U.PORTO

FEP FACULDADE DE ECONOMIA
UNIVERSIDADE DO PORTO

INFORMATION DIFFUSION IN SOCIAL NETWORKS

Patrícia Coelho Carvalho

Dissertation
Master in Modelling, Data Analytics and Decision Support System

Supervised by
João Manuel Portela da Gama

2019

Bibliographic Note

Patrícia Coelho Carvalho is a Portuguese citizen, born on 4th of august of 1986 in Santa Maria da Feira. In 2004, she joined School of Economics and Management, University of Porto (FEP) and graduated in Economics in 2009. Also in 2014, she enrolled in the Master in Modelling, Data Analytics and Decision Support System at FEP.

Started her professional activity at Deloitte in the accounting area and it was in this that developed her professional career. Currently is part of the financial department of Grupo Desfo, responsible for the accounting department of the logistics companies Forcargo and Transnautica.

Married and has two children.

Acknowledgements

First of all, I would like to thank Prof. João Gama who encouraged me, never let me give up and gave me all the technical support and, not only, that I needed.

Thanks also to my company and co-workers, which offered me all the conditions and helped to develop the thesis.

A special thank you to my family, my mother for the time she spent with my children and all the encouragement, my children for the time I stole from them for not being together and my husband for all the patience, support, encouragement, that never let me give up when even he believed more than me that it was possible.

Resumo

Com a crescente utilização das redes sociais "online", estas tornam-se uma forma para a difusão de informação "boca-a-boca", ou marketing viral. Neste trabalho vamos a analisar como maximizar esse efeito de difusão e como selecionar nós influentes na rede de forma a tornar a difusão mais rápida e eficaz.

Palavras-chave: Redes Sociais, Informação, Difusão, Cascata, Maximização, Nós influentes

Abstract

With the increasing use of online social networks, they become a way of spreading word-of-mouth information, or viral marketing. In this paper we will look at how to maximize this diffusion effect and how to select influential nodes in the network to make diffusion faster and more effective.

Keywords: Social Networks, Information, Diffusion, Cascade, Maximizing, Influential nodes

Index

1 Introduction

Social networks are characterized by the set of relationships between various individuals. We can simply put a social network on a diagram, where the nodes are the individuals and the relationships between them are represented by lines. Analysis of these diagrams allows you to identify groups, subgroups, strong and weak links or patterns.

Nowadays, many of these social networks go through online networks such as Facebook®, Twitter®, Instagram®, LinkedIn®, etc.

The rapid growth of these social networks has become important for marketing. Advertising on social networks is a fast and cost-effective method of reaching the target audience. Therefore consumers are influenced by their peers. By being aggregated according to your interests it is possible to identify the markets you want to reach. The main goal is to get a particular message to reach the fastest number of users as quickly as possible.

Despite being a cheap way of propagation, it could be difficult to control its consequences. (Birke, 2013)

2 Social Networks

A network is defined by a set of nodes that are linked together. A social network links individual with common tastes, preferences or characteristics. Online social networks have changed the whole landscape of relationships between people. It is not necessary to be in the same physical space to communicate and influence. Networks such as Facebook®, Twitter® or Instagram® are powerful tools of influence. A public node that went to see a movie, conveys that it likes and all the nodes that are connected to it can end up seeing the same movie (or not), influenced by the opinion of the first one. In turn they will influence other nodes.(Zafarani, Abbasi, & Liu, 2014)

This propagation becomes more or less rapid according to the number of nodes connected to each other and the strength they have in the network.

A social network can be homogeneous, where all individuals are all of the same type, or heterogeneous where nodes are of different types.(Chen, Lakshmanan, & Castillo, 2013)

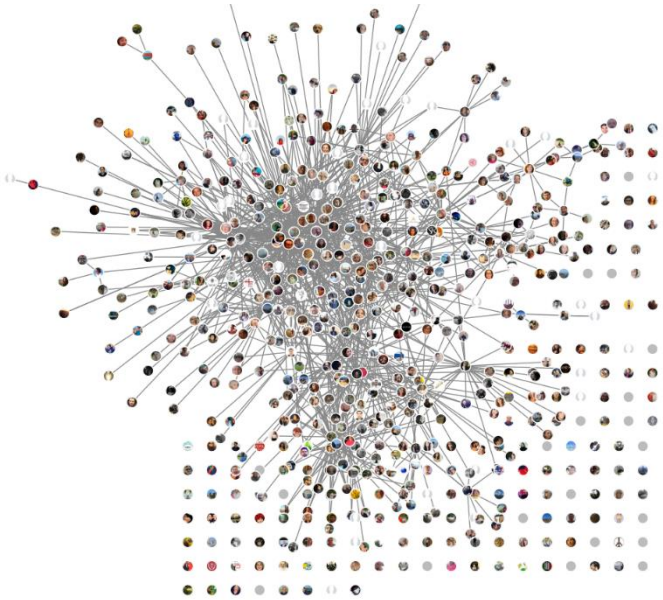


Figure 1 – Facebook network

2.1 The network value

A network is characterized by a graph (G) in which the vertices (or nodes), (V) are connected by the edges (E).

The number of nodes is n = | V (G) | and the number of edges is m = | E (G) |(Diestel, 2000).

The social network can be undirected or directed, this one happens when one node is followed by several without following the others.

The maximum number of edges in each type of network is given for m_{max}:

$m_{max} = \frac{n\,(n-1)}{2}$ –> in undirected networks

$m_{max} = n\,(n-1)$ –> in directed networks

2.2 Measures

The importance of network actors (nodes) can be calculated by their centrality in the network, identifying the main agents.

Degree

Degree measures the involvement of a node in the network and can be given by the number of incident edges in each node. If k_v is the number of edges of given node and could calculate in this way:

$$k_v = \sum_{j=1}^{n} a_{ij}\,, 0 < k_i < n$$

This measure is quite simple to calculate but doesn't consider the global network because calculate is local, for each node (Gama, Carvalho, Facelli, Márcia, & Lorena, 2012).

Betweenness

The extent to which a node is between the other nodes of the network, as measured by the average of the shortest path length between one node and the other nodes of the network. The nodes with higher b_v values are nodes with critical roles and could be calculate with this equation:

$$b_v = \sum_{s,t \in V(G) \setminus v} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

Where σ_{st} represent the number of shorter paths between edge s e t and $\sigma_{st}(v)$ the number of shortest paths between s and t passing through node v

Closeness

The Closeness (CL_V) measure the global position of a given actor (node) on the network, based on the number of times a node is crossed by each shortest path, the smallest number of routes between nodes. It is a point of rapid diffusion and is calculate with this equation:

$$CL_V = \frac{n-1}{\sum_{u \in V(G) \setminus v} d(u,v)}$$

Where $d(u,v)$ represent the length of shortest path between node u and v.

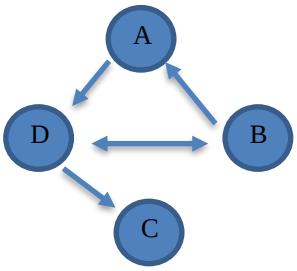
Eigenvector

This measure calculates the centrality of a given social actor, by the distance between a given initial node and a more distant node (Phillip Bonacich, 2012). This measure is a more elaborate version of the degree measure, since it assumes that not all links on a given node are equally important, taking into account not only the quantity but the quality of the connections. (Gama et al., 2012).

$$x_i = \frac{1}{\lambda} \sum_{j=1}^{n} a_{ij} x_j$$

Where x_i/x_j is the degree of node $i/\ j$, a_{ij} is equal 1 if node i and j are connect for one edge and equal to 0, if they didn't connected, λ is the highest eigenvalue of matrix A. Matrix A is how we can represent all path between in one node and another. If node A are connected with be assume value 1, if not assume value 0 (Gama et al., 2012):

Nodes	A	B	C	D
A	-	0	0	1
B	1	-	0	1
C	0	0	-	0
D	0	1	1	-



Node Source	Node Target
A	D
B	A
B	D
D	B
D	C

Figure 2 - Matrix and list of a direction network

Clustering

Friends of a given node can also be friends with each other. The clustering coefficient (c_i) compares the number of links with other nodes to the potential number of links with the other nodes in the group.

$$c_i = \frac{2|e_{jk}|}{k_i(k_i - 1)} : v_j, v_k \in N_i, e_{jk} \in E$$

Where N_i is the neighborhood of node v_i, e_{jk} represents the edge that connect to the node $v_j$to the node v_k, k_i is the degree of node v_i, and $|e_{jk}|$ indicates the edges between nodes of neighborhood of v_i.(Gama et al., 2012)

3 Information Diffusion Models

We can analyze two types of diffusion networks: Epidemics and Propagation and Dynamic Opinion.

3.1 Epidemics and Propagation

Epidemic can be a disease and its spread from individual to individual, or a computer virus, the spread of a computer innovation or even the social spread of a certain behavior or idea. Different elements determine the patterns by which epidemics spread, which can be distinguished by a single variable: degree of activeness. (Rossetti et al., 2018)

3.2 Dynamic Opinion

There are many studies and models that attempt to explain the way opinion is spread. Many of these models are defined as graphs, where nodes characterize individuals who are linked together where they are or are not influenced by each other. The state of a node can be described by discrete or continuous variables, representing for example the probability of choosing or not choosing an option.

Node status may change over time, unlike epidemic and propagation models.

The networks can be represented by a time network, which describe a dynamic structure where nodes and links can appear and disappear at any given moment.

We could have an instant network whose correspond to the snapshot of a network at any time t.

3.3 Static Models

SI Model

The model introduced by (W. O. Kermack, 1927), suggests that during the epidemic process a node changes from susceptible state (S) to infected state (I), that is, a certain susceptible node in contact with an infected node becomes if he himself is infected with a probability β.

