Second Assignment: Wikipedia Norms

Puerta de Toledo

Master in Big Data Analytics

Network analysis and data visualization

Ion Bueno Ulacia NIA 100364530

Daniel Martín Cruz NIA 100384121

Ion Bueno Ulacia NIA 100364530
Daniel Martín Cruz NIA 100384121

uc3m | Universidad Carlos III de Madrid

Contents

List of Figures

Ion Bueno Ulacia NIA 100364530
Daniel Martín Cruz NIA 100384121

In this report we will continue exploring the network we studied in the first part of the course based on Wikipedia Norms and how they were related. As a reminder, this is a directed network with 1.881 nodes and 17.235 edges. To be more precise, our study will be focused on the giant component we encounter on this network, composed by 1.872 nodes.

This report is divided in three main parts: link prediction algorithms, community detection and spreading process. In sections 2.1 and 3.1, some plots are presented to better understand the study.

1. Link prediction algorithms

We will start the second part of the analysis of the *Wikipedia Norms* network studying different link predictions algorithms that, according to some metrics, are links somehow probable to appear in the network at a given time in the future.

Most of the algorithms that will be explored in this section have been implemented to work over the whole network but, in order to favour an easy interpretation of them, in this report we will only consider the most probable future links of a single node, the one tagged with *Page.ID* = 1.

1.1. Degree Homophily

The first and most basic algorithm that we will explore is degree homophily that basically consists on estimating the probability of two nodes to be linked with the absolute difference of their degrees, being most likely to be linked if their degree is equal.

$$S(x, y) = -|deg(x) - deg(y)| \tag{1}$$

This method may make sense if we have a highly assortative network which, as we saw in the first part of the analysis of the network, is not the case with the Wikipedia Norms network.

As we indicated, we will use this algorithm to predict new links for the node with *Page.ID* = 1. These are the most likely links for this node:

Id	Node	Proximity
10	A Side of Chips	0
22	About the Sandbox	0
91	Anti-Wikipedianism	0
144	Authority control	0
171	Avoiding untrue text in articles	0

Table 1: Most probable links using degree homophily

Here we have left only 5 possible links but all those nodes that have the same degree as our reference will have the maximum possible proximity.

1.2. Graph Distance

The next algorithm we are going to study uses distance between nodes as metric to measure probability of a future link to exist. The idea behind it is that, friends of our friends may be our direct friends in the future. This would be the resulting formula to estimate similarity.

$$S(x, y) = -d_{x,y} \tag{2}$$

Let us take a look now to the most probable links if we use this algorithm.

Id	Node	Proximity
284	Citing textbooks	-2
524	Editor	-2
664	Good-faith Googling	-2
833	Josh Billings	-2
1267	Proposed guideline for lists of people by ethnicity, religion, and other cultural categorizations	-2

Table 2: Most probable links using graph distance

Here we have the nodes that are at a distance of 1 from our reference node or what is the same, the neighbors with a step of 2 from the reference node.

1.3. Common Neighbors

In order to continue with the study of link prediction algorithms using the Wikipedia Norms network we will explore some algorithms based on common neighbors as metric the estimate probability for future links. The most simple algorithm we can implement that uses this idea is by estimating the probability using the number of common neighbors. With this idea, this would be the formula used to calculate proximity:

$$S(x, y) = |V(x) \cup V(y)| \tag{3}$$

Id	Node	Proximity
98	April Fool's Main Pages	9
284	Citing textbooks	9
319	Conflict of interest limit	8
385	Deletion of newly created pages	8
542	Emoticons	8

Table 3: Most probable links using common neighbors

1.3.1. Jaccard's coefficient

A problem of the two previously analyzed algorithms is that they may favour the connections with those nodes that present a higher connectivity. In order to avoid this, we will correct this number of common neighbors using

the total number of neighbors that those two nodes have. This would be the resulting formula.

$$S(x,y) = \frac{|V(x) \cap V(y)|}{|V(x) \cup V(y)|} \tag{4}$$

The results obtained using this prediction link algorithm are the following:

Id	Node	Proximity
98	April Fool's Main Page	108
1137	Obvious sock is obvious	82
342	Copyright on emblems	78
366	Customizing watchlists	77
1572	The logic behind Wikipedia policy	74

