Social Network Analysis on Airbnb

Khashayar Abtin
k.abtin@studenti.unipi.it
Student ID: 545506

Monia Bennici
m.bennici4@studenti.unipi.it
Student ID: 534792

Kamran Mehravar
K.mehravar@studenti.unipi.it
Student ID: 628053

Gianmarco Pepi
g.pepi2@unipi.it
Student ID: 531425

ABSTRACT

In the following we will analyse the network created from Airbnb, capturing the main measures and trying to understand some aspect, applying some community discovery techniques and the simulation of diffusion model. In particular, we will implement, apart from the algorithms given in the libraries, a new algorithm based on the spectral clustering. Moreover, we would like to capture the so called "sockpuppet" phenomenon, that is all users that create more than one account to deceive the restrictions given in the platform. All the notebooks are stored in github repositories. [1]

KEYWORDS

Social Network Analysis, Sockpuppets, Airbnb

ACM Reference Format:
Khashayar Abtin, Monia Bennici, Kamran Mehravar, and Gianmarco Pepi. 2021. Social Network Analysis on Airbnb. In *Social Network Analysis '21.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Airbnb is one of the biggest platform of house sharing, in which people can rent rooms of their own house for short terms, that was born in order to let them make some money. Of course this is a problem for most of the hotels owners and this is the reason why some cities decided to restrict the maximum amount of time a host can rent his house, that is usually 60-90 days per year depending on the city. Our network is created using as nodes hosts and guests registered in the platform in the area of Amsterdam. In particular, hosts are those who rent the rooms, guests are users who pay to use the room and leave the reviews of their experience in that house and with that host. So, our net is a bipartite graph, in which every guest can be connected only to hosts and the edge represent the number of time the guest has left a review.
After collecting the useful data we compared our network with random networks and then we did 3 analytical tasks.
Our work is diveded as follows: in section 2 we explain how we crawled the data, in section 3 there are the main network characterization, in section 4 we apply community discovery algorithm, in section 5 we simulate different diffusion models on or network and in section 6 we implement and apply the spectral clustering and finally we tried to solve the open question which is discovery the sockpuppet accounts in section 7.

2 DATA COLLECTION

To extract data from Airbnb there is no free API. That's why we had to use more techniques to have them. To obtained an interesting network for the analysis we have to collect data in which there are some reviewers that are connected to different houses. The issue is that Airbnb provides to users only 300 listings per City at a time, and this was a problem because in that way we could obtain as much isolated components as the number of houses in the net. So we decided to downloaded a csv file with Amsterdam's reviews since 2009 from InsideAirbnb (http://insideairbnb.com) and then we filtered only reviewer data who left more than 7 reviews in Amsterdam houses. From this data we selected the listing_id of interesting houses, that are the ones reviewed by these users. This let us create edges also between different components. Then we decided to scrape house's data of related interesting listings from Airbnb website and finally we scraped the data of all the reviewers who left the review.

[1] **Project Repositories**
Data Collection: https://github.com/sna-unipi/data-collection
Analytical Tasks: https://github.com/sna-unipi/analytical-tasks
Report: https://github.com/sna-unipi/project-report

SNA '21, 2020/21, University of Pisa, Italy

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$0.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Selected Data Sources

From Data Scraping we obtained two dataframes: one for reviewers and one for listings.
Then we trasformed and cleaned data using Pandas to merge 2 dataframes and we obtained the following csv files to import as networks:

- weighted_net_amsterdam.csv
- temporal_binetwork_amsterdam.csv

In the first we have reviewer_id, listing_id and the weight that represent the number of reviews the reviewer left to the listing. The first two attributes represent the reviewer and the listing node respectively. We used this file for various tasks.
The second file contains reviewer_id and listing_id as before and a new attribute with the date in which the reviewer left his review. We used this file to create series of network snapshots to make some dynamic task.

Crawling Methodology and Assumptions. From the set of listing_id ,obtained with data manipulation of csv file downloaded from InsideAirbnb, we crawled information regarding a set of 156 listings.
We used Beautifulsoup library to extract the static HTML of Airbnb webpage but unfortunally detailed information of listings are loaded into webpage using Javascript so we had to use selenium and its webdriver for chrome because the JS elements take some seconds to loads their content. Through different get functions we extracted various information for any listing like link, title, facilities, rating and review number. Then we stored this information in a dataframe.
Starting from the links in the crawled dataframe we extracted reviewer informations using BeautifulSoup and Selenium to simulate human-like mouse movements, in particular we wrote a function to scroll the webpages containing reviews. As before we stored information obtained in a dataframe.

