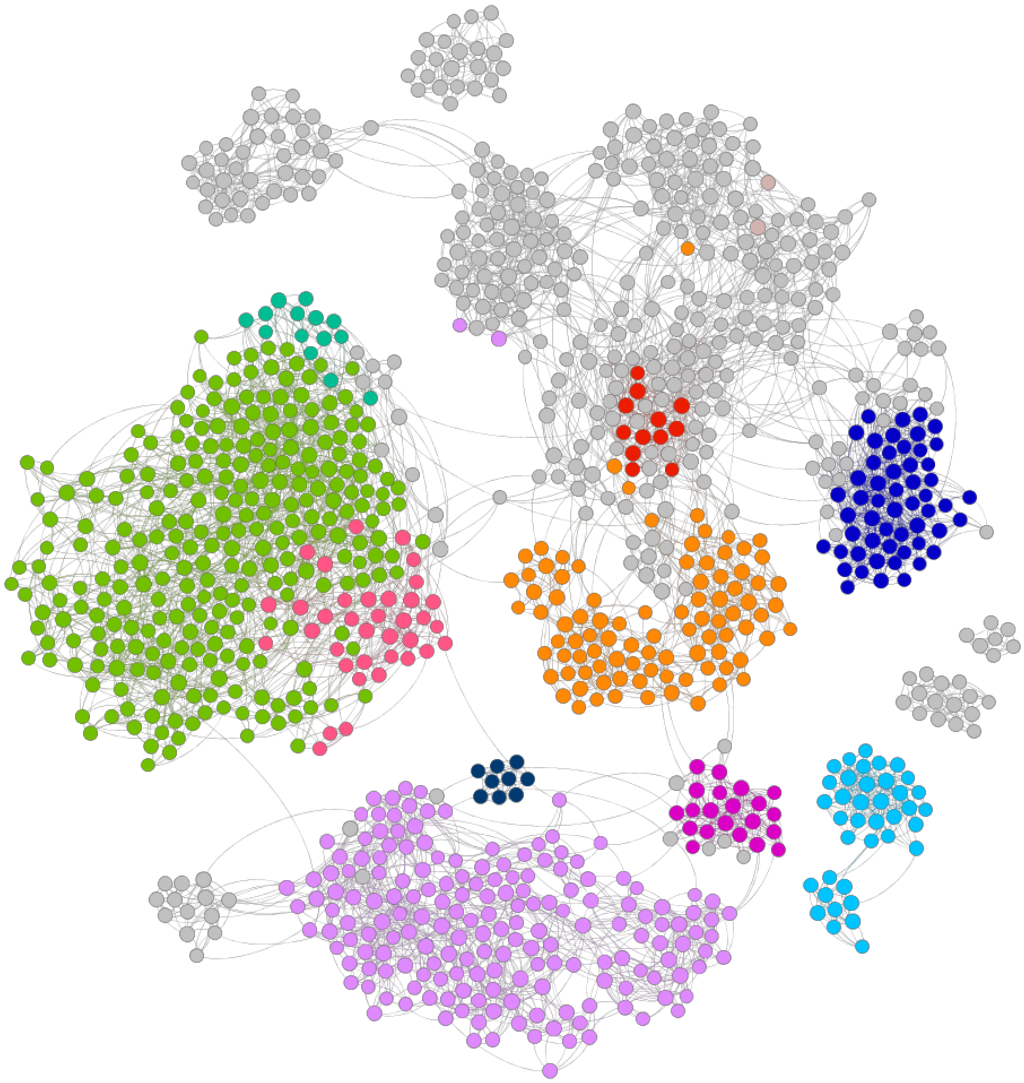
Network Analysis of Word Vectors



A Project report in "Social Network Analysis"

Abhinav Dwivedi

"569324"

Introduction

Word2vec is a group of related models that are used to produce word embeddings. These models are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words. Word2vec takes as its input a large corpus of text and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space. Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located in close proximity to one another in the space.

Based on Approximate Nearest Neighbours approach, the similarity between the words is measured. The words having similar contexts are located nearer. Using this idea, creating the network or words is easy.

In this project, for every words in vocabulary, top 20 most similar words are considered. Those words are then filtered (by the criteria defined later). The final network is then trimmed and analysed.

Data

I have used the pre-trained Word2Vec embedding model. The model was trained during my previous project in 'Text Analytics'. It's trained on 'Sentiment140' dataset with 300 dimensions. The 'Sentiment140' dataset is a collection of 1.6 million tweets.

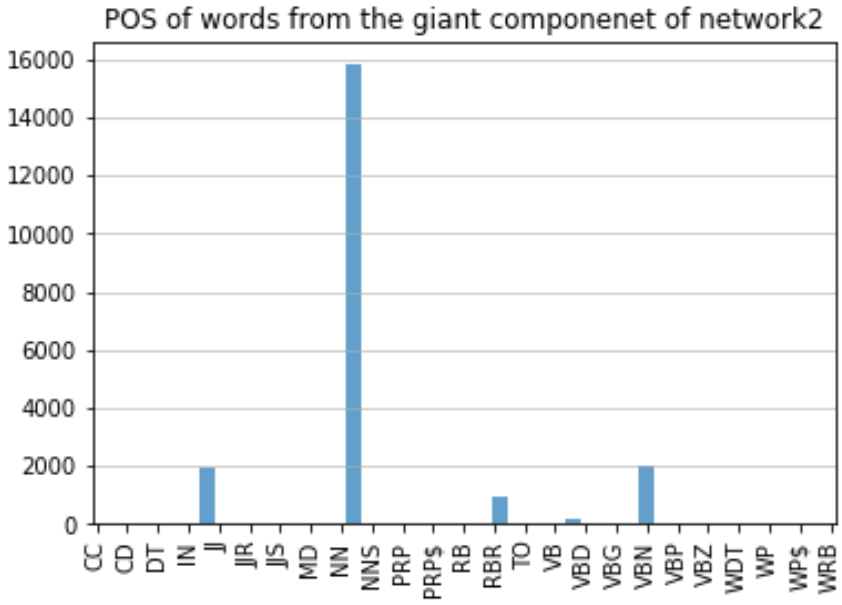
Network

The network G = (V, A) is an undirected graph.
"top20" is the list of candidates for connecting edges, are the top 20 similar words per word in vocabulary as classified by vector space. [1]
"distTop20" is the distance of respective words from the main node in the vector space.
"meanDist" is calculated by taking the mean of "distTop20". "maxDist" is the maximum distance in "distTop20".
The connections/edges are made iff the distance of the candidate word is more than "meanDist".

The weight ∈ {1,2,3}, which depends upon the breakpoint [meanDist, (meanDist+maxDist)/2, maxDist]

The network created initially, **network1**, was composed of 288538 nodes (equals the words in vocabulary) and 1913314 edges. To reduce the processing load, I decided to further trim the network.
In order to trim, I downloaded the 'words' corpus from nltk.corpus. The nodes (i.e. words from word vectors) which were not found in nltk.corpus.words were removed. This step significantly cleaned the network as it removed all the abbreviations and meaningless words (e.g. xxxxadfad, idontknow etc.). The network, **network2**, was reduced and the new size obtained was 31458 nodes and 52500 edges in total.
Not only it reduced the processor load, trimming meaningless words changed the whole distribution of the network (to be discussed later).

There was only one big connected component in the large network, consisting of ~38k nodes.
After trimming, it ended up with 8256 connected components. The biggest of them was of 20919 nodes and 46863 edges. By tagging parts of speech to all the nodes from the largest connected component, it appears that it is mostly composed of nouns (classified as NN : Noun, singular or mass by nltk pos-tagger) of them are nouns. The histogram is given below.



The diameter of the giant component obtained was 29 with average shortest path length 7.61.

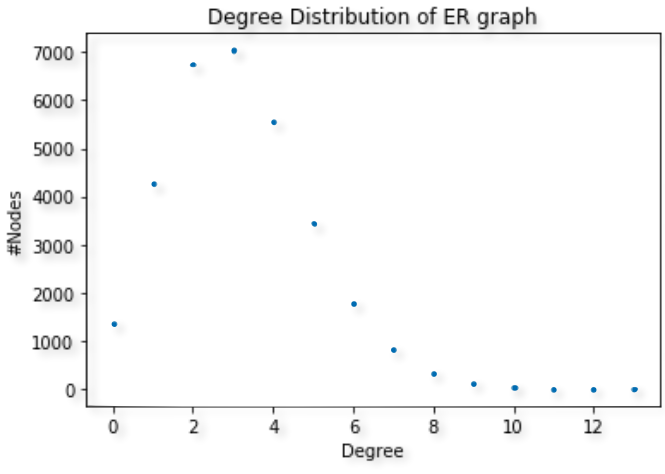
Network Metrics

The basic network metrics are given in the table below:
Network1 : Big network trained on Sentiment140 dataset.
Network2 : Created after removing words from originalNet
ER : Erdos-Renyi random graph
BA : Barbasi-Albert random graph

	Network1	**Network2**	**ER**	**BA**
Nodes	288538	31458	31458	31458
Edges	1913314	52527	49538 (p=0.0001)	125816 (m=4)
Density	4.5963	0.0001	0.0009	0.0002
Average Degree		3.342	4.12	
Average Weighted Degree		4.82	-	-
Average Clustering Coefficient	0.171370	0.0655	0.00009	0.0027
**some values are omitted due to time constraints or processor				

Degree Analysis

If we draw the degree distribution of the Network1 (original network without trimming), it seems to follow the Erdos-Renyi random distribution. In the



