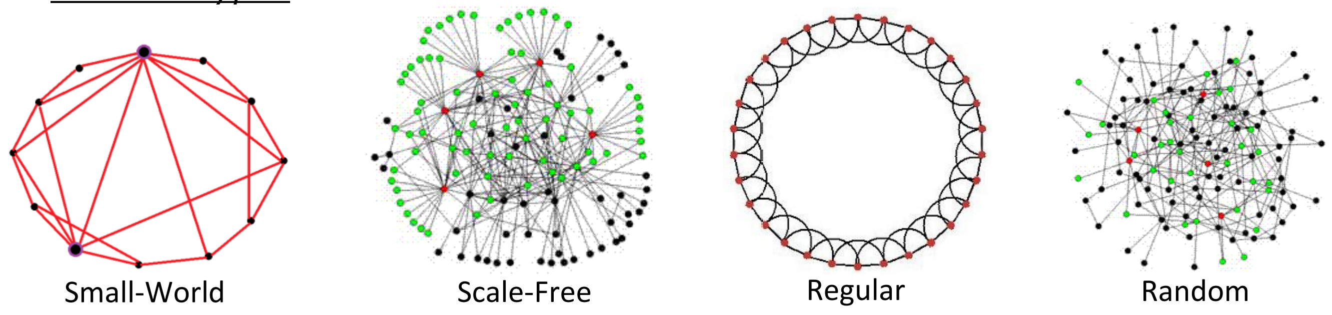


Figure 1. From left to right: small-world network, scale free networks, regular networks, and random networks

Four common types of networks exist as shown in Figure 1.[xi][xii] A random network consists of nodes where each node pair is connected with a random probability. A small world network represents a graph where most nodes are not connected to their neighbors but they can reach their neighbors by a small number of hops along the network[xiii]. Scale-Free networks have degree distributions which follow a power law[xiv]. This means the number of neighbors per node across the population follows a power law[xv]. Regular networks are engineered or follow a pattern. in the connections between nodes.

C. Model Output

The goal of the agent is to learn how to manage a community so as to be able to have a productive group over a period of 20 days. The outputs of the model will be:

- The final state of the graph compared to the seed graph
- The success rate of connection compared to a completely random model.
- The number of iterations before reaching optimal network parameters
- The percentage of iterations where the model is trending near or in the destined state

II. Analysis

Analysis consists of Data Exploration, Exploratory Visualization, Algorithms and Techniques, and Benchmark.

Data Exploration
The input data for this project was derived from the real data of the BioBreaks Slack group. The Slack group consists of 49 members with 10 channels with over 600 messages. The statistics for the dataset are given below.

Number of Users: 50
Number of Messages: 239
Number of Tags:
Seed Graph Stats:

```
    {
      'avgpathlength': 1.0,
      'radius': 1,
      'center': [0, 1, 2, 3, 4],
      'components': 1,
      'density': 1.0,
      'nodecount': 5,
      'avgcluscoeff': 1.0,
      'nodeconnectivity': 4
     }
```

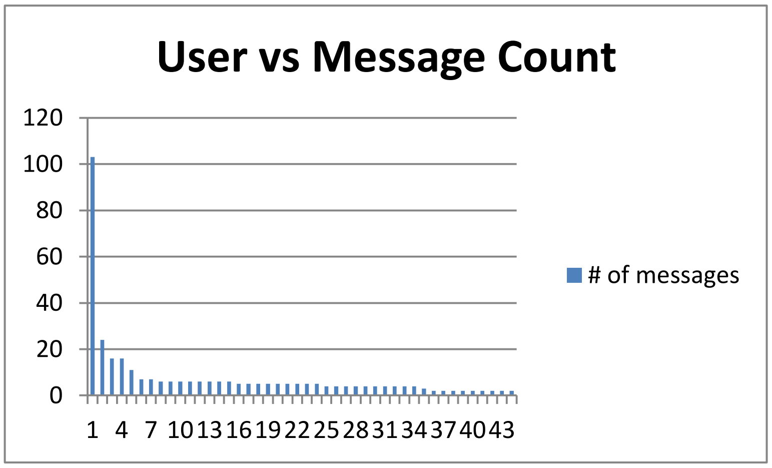


Figure 2. This graph shows each user in BioBreaks and the number of messages sent.

An interesting thing to note about this group is that the group's messages are dominated by 1 person with over 50% of the messages. This is typical for a group, representing a scale free network with a power law distribution. Similar phenomena can be viewed throughout the other statistics. This data is useful for understanding how a group might grow when interactions are minimized, new user recruitment is average, and the job of creating connections is limited to only one user. Hence, this data serves two purposes: First, it helps to give a sense for the initial statistics of a starting Slack community and will serve as a benchmark to compare when the simulated model reaches a size of 50.

The initial graph will be generated using a similar approach to Zeng and Sheng[xvi]. They use a stratified random sampling algorithm to generate a power-law model with a fat-tail. Percey takes a random sampling of the BioBreaks community graph in order to create the seed graph. The resultant graph has 5 users in it initially with perfect connections (each user is connected to all other users).

