Complex Networks - Project Report

Analysis of LastFM Asia social network

Teddy ALEXANDRE

June 2023

Table of contents

1 Introduction **2**

2 Network metrics **2**

2.1 Number of nodes and edges 2

2.2 Connectivity 2

2.3 Average degree 2

2.4 Centrality 2

2.5 Clustering coefficient 3

2.6 Longest shortest path and graph diameter 3

2.7 Degree distribution and scale-free property 3

2.8 Small world property 4

3 Community detection **5**

3.1 Optimization-based approach : the Louvain algorithm 5

3.2 Inference-based approach : the degree-corrected stochastic block model 6

4 Epidemics simulation - the SIR model **8**

5 Conclusion and discussion **10**

1 Introduction

The information about the graph chosen for the project can be found on this webpage : `http://snap.stanford.edu/data/feather-lastfm-social.html`. More specifically, the graph analyzed here is a **social network of LastFM users** which was collected from the public API in March 2020.

- Nodes are LastFM users from Asian countries
- Edges are mutual follower relationships between them.

The vertex features are extracted based on the artists liked by the users. On this graph, we can perform multinomial node classification, community detection, network visualization and link prediction. Some statistics are provided on the webpage, some of which will helped me comparing with the results obtained in the notebook. The data I worked on is contained in the file "lastfm_asia_edges.csv", that can be found attached to the report and the code.

2 Network metrics

In this section I will provide an analysis of the graph's structure, looking at some characteristics such as the graph's "size" (number of nodes and edges, graph diameter etc...), its degree distribution, the centrality measure of every node, or other properties (scale-free, small-world...)

2.1 Number of nodes and edges

The graph contains 7624 nodes and 27806 edges, which is quite consequent and makes us consider this graph as big. But both measures are not sufficient to characterize the graph, so we consider other metrics. Moreover, the graph is not dense ($9.5 * 10^{-4}$).

2.2 Connectivity

The graph has one connected component, the whole graph itself. Thus, the graph is **connected**, meaning that every pair of nodes (LastFM users) can be connected to each other by a finite amount of transitions.

2.3 Average degree

This measure provides information about the amount of links that a node in the graph is expected to have. Computing it, the average degree of the graph is about **7.29** : picking a node randomly, we expect it to have around 7 connections. Of course, some nodes will be more "central" than others, and some will be more "marginal".

2.4 Centrality

This tackles the observation made above. In a graph, there are many ways to evaluate the centrality of each node. In this project, I considered the **closeness** centrality, to unveil the nodes that are "close" to others. The closeness of a node

is the reciprocal of the sum of the sum of the length of the sortest path from this node to the others. When computing the code, I found that the average closeness of a node in the graph is about 0.2, with values between 0.1 and 0.3.

2.5 Clustering coefficient

The clustering coefficient of a node measures how much its friends are linked each other. The average clustering coefficient of a node in the graph is **0.22**.

2.6 Longest shortest path and graph diameter

In order to complete the analysis of how big the graph is, I computed for each node, the longest shortest path (to every other node) and the graph diameter, which is the maximum of those quantities. Of course, the diameter is larger than the average longest shortest path (mean of every node's LSP), but it is only a **few units above** (diameter of 15 against an average at about 10.45).

2.7 Degree distribution and scale-free property

Plotting the degree distribution with a histogram, I was able to unveil a decreasing sequence. I then asked myself if the degree distribution could not be modeled by a power law, which would make the graph have the scale-free property.