3 NETWORK CHARACTERIZATION

Once the dataset has been crawled, we could create the graph. To create and analyse the graph we used Networkx library. What we created is a weighted bipartite graph, where the nodes are guests and hosts, the edges are created if a reviewer leaved a review to a house and the weights used are the number of times a reviewer has left a review to a listing. The graph has 11266 nodes and 11265 edges, and the average degree is 1.998. The degree distibution has not a poisson distribution and this is another sign that it is a real network. Looking more specifically, most of the nodes have a degree equal to 1 but a small part has high degree (figure 1).

The most logical explanation to this is that the nodes with high degree represent the listings, instead the others are the reviewers. In fact, in the graph there are only 156 listings

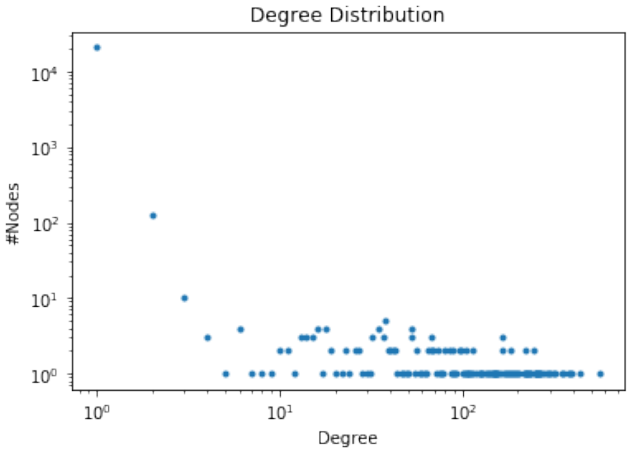


Figure 1: Degree Distribution

and 11110 reviewers.
Average clustering coefficient is 0.9706 and in figure 2 we show the Clustering Coefficient distribution on Giant Component where we can see that most of clustering coefficient are between [0.9,1.0] interval, as we could expect on real networks.

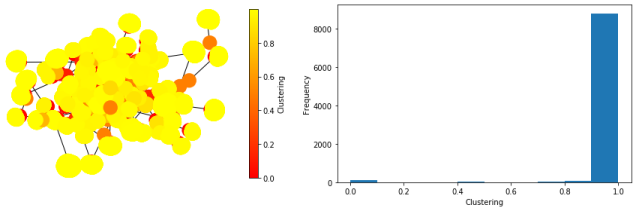


Figure 2: Clustering Coefficient on Giant Component

We have a sparse graph, with density equal to 0.0065 and the number of connected components is 9. We found a giant component having 9031 nodes. Due to its bipartition, the maximum length of cliques found is 2. The graph is non connected, so we computed the average shortest path length only on the giant component, founding it equal to 10.622. The closeness centrality of all nodes is 0.27 and the degree centrality is 1.99, same as average degree and it is in fact the number of neighbours a node has.
Since our graph is bipartite, we decided to compare it with a bipartite version of artificial graphs. We used *random graph, configuration model* and *preferential attachment graph* giving in input parameters such that we obtained an artificial graph with a structure similar to the real network, that is with same number of nodes and edges. In table 1 we have summarized features of real and artificial bipartite networks where the column name have the following meaning:

- AD : Average Degree

- ACC : Average Clustering Coefficient
- NCC : Number of connected components
- ASP : Average Shortest Path Length
- D : Density

Graph Type	AD	ACC	NCC	ASP	D
Real Network	1.9998	0.9706	9	10.62	0.0065
Random Graph	1.9831	0.3218	4013	4.74	0.0064
Pref. Att. Graph	2.0014	0.9648	82	6.26	0.0065
Config Model	1.9982	0.9708	80	5.18	0.0065

Table 1: Comparison with bipartite artificial networks

Then , we applied the models seen during the course to compare the projection of the graph with artificial networks and in table 2 we can see the results.

Graph Type	AD	ACC	NCC	D
Projected Graph	226.9941	0.9981	9	0.0204
Barabasi	0223.7014	0.0573	1	0.0201
Watss-Strogats	10	0.6572	1	0.0203
Erdos-Renyi	222.0592	0.02	1	0.0199

Table 2: Comparison of the projection with artificial networks

Comparision with bipartite artificial networks

To obtain bipartite artificial networks with a structure similar to the real network we had to set parameter as follows:

- **random_graph:** The sizes of the classes of nodes and the probability for edge creation estimated as $p = \frac{<k>}{N_1} + \frac{<k>}{N_2}$, where N_1 is the number of nodes in the smallest class and N_2 is the number of nodes of the largest class.
- **configuration_model:** Degree sequences of the two classes and, since the sum of the two sequences is not equal, we had to set create_using = Graph() to avoid to obtain a multigraph as output.
- **preferential_attachment_graph:** The degree sequences of classes with fewer nodes and the probability that a new bottom node is added, estimated as follows : $p_1 = 1 - \frac{N_1}{N_2}$. As before we had to set create_using = Graph().