Table 4: Most probable links using Jaccard's coefficient

1.3.2. Adamic-Adar coefficient

If in the case of the previous algorithm we wanted to avoid favouring those nodes with a high degree, we will put the focus now on limiting the influence of neighbors with a high connectivity. The approach will be similar to the one we did in the case of common neighbors but each common neighbor will be weighted inversely proportional to its degree. This is the formula used to calculate the similarity between two nodes x and y.

$$S(x,y) = \sum_{z \in V(x) \cap V(y)} \frac{1}{\log |V(z)|} \tag{5}$$

Id	Node	Proximity
98	April Fool's Main Page	1.68
1605	Toby	1.48
1137	Obvious sock is obvious	1.47
385	Deletion of newly created pages	1.45
542	Emoticons	1.45

Table 5: Most probable links using Adamic-Adar coefficient

1.4. Hitting Time

This link prediction algorithm does not return a deterministic output as the ones we have been studying. The proximity metric now equals to the number of steps that a random walk needs to reach a determined node y, starting at another node x. As we have that, the smaller the number of steps, the better, we will change the sign of this number of steps:

$$S(x,y) = -H_{x,y} \tag{6}$$

Where H is the number of steps between x and y.

In order to implement this link prediction algorithm we have defined our own `randomWalk(graph, start,`

`end, count=0, max_steps=1000)` function that is slightly different to the one contained in the `igraph` library. Our function returns the number of steps between two given nodes. We have designed this as a recursive function that has two *stop conditions*. First one is that the `start` node is not the same node as the `end` node. Second one is to have already performed the maximum number of steps in order to avoid an infinite recursion. If neither of these conditions is true, the function gets all the *outgoing* neighbors of a given node and randomly selects one of them as `next_step`. Then, the functions returns a call to itself in the following way:

```
randomWalk(graph, next_step, end, count+1, max_steps)
```

As we said, the result given using this algorithm is not deterministic as all the previous algorithm. For this reason, we have executed 5 simulations of random walks between every pair of nodes analyzed. We have obtained only two links with an average number of steps that does not equal the `max_steps` we defined. Here are these two nodes:

Id	Node	Proximity
1268	Proposed naming conventions and guidelines	-2.33
560	Evaluating Wikipedia as an encyclopedia	-8.33

Table 6: Most probable links using hitting time

1.5. Conclusions

Watching at the results generated by each of these algorithms, we see there exists some links with a highly likelihood of being present on the network. For example, the cases of *April Fool's Main Page* that appears suggested by three algorithms or *Obvious sock is obvious* and *Citing textbooks* that appear as output of two link prediction algorithms.

2. Community Detection

In the first assignment, we employed Gephi tool to analyze and obtain the possible communities in the network. As it can be seen in figure 1, we detected 7 main communities applying *ForceAtlas2*.

Figure 1: Detected communities employing Gephi

In this work, R functions are used to detect communities. Mention the network is directed, so we were not able to employ some algorithms as Louvain due to some errors in the code. We tried the following algorithms:

1. Community structure detection based on edge betweenness: many networks consist of modules which are densely connected themselves but sparsely connected to other modules.
2. Finding communities based on propagating labels. It works by labeling the vertices with unique labels and then updating the labels by majority voting in the neighborhood of the vertex.
3. Community structure via short random walks. This function tries to find densely connected subgraphs via random walks. The idea is that short random walks tend to stay in the same community.
4. Infomap community finding. Find community structure that minimizes the expected description length of a random walker trajectory.

The first two provided us one huge community and many others with only one or two nodes, so taking into account the analysis performed in the first work, it does not make any sense to select one of these. The ones corresponding with the random walk detected three huge communities and then the number of nodes per community was slightly decreasing. These configurations fit better with the previous results.

We finally selected the communities obtained by the third algorithm, with a total of 216. For the spreading process analysis (section 3) as well as the following barplot (figure 2), only the 7 largest are selected, representing the 78% of the whole network. Their sizes are presented in table 7.

