Analysis of Large Scale Social Networks
Analytics Project using R and Gephi

Annelien Morlion (r0702844)
and
Laavanya Wudali Lakshmi Narsu (r0690809)

Master in Artificial Intelligence

June 2018

Contents

1 Overview **2**

2 Centrality Measures **3**

2.1 Local Centrality Measures . . . 3

2.1.1 Degree . . . 3

2.1.2 Closeness . . . 4

2.1.3 Betweenness . . . 5

2.1.4 Eigenvector Centrality . . . 5

2.1.5 PageRank Centrality . . . 5

2.1.6 Correlations among different local centrality measures . . . 6

2.2 Global Centrality Measures . . . 7

2.2.1 Density . . . 7

2.2.2 Global Clustering Coefficient . . . 7

2.2.3 Average Degree . . . 7

2.2.4 Degree Distribution . . . 7

2.2.5 Average Path length . . . 7

2.2.6 Network Diameter . . . 7

2.3 Central Nodes using several methodologies . . . 8

3 Community Detection **9**

3.1 Clustering based on weakly connected components . . . 9

3.2 Clustering based on strongly connected components . . . 9

3.3 Clustering based on leading eigenvector of the community matrix . . . 9

3.4 Clustering based on Louvain Method . . . 10

3.5 Clustering using Gephi . . . 10

3.6 Clustering the reduced network with two different clustering methods (leading eigenvector method vs Louvain method) . . . 10

4 Appendix - R Script **11**

Chapter 1

Overview

The topic of this project is the who-trust-whom social network of Epinions.com. This consumer review site is a shopping search engine that helps compare products, prices and stores online. The members of the site can decide whether to “trust” each other and the interaction between all these trust relationships forms the “Web of Trust”. This Web of Trust is combined with review ratings to determine which reviews are shown to which user.

In this project, we use R (package iGraph) and Gephi software for analyses and visualizations. The layout algorithm used for visualisations is ForceAtlas2.

The who-trust-who social network is :

- Unweighted
- Directed
- 75879 Nodes
- 508937 Edges

Chapter 2

Centrality Measures

2.1 Local Centrality Measures

2.1.1 Degree

- Degree: range [1, 3079], mean=13.4, median=2

  Node 18 represents the user with the highest overall degree.
- In-degree: range [0, 3035], mean=6.7, median=1

  Node 18 is the node with the most incoming edges (3035), which implies that this user is trusted by a lot of other users. While this user is trusted by 3035 others, however, he/she only trusts 44 users.
- Out-degree: range [0, 1801], mean=6.7, median=1

  Node 645 is the node with the most outgoing edges (1801), which implies that this user trusts many other users. On the other hand, this user trusts 408 others (Figure 2).

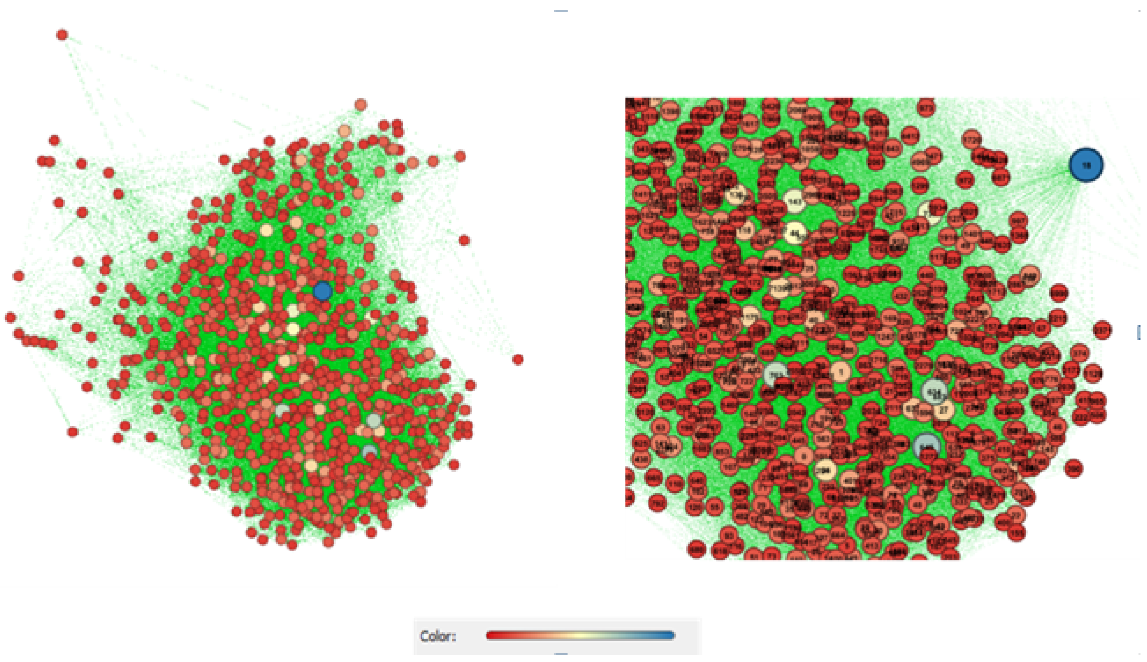


Figure 2.1: Overview of the nodes with highest degree (the nodes with degree below 210 were filtered out). The color coding for the nodes goes from red (low degree) to blue (high degree). The edges are green. Node 18 is the node with the highest degree (dark blue node).