Exploratory Visualization

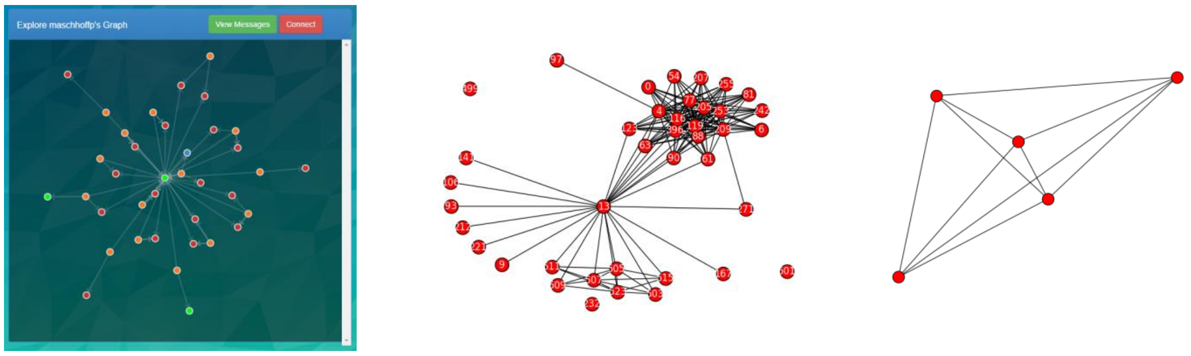


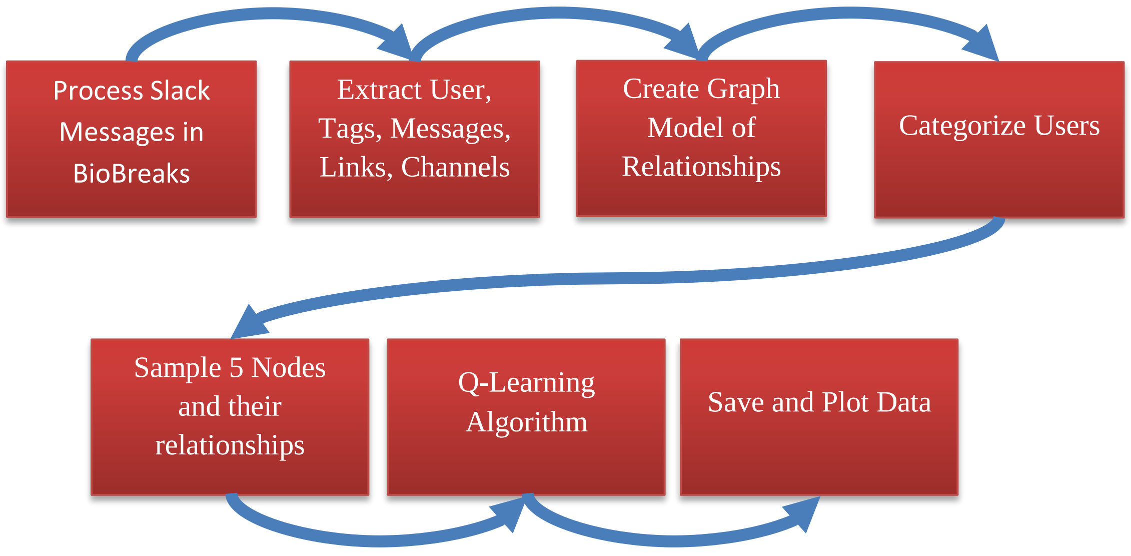
Figure 3. From left to right this figure visualizes a single user graph on Percey.herokuapp.com/demo, the user connections graph of the community, along with the seed graph for the simulation.

The three pictures in Figure 2 give great indication of the complexity of networks. The first picture shows an individual user's connections to messages (orange), links (blue), tags (red), and other users (green). The scope of the project limits monitoring to understanding the relationships between tags and users within the community. Figure two shows the connections formed between users in the community. User 13, the owner and creator of the group, serves as a connection hub. Finally the third picture shows the perfectly seeded graph where all nodes are connected to each other. What this graph doesn't show is the tag attributes of each node which reflect 3 interests (tags) per user.

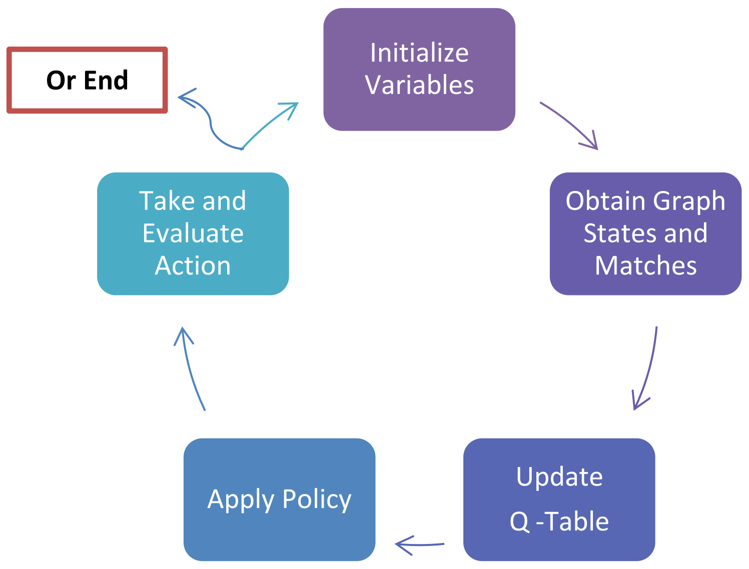
Algorithms and Techniques

In order to solve this problem Reinforcement Learning and Q-Learning are used. The technique workflow is as follows:

Overall Model


Process Slack Messages in BioBreaks
Extract User, Tags, Messages, Links, Channels
Create Graph Model of Relationships
Categorize Users
Sample 5 Nodes and their relationships
Q-Learning Algorithm
Save and Plot Data

Q-Learning Algorithm Overview



The Overall Model shows the process for building, and evaluating the model. The most important part of the model is the Q-Learning algorithm. I will discuss the approach taken for Obtaining Graph States and Matches, Updating the Q-Table, and Taking Action.