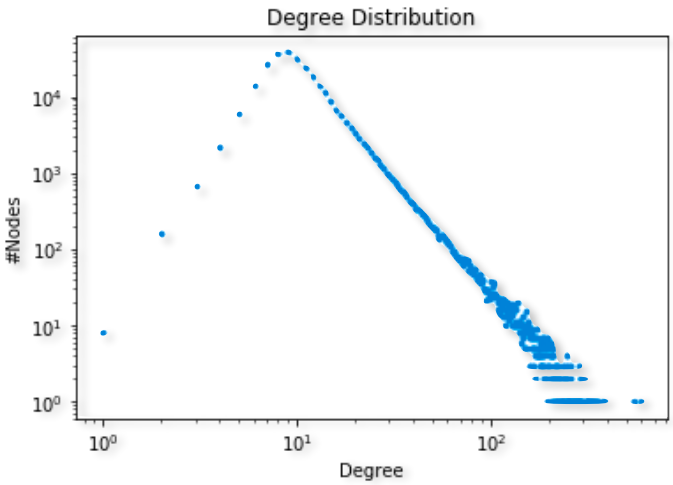
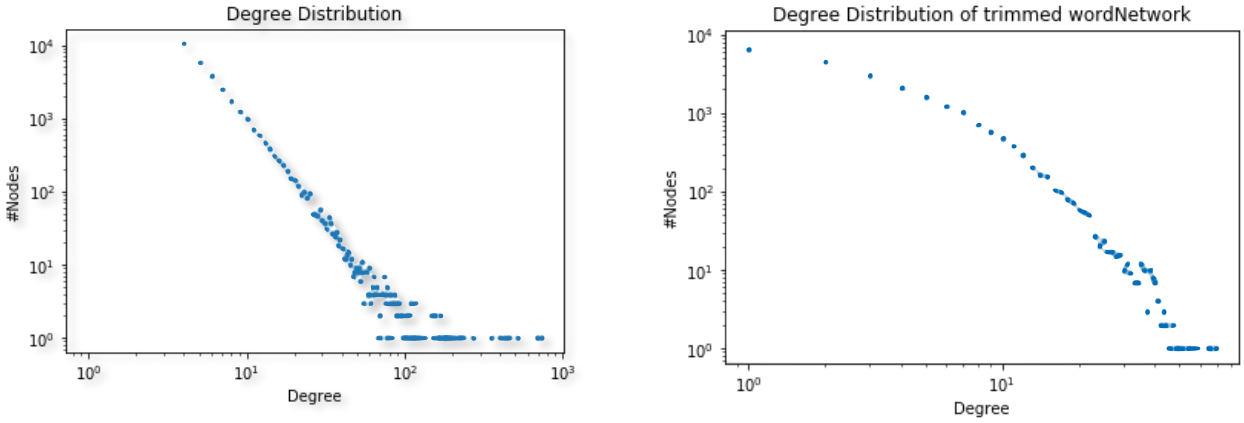


diagram above, image on the left is the degree distribution of Erdos-Renyi graph (with p=0.0001). On the right, is the degree distribution of Network2. However, when we trim the network by removing the meaningless and out-of-dictionary words, it changes to scale-free (Barbàsi-Albert). Given in the image below, the graph on the left is Barbàsi-Albert and on the right is Network2, ie. trimmed network from the above.



Given in the table below, is the list of top 10 words by their degree centrality from Network2.

Node	Degree Centrality	Degree
arm	0.0021	69
skirt	0.0020	66
Chicken	0.0020	65
Purple	0.0018	59
dog	0.0018	58
Spinach	0.0017	56
Foot	0.0017	55
Break	0.0016	54
Salad	0.0016	54
Chocolate	0.0015	50

Eigenvector Centrality

Eigenvector centrality is the measure of influence of a node in a network. Eigenvector centrality computes the centrality for a node based on the centrality of its neighbors. The eigenvector centrality for node i is

$$\mathbf{Ax} = \lambda\mathbf{x}$$

where A is the adjacency matrix of the graph G with eigenvalue λ. By virtue of the Perron–Frobenius theorem, there is a unique and positive solution if λ is the largest eigenvalue associated with the eigenvector of the adjacency matrix A.

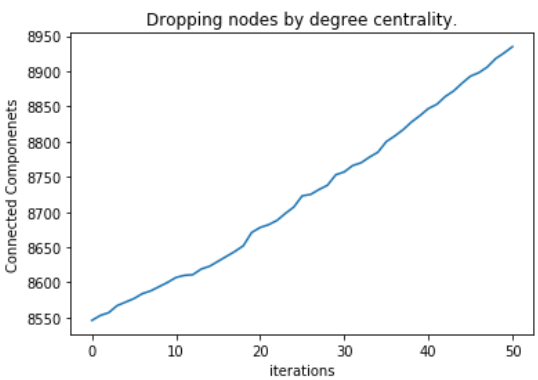
The table below shows the top 5 nodes as per Eigenvector centrality from Network2.

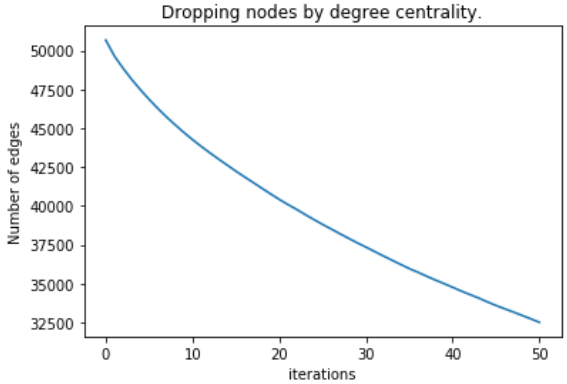
Node	Eigenvector Centrality
arm	0.33
foot	0.28
leg	0.25
shoulder	0.22
knee	0.21

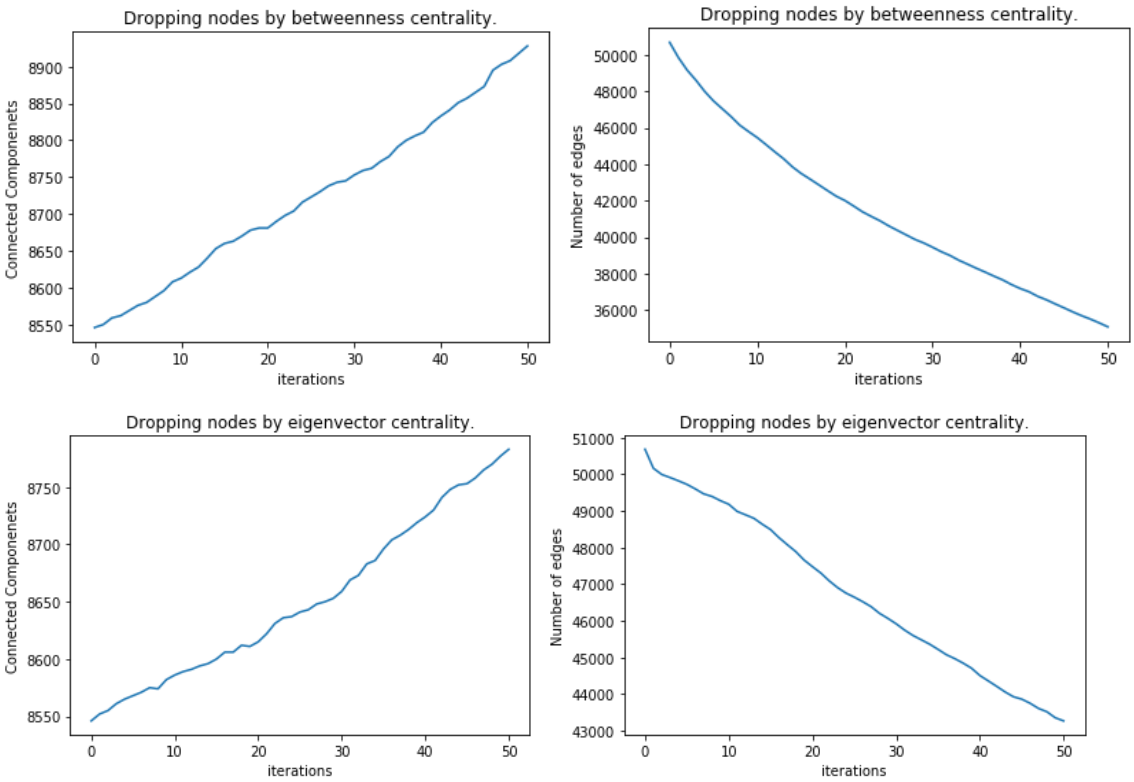
Network Resilience

In this experiment top 20 nodes were dropped per iteration for up to 50 iterations (network2). The number of connected components and number of edges of the giant subcomponent is then measured and plotted below. The nodes were dropped by decreasing order of top nodes on the basis of :

a) Degree Centrality
b) Eigenvector Rank
c) Betweenness Centrality







Community Discovery

Girvan-Newman Communities:

There are total 8547 obtained communities in network2. The biggest was them was composed of 21021 nodes. The second biggest was 34, followed by 20, 19, 14 ….
Total 8200 communities were composed of only two or only one nodes.