From the moment the node becomes infected it is always infected, being the only possible transition from S −> I.

SIR Model

The SIR model was introduced by (W. O. Kermack, 1927), being an extension of the previous one. In this model and during an epidemic process, a node may have the state from susceptible (S) to infected (I) and then removed (R)

A node becomes infected with probability β and is removed with probability γ, with the only possible transition being S −> I −> R.

Independent Cascades Model (IC)

Introduced Kempe in 2003, this model defines its diffusion process starting with an initial set of active nodes. A node is active when it is susceptible to being infected with the information, and the model begins with a certain number of infected nodes. At each interaction a node can activate a neighbor node with a given probability. The process ends when no more nodes can be activated. Thus during the process, a particular node can be susceptible (S), infected (I) or removed (R).

"Each node v chooses a threshold $\theta_v\theta_v$ uniformly at random from the interval [0, 1]; this represents the weighted fraction of v's neighbors that must become active in order for v to become active. Given a random choice of thresholds, and an initial set of active nodes A_0A_0 (with all other nodes inactive), the diffusion process unfolds deterministically in discrete steps: in step t, all nodes that were active in step t − 1 remain active, and we activate any node v for which the total weight of its active neighbors is at least $\theta_v\theta_v$" (David Kempe, Jon Kleinberg, 2002):

$$\sum_{w \text{ active neighbor of } v} b_{v,w} \geq \theta_v.$$

Independent Cascade Model based on opinion change (IMIC)

This model (Wang, Jin, Lin, Cheng, & Yang, 2016) shows the process by which users build their opinion. With a diffusion of information that they want to view as recent opinions and with input from new users, they will influence the others, and may or may not change their mind. This change is calculated through probabilities.

We then have the actual application of these methods. The independence cascade model is used for information diffusion and the IMIC model to calculate the maximum positive influence. That is, the former allows us to explain how the builder or user uses their opinions about the information dissemination process, or the latter allows to find users who have the maximum positive variation.

Continuous-Time Markov Chain Model (CTMC)

Introduces an improvement to the method described above. They also prepare a new metric for ordering nodes – SpreadRank (Zhu, Wang, Wu, & Zhu, 2014) -, which takes into account the diffusion ability of each network node, making it more efficient than distance-based in centrality.

Marketing Influential Value (MIF)

One of the means used for enterprise level dissemination is blogging. Identifying an influential blogger becomes a unique opportunity for sales and promotion. In the article "Discovering influencers for marketing in the blogosphere" (Li, Lai, & Chen, 2011) the model is developed for gauging the power of influence and identifying in fluent bloggers - Marketing Influential Value (MIF). Three dimensions of blogs are analyzed to find the power of influence and identify influential bloggers. Based on the network, content and activity on the network, it uses an artificial neural network to discover the blogger's potential.

3.4 Dynamic Models

The difference to static models, is that typology of dynamic networks changes over times and nodes and/or edges may come and go.

In "Diffusive Phenomena in Dynamic Networks: a data-driven study" article, authors "compare the behaviors of classical spreading models when used to analyze a given social network whose topological dynamics are observed at different temporal- granularities."

(Milli, Rossetti, Pedreschi, & Giannotti, 2018)

SI Model

In this model, during an epidemic process, a node can change its status only from Susceptible (S) to Infected (I). Thus, the graph is characterized by a non-empty set of infected nodes.

SI assumes that if, during a generic iteration, a susceptible node contacts an infected one, it becomes infected with probability β, that is, once a node is infected, it remains infected (the only allowed transition is S → I).

The implementation of this model assumes that the process occurs in a directed / undirected dynamic network. (Milli et al., 2018).

SIR Model

SIR assumes that if, during a generic iteration, a susceptible node comes into contact with an infected one, it becomes infected with probability β, than it can be switch to remove with probability γ (the only transition allowed are S→I→R).

4 Study of a network

We will apply the referenced measures in a real database in order to identify nodes that can accelerate or enhance the diffusion of information in a network.

Through the Netvizz application you can obtain data from a Facebook® network.

I chose to study one of the most visited lifestyle blogs in Portugal through its Facebook® network. With a total of 255,000 followers and more than 250,000 likes, its promoter is chosen by various brands to be disseminated through social networks.

At http://lookup-id.com/# I find the Facebook® page ID and with the Netvizz app at https://apps.facebook.com, I make the download of the network to be read through Gephi.

The network represents Facebook® pages linked through the "likes" to each other. The database consists of the ID (our nodes), page name, username, category according to Facebook® rating, number of posts per hour, number of likes the page receives, metric used by Facebook® to measure the number of people talking about a particular page, whether or not users can post to that page and the home page link. The edges are represent by the source edge and target edge.

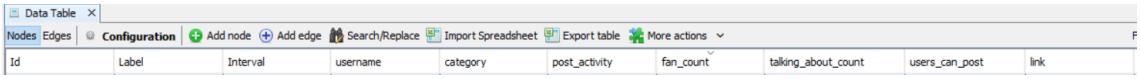


Figure 3 - database fields

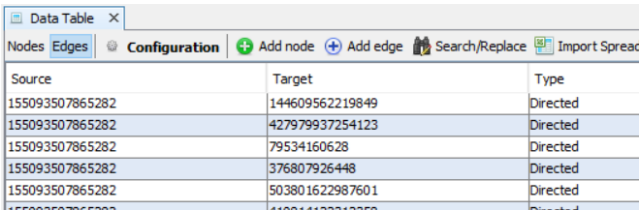


Figure 4 - edges of network

This network has 465 nodes and 4.095 edges, and we can visualize in Gephi. We are manipulating the different levels of network to obtain the information that we need to analyze the propagation of information in network.

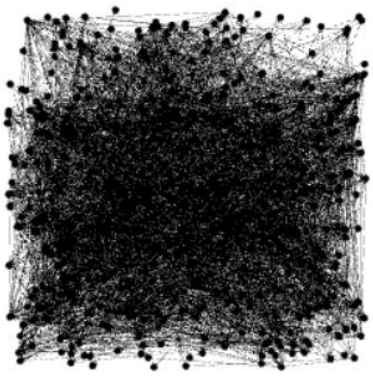


Figure 5 - Network in Gephi

After changing the layout to ForceAtlas 2 at a scale of 100 we were able to change the display of the network from a black spot to a setting where we can better understand the density of some nodes and how far they are apart by how they are connected between each other. The further the nodes are from each other means that there are fewer links between them.

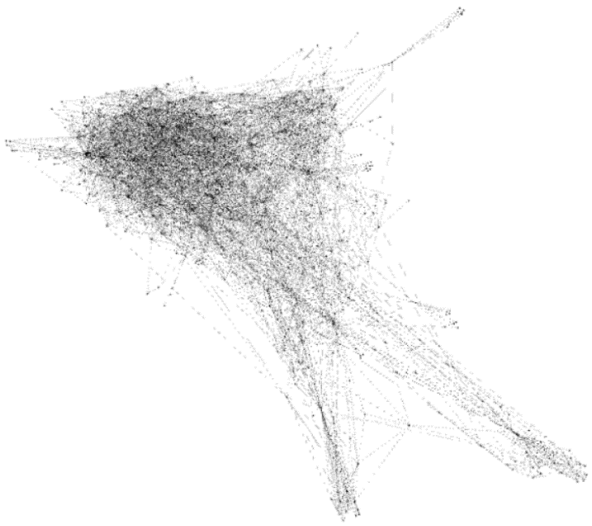


Figure 6 - Network in ForceAtlas 2 layout

The network can be manipulated to highlight the most important nodes or connections in the network. Given the initial criteria of post activity, fan account, talking_abou_count or even creating groups according to category, we can change de layout of network to evidence these criteria.

Let's start by showing which nodes with the most fans. So, the bigger the node the bigger the number of fans this page has.

And color-scale the nodes with the most activity. The more reddish the nodes appear, the greater their activity in terms of network publications.

Figure 7 - nodes with more activity and more fans number

4.1 Statistics – Database Analysis

Modularity

This algorithm allows to detect communities from the weights. It is a -1 to 1 scale that measures the density of links within communities and compares with links between communities (Blondel, Guillaume, Lambiotte, & Lefebvre, 2008).

In Gephi the normal value is 1 and the higher this value, the fewer communities are detected but the larger.

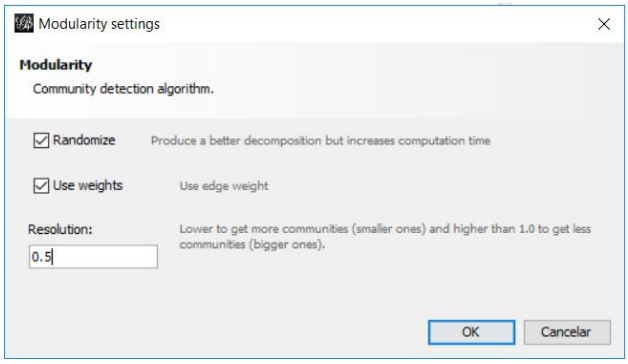


Figure 8 - Modularity criteria in Gephi

With resolution of 0,5 we find 22 communities, with 1 of resolution obtain 8 communities and detect 4 with 2 of resolution.