The goal of the Q-Learning algorithm is to select actions that will return a positive reward while exploring the possibility space and learning overtime. The state-space parameters used for the Q-Learner include *graphStats*, which is made up of all the graph statistics of the network, and the previous *action* taken to reach the current state. The possible actions for the model to take on the network include:

Connect: Connects 2 users using a matching function (Random or Percey)
Disconnect: Disconnects two users
Remove Node: Removes a user
None: No action is taken

The input parameters to initiate the growth of the network are *repeat*, which is the number of iterations the model will run for, *matchrate*, which is the number of internal matches identified per iteration, *growthrate*, which is the number of new nodes added to the network per iteration, and *random*, which is a Boolean defining whether the model selects and matches nodes randomly or uses the matching function. A sample visualization of the growth of the network over iterations is shown in Appendix B.

Obtaining Graph States and Matches

Each state is saved based on the graph statistics generated per network, as outlined in *Metrics>Network*. The state-space parameters include *graphStats* and *self.old_action* as shown in Figure 4. In order to minimize the number of states generated and ensure state learning each current graph statistic parameter is compared to its corollary in *old_graph_Stats* and given a value of -1, 0, or 1. This discretizes the continuous graph statistics. Possible drawbacks for this approach are that the algorithm reaches a local minima in which it revolves around certain network properties or the q-learner does not reach does properly value the goal graph state. The code is shown below:

```
graphStats = self.normalizeStats(graphStats)
temp = graphStats
if self.old_graph_Stats != None:
    for attribute in graphStats:
        if graphStats[attribute] > self.old_graph_Stats[attribute]:
            graphStats[attribute] = 1
        if graphStats[attribute] < self.old_graph_Stats[attribute]:
            graphStats[attribute] = -1
        if graphStats[attribute] == self.old_graph_Stats[attribute]:
            graphStats[attribute] = 0
    self.state = (graphStats, self.old_action)
```

Figure 4. Obtaining Graph States

Update Q-Table

The q table holds the information on the relative value of a given action in a given context. It takes into account the past actions rewards, and the current state to determine the q value. It can be considered as the action reaction index for finding the best action given a state and previous performance. The formula for q – value is shown below along with the accompanying code:

$$\boldsymbol{q(oldindex, action) = \alpha * (R + \gamma * qMax - q(oldindex, action)}$$

where α = alpha, R =the reward, γ = gamma, $qMax$ = the maximum q for that state, and $q(oldindex, action)$= the q value we are updating

```
self.update_q(self.old_state, self.old_action, self.old_reward, self.state)
…
def update_q(self, oldstate, actionlist, reward, newstate):

    qMax = self.maxqs(newstate) # Finds qMax for a given state
    oldindex = self.states.index(oldstate)

    for action in actionlist:
        action = action[0]

        if oldstate in self.states and self.q[oldindex][action] is not None:
            self.q[oldindex][action] += self.alpha * (reward + self.gamma * qMax
        else:
            self.states.append(oldstate)
            self.q[oldindex][action] = reward
```

Figure 5. Updating the q_table

Take and Evaluate Action

Once the q table is updated the next step is finding connections within the network and connecting the newly added nodes. Connections are determined based on similarity in tag interest and an affinity towards nodes with a higher UserScore. In the following code we compare the tags of two users to determine their closeness and then sort the possible connection by closeness and userscore.

```
# Check if nodes have same tags
for option, key in enumerate(allnodes):
    close = 1 - spatial.distance.cosine(graph.node[node]["tags"], graph.node[option]["tags"])
    matchray[option] = {"spdist": close, "userscore": graph.node[option]['userscore']}
    if key > len(allnodes):
        break

# Check if nodes have high degree amongst tags
sorted_matchray = sorted(matchray.items(), key=lambda x: (x[1]['spdist'],x[1]['userscore']))
```

Figure 6. Matching users based on tags and UserScore

Benchmark

In order to evaluate the effectiveness of the agent, Percey, I compared the agent's outcomes to a purely randomized selection and intervention process. Zeng and Sheng used similar randomized sampling and comparison on their simulation trials. With an intelligent agent focused on improving the characterizing properties of graphs, the clustering coefficient and path length, the final graph and majority of the states should have characteristics that reflect an optimal graph. An optimal graph in the context of a Slack community places an emphasis on equality of degree distribution i.e. incoming nodes would be able to reach a higher level of connectedness. There would be increased participation amongst members, and a closer knit community. These qualities translate to quantitatively comparing the final graphs' statistics and noting the degree to which one trumps another. The better model will generate graphs with a smaller radius, diameter, and components and a higher centrality, clustering coefficient, node connectivity, and eigenvector.

III. Methodology

Data Preprocessing

The seed data was sampled from the BioBreaks graph data. The BioBreaks graph data was generated by using the slack API to process the individual messages in the BioBreaks group. Using the NLTK, important entities per message, like Links, Channels, and other Users were extracted per message.

I also used part of speech tagging to extract the top 3 noun keywords (Tags) for each message. With the entities identified relationships were created in a neo4j GraphDB. Some example relationships created included, "(Message)-[By]-(User)", "(User)-[Mentions]-(Tag)", "(User)-[Mentions]-(User)", and "(Message)-[Includes]-(Tag)". A more detailed explanation can be found in app.py and Figure 7. Afterwards, sampling from the BioBreaks graph maintained the distribution of user types within the 5 original nodes.

```
# Parses the keywords from a given Slack message
def parseTags(msg):
    tagged_msg = pos_tag(msg.split())
    BAD_CHARS = ":.!?,\'\""
    allnouns = [word.strip(BAD_CHARS) for word, pos in tagged_msg if (pos == 'NNP' or
pos == 'NN') and len(word) > 2]
    return allnouns
```

Figure 7. Creating a graph representation of a Slack community requires extracting keywords from messages

Implementation

Implementing the algorithms brought on a set of unique and interesting challenges. One challenge was finding a way to discretize individual states. Each state is made of up of the possible node matches to act upon and the graph's statistics. Each of the graph's 6 properties are continuous whereas the action has only 4 options. I considered three approaches. The classical approach is to minimize the number of states by splitting the continuous variables into a chosen number of ranges. Another approach is to combine the features into one score that reflects the graph state. I believe this method oversimplifies the information and still requires splitting the final score into ranges. My chosen approach was to reflect the directional change of each graph property. If the property increased relative to its previous state, it was given a value of 1. A similar strategy was employed for no change (0) and a lower value (-1).