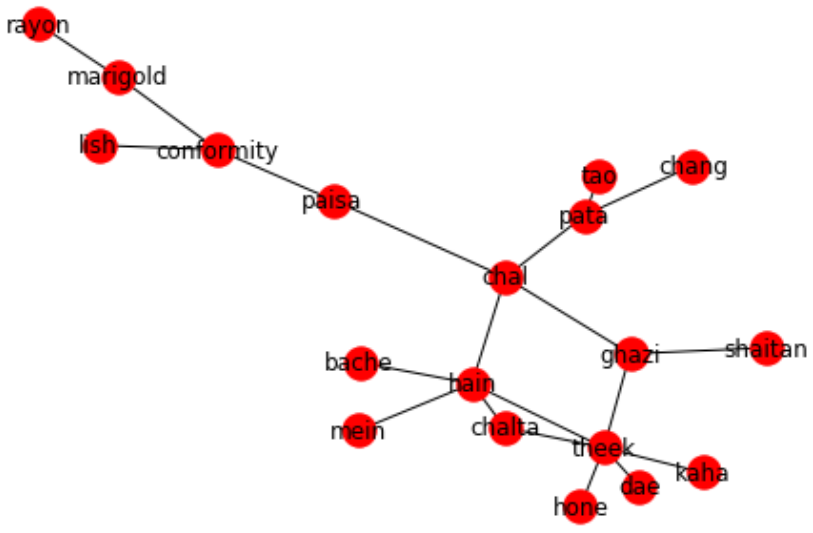


Fig. : Random Girvan-Newman community of 19 nodes. Almost all the words here are Hindi language words which are classified as similar words since they appeared in English corpus.

Louvain:

The number of communities obtained by Louvain method are 8628 with 8200 communities composed of only two or one nodes (just like Girvan Newman, only the number of bigger communities is more).

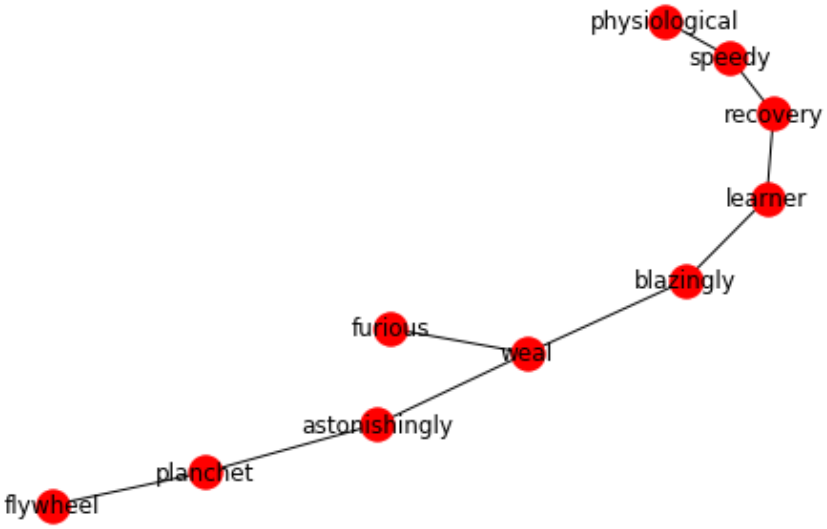


Fig. : Random louvain community of 10 nodes.

Label Propagation

The number of communities obtained with this method was 12957. The number of smaller communities out of which was 10331. The biggest community was composed of 371 nodes. Given in the image below, the distribution of top 100 communities is shown.

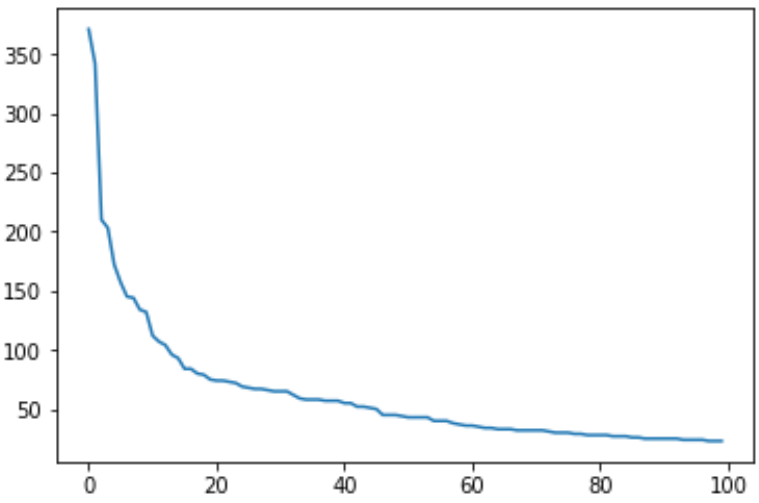


Fig. : Distribution of number of nodes of top 100 communities by label propagation.

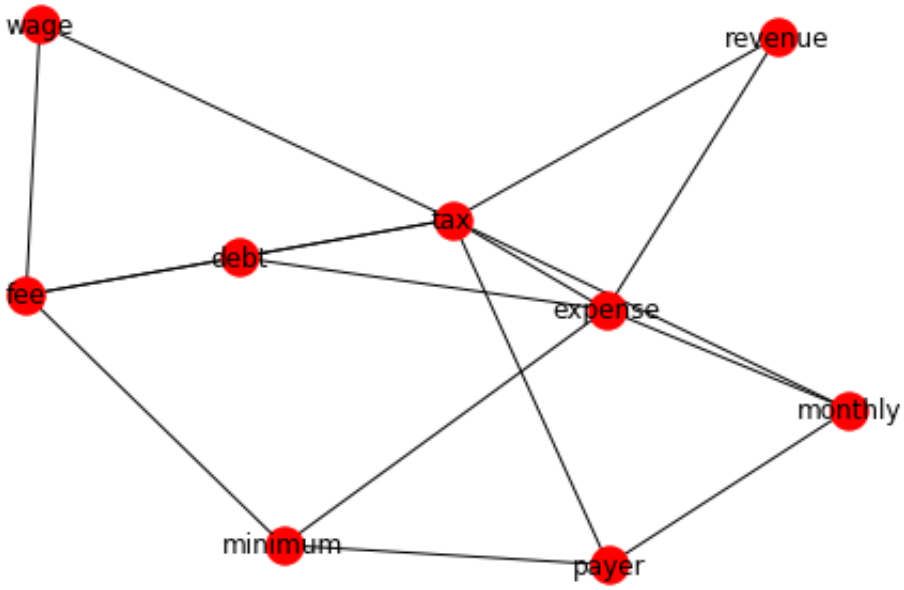
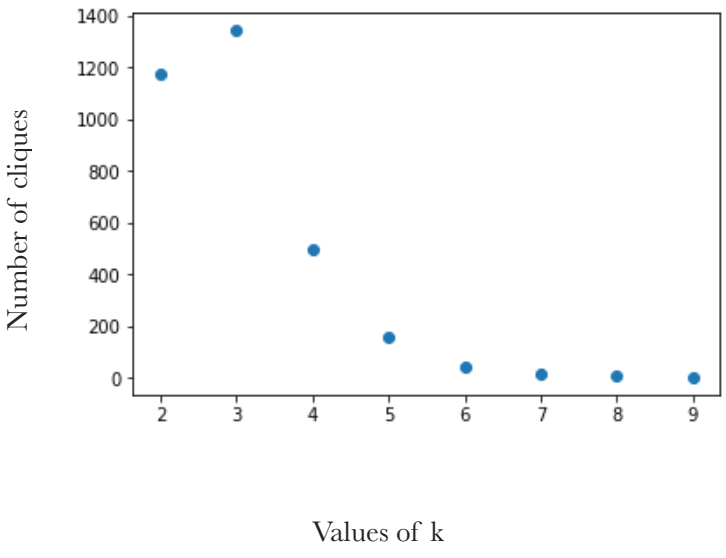


Fig. : A random community of 9 nodes obtained by label propagation.

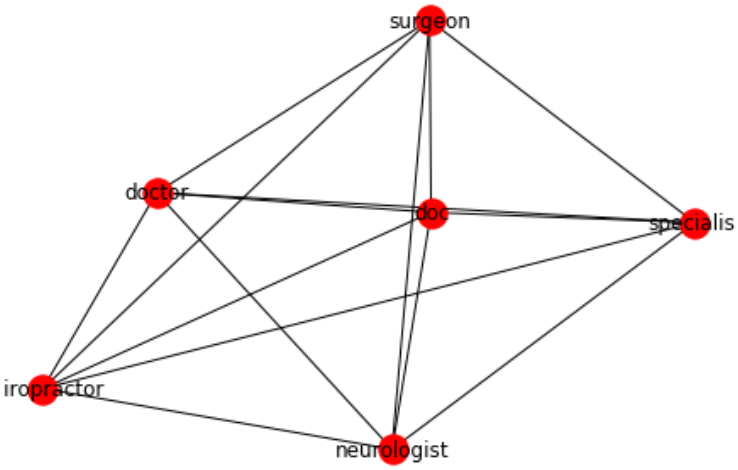
k-Cliques:

The number of cliques found wrt value of k is given in the table and plotted in the image below.