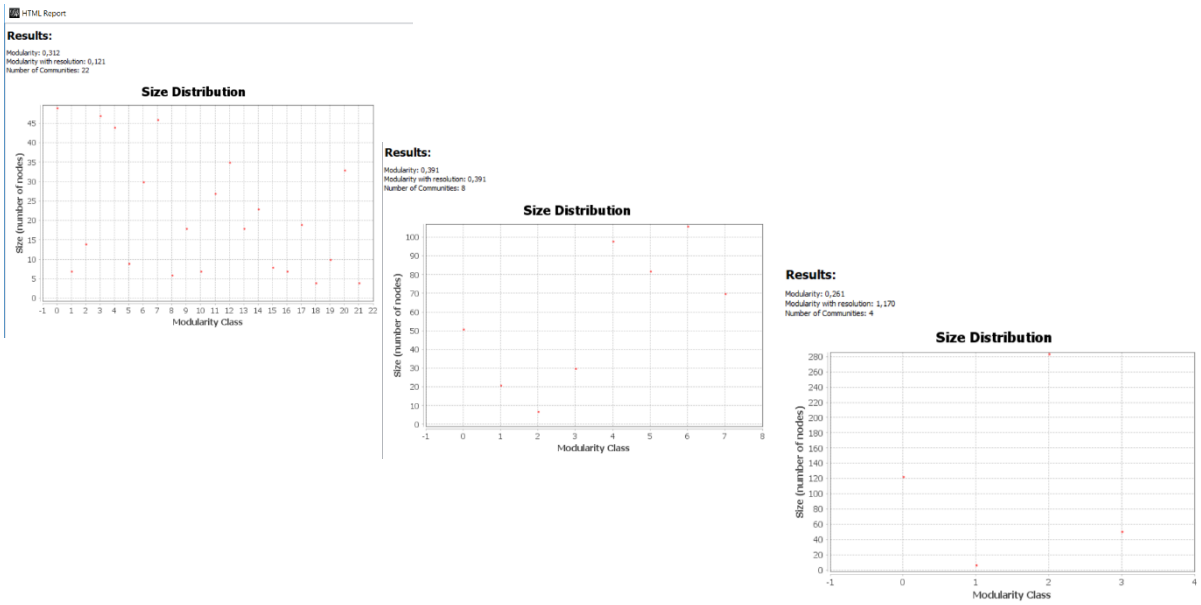


Figure 9 - Communities results with different resolution

We will choose the resolution of 1 and apply modularity to our network to show the 8 communities.

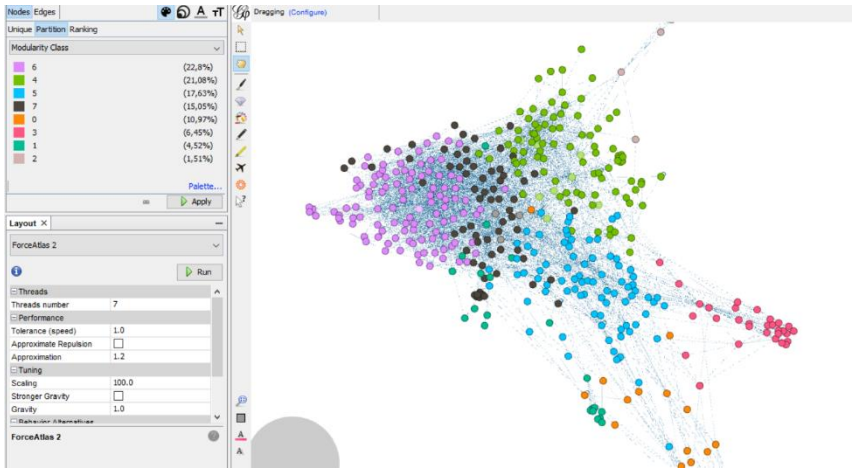


Figure 10 - Network with 8 communities

The purple color represents de biggest community with 22,8% of nodes. In opposite is the community 2, with light pink color, we have only 1,51% of nodes. It's the smallest community in network.

4.2 Averange Degree

Degree is given by the number of links a given node has with other nodes(Freeman, 1978). a node with a high degree is a node with some relevance in the network because it has more connections with other actors in the network and therefore can pass the message to more people.

In the analyzed network we obtained an average of connections per node of 8,806, that is, on average each node has 8,806 connections.

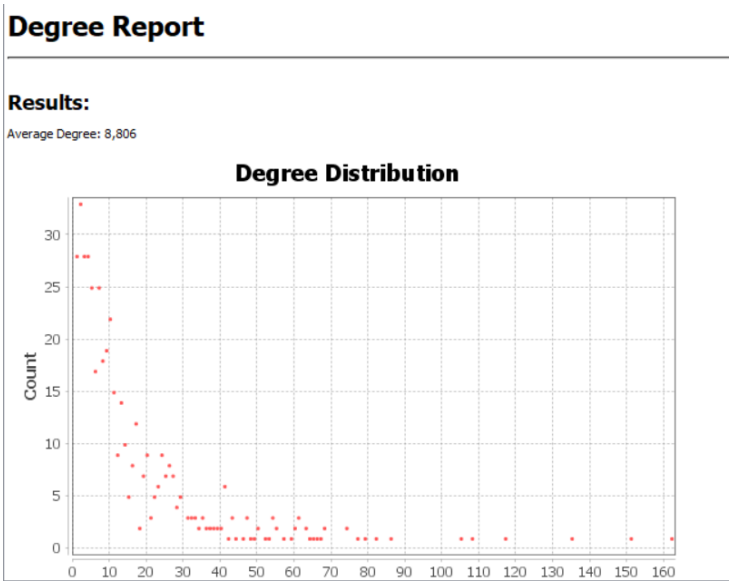


Figure 11 - Degree Distribution

There are 6 nodes with more than 100 links and have 28 with only 1 connection. The nodes with higher degree could be considerate nodes important to diffusion of information. To evidence the nodes with high degree and keeping the discretion of communities, we will modulate the network to show this.

Figure 12 - nodes with high degree on each community

4.3 Network Diameter

The measures of centrality are based on the distance between the nodes.

With betweenness centrality (Brandes, 2001) we measure how many times a node appears on the shortest path in the network. It shows us that nodes function as bridges between other nodes in the network and allows us to identify all shortest paths and count how many times each node appears on that path.

The closeness centrality measure (Brandes, 2001) calculate the average of distance from a given node to the remaining nodes of the network from which we can rake the nodes with the shortest distances from remaining nodes of the network. We can find the nodes in the best position to influence the remaining network.

The eigenvector centrality measures the importance of a node in the network based on the connections between the nodes. We get influential ones across the network and not just those who are directly connected with you.

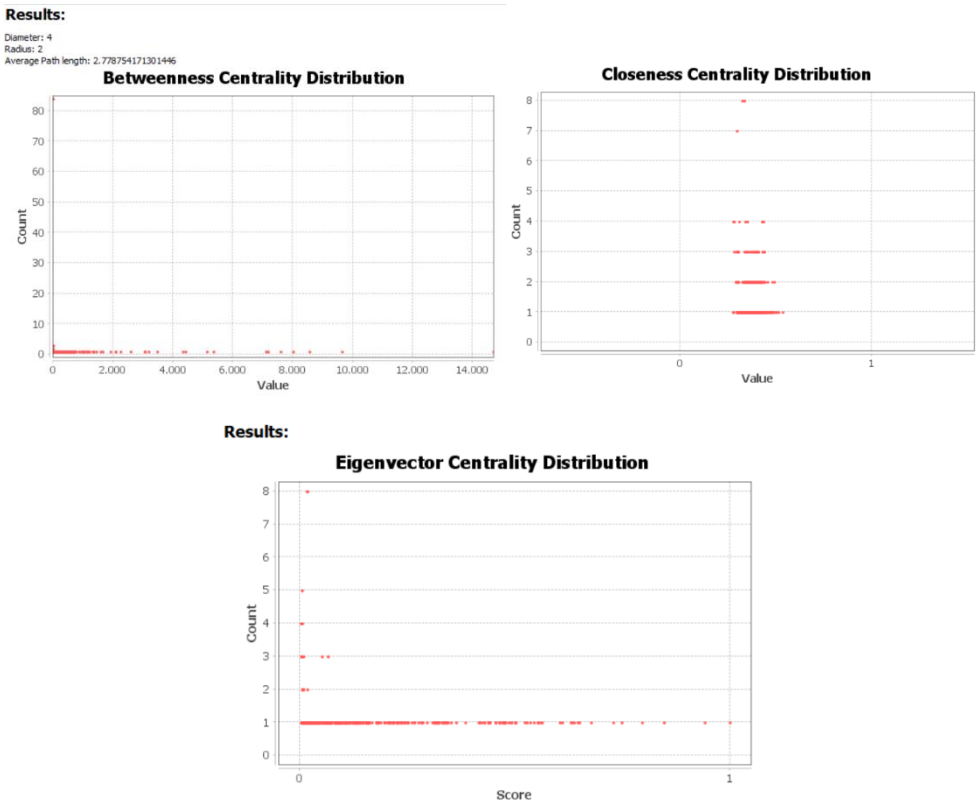


Figure 13 - Betweenness, Closeness and Eigenvector Centrality

If we highlight the nodes with greater betweenness centrality, we find in orange, blue, green, purple and grey community some nodes that stand out.

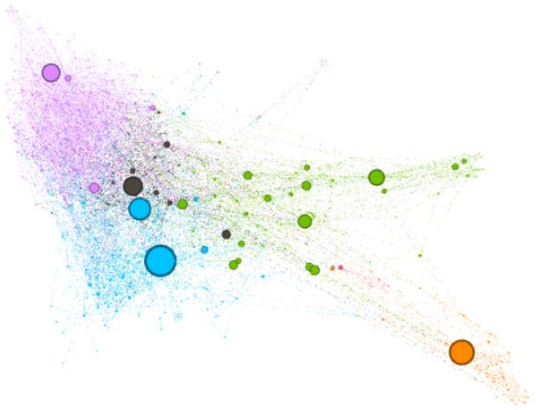


Figure 14 - Nodes with highest betweenness

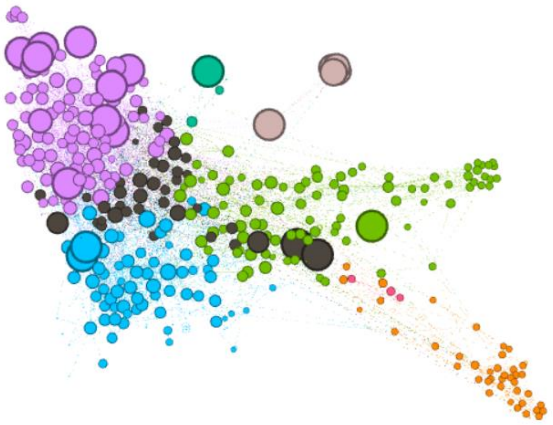


Figure 15 - nodes with higher closeness centrality

With closeness centrality we could conclude that have many nodes with values very close, which makes it difficult to identify the most important nodes in the network.