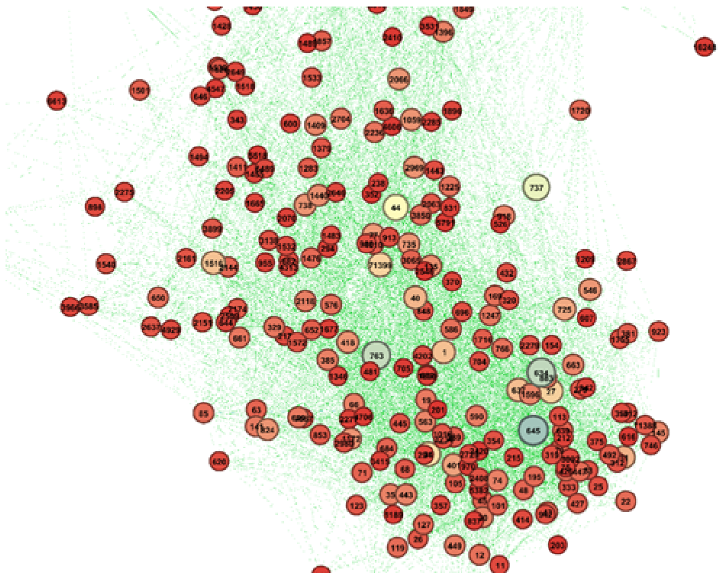


Figure 2.2: Overview of the nodes with the highest out-degree (nodes with an out-degree below 175 were filtered out). Color coding goes from red (low out-degree) to blue (high out-degree). Node 645 is the node with the highest out-degree. Visualization with Gephi (using ForceAtlas2).

2.1.2 Closeness

Closeness is a measure of how long it will take to spread information from the node to all other nodes sequentially.

- In-closeness: degree to which a node can be easily reached FROM the other nodes (i.e. using edges coming in towards the node); the inverse of the average length of the shortest paths from all the other nodes in the graph. The user represented by node 7437 has the highest in-closeness, and therefore is the node that is the most easily reachable from the other nodes.

  Range: [1.736851e-10, 6.808817e-10]

  Median: 6.784961e-10
- Out-closeness: degree to which a node can easily reach other nodes (i.e. using edges going out of the node); the inverse of the average length of the shortest paths to all the other nodes in the graph.

  Range: [1.736851e-10, 4.676034e-10]

  Median: 4.672405e-10
- All in- and out-closeness values are very small, this measurement can however be biased by the size of our network because if certain nodes cannot be reached from other nodes, R takes the total number of vertices as shortest path length and because our network is very big, this 'shortest path' - a part of the denominator for closeness - is very large).
- The fact that the median in both cases is a lot closer to the upper bound of the range indicates that most of the nodes have a closeness that is more similar to the 'best' node (upper bound of the range: a larger closeness value due to a smaller denominator in the fraction (thus smaller lengths of shortest paths)).
- We also calculated closeness with Gephi 2.3, which resulted in easier to interpret values (range: [0,1]) but still confirms the above points: the majority of the nodes have a closeness very close to 1 -> so their shortest paths to/from other nodes are small and thus the connection to/from other nodes is quite efficient. Some values, however, have a value close 0 which implies that they cannot reach or cannot be reached from some other nodes (or at least it takes a very long

time).

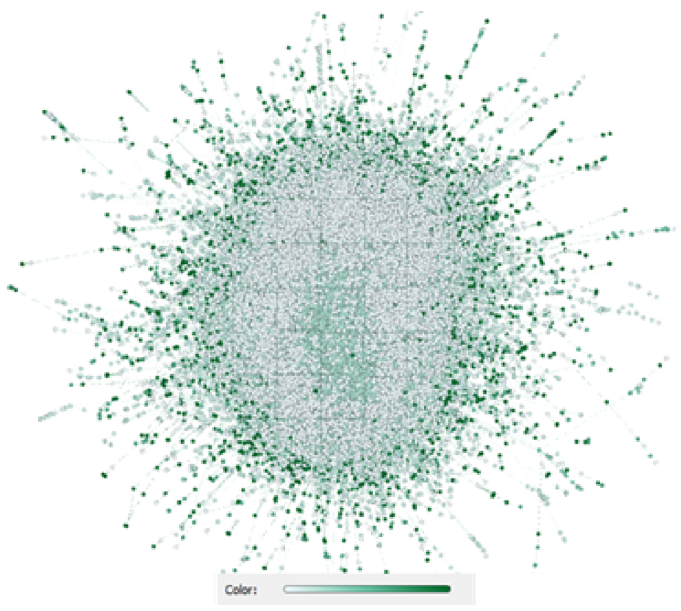


Figure 2.3: Overview of the closeness of different nodes in the network (nodes with a closeness smaller than 0.146 were filtered out for better visualization). Closeness of the nodes. Color code: lightest green (closeness=0) to darkest green (closeness=1). Visualization with Gephi (using ForceAtlas2).