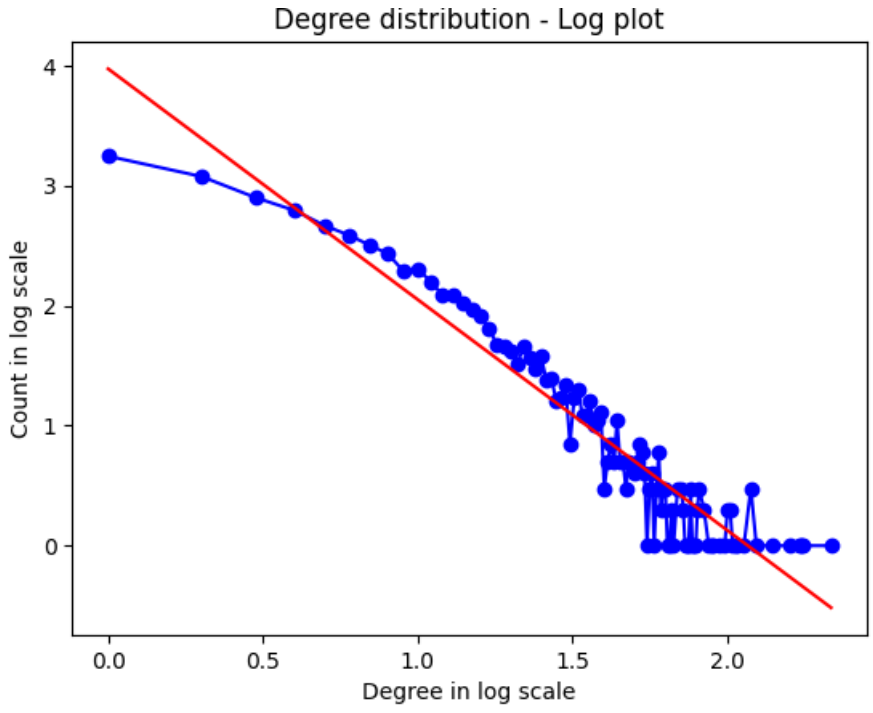

Figure 1: Power law guess : linear regression

When plotting the logarithm of the degrees, I saw that we were not far from having a linear trend on the distribution, and computing with linear regression

tools, I derived a slope that was close from -2 : $P(k) \approx k^{-2}$. We can consider that the graph is not far from a **scale-free** model.

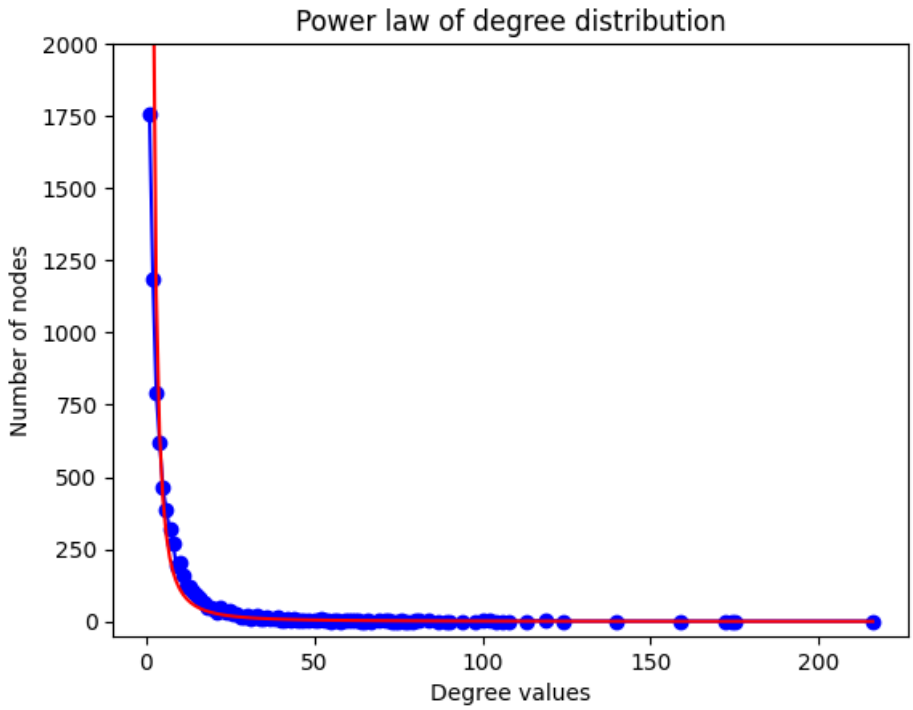

Figure 2: Power law guess : fit to the degree distribution

2.8 Small world property

The small world property refers to the ability to transition to every pair of nodes that are not neighbors in a small number of steps. To get this property, the average path length must be dominated by the logarithm of the number of nodes. When computing both measures, we see that the average path length (about 5.3) grows no bigger than the logarithm of the number of nodes (about 12.9). Hence, the graph can be considered as **small world**.