Another challenge was tuning the input parameter matches. This parameter determines the number of matches that will be considered per day. Initially I believed this number should be lower to reflect reality considering the fact that a Slack admin has a limited amount of time to engage with the community. However upon using a constant match rate I realized connection within the community needed to be separated into two components: connecting new individuals to people within the community and connecting individuals within the community.

Likewise, initially I believed network statistics could be completed on the entire graph irrespective of its composition. For some of the properties like average clustering coefficient, I realized the graph had to be broken into components before computing the dominant component's properties. Other challenges included selecting the best data structures and normal debugging.

Refinement

In order to refine my initial approach I addressed the challenges outlined in *Implementation*. The parameters of interest when refining the model included the growth-rate, number of repetitions, and the initial number of matches.

I took an iterative approach to tuning these input parameters to obtain the best model. I varied the growth rate and initial number of matches from 1 to 10 in increments of 1 and the repeat rate from 2 to 20 in increments of 2.

Implementing these refinements required placing the main agent call within a function and running the various permutations as seen the function experiment(). An output of experimentation can be viewed in Appendix A.

IV. Results

Model Evaluation and Validation

After refinement the final chosen model comprised of the best combination of parameters. I took into account the need for the parameters to be realistic when picking the final combination. This took away some interesting choices. Ultimately I was looking for the most generalizable model. The chosen input parameters (repeat = 20, growthrate=2, matchrate = 5) lead to a model that on average better in 5 of 6 graph properties against the random network. The data showed an increased repeat, or the number of iterations for the model to work, correlated with higher density network and shorter path lengths. Appendix A shows the full table of experiments and the properties of the resultant networks. Figure 4 shows how the model fared against the randomized trial overall.

Overall Comparison, Averages over 52 runs (26 Random, 26 Model)

	repeat	growthrate	matchrate	radius	avg path length	node connectivity	density	node count	avg. cluscoeff
Model	16.538	3.077	5.154	**3.577**	**3.019**	0.942	**0.208**	**27.212**	**0.110**
Random	16.560	3.120	5.180	3.700	3.120	**0.960**	0.197	28.240	0.114

Figure 8 Overall Model Comparison for the Model and Random Networks. Repeat, Growthrate and Matchrate are the inpute parameters while the columns to the left reflect the network's average properties

For all of the above properties except for node connectivity the average model network showed an improvement over a random network. A smaller radius, path length, and clustering coefficient indicates a tighter knit community in which information dispersion occurs faster. The same can be said for a greater density. The model network has a lower node connectivity, indicating that there are fewer connections. However this can be attributed to the greater selectivity of the model network and unwillingness to connect willy-nilly.