2.1.3 Betweenness

- Betweenness relates to the number of times a node acts as a bridge along the shortest path between two other nodes.
- The betweenness ranges from 0 to 95831448 with a median of 0 (indicating that more than half of the nodes do not act as connectors for others, which is what we expect in this network).
- Nodes 44, 763, 634 represent the users with the highest betweenness2.4 which implies that these are the top users responsible for bridging the trust between all the members of the network.
- We also notice that there are a very nodes that have a high value of betweenness which indicates that very few members act has bridges of trust between different groups.
- There is a positive correlation (0.75) between the degree of nodes and their betweenness (2.5 left). Although this correlation is not perfect, it indicates that users with a larger degree are more likely to act as hubs for connecting other users.

2.1.4 Eigenvector Centrality

- The eigenvector centrality is a measure of the structural importance of a node proportional to the structural importance of their neighbourhood (the value is proportional to the frequency with which that node is visited during a random walk).
- There is an almost perfect positive correlation (0.94) between the eigenvector centrality and the degree (2.5 middle) which implies that the higher the degree of a node (the more direct trust connections a member has, the more is his/her influence in the Epinions Network).

2.1.5 PageRank Centrality

- A high Pagerank centrality indicates a strong influence over other nodes in the network.

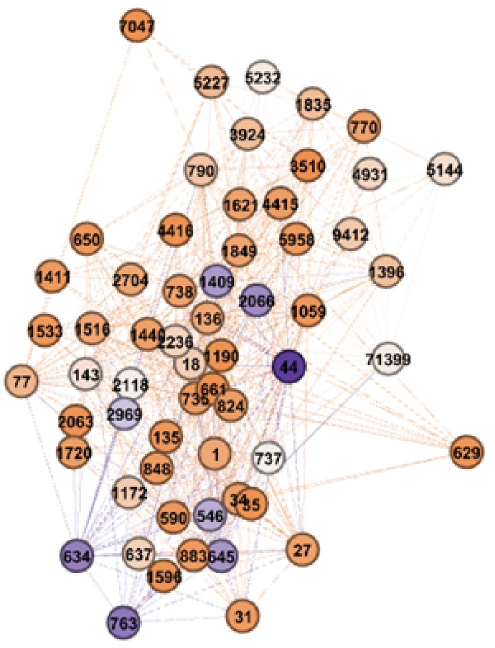


Figure 2.4: Overview of the nodes with a large Betweenness value. The nodes in dark purple (44,763, 634) are representative of the members with the highest Betweenness values. (Visualised using Gephi)

- In the Epinions network, nodes 18, 737, 118 have the highest pagerank centrality. Node 18 also has the highest eigenvector centrality, which is expected because both Pagerank and Eigenvector centrality calculate the influence of a node within its network, the difference being Pagerank takes direction into account.
- There is a positive correlation (0.85) between the PageRank and the degree shown in 2.5 right. The correlation between PageRank and the betweenness, however, is a lot lower (0.56).

2.1.6 Correlations among different local centrality measures

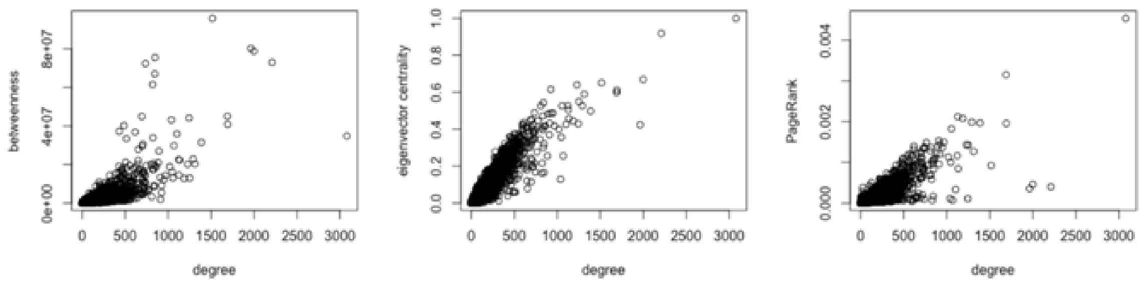
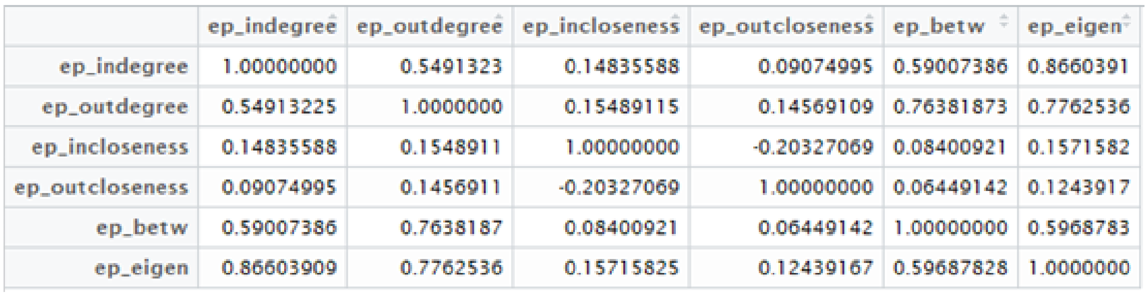