Parameter	Number of nodes
Whole network	1.872
Community 1	508
Community 2	471
Community 3	298
Community 4	60
Community 5	47
Community 6	47
Community 7	40

Table 7: Number of nodes in the 7 largest detected communities

2.1. Visualization Network Statistics

In order to analyze the network, we have plotted some network statistics using `ggplot`. In figure 2, we have a barplot representing the average out degree in the 7 largest communities. The biggest communities obtained a higher value, as was likely, since there are more possible connections.

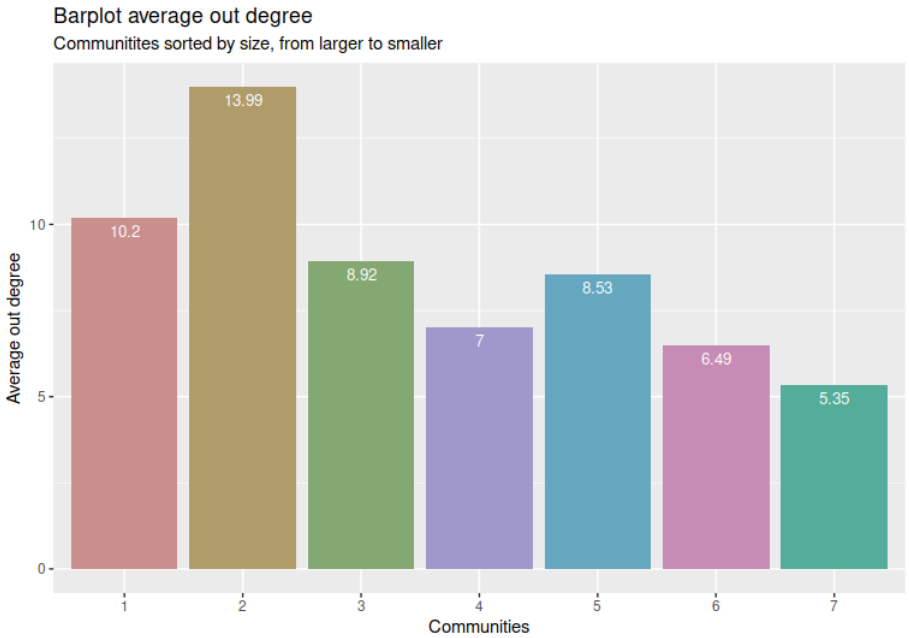

Figure 2: Barplot of the average out degree in the 7 largest communities

In figure 3, a bubble plot is shown in which different network statistics are studied. In this case, all the detected communities (216) are presented. The x-axis corresponds with the average betweenness and the y-axis with the average degree, considering in and out degree. In addition to that, the average closeness is presented with the blue color and the the communities size as the bubbles size in logarithmic scale.