3 Community detection

In this section, we want to unveil structures of communities inside the graph. I used two approaches to get communities : one is optimization-based, the Louvain algorithm, and the other one is inference-based, with the degree-corrected stochastic block model. They both required import of extern packages of Python (*community* and *graph-tool*).

3.1 Optimization-based approach : the Louvain algorithm

The Louvain algorithm is an algorithm that can be used to reveal the community structure of a network. It is **powerful** and **quite fast** in terms of execution (in $O(NlogN)$, with N the number of nodes), since it is based on optimization. Indeed, the quantity that we desire to maximize in this algorithm is the **modularity**. The higher the modularity is, the better the communities will reveal at the end of the algorithm. It is a greedy algorithm : the best local maximum will be considered, which means that we are not sure that the graph obtained at the end is the one that reveals the best the community structures.

The communities obtained change after each iteration : they may even have nothing in common, but the modularity values tends to be the same every time. Hence, there is no general solution but only approximative solutions, since we are in an optimization problem.

Summing things up, there is here a tradeoff between quick execution and highlighting conveniently the different communities.

With the graph of LastFM users, I obtained about 25 and 30 communities, with a modularity about 0.81, which is quite close to 1. Considering the size of the graph, I think it is a rather good amount of communities, even though the number of communities can still be increased, even more if we want smaller communities.

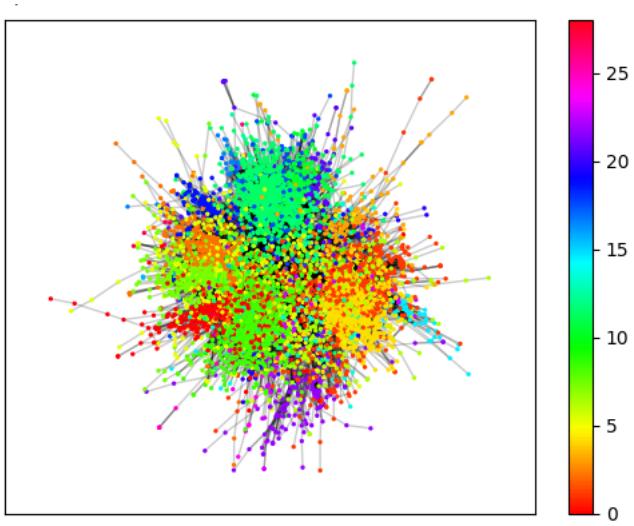

Figure 3: Communities with Louvain algorithm

3.2 Inference-based approach : the degree-corrected stochastic block model

The former approach was optimization based, I took a different here : the Bayesian inference approach. Here, I employed the **degree-corrected stochastic block model** to compute communities on an inference-based approach.

In the code, we find that the degree-corrected stochastic block model finds a **larger number of communities** than the Louvain optimization-based approach. However, the computations take much more time than the optimization-based algorithm, and finds communities at different levels of depth. The algorithm considers initially a node as its single community, then enlarges to higher levels regrouping nodes that are "close" to each other.

The latter algorithm is practical because it reveals a **hierarchical structure** of the graph. Indeed, we see several groups englobing other groups, with the minimized nested block model.