Figure 2.5: Different centrality measures plotted against the respective degree. Left: betweenness vs. degree (correlation = 0.75); Middle: eigenvector centrality vs. degree (correlation = 0.94); Right: PageRank vs. degree (correlation = 0.85). Visualization in R.



	ep_indegree	ep_outdegree	ep_incloseness	ep_outcloseness	ep_betw	ep_eigen
ep_indegree	1.00000000	0.5491323	0.14835588	0.09074995	0.59007386	0.8660391
ep_outdegree	0.54913225	1.0000000	0.15489115	0.14569109	0.76381873	0.7762536
ep_incloseness	0.14835588	0.1548911	1.00000000	-0.20327069	0.08400921	0.1571582
ep_outcloseness	0.09074995	0.1456911	-0.20327069	1.00000000	0.06449142	0.1243917
ep_betw	0.59007386	0.7638187	0.08400921	0.06449142	1.00000000	0.5968783
ep_eigen	0.86603909	0.7762536	0.15715825	0.12439167	0.59687828	1.0000000

Figure 2.6: Correlations between the different local centrality measures.

2.2 Global Centrality Measures

2.2.1 Density

Density of a network gives an idea of the number of potential connections that are actual connections in the network. If all the nodes are connected to each other, the network is dense (density is 1), if no nodes are connected the density is 0. For our network, the density is only 8.83774e-05, which indicates that the network is very sparsely connected.

2.2.2 Global Clustering Coefficient

The global clustering coefficient is the ratio of the number of closed triplets (triangles) and all triplets in the graph (open and closed) and gives an indication of clustering. In our graph, this global clustering coefficient is very low: 0.06567883.

2.2.3 Average Degree

The average degree is 13.4 (average in-degree and out-degree: 6.7059). However, this average can be misleading because of a few nodes with a very high degree (which is the case in our graph). The median for example is only 2 for the overall degree (and 1 for the in-degree and out-degree). Therefore, it is better to look at the degree distribution.

2.2.4 Degree Distribution

As can be derived from the distribution plots 2.7 and the median (2 for overall degree and 1 for in-degree and out-degree), most of the users trust very few other users. The degree distribution shape (power-law) is consistent with a scale-free network.

2.2.5 Average Path length

The average path length of this network is 4.754723.

2.2.6 Network Diameter

The network diameter of this network was found to be 16, which means that the two most distant nodes in the network are 16 hops away from each other, which is the longest shortest path in the network.

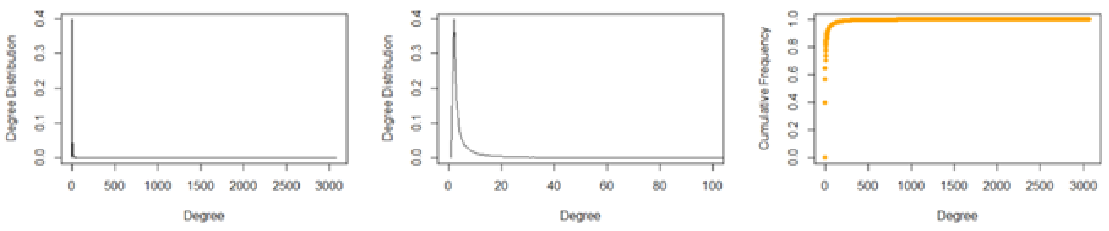
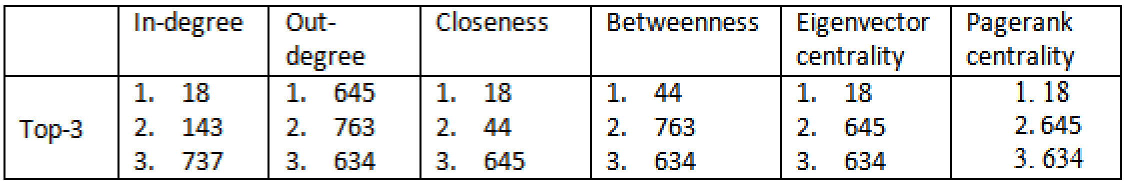


Figure 2.7: Degree distribution of the network. Left: overall degree distribution; Middle: zoomed in to the nodes with degree between 0 and 100; Right: Cumulative curve for the distribution. Most of the nodes have a very low degree (so those users trust and/or are trusted by a limited number of other users), between 1 and 10, but there are some nodes with a very high degree (up to 3079).