k	#cliques
2	1171
3	1342
4	494
5	155
6	44
7	17
8	6
9	3

Fig. Number of cliques by values of k



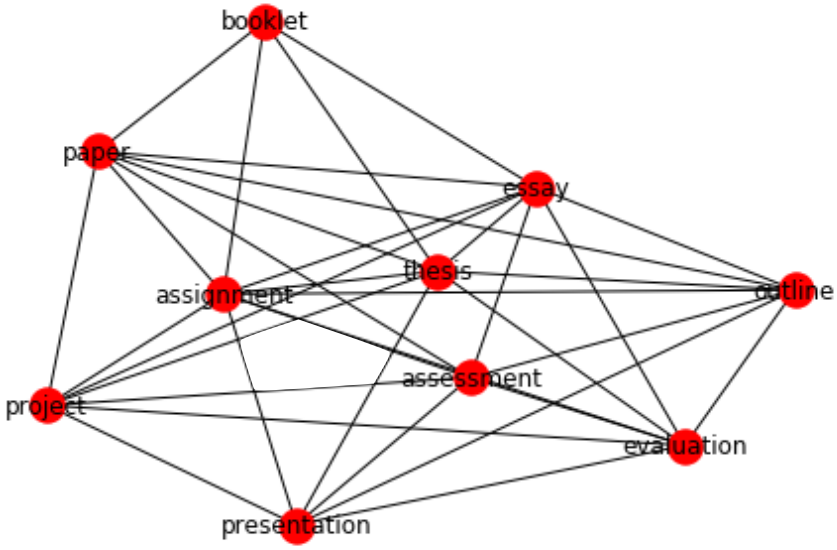
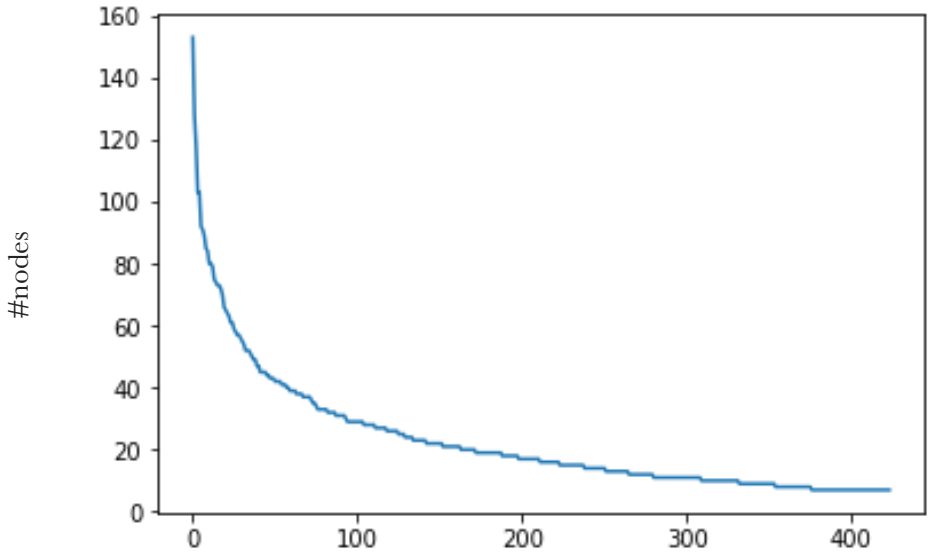


Fig. Random cliques for k=7 (top) and k=5(right)

Demon:

Demon discovered 424 communities with largest community of 153 nodes and smallest community of 7 nodes. The distribution of number of nodes per community obtained is shown in the image below.



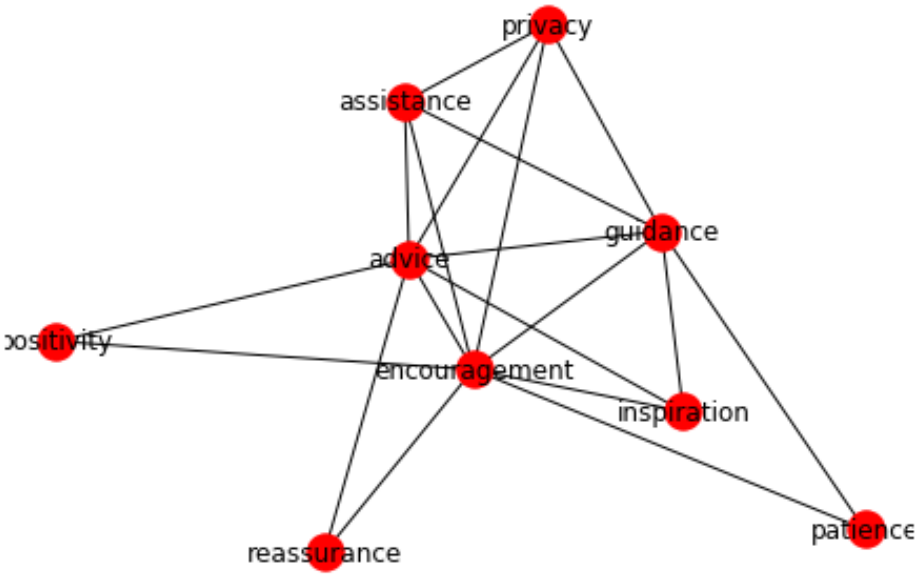


Fig. Random demon community of 9 nodes

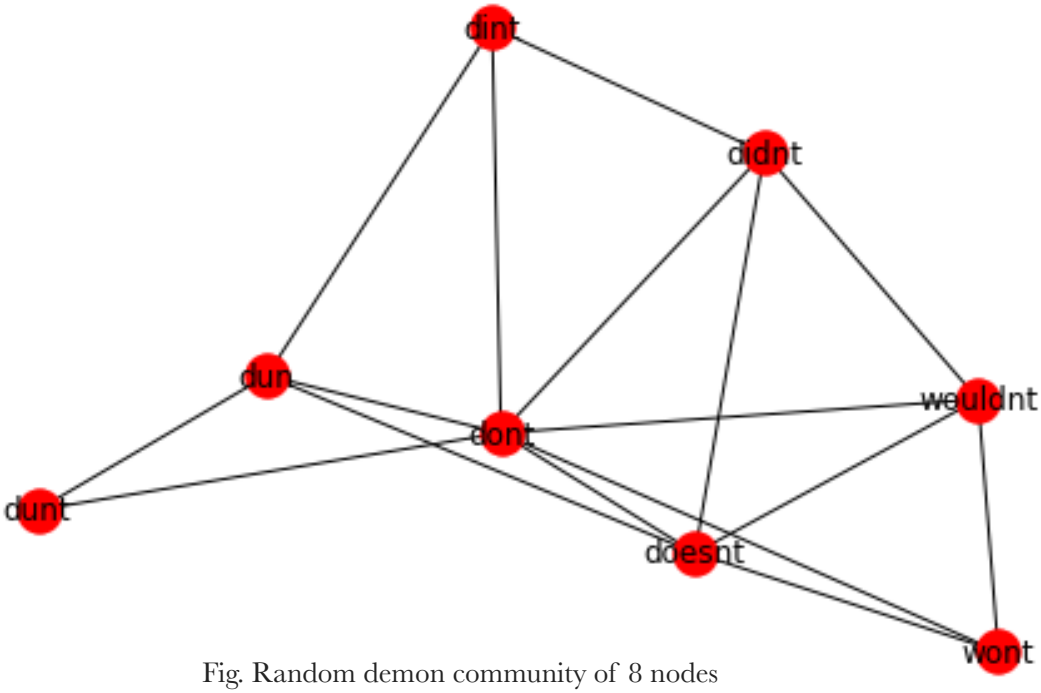


Fig. Random demon community of 8 nodes

Normalized Mutual Info Score

The normalized mutual info score is given in the table below.

	LabelPropagation	Louvain	Girvan-Newman
LabelPropagation	1	0.7902	0.6430
Louvain	0.7902	1	0.7779
Girvan-Newman	0.6430	0.7779	1

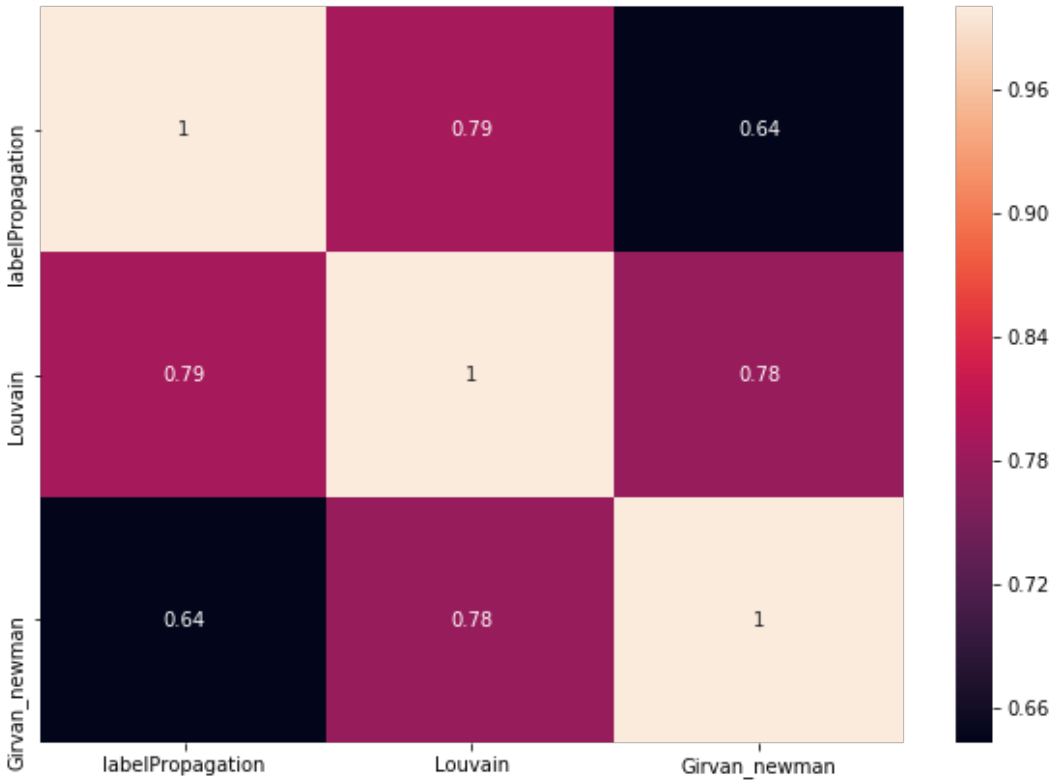


Fig. Heatmap of NMI Score

Partition Quality

The results are given in the tables below.