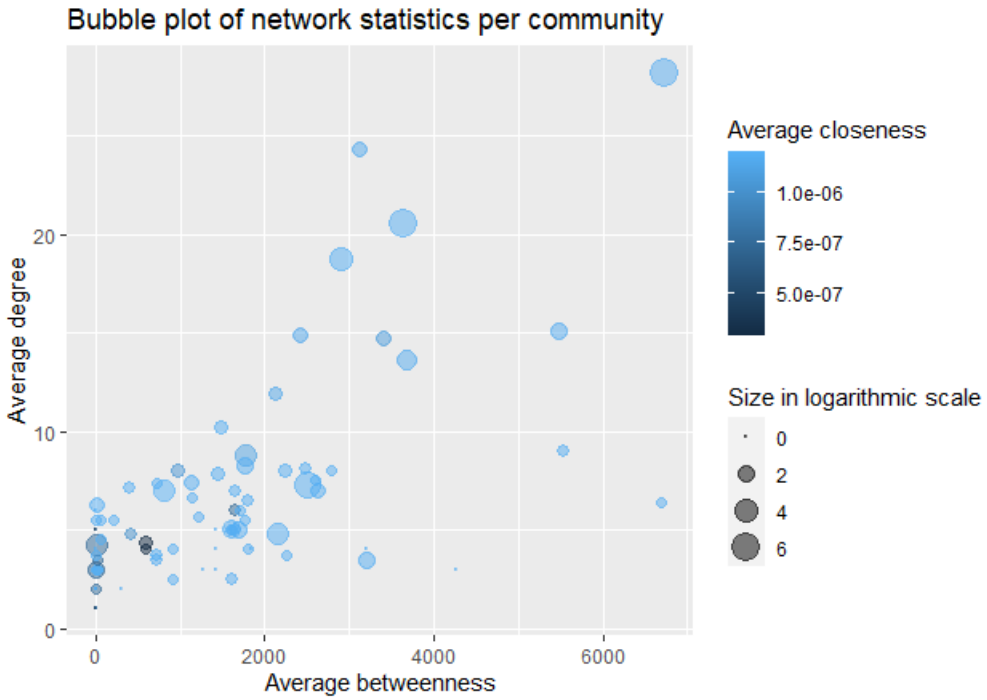

Figure 3: Bubble graph of network statistics

The average betweenness and average degree are positive correlated, as well as the community size, larger communities tend to have larger degree and betweenness. In addition to that, we can appreciate that the average closeness is similar in almost all communities, excluding a few small in size. We can consider than larger communities are more centered according to these statistics.

3. Spreading Process

We have chosen the **SIRS model** to simulate a spreading process. Three states are defined: **susceptible**, **infectious** and **recovered**. As it has been explained in class, the individuals are born into the simulation as susceptible and can be infected with probability β by a network neighbor. They can become recovered with probability γ, but in this state they can turn susceptible again with probability ξ.

The whole process is going to depend on which nodes are initially infected as well as the defined probabilities and the number of steps. The work has been done fixing these parameters, table 8, thinking it as Covid-19 spreading, where it was easy to get infected and became recovered after a time. As few individuals were infected again, meaning that they were susceptible, the probability ξ is not very large, since this situation was not very common.

Parameter	Value
β	0.9
γ	0.6
ξ	0.3
time steps	30

Table 8: Parameters employed for the spreading process

We perform different experiments changing which nodes are infected at the beginning, in order to know which are the best nodes for the spreading process:

- The first 100 nodes with minimum eccentricity in the whole network.
- The nodes with minimum eccentricity per community, a total of 7.
- The first 100 nodes with maximum out degree in the whole network.
- The nodes with maximum out degree per community, a total of 7.
- The first 100 nodes with maximum betweenness in the whole network.
- The nodes with maximum betweenness per community, a total of 7.
- The first 100 nodes with maximum closeness in the whole network.
- The nodes with maximum closeness per community, a total of 7.

In theory, considering these metrics the infection may spread better, since we consider nodes which have a main role in the network due to their connections or positions. In addition, we also tried the contrast, for example nodes with minimum out degree, in order to know if the initialization was significant.

However, there was not a huge difference. In the cases were 100 nodes are selected, the infected ratio for the whole network is larger at the beginning, as expected. When specific nodes from every community are selected, the ratio per community is very high in some of them at certain steps, but the ratio in the whole networks is smaller, since only a few nodes are used.

Nevertheless, the behaviour is almost the same in all the configurations (it will be explained in section 3.1). A significant difference is given if more or less initial nodes are used, for example, with the same configurations and only 10 nodes instead of 100 the infected ratio was much smaller at the beginning.

Finally, for section 3.1, we have chosen as initial nodes the first 100 with maximum out degree in the whole network, which seems to be the best configuration for the spreading process in terms of infected ratio.

3.1. Visualization Spreading Process

In figure 4, it is visualized the evolution of infectious, recovered and susceptible individuals.