In figure 3 we can see the degree distribution of real and artificial networks and it seems that real networks has similar distribution to the configuration_model and preferential_attachment graph, unlike the random_graph. This is confirmed by "two sample Kolmogorov–Smirnov" statistical test in which we can reject null hypothesis in random_graph because the KS statistic is higher than p-value meanwhile the other 2 artificial degree distributions have an higher p-value compared to KS statistic.
As we shown in table 1 the average clustering coefficient is small in a random graph and large in preferential attachment graph as expected.
Unlike what we expected in configuration model, clustering coefficient is as large as in real network: this is likely due to the fact that the two classes are unbalanced in the number of nodes.
Number of connected components in artificial networks is much more greater than in the real one while density is very close to the real network density.
Unlike the projected case, we were able to compute the average shortest path length. The artificial networks average distances are all around 5 while the real one is 10.62.

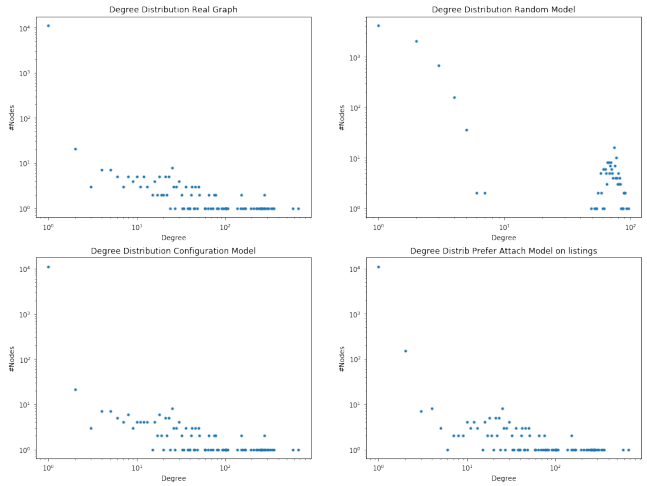


Figure 3: Comparison of degree distriubtions

Comparison of the projection with artificial networks

We created a projection of the graph on the set of nodes representing the reviewers. Also in this case, since the projection is not a connected graph, we extracted the giant component to measure the average shortest path length. Unfortunatly, the net has too much edges and the computation is too time consuming. We compared the projection with the following artificial net:

- Erdos-Renyi: Given an Erdos-Renyi graph of the form
$$G(N, p)$$
we expected to have certain values: for example, we know that the degree distribution follows the Poisson distribution, the clustering coefficient is given by $\frac{<k>}{N-1}$,

the connected components depend on the regime and the path length is usually small. Our random model, built taking into account the values of the projected graph (on the reviewers), was given as N the number of nodes and as p

$$p = \frac{< k >}{N - 1}$$

. Looking at the results, our graph follows these rules and in fact we have $p > \frac{1}{N}$ and so just one giant component,the average clustering coefficient is 0.02 and the degree distribution is Poisson as we can see in figure 4 (however, to have more omogeneous results in figure 5 is represented the log version of the graph).