In turn, if we highlight the nodes with the highest eigenvector centrality, we realize that most of the most influential nodes in the entire network belong to the purple community.

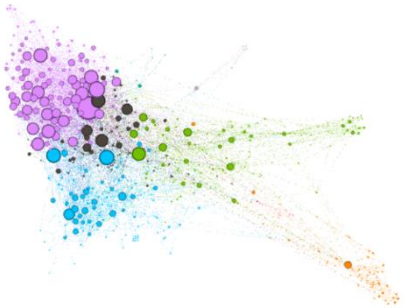


Figure 16 - nodes with highest eigenvector centrality

5 Aplicação de Modelos à base de dados

The NDLIB is a tool that allows analysis social networks and represent a multidimensional solution for epidemic diffusion simulation. Developed by Giulio Rossett, is a Python software package that allows to describe, simulate, and study diffusion processes on complex networks. (Rossetti, 2019). The goal is built upon the NetworkX python library and is intended to provide:

• tools for the study diffusion dynamics on social, biological, and infrastructure networks,

• a standard programming interface and diffusion models implementation that is suitable for many applications,

• a rapid development environment for collaborative, multidisciplinary, projects., (Rossetti, 2019)

We chose this tool deteriorating from others because it allows quickly and easily change the initial parameters of the models as well as view the results.

In (Rossetti et al., 2018) we can achieve the differences in each libraries and tools on network analysis.

Name	Lang.	Epi.	Op. Dyn.	Viz.	Dyn. Net.	Exp. Server	Viz. Platform	Ext.	Net. Model	Active	License
NDLIB	Python	✓(14)	✓(5)	✓	✓	✓	✓	✓	NetworkX DyNetX	✓	BSD
epigrass	Python	✓(8)		✓			✓	✓	NetworkX	✓	GPL
GEMFsim	Python	✓(4)						✓	NetworkX	✓	-
Nepidemix	Python	✓(3)					✓	✓	NetworkX	✓	BSD
EoN	Python	✓(3)							NetworkX	✓	MIT
epydemic	Python	✓							NetworkX	✓	GPL
ComplexNetworkSim	Python	✓(3)		✓	✓			✓	NetworkX		BSD
nxsim	Python	✓(3)		✓	✓			✓	NetworkX		Apache
EpiModel	R	✓(3)		✓			✓	✓	igraph	✓	GPL
RECON	R	✓		✓			✓	✓	adhoc	✓	Various
sisspread	C	✓(3)							adhoc	✓	GPL
GLEaMviz	C++ Python	✓		✓		✓	✓	✓	adhoc	✓	SaaS

Table 1 - Diffusion Libraries and Tools. A qualitative comparison of network diffusion related libraries and tools. Within brackets the number of diffusion models, if any, natively implemented within each library.

NDLIB allow apply the models in a random network with network that is another Python package. We will apply the models SI, SIR and Independent Cascade in our database.

5.1 Database import

First will install the packages mandatory to develop all codes: Ndlib, pandas, networkx, matplolib.

After, in Python we need import all models that we will need:

```
import networkx as nx
import matplotlib.pyplot as plt
import pandas as pd
import ndlib.models.ModelConfig as mc
from ndlib.viz.bokeh.MultiPlot import MultiPlot
from bokeh.io import output_notebook, show
from ndlib.viz.bokeh.DiffusionTrend import DiffusionTrend
import ndlib.models.epidemics.SIModel as si
import ndlib.models.epidemics.SIRModel as sir
import ndlib.models.epidemics.IndependentCascadesModel as ids
```

To import our databases:

```
#Our data
vm = MultiPlot()
g = nx.DiGraph()
df = pd.read_csv('C:\\Users\\networkx.csv', header=0)

for row in df.iterrows():
    print(row[1].Source)
    g.add_node(row[1].Source)
    g.add_node(row[1].Target)

for row in df.iterrows():
    g.add_edge(row[1].Source,row[1].Target)

nx.draw(g,with_labels=False)
plt.draw()
plt.show()
```

And visualize our network through Python. The layout isn't so friendly like Gephi but the objective isn't seeing the network but analysis the results of models.

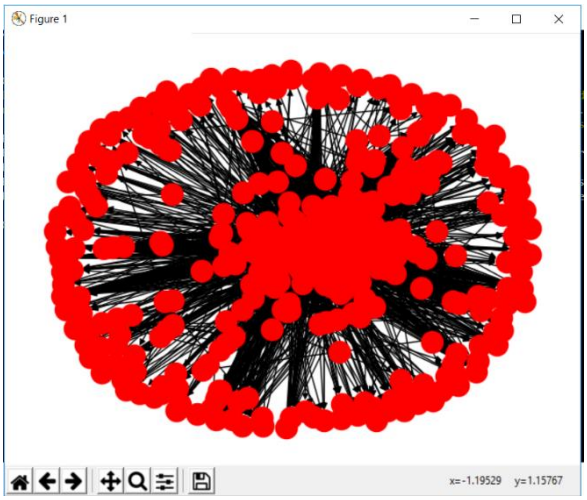


Figure 127 - our network in Python layout

With application of models, we will infect a percentage of initial nodes and see how the propagation in network is, that is, how many nodes are infected and how long does it take.

First, we will apply the models with standard parameters, and after we will choose the initial nodes will be infected to show if diffusion is bigger and or more quickly.

5.2 SI Model

The number of initial infected nodes is 5% and we will assume a probability of infection (β) of 1%. Apply the model with 200 iterations.

Code on Python:

```
#SI Model

# Model selection
model = si.SIModel(g)

# Model Configuration
cfg = mc.Configuration()
cfg.add_model_parameter('beta', 0.01)
cfg.add_model_parameter("fraction_infected", 0.05)
model.set_initial_status(cfg)

# Simulation execution
iterations = model.iteration_bunch(200)
trends = model.build_trends(iterations)

# Visualizion
viz = DiffusionTrend(model, trends)
```

```
p = viz.plot(width=400, height=400)
show(p)
```

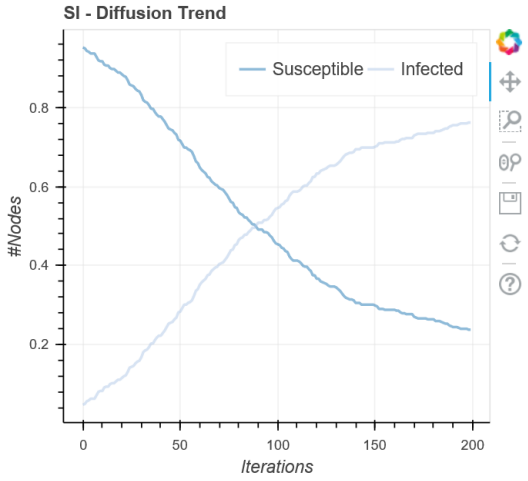


Figure 18 - SI model

We conclude that over the interactions the infected nodes are increasing, and after 200 interactions we have about 80% of the infected nodes.

5.3 SIR Model

This model consider that nodes could be removed of network. So we must define a percentage to probability of the nodes be removed (γ).

Equal to the previous model, we start with 5% of initial infected nodes, and the probability of infected a new node (β) is 1%. To γ we will assume a 0,5% of probability of a node be removed.

Code on Python:

```
#SIR Model

# Model selection
model = sir.SIRModel(g)

# Model Configuration
cfg = mc.Configuration()
cfg.add_model_parameter('beta', 0.01)
cfg.add_model_parameter('gamma', 0.005)
cfg.add_model_parameter("fraction_infected", 0.05)
```

```
model.set_initial_status(cfg)

# Simulation execution
iterations = model.iteration_bunch(200)
trends = model.build_trends(iterations)

# Visualizion
viz = DiffusionTrend(model, trends)
p = viz.plot(width=400, height=400)
show(p)
```

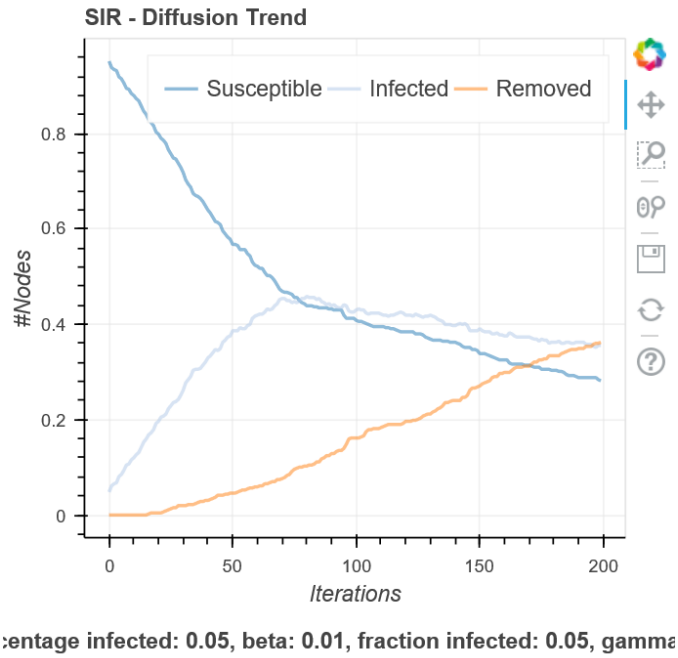


Figure 19 - SIR model

With this model the total of infected nodes is lower than first one. In 200 interactions we have less than 40% nodes that be infected and still have nodes susceptible of infection, about 30%. The process ends when all nodes infected are removed and same nodes never been infected.