2.3 Central Nodes using several methodologies

We notice that different methodologies give us different central nodes. The most correlated centrality measures are overall degree, closeness,eigenvector centrality and PageRank centrality whereas betweenness seems to differ a little. This discrepancy in measures can be due to the structure of the network. For example, a node that has a high in-degree centrality (18) may not be strategically located in order to bridge connections between different groups of people which may result it having a comparatively low betweenness. But if we see nodes 763 and 634, they are the most central nodes according to out-degree and betweenness, which implies that Out degree and Betweenness have a higher correlation than In degree and betweenness.



	In-degree	Out-degree	Closeness	Betweenness	Eigenvector centrality	Pagerank centrality
Top-3	1. 18 2. 143 3. 737	1. 645 2. 763 3. 634	1. 18 2. 44 3. 645	1. 44 2. 763 3. 634	1. 18 2. 645 3. 634	1. 18 2. 645 3. 634

Figure 2.8: Top 3 central nodes using various centrality measures

Chapter 3

Community Detection

3.1 Clustering based on weakly connected components

Only two clusters are detected: the first (and largest) one contains 75877 nodes (99.997 percent of all the nodes in the network), the second contains only two nodes (user 74845 and 71749). This does not seem to be a very useful clustering.

3.2 Clustering based on strongly connected components

Here, 42176 clusters are formed of which 41112 clusters contain only 1 node and the largest cluster contains 32223 nodes (42.466 percent of all the nodes in the network). We looked more closely to this largest cluster by reducing the size of the network to the nodes that are part of this cluster 3.1. The nodes in this cluster indeed seem to be highly inter-connected.

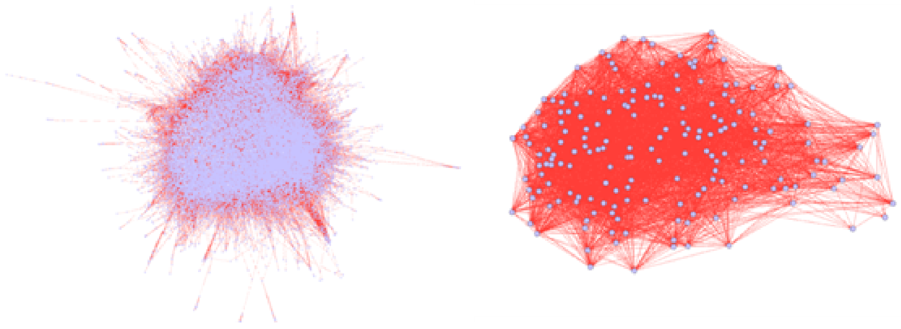


Figure 3.1: Visualization of the largest cluster (based on strongly connected components clustering). Nodes: blue; edges: red. This cluster contains 32223 nodes (left); Right: zoomed in to visualize the many connections between the nodes. Visualization with Gephi (using ForceAtlas2).

3.3 Clustering based on leading eigenvector of the community matrix

- This clustering resulted in 5 clusters with respective sizes of 15948 nodes, 2 nodes, 4182 nodes, 9512 nodes, 46235 nodes.
- This seems to be a more balanced clustering than the previous two methods where either almost all nodes ended up in one cluster (weak clustering) or with a lot of single-node clusters (strong clustering).

- Perhaps interesting to note is that users 74845 and 71749 end up together again in a 2-node cluster just like in the clustering based on weakly connected components. This indicates that those two users are in a way separated from the rest.
- The modularity of this clustering is 0.3126266. The low value implies that nodes from the different modules still have a lot of connections within each other.

3.4 Clustering based on Louvain Method

- This clustering resulted in 1629 clusters.
- the modularity of this clustering is 0.435. This has a higher modularity than the leading eigenvector method. Although we cannot say if it forms better clusters because the number of clusters are much larger than that in the leading eigenvector method.