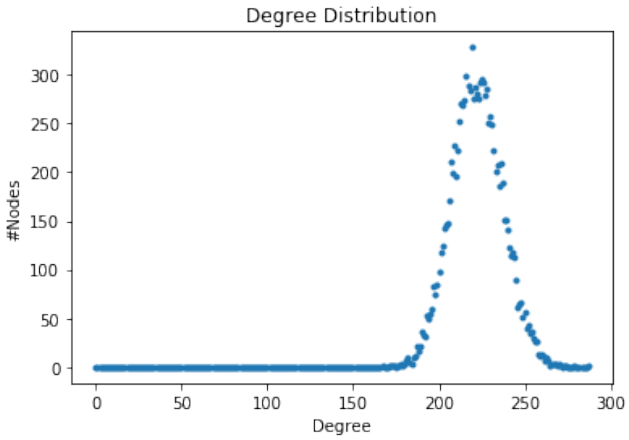


Figure 4: degree distriubtions of the Erdos-Renyi artificial model

- We also built a Barabasi-Albert artificial network, giving in input the number of nodes of the projected graph and a probability equal to $\frac{numberofedges}{numberofnodes}$ that a new node is added to the net, in a way that we obtained an artificial net that has same number of nodes and edges of the original one. What we expected is what we had, and infact the average clustering coefficient is very small and the degree distribution is not similar to a Poisson curve.
- For the Watts-Strogratz model we gave $k = 227$ (in the model k is the number of nearest neighbours and we estimated it taking into account the average degree) and as probability of rewiring p we used the same as in the Erdos-Renyi model, so $p = 0.02$. In this case, despite the clustering coefficient is quite high and the number of connected component is 1, the degree distribution is not described by the Poisson curve.
- Finally, we built also the configuration model giving as input the sequence of degrees of the projected network. Below the comparison of the degree distribution.

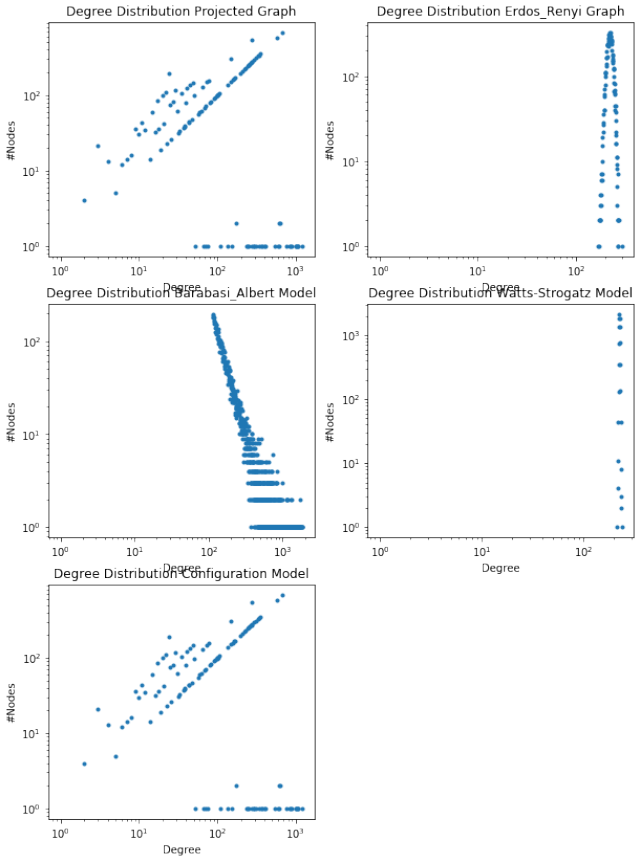


Figure 5: Comparison of degree distriubtions with projection of the graph

4 TASK 1: COMMUNITY DISCOVERY

In order to apply community discovery algorithms we projected the bipartite graph using two set of nodes: the first one containing the reviewers, the second containing the houses. To make the projection we used a weighted function, keeping the original weight function.
In order to understand how the community discovery algorithms work, we used different approaches, applying the algorithm both on bipartite graph and in more than one projection to compare them.

CD on the bipartite graph

First thought was to apply the CD algorithms to the bipartite graph, but this was not a good idea, since the assortativity of the communities is for all cases under -0.05, meaning that the nodes belonging to a community are different one from

the other. However the modularity is 0.97 indicating that the communities are internally densely connected w.r.t. the other communities. A good explanation to this behaviour could be the weight of edges in communities. The algorithm does not seem to capture the bipartite nature of the graph and infact, using the bimlpa algorithms for bipartite network the modularity is negative and this can better explain the negative assortativity.

CD on the projection of the graph on reviewers

We tried to apply 3 different algorithms to the projected graph on reviewers.

- Louvain: this algorithm creates 88 communities, with high difference in size (fig 6). The computed Newman-