5.4 Independent Cascades

As in the previous ones, we will apply the IC model with an initial percentage of infected nodes of 5%, and we will analyze how the information is propagated, that is, see after 200 interactions how many nodes are infected and how many are removed.

Python code:

```
#IndependentCascades Model

# Model selection
model = ids.IndependentCascadesModel(g)

# Model Configuration
cfg = mc.Configuration()
cfg.add_model_parameter('fraction_infected', 0.05)

# Setting the edge parameters
threshold = 0.1
for e in g.edges(): cfg.add_edge_configuration("threshold", e, threshold)

model.set_initial_status(cfg)

# Simulation execution
iterations = model.iteration_bunch(200)
trends = model.build_trends(iterations)

# Visualizion
viz = DiffusionTrend(model, trends)
p = viz.plot(width=400, height=400)
show(p)
```

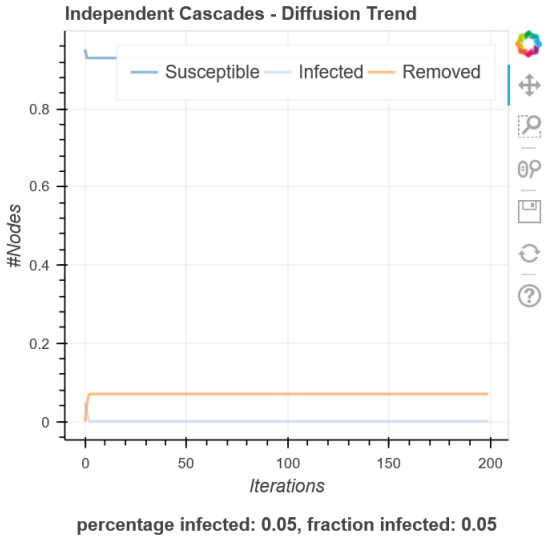


Figure 20 - Independent Cascade model

After 200 interactions we find that in the first interactions the number of infected nodes is removed and only the susceptible ones are left, that is, in this model there is no propagation of information since there are very few infected nodes. The database used despite being a

real database has few nodes for application of this model, which does not allow to draw large conclusions regarding the dissemination of information.

5.5 Dynamic Network Models

With NDLIB we will apply dynamic networks to our database.

First, we need import some newest packages, so introduce in Python the following code:

```
import dynetx as dn
import ndlib.models.dynamic.DynSIModel as dm
from past.builtins import xrange
```

SI Model

In the code below is shown an example of instantiation and execution of a DynSI simulation. We set the initial set of infected nodes as 5% of the overall population and a probability of infection (β) of 1%.(Rossetti, 2019)

```
#SIModel
dg = dn.DynGraph()
for t in xrange(0, 3):
dg.add_interactions_from(g.edges(), t)

# Model selection
model = dm.DynSIModel(dg)

# Model Configuration
config = mc.Configuration()
config.add_model_parameter('beta', 0.01)
config.add_model_parameter("fraction_infected", 0.05)
model.set_initial_status(config)

# Simulate snapshot based execution
iterations = model.execute_snapshots()
# Simulation interaction graph based execution
iterations = model.execute_iterations()

trends = model.build_trends(iterations)
viz = DiffusionTrend(model, trends)
p = viz.plot(width=400, height=400)
show(p)
```

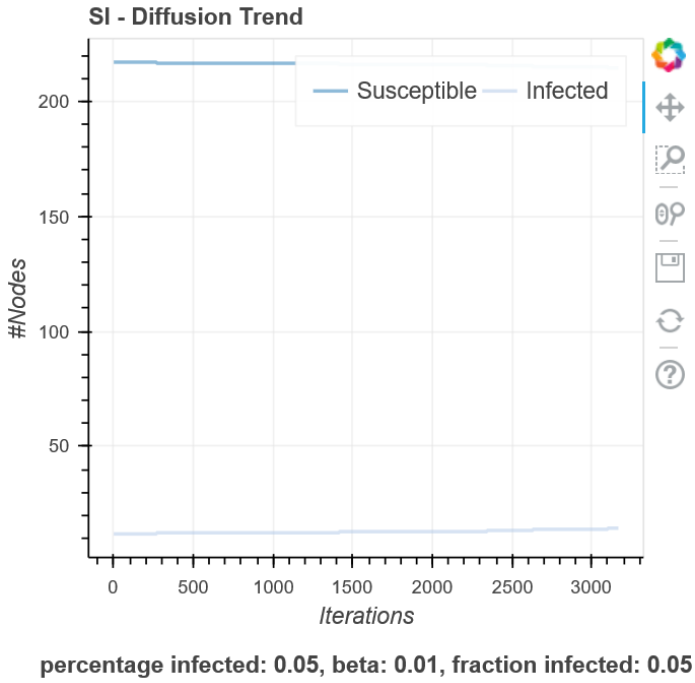


Figure 21 - SI model in dynamic network

In this typology of diffusion, we conclude that didn't exist diffusion, because all nodes still susceptible and few nodes are infected.

SIR Model

In the code below is shown an example of instantiation and execution of an simulation, we set the initial set of infected nodes as 5% of the overall population, a probability of infection (β) of 1%, and a removal probability (γ) of 0,5% (Rossetti, 2019).

Must import the dynamic model to SIRModel, and apply the model:

```
import ndlib.models.dynamic.DynSIRModel as ds

# Dynamic Network topology
dg = dn.DynGraph()
for t in xrange(0, 3): dg.add_interactions_from(g.edges(), t)

# Model selection
model = ds.DynSIRModel(dg)

# Model Configuration
```

```
config = mc.Configuration()
config.add_model_parameter('beta', 0.01)
config.add_model_parameter('gamma', 0.005)
config.add_model_parameter("fraction_infected", 0.05)
model.set_initial_status(config)

# Simulate snapshot based execution
iterations = model.execute_snapshots()

# Simulation interaction graph based execution
iterations = model.execute_iterations()

# Visualizion
trends = model.build_trends(iterations)
viz = DiffusionTrend(model, trends)
p = viz.plot(width=1000, height=1000)
show(p)
```

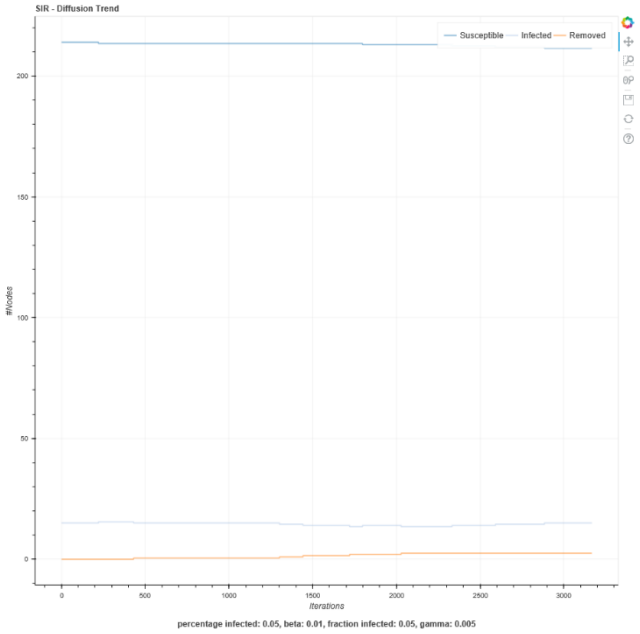


Figure 22 - SIR Model in Dynamic network

Like in SI Model, when we apply the dynamic network a few nodes are infected and all most still susceptible to infected.

The NDLIB didn't have the dynamic network to apply to the Independent Cascade Model.

The analysis of dynamic networks isn't the aim of this work but show the potentials of NDLIB tool.

6 Diffusion Trend Comparison

Other functionality of NDLIB is be able to show in one graph two models. We will compare the number of infected nodes between SI model and SIR model ate the end of 200 interactions.