Justification

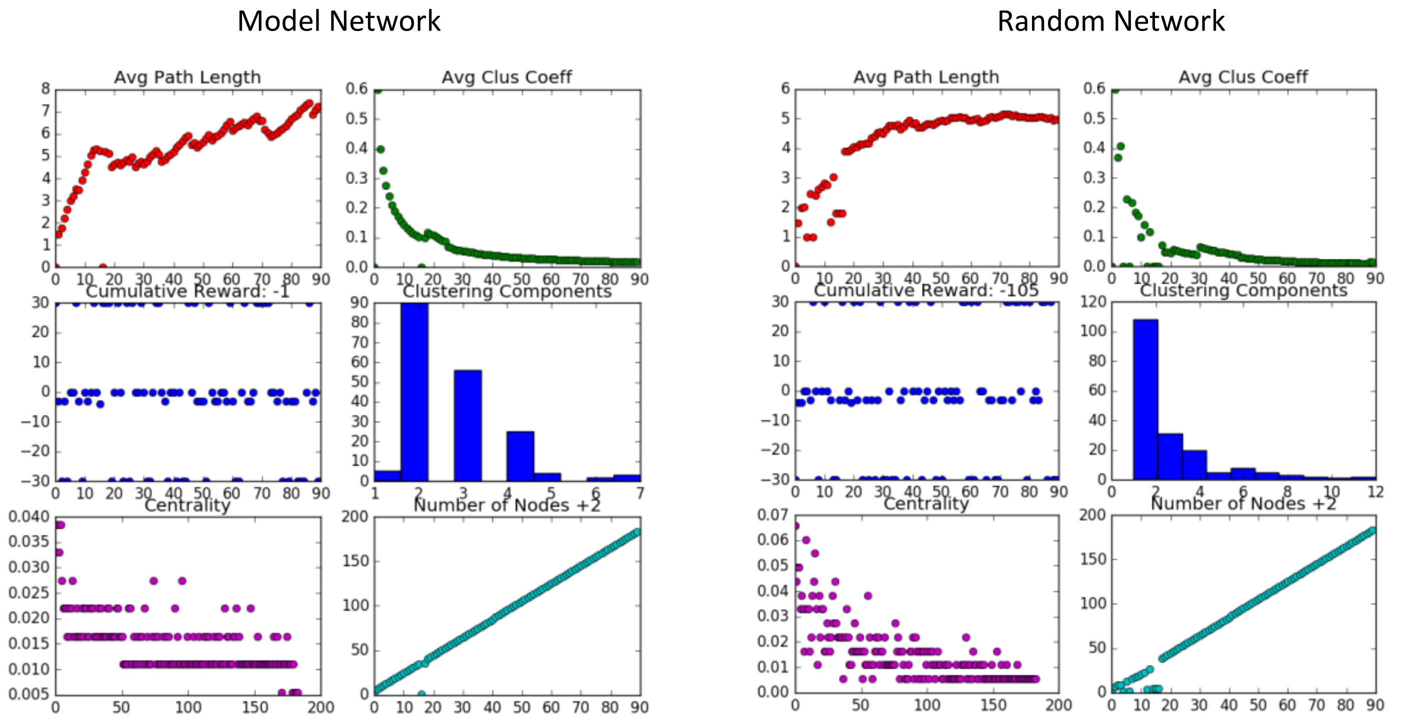


Figure 9. Trial Results for a 90 day/iteration repeat, Growthrate of 2, and matchrate of 3. All graphs with the exception of the Clustering Components Graph shows iteration vs. graph statistic (Avg. Path Length, Avg. Clustering Coefficient, Reward, Centra

The resultant statistics after running the model and random benchmark for a single trial are shown in Figure 5 on the left and right respectively. Focusing on the clustering components emphasizes an important difference between the two resultant networks. The distribution of number of connections in the model network skews to the right whereas in a random network fewer nodes have more than 2 connections. This reflects the model's ability to connect individuals both successfully and in a distributed manner which elevates the performance of the entire group.

V. Conclusion

Free-Form Visualization

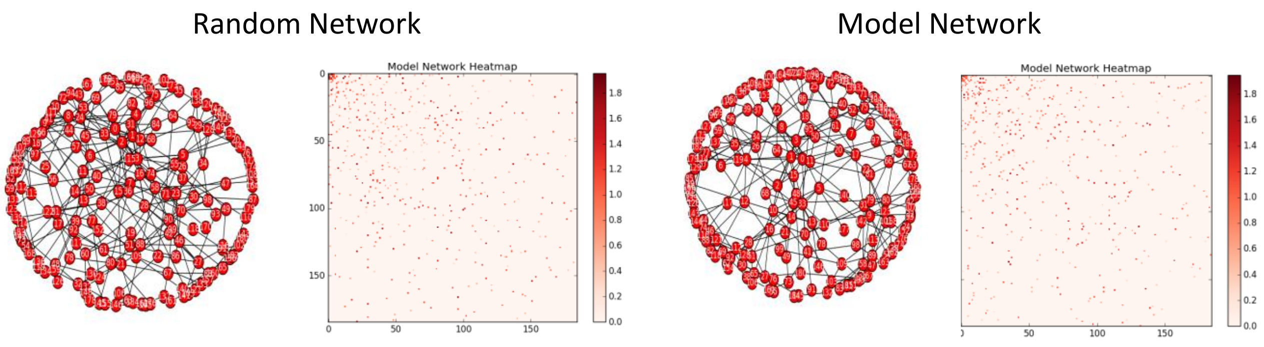


Figure 10. Random and Model Network Graphs and heat maps.

When working with networks, graphs, and communities it is very easy to get caught up in the surface observations. For example, the figure above shows two networks. One looks denser than the other. What really matters about these graphs is how they got to this state and what their current structure indicates about the activity of the group. A simple observation such as the density does not tell the whole story. The random graph on left, although more dense was built in a way in which individual user preferences were not taken into account and the linkages between users are weak. A denser network has a higher clustering coefficient and shorter path length, but the network on the right exhibits better engagement properties due to the quality of the connections. The heatmap reflects the quality of connections between two users in the community. This quality is considered to be the maximization of UserScore and the interest similarity score through tags. As seen more clearly in Appendix C., the quality of connections within the first 50 nodes of the Model Network is slightly higher than for the random Network. This is in line with the tendency for humans to connect with the more established players and hubs when entering a new network. The point to be taken is that the outcomes of a directed, selected, and considered evolution of a network may lead to a non-intuitively structured network. This solidifies the need for a solution that can comprehend hyper-locally throughout the network. The suggested model, Percey, is fully capable of processing individualized contextual information in order to take actions that will improve the network.

Reflection

The process for this project was as follows:

1. Create BioBreaks Slack, Obtain Users, Engage with them
2. Connect to Slack API and get Slack community messages
3. Extract Entities from Messages (NLTK) and generate a network relationships between them
4. Sample a seed graph from the Graph database
5. Create the Network growth Environment
6. Run the network through multiple environments
7. Measure Results

The most difficult parts of this process were creating the initial relationship model and building out a robust network environment. Using the natural language toolkit was a first for me. However, I was able to wrap my head around CYPHER queries and handling the database over a period of time. Building the network environment was difficult because problems would exacerbate and have huge effects on the network growth. It also required a different kind of thinking in which the problem was nuanced and the goal was always to find the best way to approximate human connections. Also, I learned to appreciate wonderful libraries like networkX with its spotless documentation and realized that it’s often better to pick an open source well documented difficult solution than to deal with having to fix bugs in an outdated alchemy.js repository. I also learned the power of networks from many interesting researchers who are working on challenging problems at scale, considering social network analysis is not my background. I really enjoyed building the visualizations both on the web and in python. I’m looking forward to handling more data and 3 dimensional visualizations with animation. I believe this type of analysis and objective focused on directing systems for a purpose transcends domains.