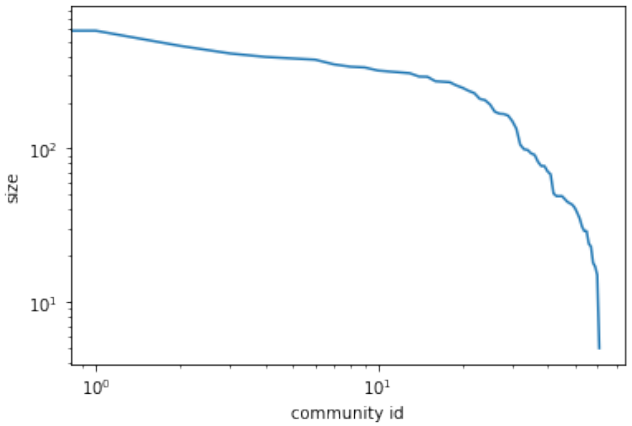


Figure 6: Distribution of size of communities

  Girvan modularity is 0.92 but internal edge density is 0.2 and degree centrality fraction is always under 0.05. It is interesting to look at the centrality of single nodes, noticing that just some of them have high centrality degree (so they are connected to a lot of nodes) and all the other have value 1. However, computing the degree distribution of communities, the graphs show an increasing function, representing a good assortativity (we do not report the plots, but they are available in the notebook). The visualization does not help to distinguish every community

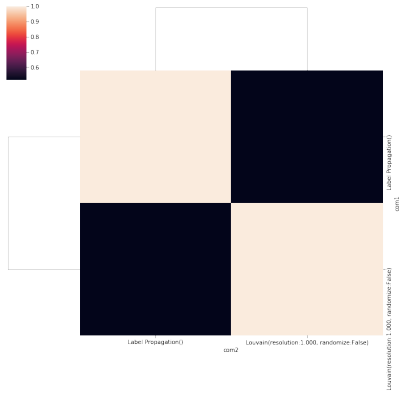
- Label propagation: 26 communities founded. Plotting the distribution of size, the minimum size is 3, the maximum is 1246. Internal density is 0.14 and the modularity is 0.93. For this algorithm, of course, we could have reached better results creating a graph with more layers in which we included also some features.
- K-cliques: after finding the cliques, we computed the average size of them, which is 73. This parameter was used in the algorithm and we found quite good results: modularity is 0.94.

After using some internal evaluation, we also compared the different methods to see how similar they are using external evaluation : by a comparison of Louvain and label propagation we can see how the two partition are not very dissimilar, since the variation of information score is 1.5 (considering that a score=0 indicates the complete overlap), and the overlapping normalized mutual information is 0.51. The confusion matrix is available in figure 7a (max value 1, min value 0.5). Comparing Label propagation and K-cliques we found results that make us think that the communities found are very different: the overlapping NMI is 0.28 .
Different is the situation comparing Louvain and K-clique: the overlapping is 0.65 so this make us think most of the communities have the same nodes. The matrix representing the overlapping NMI is represented in figure 7b (maximum value is 1, minumum value is 0.68)

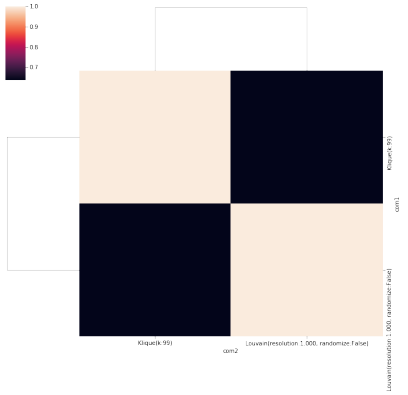
CD on the projection of the graph on hosts

Making a projection considering only the nodes corresponding to hosts, we obtain a graph with 156 nodes and 701 edges. There are 9 connected components but one of them is a giant component containing 117 nodes and 596 edges. Since a lot of nodes are left outside the Giant component we focused our analysis on the complete graph.

- Also in this case we applied Louvain algorithms, finding 19 clusters. Computing their assortativity, we noticed that only 2 clusters have a positive value, while all the other have value nan. So, we looked at their degree, discovering that, apart from the two clusters, all the nodes inside the clusters have equal degree.
  In fig 8 we plotted the found communities.
- Label Propagation gives us similar results: the number of communities founded is 17 and in this case only 4 of them have positive assortativity, all the other nan. The modularity in this case is 0.9 and the average size is 8.6
- This time we tried to apply Angel algorithm, giving as threshold 0.30 because was the one returning the best modularity: 0.92. The function return a frozen graph so we could not visualize it.
  Comparing these methods of clustering we noticed how all of them agreed: the overlapping measure is 0.83 in the case of Lovain-label propagation and 0.94 in the case Louvain-angel.



(a) NMI matrix for LP and louvain



(b) NMI matrix for k-cliques and louvain

Figure 7: Overlapping NMI matrix

5 TASK 2: SPREADING

As we have already said our bipartite network represents the reviewers and a listings, but on Airbnb platform you can left a reviews on a listing only if you stayed there. So the projection on reviewers class is a network in which the nodes represent of course the guests and an edge between two node means that they stayed in the same house so they had a contact with the same host.