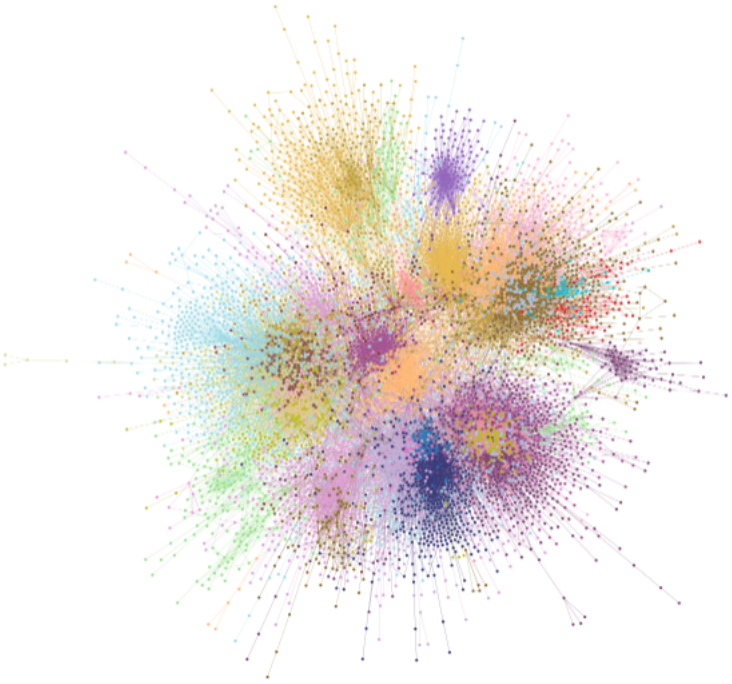

Figure 4: Communities with DCSBM algorithm

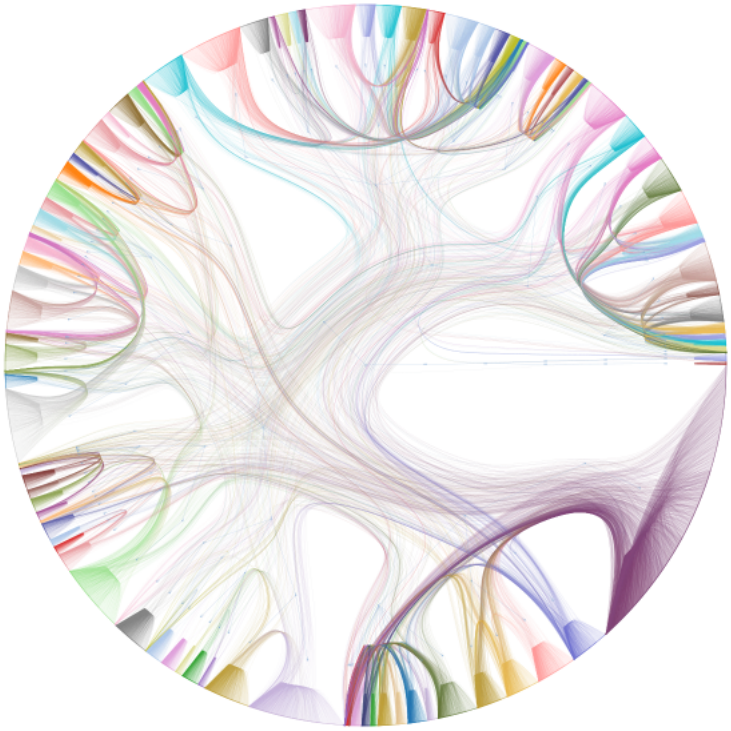

Figure 5: Hierarchical structure with DCSBM algorithm

4 Epidemics simulation - the SIR model

Here, I simulated the **SIR model** (susceptible, infected, removed) on the graph. Initially, most nodes are susceptible, the other fraction is infected and no one is removed. The propagation occurs through iterations, where infected nodes will infect their neighbors with a certain rate β. Then, after some time, the infected people will get completely removed from the epidemic and not get infected again, with a rate γ.

I ran a few simulations on the graph, with differences between the initial setting of infected nodes. The results show almost the same tendancy, with a phase of heavy infection followed by a long sequence of recovery. The epidemic threshold is obtained with $\beta = \gamma = 1$.

Moreover, the observations tell that initializing infected nodes from the same community makes the propagation slightly faster than initializing 1 node infected in its community. A random node is much probable to get infected by nodes from its community than from another, since he shares more links with nodes from the same community.