Providing valuable connections to a Slack community is hard for moderators, especially for larger groups. Percey creates a relationship graph of the group based on messaging in order to suggest the best user matches for increased group engagement, and performance over time.

Improvement

One of the challenges of building a simulation is that you have to find the sweet spot between reality and limitations. Many opportunities to improve this model for simulating directed evolution include (from perceived simplicity to most complex):

1. Consideration of member defection rate

2. Better characterization of graph states
3. Increasing the quantity of the initial seed data
4. Better modeling of user interaction patterns

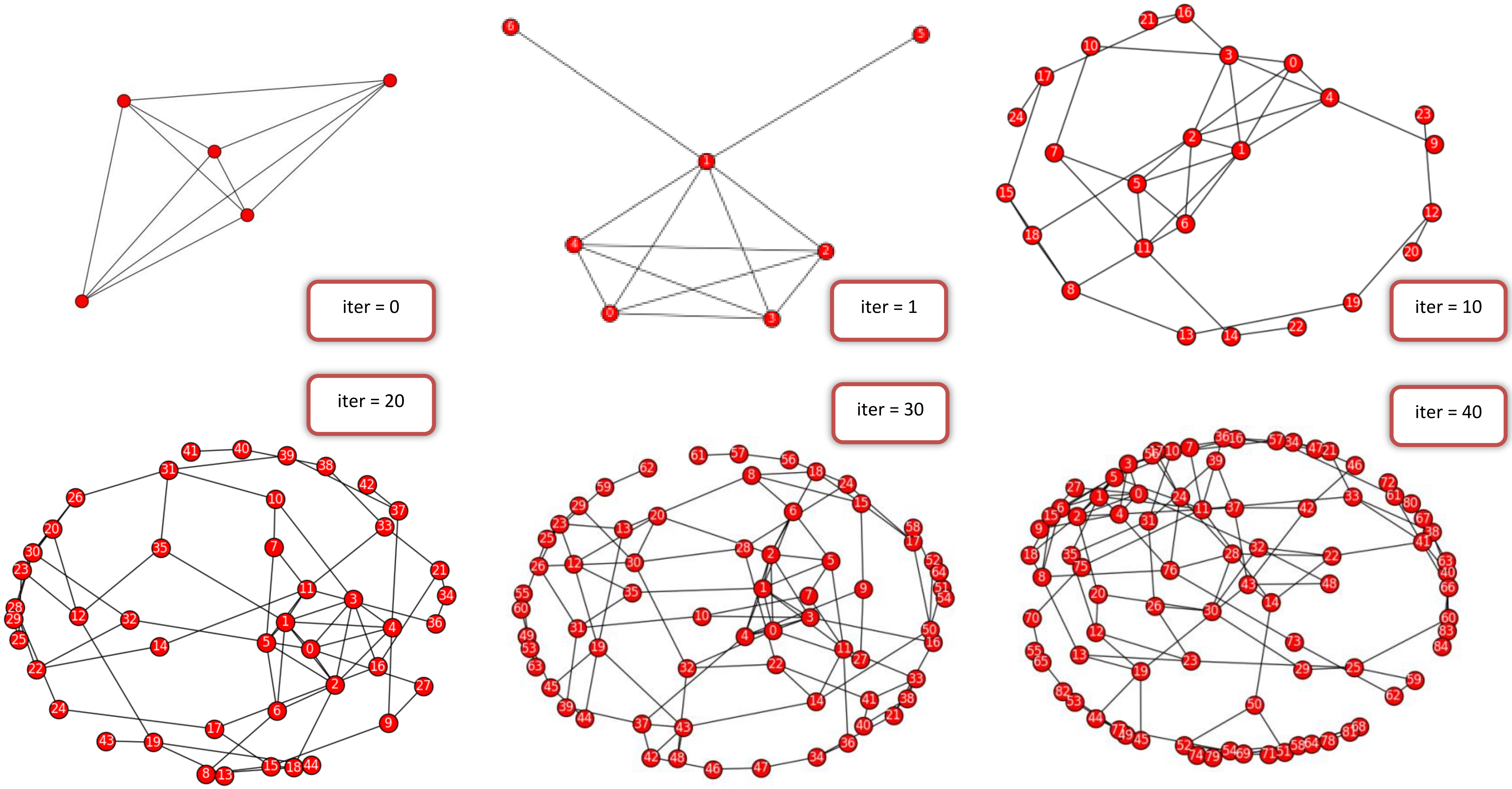
For (1), I did not have a basis for which to determine this rate. Implementing (2), requires dealing with discretization in a more classical manner requiring more algorithms to handle the uncertainty. (3), is out of my hands and reflects the state of social network analysis research. There are many datasets for social network analysis. Few researchers focus on the evolutionary aspect of them nevertheless in a Slack context. With time, more data about this approach will be obtained and shared. Ideally the resultant data will provide fundamental insights into the probabilistic behavior of humans when they are provided with directed interaction in an online community (4). When dealing in the extremely niche sphere of social network analysis and evolution of online forums like Slack by default, my model stands out due to lack of other competitors. However, I look forward to replicating the rigor and thoughtfulness of Zeng and Sheng to improve the model. As social engineering transitions from a lethal weapon employed by hackers to an open tool to be utilized by the general population, the opportunities to improve on this model will explode. Hopefully, my model's effectiveness explodes as well.