	min	max	avg	std
Index				
Internal Density	0.0	0.000000	0.000000	0.000000
Edges inside	0.0	0.000000	0.000000	0.000000
Average Degree	0.0	0.000000	0.000000	0.000000
FOMD	0.0	0.000000	0.000000	0.000000
TPR	0.0	0.000000	0.000000	0.000000
Expansion	0.0	69.000000	3.342644	4.571022
Cut Ratio	0.0	0.002193	0.000106	0.000145
Conductance	0.0	1.000000	0.765494	0.423690
Normalized Cut	0.0	1.000656	0.765526	0.423707
Maximum-ODF	0.0	69.000000	3.342644	4.571022

Average-ODF	0.0	69.000000	3.342644	4.571022
Flake-ODF	0.0	1.000000	0.765494	0.423690

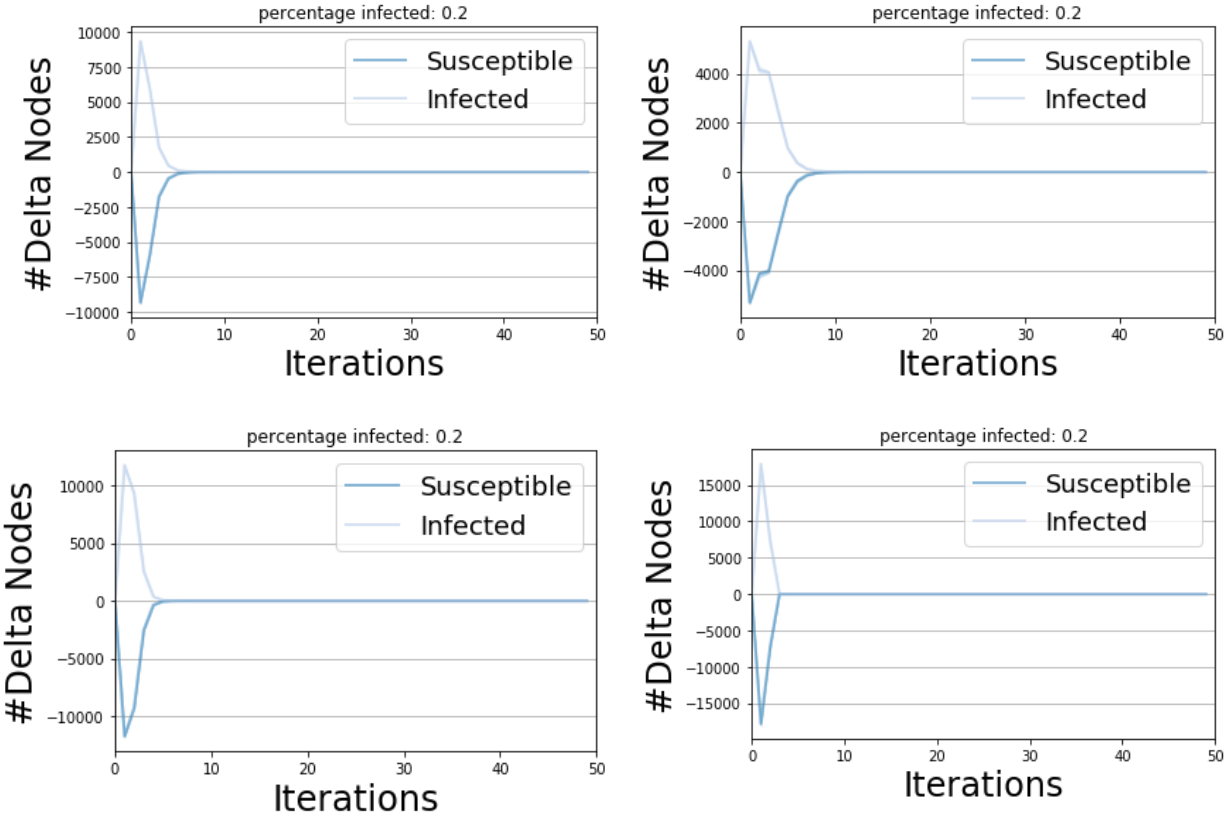
Index	value
Modularity (no overlap)	-1.110813e-07

Spreading

Threshold Model

Epidemic threshold is the critical number or density of susceptible hosts required for an **epidemic** to occur. There's also a percentage of initially infected nodes, that are the beginning of the epidemic diffusion.

The initial nodes infects the neighbouring nodes only if the infected neighbours are more than the threshold.
In this experiment, the percentage of infected nodes was kept to 20%. The result of testing the model on three networks are given in the images below.



Fig_Top Left: Network2, beta=0.05; Top Right: Network2, beta=0.005
Bottom Left: ER, beta=0.05, Bottom Right, BA, beta=0.05

SI Model

This model is characterised by two states: susceptible (S), infected (I). The infection rate is given by β. In this model the epidemic rate has an exponential growth due to the fact that all the nodes of the network are considered as susceptible, so all the nodes will be infected.

The plots are drawn below. The plots show the exponential growth in infection. In the scale-free network the growth is faster due to the presence of hubs.

The model was tested on beta 0.05 and 0.005.

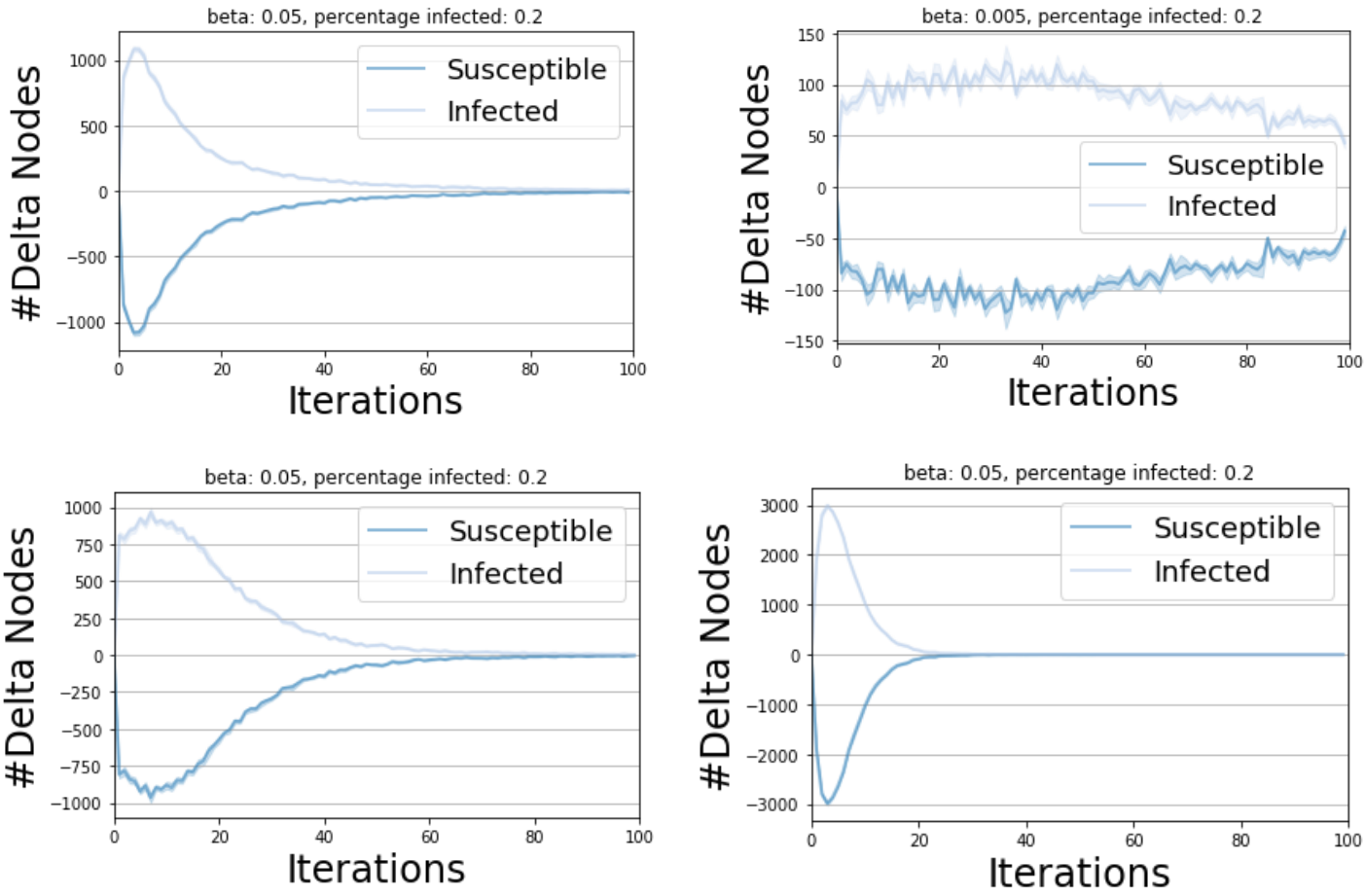


Fig.
Top left: Diffusion trend for Network2, beta=0.05
Top Right: Diffusion prevalence for Network 2, beta=0.005
Bottom Left: Diffusion trend for ER, beta=0.05
Bottom Right: Diffusion trend for BA, beta=0.05

SIS Model

This model is described by the states: susceptible infected (SI), where an infected node can also become susceptible (S) again. β here is infection rate and μ is recovery rate.
With these values the basic reproductive number λ can be calculated, the virus reproductive rate

$\lambda = \beta/\mu$, λ indicates the average number of nodes infected, generated by a single infected node in a completely susceptible society.

The plots are below along with the values of parameters.