3.5 Clustering using Gephi

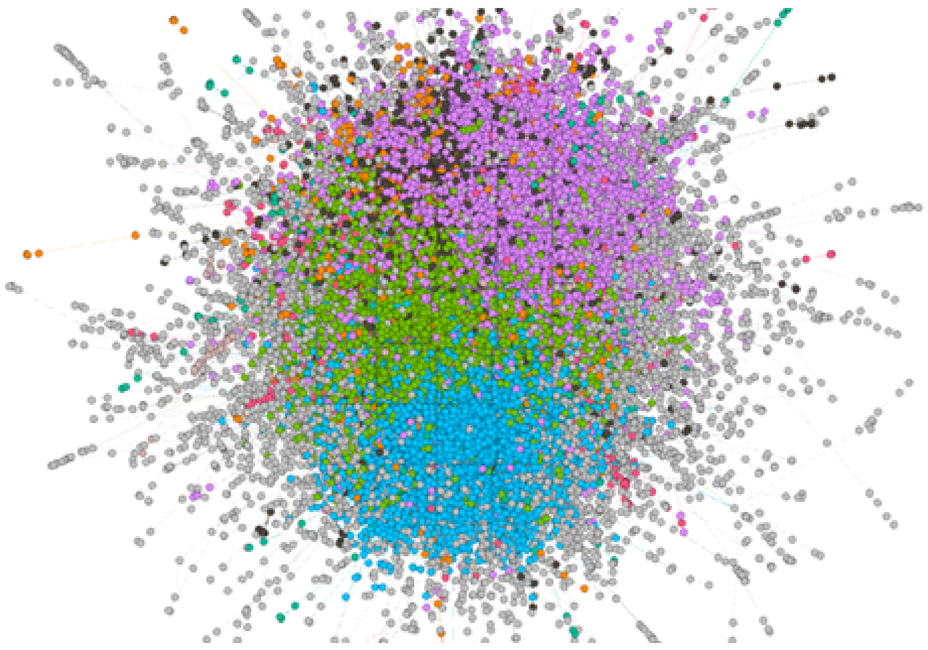


Figure 3.2: 0.441 Modularity Visualization with Gephi (using ForceAtlas2).

3.6 Clustering the reduced network with two different clustering methods (leading eigenvector method vs Louvain method)

- We elaborated the clustering of the reduced network (i.e. the network that only contains the nodes of the largest cluster based on the strongly connected clustering).
- With the leading eigenvector method, the reduced network is further separated into 5 communities (respective sizes: 6654, 3895, 3623, 16560, 1491 nodes).
- With the Louvain method, 425 clusters are formed.
- The modularity of the leading eigenvector clustering is 0.2998961, that of the Louvain clustering is 0.4251552, which implies that the Louvain clustering does a better job at finding more isolated modules than the former method of clustering.