Appendix A.

model	repeat	growthrat	matchrate	radius	avgpathle	nodeconn	center	density	componer	nodecoun	avgcluscoe
random	2	2	5	2	2.095238	1	[1]	0.333333	1	7	0
random	4	2	5	2	2.055556	1	[4]	0.333333	1	9	0.340741
random	6	2	5	3	2.285714	1	[0, 1, 2, 3,	0.238095	1	15	0.274603
random	8	2	5	0	0	0	[11]	0	1	1	0
random	10	2	5	3	3.003953	1	[8, 12]	0.114625	1	23	0.135197
random	12	2	5	4	3.287749	1	[3, 6, 7]	0.105413	1	27	0.103439
random	14	2	5	4	3.219355	1	[0]	0.096774	1	31	0.128264
random	16	2	5	4	3.670588	1	[3]	0.068908	1	35	0.069932
random	18	2	5	1	1.333333	1	[12]	0.666667	1	3	0
random	20	2	5	3	3.164925	1	[1]	0.073171	1	42	0.039985
model	2	2	5	2	1.47619	1	[0, 1, 2, 3,	0.571429	1	7	0.6
model	4	2	5	3	2.309091	1	[0, 1, 2, 3,	0.290909	1	11	0.345455
model	6	2	5	3	2.371429	1	[7, 9, 10]	0.257143	1	15	0.333333
model	8	2	5	4	2.654971	1	[0, 1, 4, 7,	0.204678	1	19	0.223058
model	10	2	5	4	3.839827	1	[0]	0.121212	1	22	0.172727
model	12	2	5	1	1	1	[19, 23]	1	1	2	0
model	14	2	5	6	3.292473	1	[0, 1, 2, 3,	0.124731	1	31	0.144547
model	16	2	5	5	3.607843	1	[0, 2, 13, 1	0.087344	1	34	0.087255
model	18	2	5	7	4.588394	1	[10]	0.078273	1	39	0.083476
model	20	2	5	5	3.537099	1	[1, 2, 3, 4,	0.074197	1	43	0.065116
random	20	2	5	3	2.782609	1	[0, 7, 9]	0.134387	1	23	0.199793
random	20	3	5	4	3.167339	1	[0, 1, 2, 3,	0.092742	1	32	0.060491
random	20	4	5	4	3.842683	1	[4]	0.059756	1	41	0.074042
random	20	5	5	5	4.086531	1	[0, 1, 2, 3,	0.050612	1	50	0.042857
random	20	6	5	0	0	0	[36]	0	1	1	0
random	20	7	5	6	5.119842	1	[0, 2, 3, 13	0.031607	1	68	0
random	20	8	5	1	1.333333	1	[23]	0.666667	1	3	0
random	20	9	5	6	4.829275	1	[0, 1, 2, 3,	0.025992	1	86	0.027014
model	20	2	5	4	2.675889	1	[0, 1, 2, 3,	0.162055	1	23	0.215942
model	20	3	5	1	1.333333	1	[23]	0.666667	1	3	0
model	20	4	5	5	4.123171	1	[14]	0.073171	1	41	0.042276
model	20	5	5	6	5.496327	1	[0, 1, 2, 3,	0.04898	1	50	0.06
model	20	6	5	7	7.223261	1	[3]	0.034483	1	59	0.025424
model	20	7	5	5	5.012291	1	[0, 1, 2, 3,	0.035996	1	68	0.047549
model	20	8	5	5	4.616883	1	[0, 1, 2, 3,	0.033151	1	77	0.037724
model	20	9	5	5	5.437756	1	[0, 1, 2, 3,	0.025992	1	86	0.024252
random	20	2	2	1	1	1	[9, 17]	1	1	2	0
random	20	2	3	3	2.608696	1	[0, 1, 2, 3,	0.134387	1	23	0.153382
random	20	2	4	4	3.245059	1	[4]	0.110672	1	23	0.090683
random	20	2	5	4	3.268775	1	[1, 3, 4]	0.118577	1	23	0.105797
random	20	2	6	3	3.146245	1	[6]	0.118577	1	23	0.115528
random	20	2	7	4	2.916996	1	[2, 4, 6, 8,	0.12253	1	23	0.17971
random	20	2	8	3	2.56917	1	[3, 4, 5, 9]	0.15415	1	23	0.132505
random	20	2	9	3	2.513834	1	[3, 6, 8, 11	0.16996	1	23	0.161594
model	20	2	2	5	3.841897	1	[5]	0.118577	1	23	0.147826
model	20	2	3	4	3.189723	1	[5, 6, 9, 12	0.126482	1	23	0.14058
model	20	2	4	0	0	0	[13]	0	1	1	0
model	20	2	5	1	1	1	[8, 20]	1	1	2	0
model	20	2	6	4	2.936759	1	[0, 4, 8]	0.166008	1	23	0.203313
model	20	2	7	6	4.667984	1	[2]	0.114625	1	23	0.130435
model	20	2	8	5	3.636364	1	[2, 4, 6]	0.142857	1	22	0.154545
model	20	2	9	3	2.571429	1	[9, 11]	0.25	1	8	0

Figure 11 Yellow: Testing Repeat, Pink: Testing growrthrate, Purple: Testing matchrate