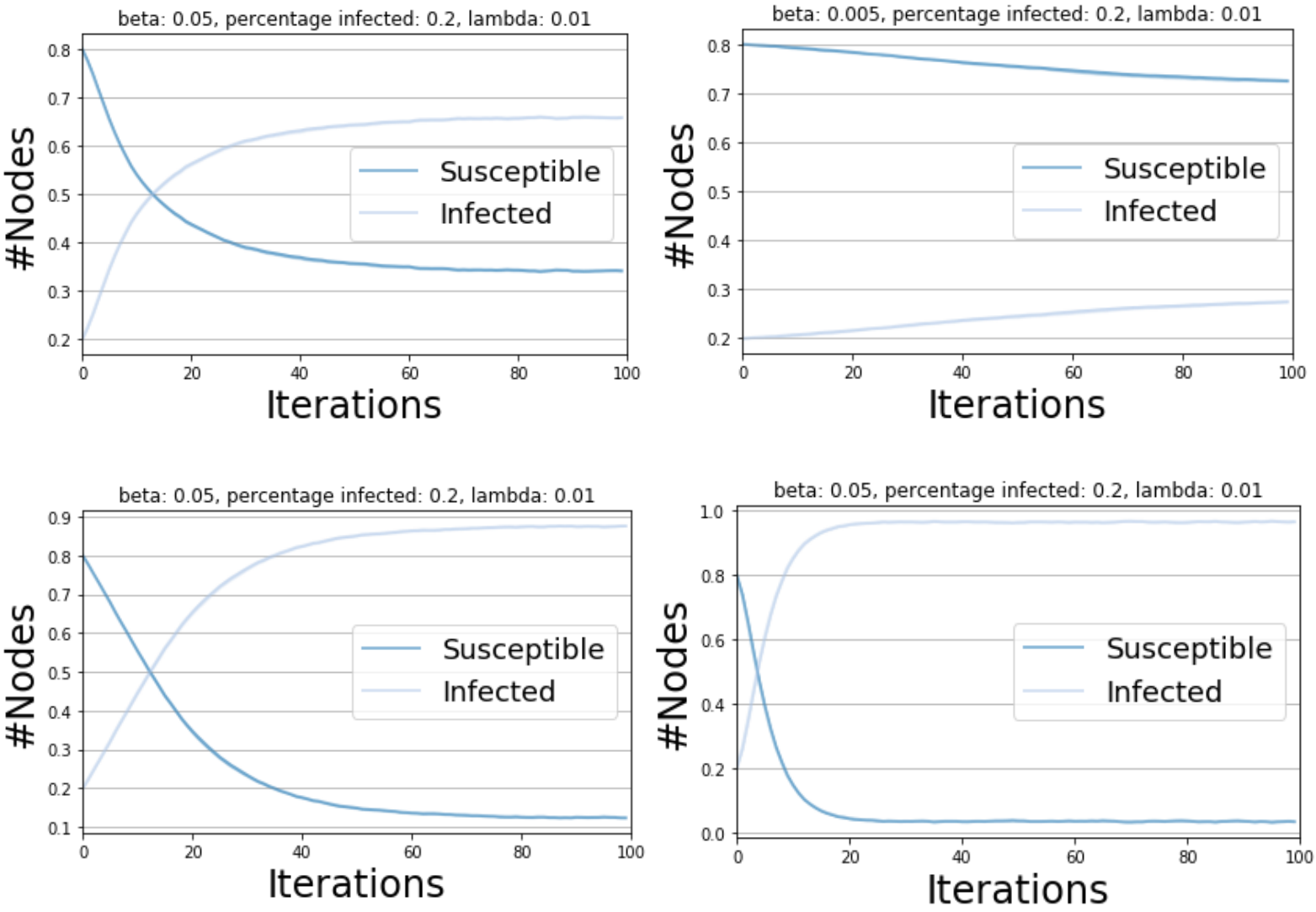


Fig.
Top left : Network2; Top Right: Network2
Bottom Left: ER
Bottom Right: BA

SIR Model

This model has three states: susceptible and infected (SI) and recovered (R). R is the state after the nodes have recovered after being susceptible and infected. Comparing the plots it can be noticed that the trend of the three curves is similar for the networks.

When the percentage of susceptible nodes decreases, then the percentage of infected nodes increases.
When the infected curve reaches the pick, it starts decreasing, letting the recovery rate curve increase.

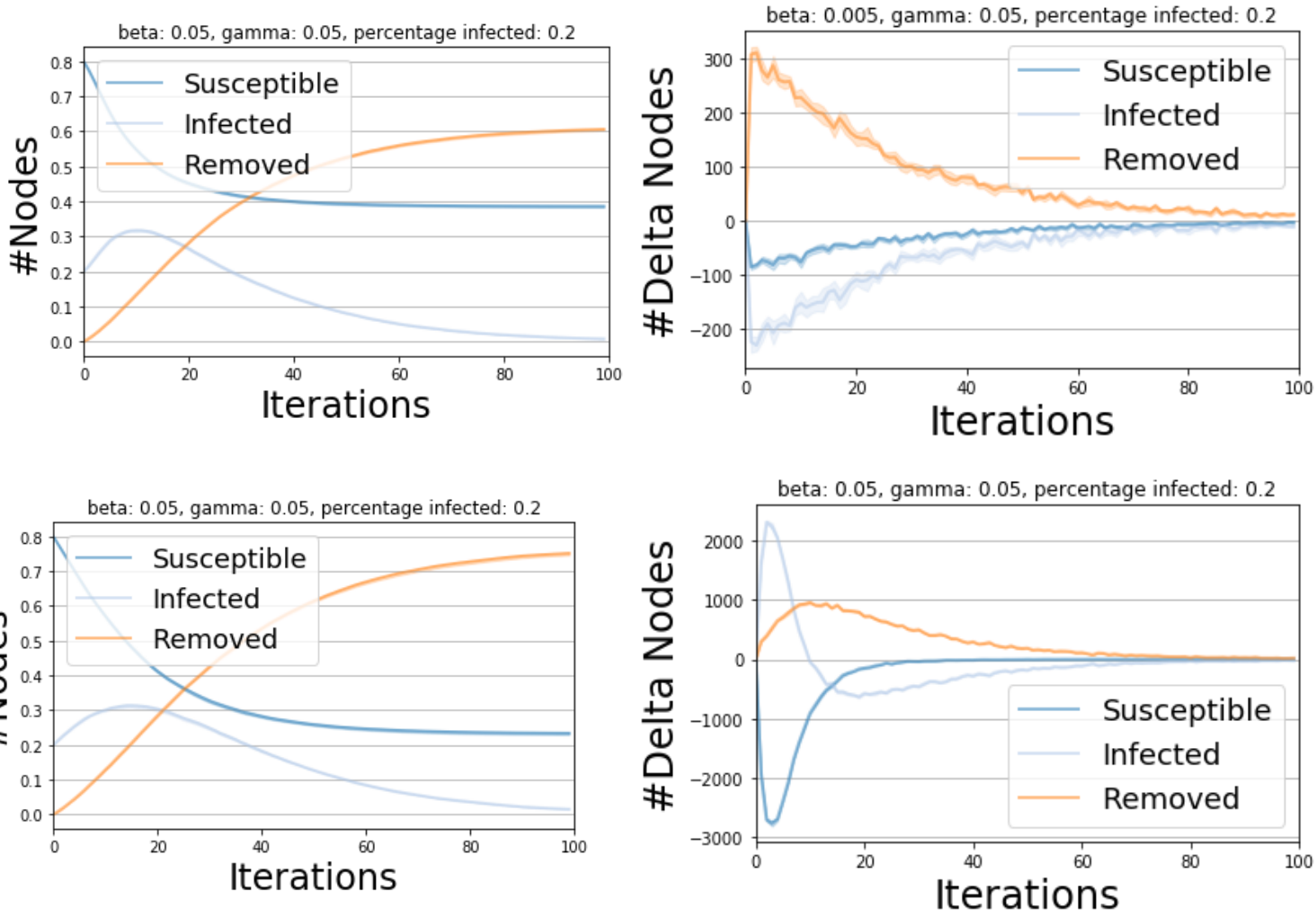


Fig.
Top Left: Diffusion trend for Network2
Top Right: Diffusion prevalence for Network2
Bottom left: Diffusion trend for ER
Bottom right: Diffusion prevalence for BA

Comparisons:

Comparisons are plotted below.