Chapter 4

Appendix - R Script

```
rm(list=ls())
#install.packages("igraph")
library("igraph")
library("ggplot2")

### Original file (soc-Epinions1.txt from
    "https://snap.stanford.edu/data/soc-Epinions1.html") is plain text format
#data <- read.table("/Volumes/Data/Annelien/KUL-MAI/Analysis of large scale social
    networks/Project/soc-Epinions1.txt", header=F, colClasses=c("character",
    "character"))
#epinions <- graph.data.frame(data, directed=TRUE)

### For later use with Gephi and for consistency, the original file was converted into
    a pajek format using a Python script.
epinions <- read.graph("/Volumes/Data/Annelien/KUL-MAI/Analysis of large scale social
    networks/Project/soc-Epinions1.net", format="pajek")

### General info
is.directed(epinions) # Network is directed
is.weighted(epinions) # Strenght of the link: higher value=stronger link -> here,
    unweighted!

#extract Vertices
V(epinions)$id
V(epinions)[1] #0
vcount(epinions) #75879

#extract edges
E(epinions)[1:10]     # 0->4  0->5  0->7  0->8  0->9  0->10 0->11 0->12 0->13 0->14
# E(epinions)[1:10]$weight # unweighted network!
ecount(epinions)      # 508837
ends(epinions,10)     # incident vertices of some graph edges (for edge 10: 0 and 14)

head_of(epinions,10)  # 'head' (head of the arrow), 'tail' (the other end)
tail_of(epinions,10)
ends(epinions,1:10)   # incident vertices of some graph edges (here edges 1 to 10)

#Degree
ep_deg <-degree(epinions) # degree of a vertex: the number of adjacent edges of the
    vertex
#ep_indegree <- degree(epinions,mode = "in") #calculates in-degree of nodes
```

```
mean(ep_deg)  #13.4118
max(ep_deg)   #3079
min(ep_deg)   #1
median(ep_deg)#2

mean(degree(epinions,mode="out")) #6.7059
max(degree(epinions,mode="out")) #1801
min(degree(epinions,mode="out")) #0
median(degree(epinions,mode="out")) #1

mean(degree(epinions,mode="in")) #6.7059
max(degree(epinions,mode="in")) #3035
min(degree(epinions,mode="in")) #0
median(degree(epinions,mode="in")) #1

V(epinions)$name[degree(epinions,mode="in")==max(degree(epinions,mode="in"))] #To get
    the node with the highest in-degree - "18" (most prominent)
degree(epinions, v=which(V(epinions)$name == "18"), mode="in")  #3035
degree(epinions, v=which(V(epinions)$name == "18"), mode="out") #44

V(epinions)$name[degree(epinions,mode="out")==max(degree(epinions,mode="out"))] #Node
    with max out-degree - "645" - most central
degree(epinions, v=which(V(epinions)$name == "645"), mode="in")  #408
degree(epinions, v=which(V(epinions)$name == "645"), mode="out") #1801

V(epinions)$name[degree(epinions)==max(degree(epinions))] #To get the node with the
    highest overall degree - "18"

#Degree Distribution
deg.dist <- degree_distribution(epinions, cumulative=T, mode="all")  # numeric vector
    of same length as maximum degree plus one.
plot( x=0:max(ep_deg), y=1-deg.dist, pch=19, cex=1.2, col="orange",
+       xlab="Degree", ylab="Cumulative Frequency")
# first element: rel.freq.zero degree vertices, second: vertices with degree one, etc.
plot(degree_distribution(epinions),type="l",xlab="Degree", ylab="Degree Distribution")
#Degree Distribution zoom
plot(degree_distribution(epinions),type='l',xlim=c(0,100),xlab="Degree", ylab="Degree
    Distribution")


#Closeness
ep_incloseness <- closeness(epinions,mode="in",weights=NA) #inverse of the average
    length of shortest paths to get TO a given node FROM all other reachable nodes in
    the network.
range(ep_incloseness)   #1.736851e-10 6.808817e-10
#ep_incloseness_est <- estimate_closeness(epinions,mode="in",weights=NA,cutoff=3,
    normalized=TRUE)
median(ep_incloseness)  #6.784961e-10
#normalised


ep_outcloseness <- closeness(epinions,mode="out",weights=NA)# #inverse of the average
    length of shortest paths to get FROM a given node TO all other reachable nodes.
range(ep_outcloseness)  #1.736851e-10 4.676034e-10
median(ep_outcloseness) #4.672405e-10

#Betweenness
ep_betunw<-betweenness(epinions, directed=FALSE, weights = rep(1,ecount(epinions)))
```

```
ep_betw<-betweenness(epinions, directed=TRUE)
range(ep_betw)          #0 95831448
median(ep_betw)         #0
quantile(ep_betw,probs=seq(0,1,0.25))
#0      25      50      75    100
#0        0        0     12405   95831448
cor(ep_deg, ep_betw)    #0.7546928
plot(ep_deg, ep_betw, xlab="degree",ylab="betweenness")

#Eigenvector Centrality
epinions_undirected <- as.undirected(epinions, mode='collapse') #Eigenvector centrality
    gives greater weight to a node the more
# it is connected to other highly connected nodes
# node's network importance.
ev_obj_ep <- evcent(epinions_undirected)
ep_eigen <- ev_obj_ep$vector
range(ep_eigen)   #0 1
mean(ep_eigen) #0.005998245
median(ep_eigen)  #0.0003800636
quantile(ep_eigen,probs=seq(0,1,0.25))
#0           25          50          75          100
#0.000000e+00 3.968753e-05 3.800636e-04 2.094332e-03 1.000000e+00
cor(ep_deg, ep_eigen) #0.9373242
plot(ep_deg, ep_eigen, xlab="degree",ylab="eigenvector centrality")

#Page Rank
ep_prk <- page.rank(epinions, directed=TRUE)
ep_pagerank <- ep_prk$vector
range(ep_pagerank)   #2.758026e-06 4.535031e-03
median(ep_pagerank)  #3.649017e-06
mean(ep_pagerank) #1.317888e-05
quantile(ep_pagerank,probs=seq(0,1,0.25))
#0           25           50           75          100
#2.758026e-06 2.758026e-06 3.649017e-06 7.095023e-06 4.535031e-03

cor(ep_deg, ep_pagerank) #0.853307
cor(ep_betw, ep_pagerank) #0.5590658
plot(ep_deg, ep_pagerank, xlab="degree",ylab="PageRank")
plot(ep_pagerank,ep_betw, xlab="PageRank", ylab="betweenness")