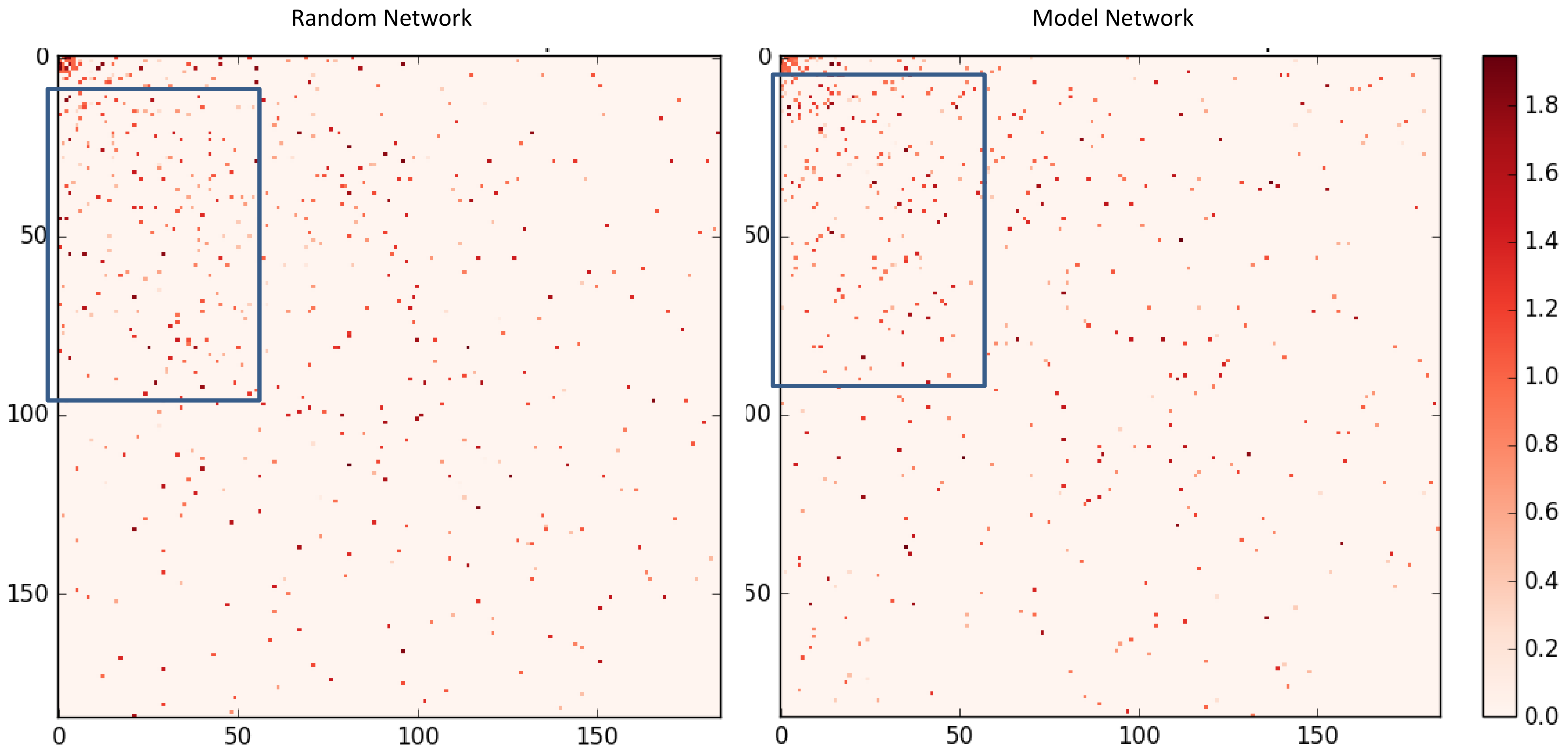
Appendix B.



Graphs show iteration 0, 1, 10, 20, 30, and 40 for the growing Model Network. Match rate = 5, Growth rate = 2

Check out a gif version of the network growth HERE.

Appendix C.


Random Network
Model Network
0
50
100
150
0
50
100
150
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0

Sources

[i] Rutledge, Kim. "Civilization." *National Geographic Society*. National Geographic, 21 Jan. 2011. Web. 21 Aug. 2016. <http://nationalgeographic.org/encyclopedia/civilization/http://nationalgeographic.org/encyclopedia/civilization/>.

[ii] Mislove, Alan. *Online Social Networks: Measurement, Analysis, and Applications to Distributed Information Systems*. Thesis. Rice University, 2009. N.p.: n.p., n.d. Web. <https://scholarship.rice.edu/bitstream/handle/1911/61861/3362365.PDF?sequence=1&isAllowed=yhttps://scholarship.rice.edu/bitstream/handle/1911/61861/3362365.PDF?sequence=1&isAllowed=y>.

[iii]http://www.imsc.res.in/~sitabhra/meetings/school10/Jari_Chennai2010_lecture2.pdf

[iv] http://www.merriam-webster.com/dictionary/simulation

[v] http://www.merriam-webster.com/dictionary/suggestion

[vi] http://www.dictionary.com/browse/social-evolution

[vii]https://www.researchgate.net/post/How_can_we_measure_node_importance

[viii]http://networkx.readthedocs.io/en/stable/_images/math/70e9b7898d39aa4caf7c7ead0c410e6cbe13bf37.png

[ix] www.Socilab.com

[x] https://networkx.github.io/documentation/networkx-1.10/reference/algorithms.html

[xi] http://barabasi.com/f/124.pdf

[xii] http://barabasi.com/f/624.pdf

[xiii] Labs, Nodus. "Types of Networks: Random, Small-World, Scale-Free." *Nodus Labs*. N.p., 05 Apr. 2012. Web. 21 Aug. 2016. <http://noduslabs.com/radar/types-networks-random-small-world-scale-free/>.

[xiv] Toriumi, F., I. Okada, and H. Yamamoto. "Classification of Social Network Sites Based on Network Indexes and Communication Patterns."*Classification of Social Network Sites Based on Network Indexes and Communication Patterns* (n.d.): n. pag. *Http://hitoshi.isslab.org/*. Web. <http://hitoshi.isslab.org/study_work/2011/swm.pdf

[xv] "Statistical Classification of Social Networks." *IEEE Xplore*. N.p., n.d. Web. 21 Aug. 2016. <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6288789&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D6288789>.

[xvi] http://crpit.com/confpapers/CRPITV144Zeng.pdf