It seemed interesting to see how a possible epidemics could spread in a network of Airbnb users. We applied the main epidemic models both on the projection of the real network on the reviewers and in the synthetic one, created using Erdos-Renyi model.

Unlike our network information we suppose that the interactions have been made in a short period of time so that if two guests have been in the same house, they can be infected one from the other.

So we simulated diffusion model like SI,SIS,SIR and Threshold model setting parameters as follows:

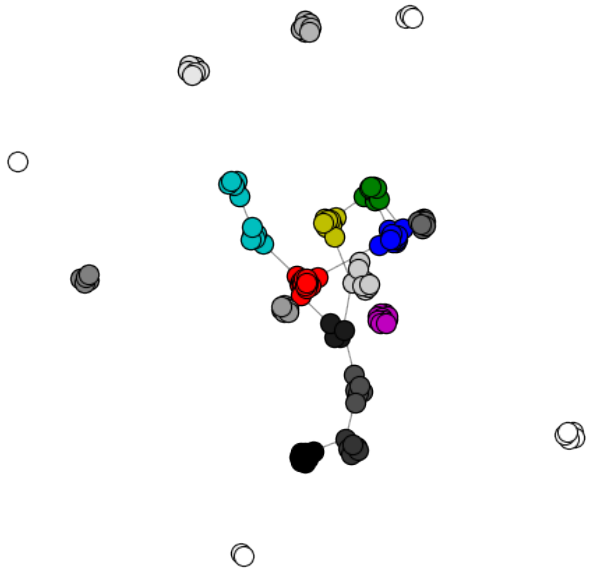


Figure 8: Visualization of Louvain communities

- **SI:** We have set the percentage of infected to 0.01 and beta to 0.0005. We chose such a low value for *beta* because in our projected network the edge between two hosts represent an indirect contact.
- **SIS:** As before we set the value of *beta* to 0.0005. We chose a very small value for *lambda* equal 0.00001 to obtain a recovery effect really little influent on epidemics diffusion.
- **SIR:** We set beta to the same value as the two previous models meanwhile for recovery rate *gamma* we chose 0.03 to highlight the recovery power to stop the epidemic.
- **Threshold model:** In this model we set the *threshold* to 0.01 and unlike previous model we had to execute a bunch of 20 model iterations for simulation.

For all of them, the percentage of infected is 0.01. As we can see, w.r.t the default value of infection rate (0.001) the spreading is much slower in our case (it arrive after 25 iteration with default values and after 50 in our model) for SIS and SI. For SIR model, since the recovery rate is high there is not a breakout of epidemics. We simulated with only 100 iterations the first three models because too expensive computationally. Then we plotted diffusion trend and diffusion prevalence for each model to try to compare them. In figure 9 we compared the infected trends and as we expected SIS and SI are very similar because the set recovery rate is not at all influential while in SIR model we chose an higher recovery rate and so the infected nodes are less and at some point decrease because recovered patients can no longer become

susceptible.
In threshold model the infected nodes grow much faster with set parameters and in fact we iterated it just 20 times, so it's not possible to compare it with the other models.

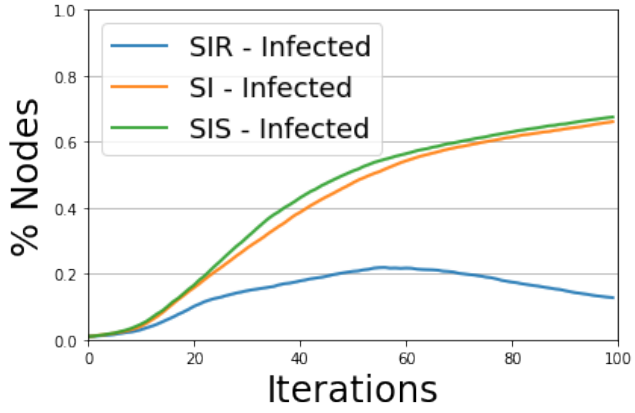


Figure 9: Comparison between probabilistic diffusion models

Finally we chose ER graph to compare the spreading with our real network. So we set the probability for edge creation as before estimating as follows $p = \frac{<k>}{N}$ in order to obtain the same structure of our real network.
Comparing synthetic graph and real one the main difference is that in random graph the epidemic spread is faster than real as expected because we built it with an higher average degree so it results very connected.