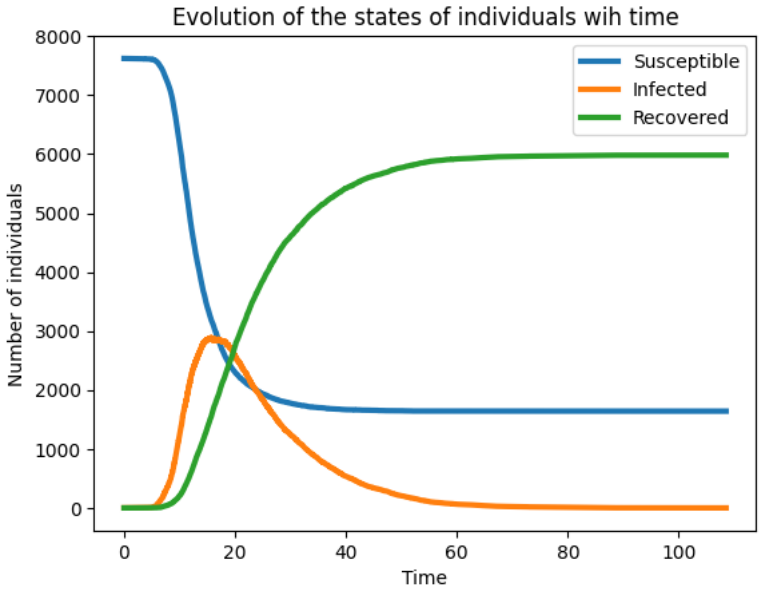

Figure 6: First plot of the evolution of epidemic through the graph

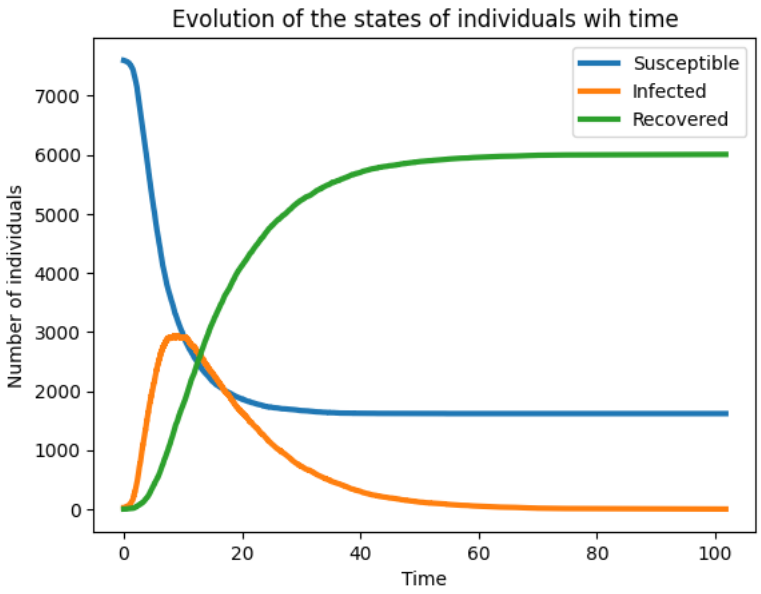

Figure 7: Second plot of the evolution of epidemic through the graph

5 Conclusion and discussion

All in all, the graph can be considered as **complex** : it has a structure that is not evident to describe and visualize (high number of nodes, large communities), so the interactions are not easy to highlight. However, some characteristics of the graph have been evidenced (small-world, scale-free) which are coherent with real world network observations, in comparison. The community detection algorithms are efficient and reveal a hierarchical structure that can be used to measure interactions between groups, and split the graph in manifold components (the graph being connected). The epidemics simulations tend to show more or less the same trend, with a first phase of heavy infections followed by a decay in term of infections.

Characteristic Functions on Graphs: Birds of a Feather, from Statistical Descriptors to Parametric Models, Benedek Rozemberczki and Rik Sarkar, 2020, pages 1325–1334, Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), ACM