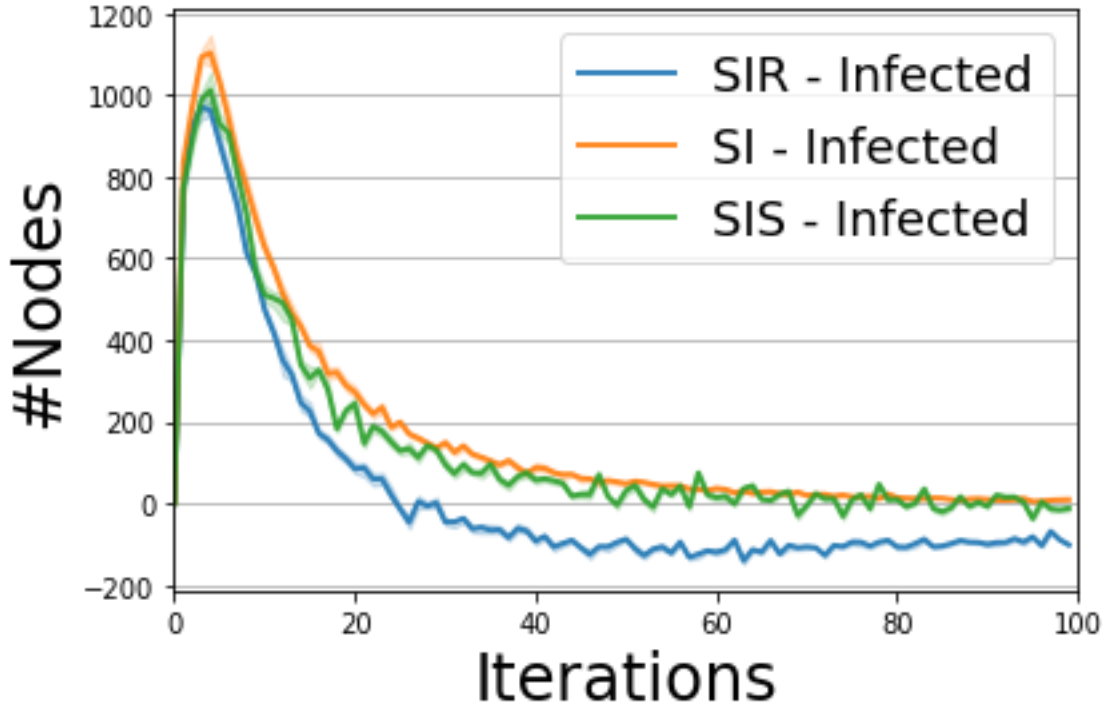


Fig. Diffusion prevalence of infected nodes in SIR, SI and SIS models

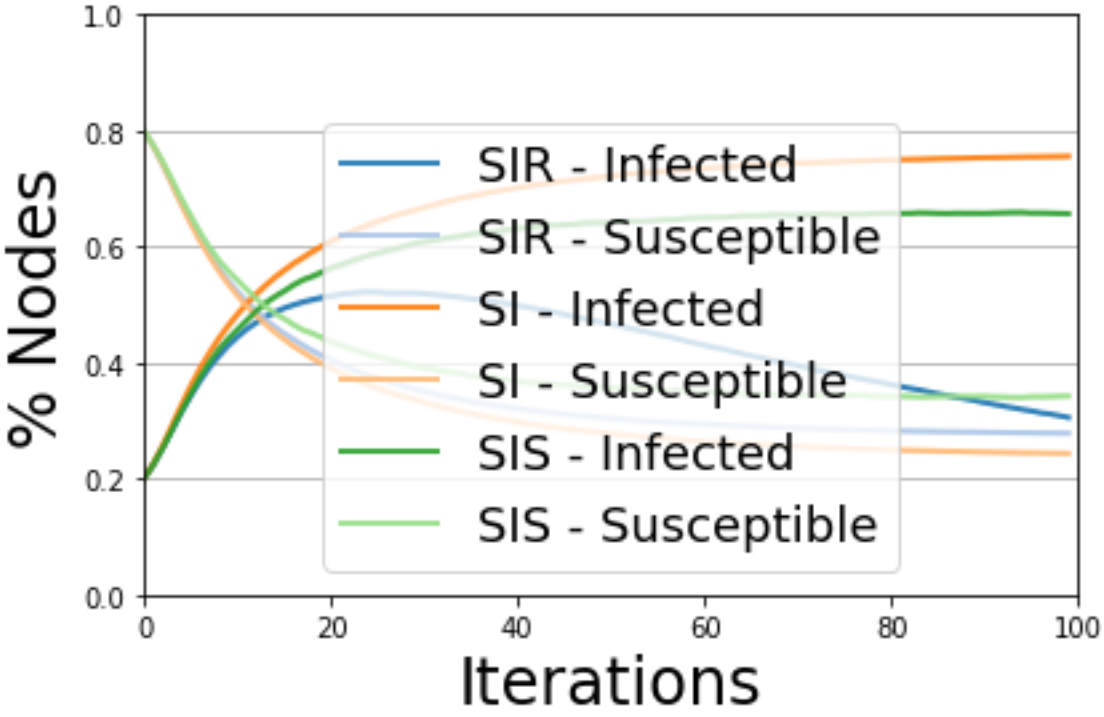


Fig. Diffusion trends in SIR, SI and SIS models

Visualizations

K-core: It's maximal subgraph of graph G in which all the nodes have degree at least k.

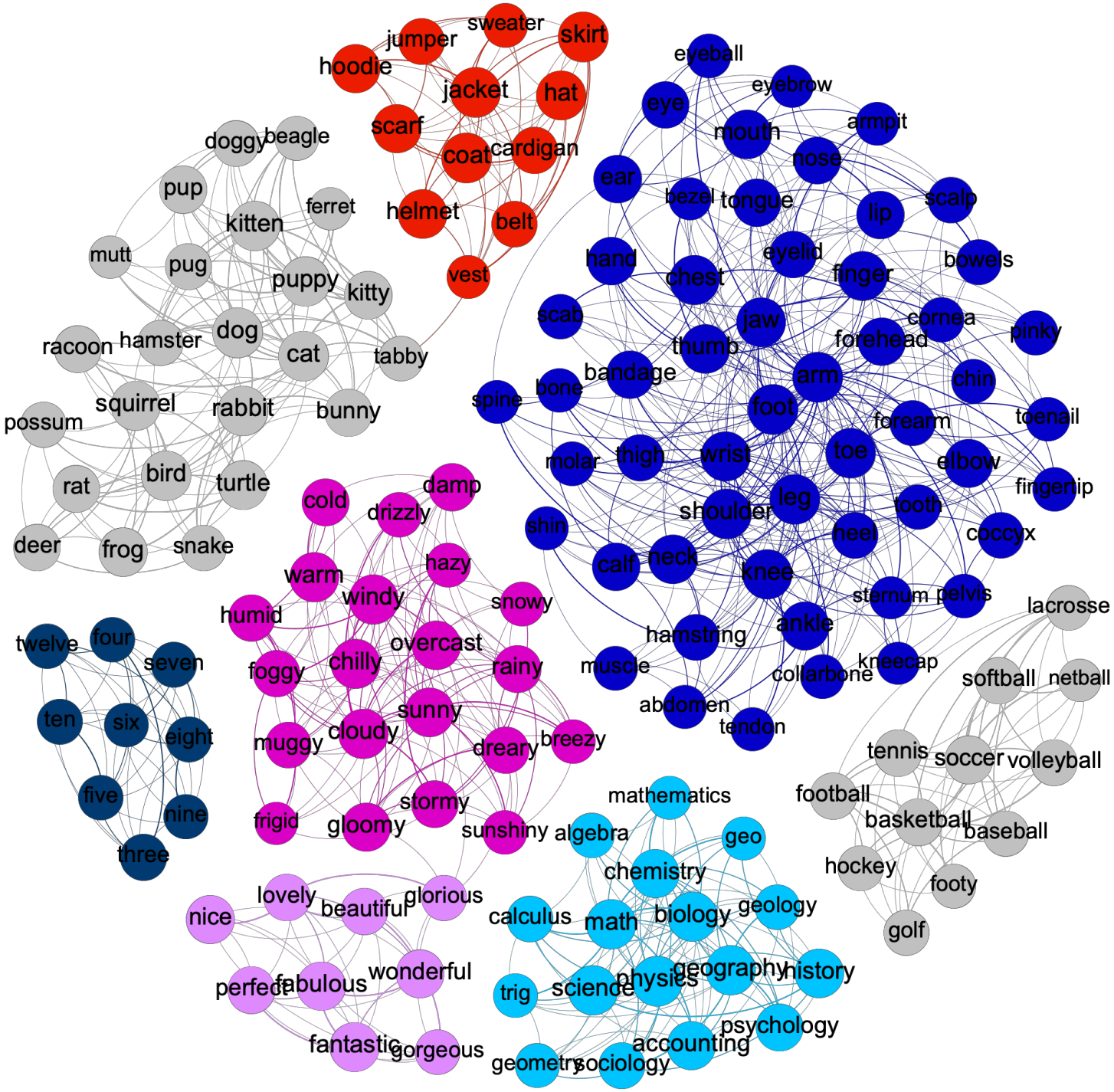


Fig. K-core of k=7

Ego networks: Ego networks consist of a focal node ("ego") and the nodes to whom ego is directly connected to (these are called "alters") plus the ties, if any, among the alters.