6 TASK 3: ALGORITHM

From scientific paper [1] we implemented a Spectral Clustering algorithm for Communities Discovery. It is based on Fielder's vector which is obtained from the eigenvector related to the second eigenvalue of the normalized *Laplaciano* and then, through fielder's vector information, it extracts the communities using Kmeans clustering.
Normalized Laplaciano is defined as follows:

$$L = D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}$$

Where **W** is the weight matrix that the algorithm calculate as the adjacency matrix in the case of standard network as input. While, when the input is a bipartite network, W is obtained by matrices multiplication between adjacency matrix and his transposed. **D** is the diagonal matrix where the element $d_i = \sum_{j=1}^{n} w_{i,j}$.
Fielder's vector f is obtained from the eigenvector related to the second eigenvalue of L v_2 as follows $f = D^{-\frac{1}{2}}v_2$.
To each node of original network corresponds a value of f, so the algorithm, unlike paper [1], sort the value in f and then it applies KMeans by choosing k with best silhouette score. We decided to add these changes to make communities discovery independent from the order of input nodes, in fact in paper [1] nodes are given in the order of belonging to the real communities so if we apply kmeans it would be simplified.
We tested the algorithm on the dataset used in paper [1] obtaining the same results and then we applied it on our network. The algorithm is implemented on *spectralCD.py* in which the function *communities* takes in input as arguments the following parameters:

- The graph *g*
- *kmax* the maximum k to evaluate the best KMeans
- *projection_on_smaller_class* a boolean value that if True then it project a bipartite network in the smallest class of node. (default is True)
- *scaler* the function to scale the fielder's vector to apply KMeans

Setting scalar function to StandardScaler we obtained 4 unbalanced communities of sizes 71, 70, 6 and 9 nodes. In figure 10 we shown how the fielder's vector separates houses in our network. In the first scatter plot seems that two classes are not well separated but zoomming scatter plot it is possible to see that those classes are quite well separated.

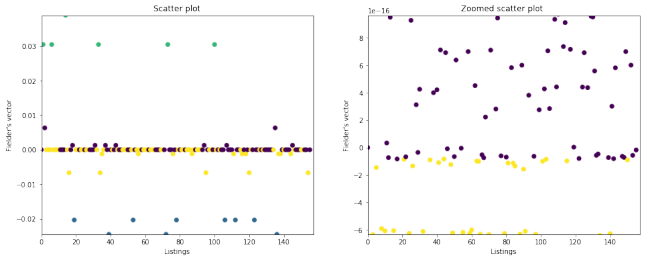


Figure 10: How Fielder's vector separates the nodes

Then we calculated some centrality measures to search the most important nodes. We extracted the three nodes with higher centrality values from each communities calculating Degree Centralities, Connectivity-based Centralities, (Eigenvector centrality and Pagerank) Geometric Centralities (closeness, harmonic and betweennes centrality). In many cases the most important nodes coincide in different measures, even in the case of degree centrality, in fact in our real network there is no issue as in other real network like twitter and wikipedia in which nodes with high degree centrality can correspond to spam or reference node respectively in twitter and wikipedia.
Finally we calculated Homophily on each communities using global approaches. So first we calculated degree assortativity coefficient for each communities and, except the third community in which every nodes has degree equal to 4, other

communities have an high assortative degree so in our communities hubs is connected with hubs. In addition we shown in fig 11 the degree correlation function of the two largest communities in which is clear the assortative behavior on communities. In fact in the log scale the K_{nn} (k) points ,that represent the average degree of neighbors of all degree-k nodes, fitted very well the bisector line.

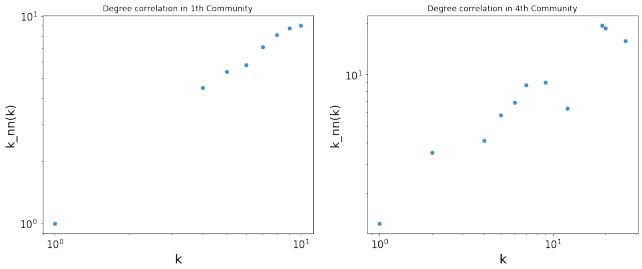


Figure 11: How Fielder's vector separates the nodes

Then we added to nodes of our graph a semantic property to calculate attribute_assortativity coefficient on rating attributes approximating the float rating to the nearest integer and obtaining a lower assortativity measure in each group. We observed that this community discovery algorithm did not separate into communities in which the nodes forms connected components so we tried to force the choice of k on values above the number of components but it did not solve the problem of connection. We concluded that this kind of approach is good for those bipartite networks in which each node of a class is connected to each node of the other groups and vice versa as in real network representing patients and genes and the edges between them represent the genetic activity as in paper [1].