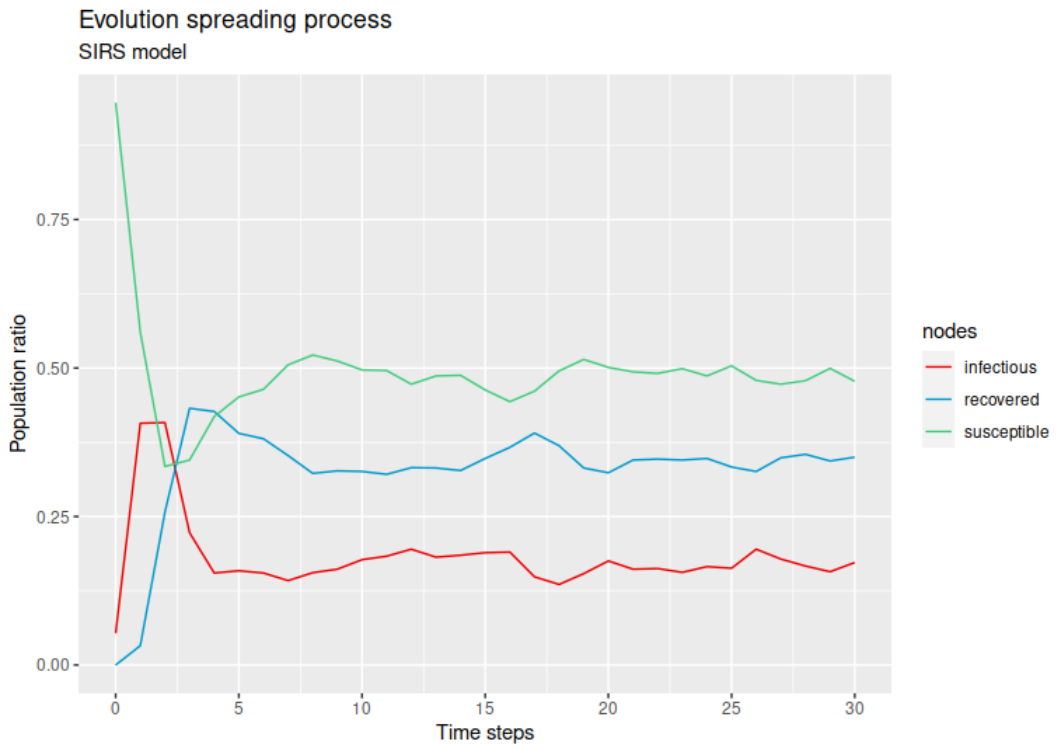

Figure 4: Evolution number of infectious, recovered and susceptible nodes in the whole network

The number of infectious is growing fast at the beginning while the susceptible one decreases. Around the second time step, the grow stops and the number of infected nodes decreases. The network begins to stabilise, and the amount of nodes is almost constant until the end for the three states.

We are interested in the infected ratio and how the communities are affected, so in figure 5 we plotted this statistic for the 7 largest communities.