Like previous codes, we need to import some packages that we need to generate the graphs, for example the Diffusion Trend Comparison.

```
import ndlib.models.ModelConfig as mc
from ndlib.viz.bokeh.MultiPlot import MultiPlot
from bokeh.io import output_notebook, show
from ndlib.viz.bokeh.DiffusionTrend import DiffusionTrend
import ndlib.models.epidemics.SIModel as si
import ndlib.models.epidemics.SIRModel as sir
from ndlib.viz.mpl.TrendComparison import DiffusionTrendComparison


#SI Model

# Model selection
model = si.SIModel(g)

# Model Configuration
cfg = mc.Configuration()
cfg.add_model_parameter('beta', 0.01)
cfg.add_model_parameter("fraction_infected", 0.05)
model.set_initial_status(cfg)

# Simulation execution
iterations = model.iteration_bunch(200)
trends = model.build_trends(iterations)

#SIR Model

# Model selection
model1 = sir.SIRModel(g)

# Model Configuration
cfg = mc.Configuration()
cfg.add_model_parameter('beta', 0.01)
cfg.add_model_parameter('gamma', 0.005)
cfg.add_model_parameter("fraction_infected", 0.05)
model1.set_initial_status(cfg)

# Simulation execution
iterations = model1.iteration_bunch(200)
trends1 = model1.build_trends(iterations)

# Visualizion
```

```
viz = DiffusionTrendComparison([model, model1], [trends, trends1])
viz.plot("trend_comparison.pdf")
```

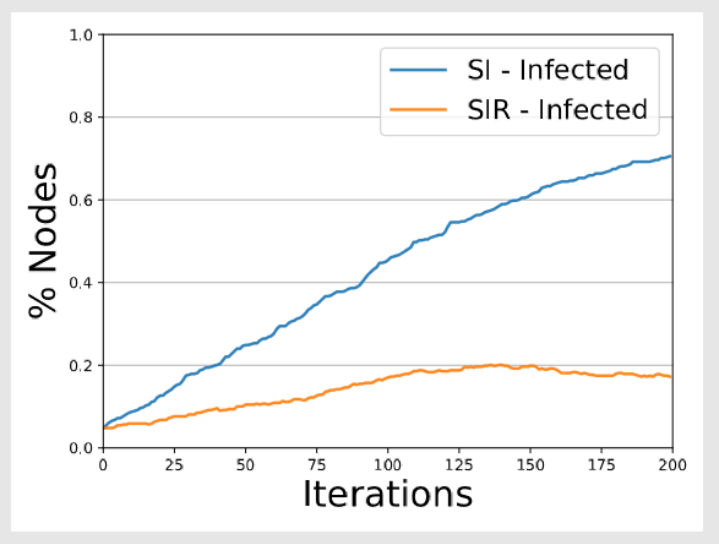


Figure 23 - Nodes infected in SI and SIR model

With this graph we can better visualize the difference between two models. The number of infected nodes in SI model is higher and in fewer interactions more nodes are infected than in SIR model. In this one some nodes are removed along the interactions, witch is closer of reality and show that the diffusion of information is more difficulty.

7 Tests

The models shown above show the propagation in the network, this is, the number of infected nodes, the initials being a random percentage.

It is now intended to demonstrate how propagation behaves if we select certain nodes, which initiate the propagation of information. As such, we will draw on the information obtained from the Gephi algorithms, and choose the nodes with the highest degree, centrality, and betweenness. We will also demonstrate what happens if instead we select the nodes with the lowest values of these criteria.

Next, let's compare whether or not it is effective to increase the number of infected initial nodes, in order to understand whether the larger the initial nodes, the greater or not, and the faster or not the propagation. We will select as infected initial nodes the first 10 that have the highest values. This represents starting with about 2% of the total nodes. The models presented earlier began with about 5% of infected initial random nodes. It will therefore be easy to see if with fewer nodes but the most important ones in the network can or cannot infect the remaining nodes faster and more effectively.

7.1 With the highest degree of centrality

Through the statistics already applied in Gephi, we obtain that the 10 degrees with the highest degree are:

Id	Degree
197086050320307	162
174665665924142	151
203067549705421	135
205724095732	117
198225803521750	108
179105525471430	105
212504785715	86
381740111841085	82
459328270852766	79
137585456260669	77
516804191670638	74
145940022138448	74

Table 2 - nodes with highest degree

They represent the nodes with the most direct links. A node with more connections has a central position in the network and is an important means of disseminating information.

To apply these initial nodes to the three models that we are studying, we need to specify the ID of nodes changing the base of code with infected_nodes for each model:

```
#SI Model

# Model selection
model = si.SIModel(g)

# Model Configuration
cfg = mc.Configuration()
cfg.add_model_parameter('beta', 0.01)
infected_nodes =
[197086050320307,174665665924142,203067549705421,205724095732,198225803521750,179105525471430,
212504785715,381740111841085,459328270852766,137585456260669]
cfg.add_model_initial_configuration("Infected", infected_nodes)
model.set_initial_status(cfg)

# Simulation execution
iterations = model.iteration_bunch(200)
trends = model.build_trends(iterations)

# Visualizion
viz = DiffusionTrend(model, trends)
p1 = viz.plot(width=400, height=400)
vm.add_plot(p1)


#SIR Model

# Model selection
model = sir.SIRModel(g)

# Model Configuration
cfg = mc.Configuration()
cfg.add_model_parameter('beta', 0.01)
cfg.add_model_parameter('gamma', 0.005)
infected_nodes =
[197086050320307,174665665924142,203067549705421,205724095732,198225803521750,179105525471430,
212504785715,381740111841085,459328270852766,137585456260669]
cfg.add_model_initial_configuration("Infected", infected_nodes)
model.set_initial_status(cfg)

# Simulation execution
iterations = model.iteration_bunch(200)
trends = model.build_trends(iterations)
# Visualizion
viz = DiffusionTrend(model, trends)
p2 = viz.plot(width=400, height=400)
vm.add_plot(p2)
```

```
#IndependentCascades Model

# Model selection
model = ids.IndependentCascadesModel(g)

# Model Configuration
cfg = mc.Configuration()
cfg.add_model_parameter('fraction_infected', 0.05)

# Setting the edge parameters
threshold = 0.1
for e in g.edges(): cfg.add_edge_configuration("threshold", e, threshold)

infected_nodes =
[197086050320307,174665665924142,203067549705421,205724095732,198225803521750,179105525471430,
212504785715,381740111841085,459328270852766,137585456260669]
cfg.add_model_initial_configuration("Infected", infected_nodes)
model.set_initial_status(cfg)

# Simulation execution
iterations = model.iteration_bunch(200)
trends = model.build_trends(iterations)

# Visualizion
viz = DiffusionTrend(model, trends)
p3 = viz.plot(width=400, height=400)
vm.add_plot(p3)

m = vm.plot()
show(m)
```

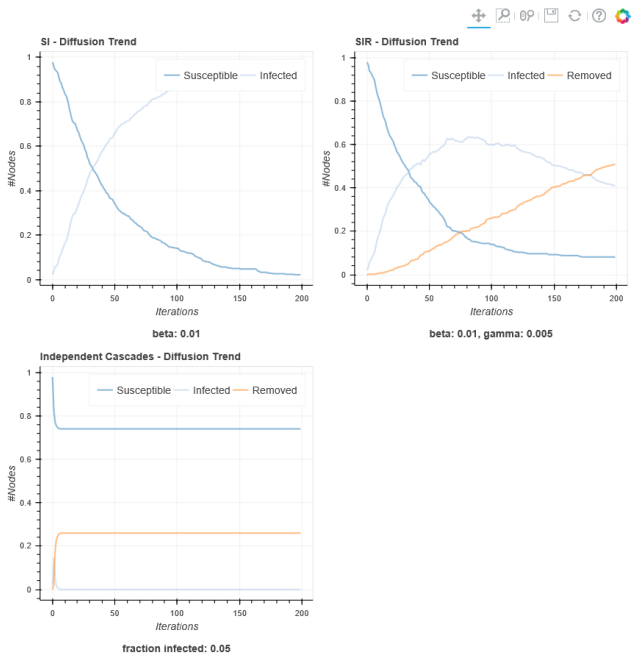


Figure 24 - SI, SIR and IC model with initial infected nodes with highest degree

Comparing with the initial results in which the infected nodes were about 5% random, we conclude that, when initially infecting the 10 nodes with the highest degree, even representing only 2% of the nodes, after fewer interactions, more nodes are infected.

In the SI model we have that before 50 interactions, about 50% of the nodes are already infected, while in the initial model only after the first 100 interactions we had more than 50% of us infected. In the case of the SIR model, we also find that after fewer interactions we have more infected nodes than if the infected initial nodes are randomly selected. The conclusion regarding the CI model is maintained, since after a few interactions no node is infected.

7.2 Nodes with greater centrality of eigenvector

Id	Eigenvector Centrality
137585456260669	1.0
35530840554	0.674619
155093507865282	0.619695
368144203347235	0.594305
10109514234	0.583549
272787167071	0.574779
197086050320307	0.57415
246510743272	0.564169
49342614805	0.557572
42933792278	0.544415
165920548508	0.530402
93042822078	0.522173
205724095732	0.511004

Table 3 - nodes with greater eigenvector

One more time, choose the 10 nodes with greater eigenvector to test if the diffusion process is more quickly than other or not and if has more nodes infected that with other nodes initial infected.

Need changing the list of infected nodes and run the code for the three models studied:

```
#SI Model

# Model selection
model = si.SIModel(g)

# Model Configuration
cfg = mc.Configuration()
cfg.add_model_parameter('beta', 0.01)
infected_nodes = [
137585456260669,35530840554,155093507865282,368144203347235,10109514234,272787167071,19708605
0320307,246510743272,49342614805,42933792278 ]
```

```
cfg.add_model_initial_configuration("Infected", infected_nodes)
model.set_initial_status(cfg)

# Simulation execution
iterations = model.iteration_bunch(200)
trends = model.build_trends(iterations)

# Visualizion
viz = DiffusionTrend(model, trends)
p1 = viz.plot(width=400, height=400)
vm.add_plot(p1)


#SIR Model

# Model selection
model = sir.SIRModel(g)

# Model Configuration
cfg = mc.Configuration()
cfg.add_model_parameter('beta', 0.01)
cfg.add_model_parameter('gamma', 0.005)
infected_nodes = [
137585456260669,35530840554,155093507865282,368144203347235,10109514234,272787167071,19708605
0320307,246510743272,49342614805,42933792278 ]
cfg.add_model_initial_configuration("Infected", infected_nodes)
model.set_initial_status(cfg)

# Simulation execution
iterations = model.iteration_bunch(200)
trends = model.build_trends(iterations)

# Visualizion
viz = DiffusionTrend(model, trends)
p2 = viz.plot(width=400, height=400)
vm.add_plot(p2)

#IndependentCascades Model

# Model selection
model = ids.IndependentCascadesModel(g)

# Model Configuration
cfg = mc.Configuration()
cfg.add_model_parameter('fraction_infected', 0.05)

# Setting the edge parameters
threshold = 0.1
for e in g.edges(): cfg.add_edge_configuration("threshold", e, threshold)

infected_nodes = [
137585456260669,35530840554,155093507865282,368144203347235,10109514234,272787167071,19708605
0320307,246510743272,49342614805,42933792278 ]
cfg.add_model_initial_configuration("Infected", infected_nodes)
```

```
model.set_initial_status(cfg)

# Simulation execution
iterations = model.iteration_bunch(200)
trends = model.build_trends(iterations)

# Visualizion
viz = DiffusionTrend(model, trends)
p3 = viz.plot(width=400, height=400)
vm.add_plot(p3)

m = vm.plot()
show(m)
```