7 OPEN QUESTION

After having an idea of what our graph looks like and what are the main characteristics, we wonder if it can be useful to find something interesting. Airbnb has always seemed a good tool to make reservations, because usually the account are verified so if you want to sign up you have to send a document. So our question was: is there any chance that we can find someone cheating? In particular, is it possible for hosts to break the rules and exploit this service to make more money? We acted in the projected graph containing only houses, so the net we used for this task is composed by 156 nodes. Our aim is to find two listings belonging to the same person and the same house. We tried to do it in different ways, exploiting the structure of the graph and the information we already knew.

Finding suckpuppets through bridges in communities

Our first though was that if a host made two listings for the same house, this means that in the net there are two nodes having different id but similar properties. This means that if we detect communities in the net, it is likely to find both of them in the same community. Applying the Louvain algorithm, we found 20 communities. So we though could be interesting to see if there is any bridge inside these communities. We found 3 bridges (for a total of 8 nodes) and this let us be able to check if there was something interesting with them: surprisingly we found that one bridge corresponded to two listing belonging to the same house. It is possible to check directly on the link of the house of Airbnb to see that both listings have same photos, same title and same host. Unfortunately the other bridges we found did not give us any information.

Finding suckpuppets through shortest path

Following the though that the more similar the nodes, the more close they have to be, we started searching for suckpuppet considering the shortest path between nodes considered "interesting", that are those forming the bridges inside the communities.We can say in advance that the found methods are not able to find directly the suckpuppets, but they let us discard the majority of the nodes and look for something interesting in the remaining ones. This time we tried applying the Label propagation algorithm on the projected graph on houses and we obtained some community. Again, we computed the bridges finding two of them (that were 2 of the 3 found before). One of the two bridges is composed by 3 nodes. So, we looked for the nodes inside the graph that have length of shortest path equal to 1 from these nodes, that, following the initial though are the one more similar. Actually, we could not find a node similar to the one we were looking for, but between the nodes having distance = 1 from it (that were only 8) two of them belonged to the same person. Of course this method is very slow and it can be difficult to apply in huge nets.

Finding suckpuppets through assortativiy measure

We used another approach to discovery sockpuppet accounts based on assortativity measure. On projected graph we added a node property as in algorithm section 6. So we extracted the rating from crawled data on listings and added to our graph its integer approximation at nearest integer.
First we calculated local assortativity of each node through Newman's assortativity [2] with Multiscale Mixing patterns library. As you can see in fig 12 in our network the range problem arises, in fact we have many nodes with assortativy less than -1.

Analyzing the two most disassortative nodes we found that they are two listings made from two different listing_id that correspond to the same house that correspond to the same sockpuppet accounts found with previous approaches.
To solve the range problem we calculated the conformity for each nodes with alpha equal to 0.5 but we couldn't find the same sockpuppet accounts because we had to calculate this measure only for Giant component but unfortunately the two sockpuppets form a single component disconnected from the rest.

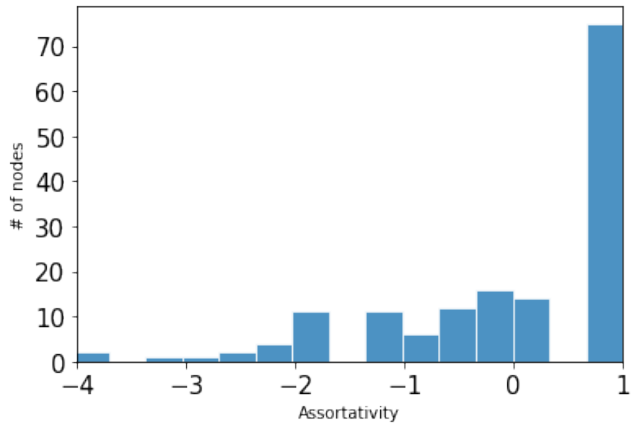


Figure 12

8 DISCUSSION

We tried to explain our net and to find a solution to the question we wondered. Of course we can make more progress on this work trying to apply the solution founded also in other networks (maybe other cities) to see if our solution are valid.
Also, a part from an analysis on houses a future work could be based on the reviewers to see if we can find cheaters between them.

REFERENCES

[1] Milla Kibble Desmond J.Higham, Gabriela Kalnaa. 2004. Spectral clustering and its use in bioinformatics. *J. Comput. Appl. Math.* 204, 1 (2004), 41–49.

[2] M. E. Newman. 2003. "Mixing patterns in networks,". *Physical Review E* (2003).