#Table with all centralities for all vertices
centrality_epinions <- data.frame(V(epinions)$name, ep_indegree, ep_outdegree,
    ep_incloseness, ep_closeness, ep_outcloseness, ep_betw, ep_eigen, ep_pagerank)

#Highest In-degree nodes
centrality_epinions[order(-centrality_epinions$ep_indegree),] #Top actors - 18(3035),
    143(1521), 737(1317)

#Highest Out-degree nodes
centrality_epinions[order(-centrality_epinions$ep_outdegree),] #Top actors - 645(1801),
    763(1669), 634(1621)

#Highest In-closeness nodes
centrality_epinions[order(-centrality_epinions$ep_incloseness),] #Top actors - 7437,
    62983, 31013

#Highest Out-closeness nodes
```

```r
centrality_epinions[order(-centrality_epinions$ep_outcloseness),] #Top actors - 69199,
    69200, 69201

#Highest Betweenness nodes
centrality_epinions[order(-centrality_epinions$ep_betw),] #Top actors - 44, 763, 634

#Highest eigen centrality
centrality_epinions[order(-centrality_epinions$ep_eigen),]
#Top actors - 18, 645, 634, 44... check other nodes in top 10

#Highes Pagerank centrality
centrality_epinions[order(-centrality_epinions$ep_pagerank),]
#Top actors - 18, 737, 118

# Generate a table of pairwise correlations of all centrality measures
correlations_centrality <- cor(centrality_epinions[,2:7])


#Density
graph.density(epinions) #8.83774e-05-- very sparse

#Global Clustering Coefficient
transitivity(epinions) #0.06567883

#Average Clustering Coefficient
transitivity(epinions, type = "average") #0.2605128

#Average Degree
mean(ep_deg) #13.4118
mean(ep_indegree) #6.7059
mean(ep_outdegree) #6.7059

#Average Path Length
mean_distance(epinions, directed=T) #4.754723


### Clustering
clu_ep_weak = clusters(epinions, mode="weak")
clu_ep_weak$no #only 2 clusters are formed
max(clu_ep_weak$csize) #75877
min(clu_ep_weak$csize) #2 (the second cluster only has two members, all the other nodes
    are in the first cluster)
which(clu_ep_weak$csize==max(clu_ep_weak$csize)) #cluster 1 is the largest cluster
index_largest_component=which(clu_ep_weak$csize==max(clu_ep_weak$csize))
which(clu_ep_weak$membership==index_largest_component) #which nodes are in the largest
    cluster
index_smallest_component=which(clu_ep_weak$csize==min(clu_ep_weak$csize))
which(clu_ep_weak$membership==index_smallest_component)
# Not a very useful clustering (only 2 clusters, one of which has only 2 members)

clu_ep_strong = clusters(epinions, mode="strong") #based on strongly connected
    components (i.e. nodes that are fully connected with each other)
clu_ep_strong$no #42176
max(clu_ep_strong$csize) #32223
min(clu_ep_strong$csize) #1
which(clu_ep_strong$csize==max(clu_ep_strong$csize)) #cluster 27256

test <- data.frame(numberclu=clu_ep_strong$no, sizeclu=clu_ep_strong$csize)
```

```
ggplot(test, aes(x=numberclu,y=sizeclu)) +
geom_point()
table(test$sizeclu) # A lot of clusters with only one member, 1 cluster with
    (32223/75879) 42.5% of the nodes (that means that all these nodes are fully
    connected with each other! huge!)
#1      2     3     4     5     6     7     8     9    15    32223
#41112  813   164   47    24    5     1     6     2     1     1

index_largest_component=which(clu_ep_strong$csize==max(clu_ep_strong$csize))
which(clu_ep_strong$membership==index_largest_component)

which(clu_ep_strong$csize==2)
sizes(clu_ep_strong)[sizes(clu_ep_strong)==2]

modularity(epinions_undirected, membership(clu_ep_strong)) #0.0185296
modularity(epinions_undirected, membership(clu_ep_weak)) #4.929253e-06

#red_ep=delete.vertices(epinions,
    V(epinions)[clu_ep_strong$membership!=index_largest_component])
#vcount(red_ep) #32223
#ecount(red_ep) #443506

red_ep=induced.subgraph(epinions,
    which(clu_ep_strong$membership==index_largest_component))
vcount(red_ep) #32223
ecount(red_ep) #443506

write_graph(red_ep, "Reduced_graph_largestcluster.net", format = "pajek")


#Removing highest in-degree node 18 and then clustering
epinions_without18 = epinions_undirected
delete_vertices(epinions_without18,c(18))
LE_ep1=cluster_leading_eigen(epinions_without18)

### Another kind of clustering for the entire network
LE_ep=cluster_leading_eigen(epinions_undirected)
membership(LE_ep)
length(LE_ep) #5
sizes(LE_ep)
# 1     2     3     4     5
#15948     2  4182  9512 46235

sizes(LE_ep)[sizes(LE_ep)>10] #clusters with more than 10 members
modularity(epinions_undirected, membership(LE_ep)) #0.3126266

V(epinions)$color=membership(LE_ep)
plot(epinions, layout=layout_with_drl, vertex.label=NA, vertex.size=4)

#Louvain clustering on entire network
LV_ep=cluster_louvain(epinions_undirected)
length(LV_ep) #1629
modularity(epinions_undirected, membership(LV_ep))  #0.435

### Here we compare two clustering methods on the reduced network (the largest strongly
    connected cluster from above)
LE_redep=cluster_leading_eigen(red_ep)
membership(LE_redep)
```

```
length(LE_redep) #5
sizes(LE_redep)
#Community sizes
#1     2     3     4     5
#6654  3895  3623 16560  1491

sizes(LE_redep)[sizes(LE_redep)>10] #all clusters
modularity(red_ep, membership(LE_redep)) #0.2998961

LV_redep=cluster_louvain(redep_undirected)
length(LV_redep) #423
sizes(LV_redep)
modularity(red_ep, membership(LV_redep)) # 0.4251552

compare(LE_redep, LV_redep,'adjusted.rand') # 0.1211313

V(red_ep)$color=membership(LV_redep)
plot(red_ep, layout=layout_with_drl, vertex.label=NA, vertex.size=4)

V(epinions_undirected)$color=membership(LV_ep)
plot(epinions_undirected, layout=layout_with_drl, vertex.label=NA, vertex.size=4)
```