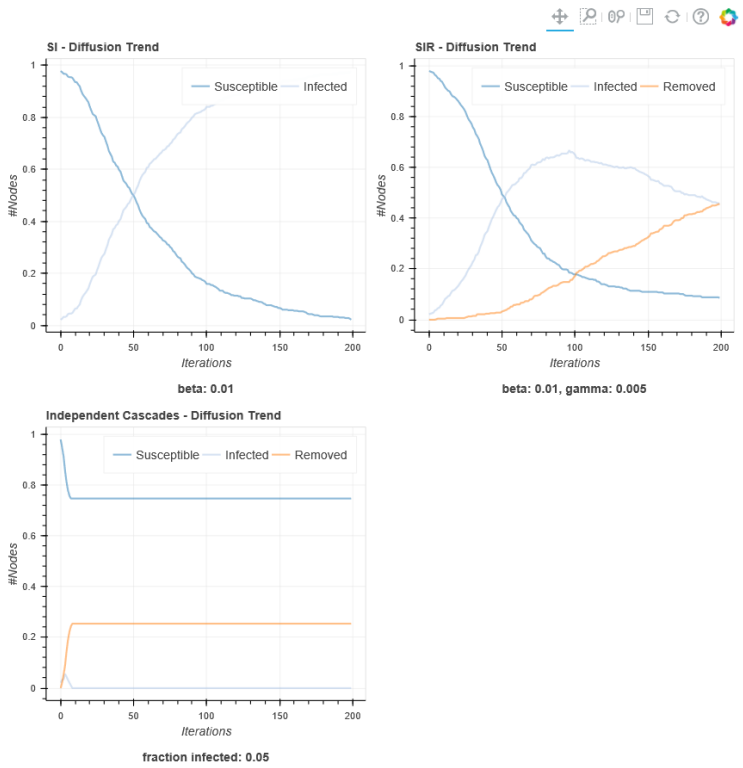


Figure 25 – SI, SIR and IC model with initial infected nodes with greater eigenvector

Regarding the SI model we find that the diffusion is not as effective as with the nodes with greater eigenvector. Only 50% of infected nodes are reached after 50 interactions, later than in the previous example. Same as in SIR model. Although there is a small increase in infected nodes the spread is slower, and more nodes are removed from the network. Again, regarding the IC model, it is inconclusive, and the results are very similar to the previous ones.

7.3 Nodes with higher Betweenness

Id	Betweenness Centrality
174665665924142	31056.238696
212504785715	24482.645185
155093507865282	21496.063975
205724095732	18256.90135
197086050320307	17463.791698
160520493840	15067.682365
376807926448	12647.480335
179105525471430	9030.39623
49342614805	8704.853457
373164443850	8573.046979
122325156660	7993.616997
116414041772996	7873.165532
187951874558978	7622.041522

Table 4 - Nodes with higher Betweenness

Here the 10 nodes selected represent the nodes that are closest to each other and not just the ones with the most connections.

```
#SI Model

# Model selection
model = si.SIModel(g)

# Model Configuration
cfg = mc.Configuration()
cfg.add_model_parameter('beta', 0.01)
infected_nodes = [
174665665924142,212504785715,155093507865282,205724095732,197086050320307,160520493840,376807
926448,179105525471430,49342614805,373164443850 ]
cfg.add_model_initial_configuration("Infected", infected_nodes)
model.set_initial_status(cfg)

# Simulation execution
iterations = model.iteration_bunch(200)
trends = model.build_trends(iterations)

# Visualizion
viz = DiffusionTrend(model, trends)
p1 = viz.plot(width=400, height=400)
vm.add_plot(p1)


#SIR Model

# Model selection
model = sir.SIRModel(g)

# Model Configuration
cfg = mc.Configuration()
cfg.add_model_parameter('beta', 0.01)
cfg.add_model_parameter('gamma', 0.005)
```

```
infected_nodes = [
174665665924142,212504785715,155093507865282,205724095732,197086050320307,160520493840,376807
926448,179105525471430,49342614805,373164443850 ]
cfg.add_model_initial_configuration("Infected", infected_nodes)
model.set_initial_status(cfg)

# Simulation execution
iterations = model.iteration_bunch(200)
trends = model.build_trends(iterations)

# Visualizion
viz = DiffusionTrend(model, trends)
p2 = viz.plot(width=400, height=400)
vm.add_plot(p2)

#IndependentCascades Model

# Model selection
model = ids.IndependentCascadesModel(g)

# Model Configuration
cfg = mc.Configuration()
cfg.add_model_parameter('fraction_infected', 0.05)

# Setting the edge parameters
threshold = 0.1
for e in g.edges(): cfg.add_edge_configuration("threshold", e, threshold)

infected_nodes = [
174665665924142,212504785715,155093507865282,205724095732,197086050320307,160520493840,376807
926448,179105525471430,49342614805,373164443850 ]
cfg.add_model_initial_configuration("Infected", infected_nodes)
model.set_initial_status(cfg)

# Simulation execution
iterations = model.iteration_bunch(200)
trends = model.build_trends(iterations)

# Visualizion
viz = DiffusionTrend(model, trends)
p3 = viz.plot(width=400, height=400)
vm.add_plot(p3)

m = vm.plot()
show(m)
```

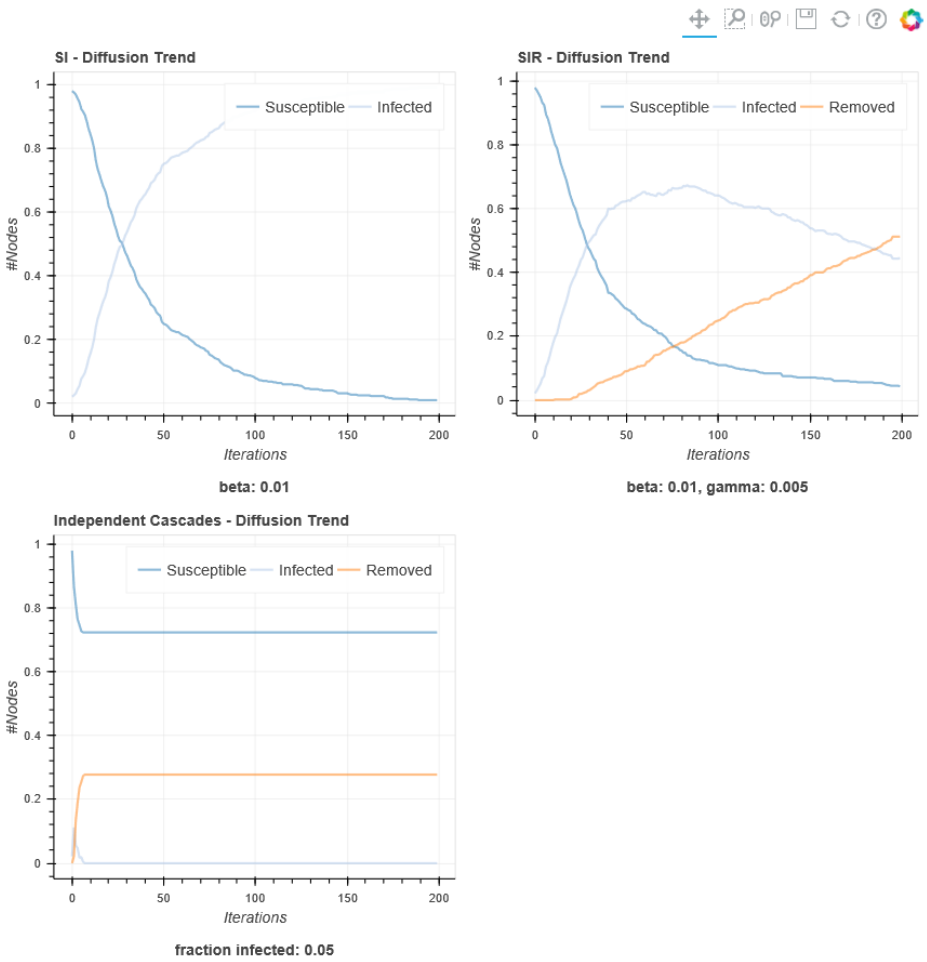


Figure 26 - SI, SIR and IC model with initial infected nodes with greater betweenness

The results are similar to those found with the application of the higher degree nodes, with both SI and SIR model improving diffusion, this is, more infected nodes faster. In SI model after 50 interactions we have almost 80% of infected nodes when with the highest nodes we had just over 60%.

7.4 Nodes with a lower degree

The difference between higher degree nodes and lower degree nodes is that diffusion is expected to be faster and reach more nodes in the network if it starts with higher degrees. This is what will be proved by choosing nodes with a lower degree.