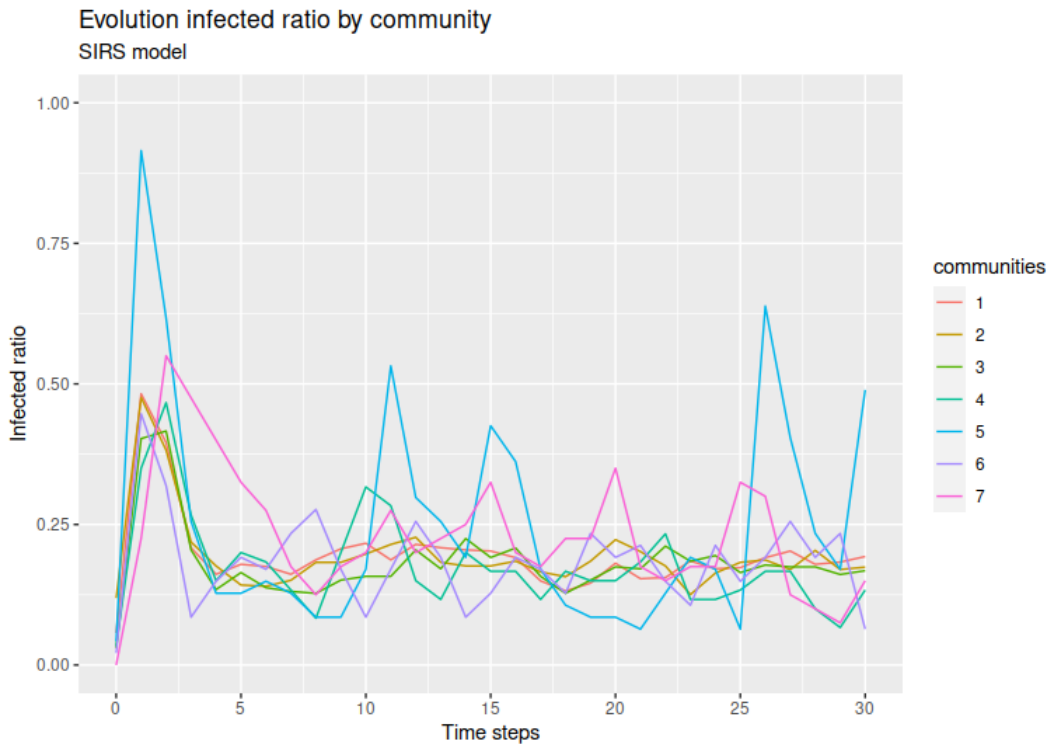

Figure 5: Evolution infected ratio by community

The behaviour is similar for almost all of them, there are some peaks and then the ratio is reduced. Mention how the community number 5 reaches a very high infected ratio with only 47 nodes. The point here is that the average out degree is really high, almost the same than community 3 with 298 nodes, as it can be seen in figure 2. This relation could also be applied for community 7, which is the second one with the highest peaks.

The last plot in figure 6 is the same that the previous image, figure 5, but overlaying a smoothing average (dotdashed line) with 95% level of confidence interval. This may be useful if we would like to know how a new community into this network could behave for a similar spreading process.

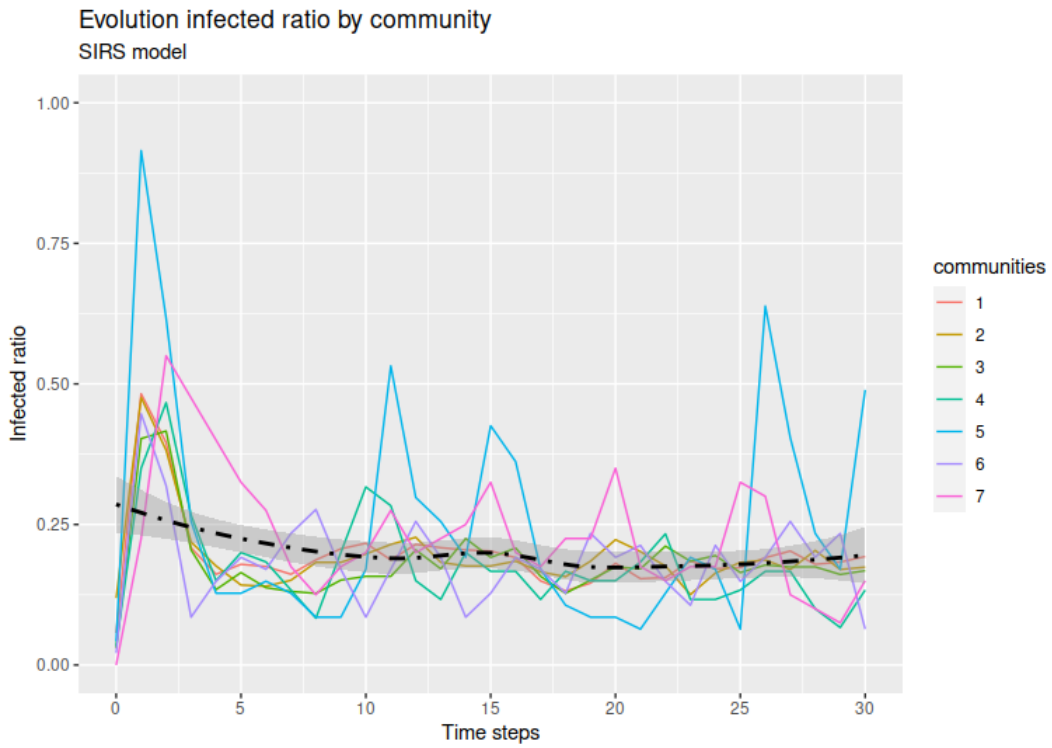

Figure 6: Evolution infected ratio by community with the smoothing average

The highest value is obtained at the beginning, when there are more infected nodes. Then it remains almost constant until the end, the same than we observed in figure 4 for the number of infectious in the whole network. For this reason, we can state that the 7 communities are well detected and provide an accurate representation of the network.