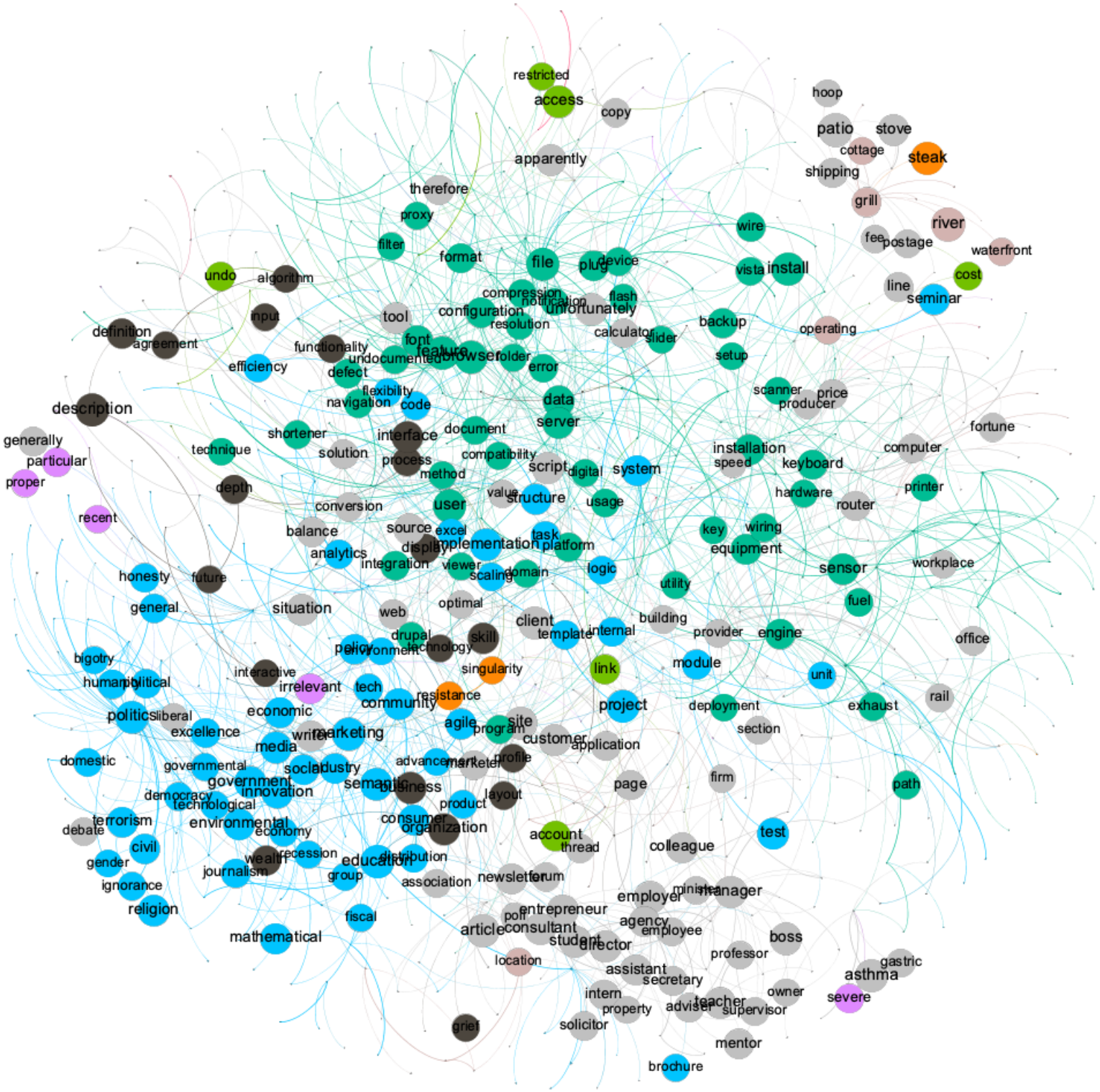


Fig. Ego network of “network” with depth 3

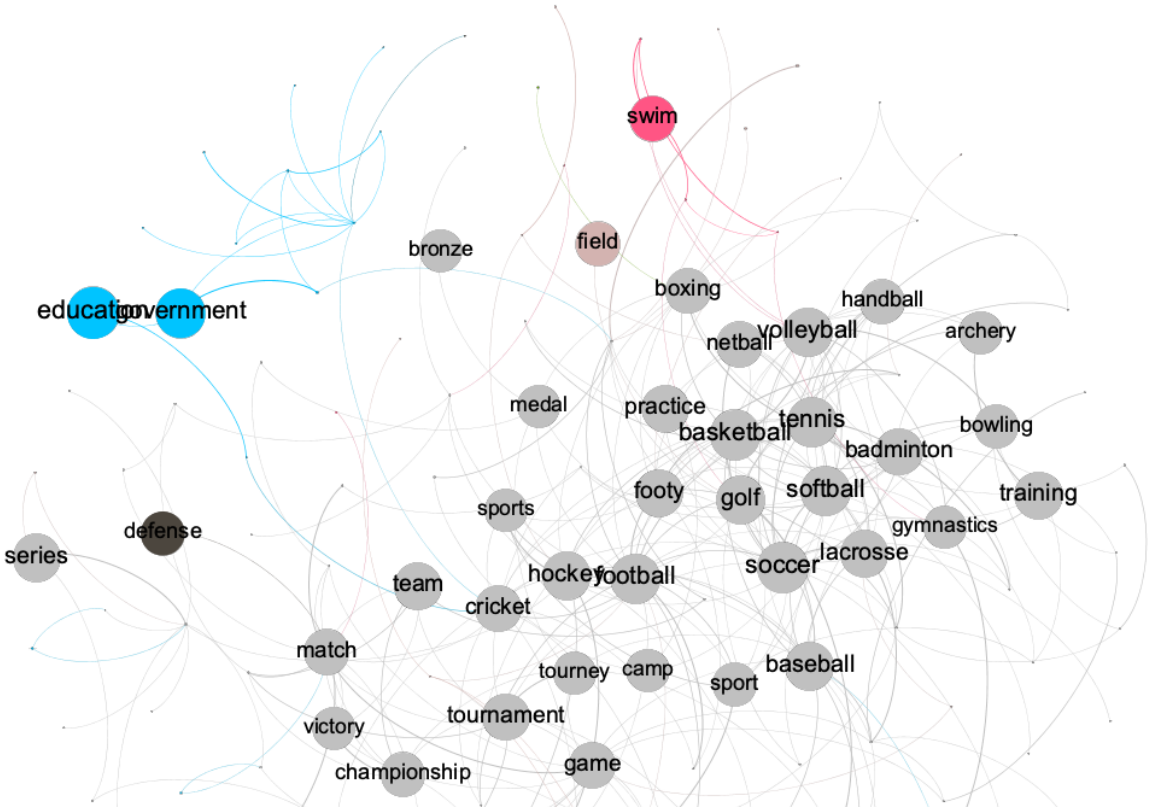
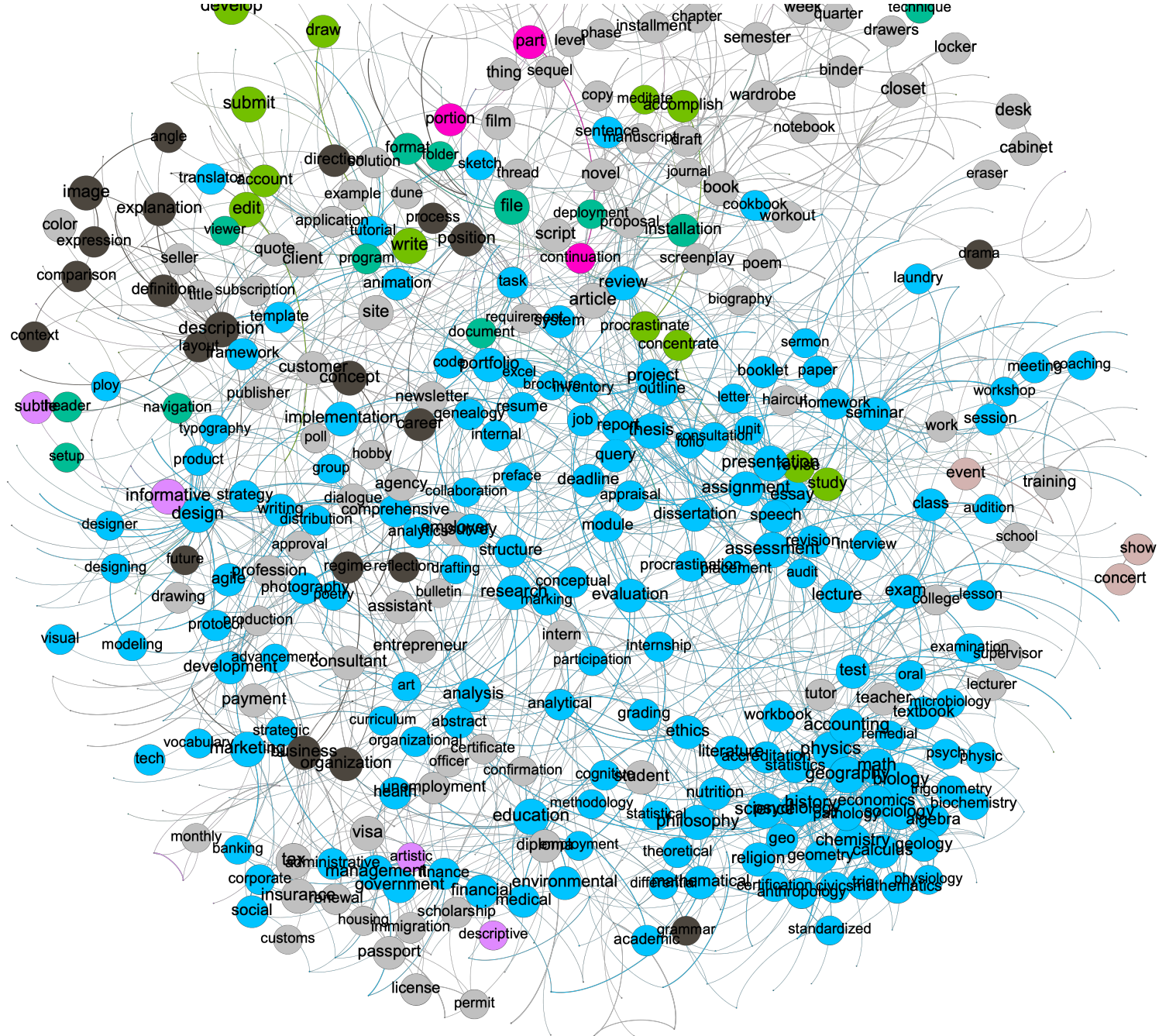


Fig. Ego network of "sports" (above) and of "thesis" (below), with depth 3



Conclusion:

Results are impressive but not that accurate since the word model was trained on reddit data which is not so informative.

Credits:

References:

Wikipedia

Social Network Analysis (Course at Università di Pisa)

Libraries and Tools:

Networkx,

Numpy,

Sklearn,

NDLib,

Pandas etc.

Gephi.

This project was submitted to Prof. Dino Pedreschi, Prof. Giulio Rossetti at Università di Pisa. This project is part of academic credits in 'Social Network Analysis' (ARS 2018).