Id	Degree
410814122312358	1
112608675420765	1
459162747586364	1
173062343381104	1
369287923423885	1
367569886919341	1
130542144256938	1
220790931331645	1
1697132070560973	1
538533919636364	1
818948298160878	1
661261670610472	1
1413536948958427	1

Table 5 - the 10 nodes with lower degree

```
#SI Model

# Model selection
model = si.SIModel(g)

# Model Configuration
cfg = mc.Configuration()
cfg.add_model_parameter('beta', 0.01)
infected_nodes = [
410814122312358,112608675420765,459162747586364,173062343381104,369287923423885,3675698869193
41,130542144256938,220790931331645,1697132070560970,538533919636364 ]
cfg.add_model_initial_configuration("Infected", infected_nodes)
model.set_initial_status(cfg)

# Simulation execution
iterations = model.iteration_bunch(200)
trends = model.build_trends(iterations)

# Visualizion
viz = DiffusionTrend(model, trends)
p1 = viz.plot(width=400, height=400)
vm.add_plot(p1)


#SIR Model

# Model selection
model = sir.SIRModel(g)

# Model Configuration
cfg = mc.Configuration()
cfg.add_model_parameter('beta', 0.01)
cfg.add_model_parameter('gamma', 0.005)
infected_nodes = [
410814122312358,112608675420765,459162747586364,173062343381104,369287923423885,3675698869193
41,130542144256938,220790931331645,1697132070560970,538533919636364 ]
cfg.add_model_initial_configuration("Infected", infected_nodes)
```

```
model.set_initial_status(cfg)

# Simulation execution
iterations = model.iteration_bunch(200)
trends = model.build_trends(iterations)

# Visualizion
viz = DiffusionTrend(model, trends)
p2 = viz.plot(width=400, height=400)
vm.add_plot(p2)

#IndependentCascades Model

# Model selection
model = ids.IndependentCascadesModel(g)

# Model Configuration
cfg = mc.Configuration()
cfg.add_model_parameter('fraction_infected', 0.05)

# Setting the edge parameters
threshold = 0.1
for e in g.edges(): cfg.add_edge_configuration("threshold", e, threshold)

infected_nodes = [
410814122312358,112608675420765,459162747586364,173062343381104,369287923423885,3675698869193
41,130542144256938,220790931331645,1697132070560970,538533919636364 ]
cfg.add_model_initial_configuration("Infected", infected_nodes)
model.set_initial_status(cfg)

# Simulation execution
iterations = model.iteration_bunch(200)
trends = model.build_trends(iterations)

# Visualizion
viz = DiffusionTrend(model, trends)
p3 = viz.plot(width=400, height=400)
vm.add_plot(p3)

m = vm.plot()
show(m)
```

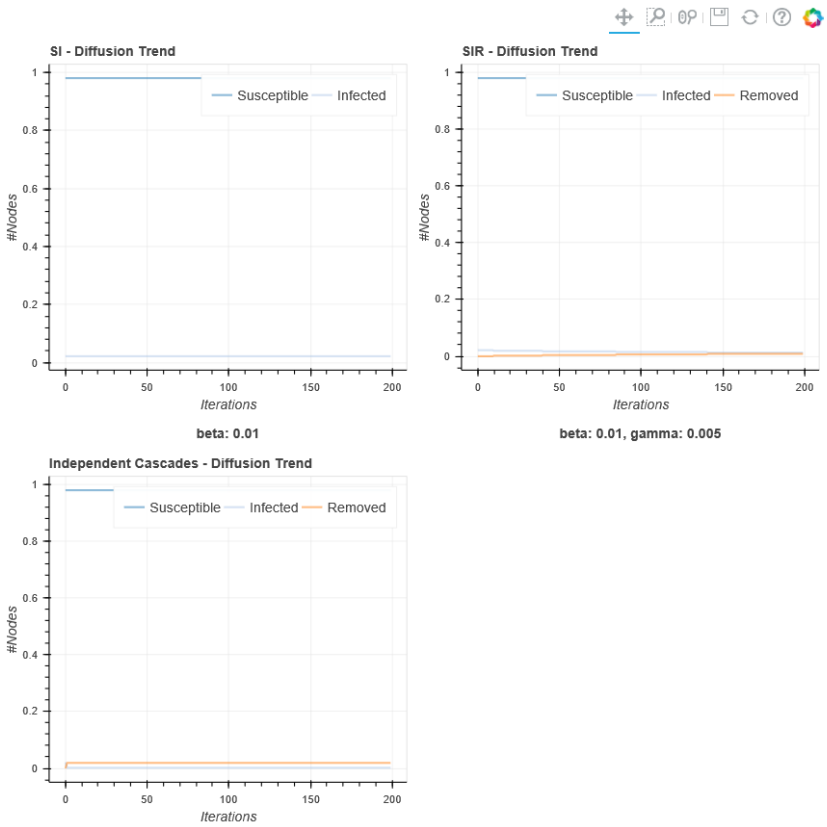


Figure 27 - SI, SIR and IC model with initial infected nodes with lower degree

As we expected, when selecting the lower degree the nodes to initial infected here is no diffusion in any of the models. In first interactions is not has nodes infected and all nodes still susceptible.

8 Conclusions and future steps

The world today finds itself with a new way of communicating making it a challenge for everyone. Companies use more efficient and cost-effective mechanisms to spread their brands and products. Social networking is a very effective means to achieve the marketing goals of many economic agents. Even in politics, culture and sport it is essential to use the digital medium to make yourself known. In this work we studied how in a well-known and generalized social network, Facebook, information is disseminated. We realize that there are more important nodes in the network and as such are faster means of information dissemination. As we can see in tables 6 and 7, in SI and SIR models the number maximum of infected nodes are bigger if the diffusion stars in nodes with greater betweenness. If the processes begin in nodes with lower degree didn't exist diffusion of information.

	random 5%	highest degree	greater eigenvector	greater betweenness	lower degree
SI	75%	90%	90%	90%	0%
SIR	45%	62%	63%	65%	0%
IC	0%	0%	0%	0%	0%

Table 6 - Max of % nodes infected in 200 interactions

	random 5%	highest degree	greater eigenvector	greater betweenness	lower degree
SI	NA	NA	NA	NA	NA
SIR	35%	50%	45%	50%	0%
IC	15%	25%	25%	30%	0%

Table 7 - Max of % nodes removed in 200 interactions

This study focused on the application to the database collected from three models of information diffusion. It would be possible to replicate and see their behavior with as many models. The NDLIB tool allows to do this study. It would also be possible to change many more elements of each model, with different initial values for both nodes and edges. For example, the passage from a susceptible to infected node is only possible from a female node, or older than 18 years. If our network has this information about nodes, we can customize the templates and check how broadcasting behaves on the network. Other interested analysis is dynamics networks. Social networks as rapidly changing topologies (Milli et al., 2018) so could be interesting studying this changings and how is influence the diffusion of information.

9 References

Birke, D. (2013). *Social Networks and their Economics: Influencing Consumer Choice*. *Social Networks and their Economics: Influencing Consumer Choice*. https://doi.org/10.1002/9781118699638

Blondel, V. D., Guillaume, J. L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, *2008*(10). https://doi.org/10.1088/1742-5468/2008/10/P10008

Brandes, U. (2001). A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, *25*(2), 163–177. https://doi.org/10.1080/0022250X.2001.9990249

Chen, W., Lakshmanan, L. V. S., & Castillo, C. (2013). Information and Influence Propagation in Social Networks. *Synthesis Lectures on Data Management*, *5*(4), 1–177. https://doi.org/10.2200/S00527ED1V01Y201308DTM037

David Kempe, Jon Kleinberg, E. T. (2002). Maximizing the Spread of Influence through a Social Network, 198. Retrieved from http://www.cs.cornell.edu/home/kleinber/kdd03-inf.pdf

Diestel, R. (2000). *Graph Theory - Diestel* (Vol. 173).

Freeman, L. C. (1978). Centrality in social networks conceptual clarification. *Social Networks*, *1*(3), 215–239. https://doi.org/10.1016/0378-8733(78)90021-7

Gama, J., Carvalho, A. P. de L., Facelli, K., Márcia, L., & Lorena, A. C. (2012). Extração de Conhecimento de Base de dados - Data Mining. *Extração de Conhecimento de Base de Dados - Data Mining*, 236-242;285-311.

Li, Y. M., Lai, C. Y., & Chen, C. W. (2011). Discovering influencers for marketing in the blogosphere. *Information Sciences*, *181*(23), 5143–5157. https://doi.org/10.1016/j.ins.2011.07.023

Milli, L., Rossetti, G., Pedreschi, D., & Giannotti, F. (2018). Diffusive phenomena in dynamic networks: A data-driven study. *Springer Proceedings in Complexity*, (219279), 151–159. https://doi.org/10.1007/978-3-319-73198-8_13

Phillip Bonacich. (2012). Power and Centrality: A Family of Measures. *American Journal of Sociology*, *92*(5), 1170–1182.

Rossetti, G. (2019). NDlib Documentation.

Rossetti, G., Milli, L., Rinzivillo, S., Sîrbu, A., Pedreschi, D., & Giannotti, F. (2018). NDlib: a python library to model and analyze diffusion processes over complex networks. *International Journal of Data Science and Analytics*, *5*(1), 61–79. https://doi.org/10.1007/s41060-017-0086-6

W. O. Kermack, A. G. M. (1927). A Contribution to the Mathematical Theory of Epidemics. *Mathematical Methods in the Applied Sciences*. https://doi.org/10.1002/mma.5067

Wang, Q., Jin, Y., Lin, Z., Cheng, S., & Yang, T. (2016). Influence maximization in social networks under an independent cascade-based model. *Physica A: Statistical Mechanics and Its Applications*, *444*(2), 20–34. https://doi.org/10.1016/j.physa.2015.10.020

Zafarani, R., Abbasi, M. A., & Liu, H. (2014). *Social media mining: An introduction*. *Social Media Mining: An Introduction* (Vol. 9781107018). https://doi.org/10.1017/CBO9781139088510

Zhu, T., Wang, B., Wu, B., & Zhu, C. (2014). Maximizing the spread of influence ranking in social networks. *Information Sciences*, *278*(3), 535–544. https://doi.org/10.1016/j.ins.2014.03.070