Heidelberg University
Institute of Computer Science
Database Systems Research Group

Lecture: Complex Network Analysis

Prof. Dr. Michael Gertz

Assignment 6

Degree Correlations and Assortativity

`https://github.com/nilskre/CNA_assignments`

Team Member: Patrick Günther, 3660886,
Applied Computer Science
rh269@stud.uni-heidelberg.de

Team Member: Felix Hausberger, 3661293,
Applied Computer Science
eb260@stud.uni-heidelberg.de

Team Member: Nils Krehl, 3664130,
Applied Computer Science
pu268@stud.uni-heidelberg.de

1 Problem 6-1 The t-Party Evolving Network Model

1. tbd

2 Problem 6-2 Friendship Paradox Follow-up

problem_2

December 9, 2021

1 Lecture: Complex Network Analysis

Prof. Dr. Michael Gertz

Winter Semester 2021/22

1.1 Assignment 6 - Degree Correlations and Assortativity

Students: Felix Hausberger, Nils Krehl, Patrick Gunther

[1]:
```
import math
import networkx as nx
import pandas as pd
import numpy as np
from scipy.special import zeta
from scipy import stats
import igraph
import matplotlib.pyplot as plt
```

[2]:
```
# load dataset
raw_data = pd.read_csv('facebook-links.txt', sep="\t", header=None,␣
 ↪names=["source", "target", "timestamp"])
```

[3]:
```
# clean dataset

# 1. remove time-stamp column
data = raw_data.drop(["timestamp"], axis=1)
```

[4]:
```
# 2. Turn the directed network into an undirected network. = done by using␣
 ↪Graph() instead of DiGraph()
# 3. remove multiple edges = done by using Graph() instead of MultiGraph()
G = nx.Graph()
G.add_edges_from(data.itertuples(index=False))
```

2 1. Compute the average number of friends and the average number of "friends of friends" (FOF).

2.1 1.1 Average number of friends

A friend relationship is described by an edge between two nodes. The number of friends of one person (node) is equal to the degree of the node. The average number of friends is then the average degree $< k >$.

[5]:
```
def calc_average_number_of_friends(G):
    node_degrees = np.array(list(G.degree))
    degrees = node_degrees[:,1]
    average_degree = np.sum(degrees) / degrees.shape[0]
    return average_degree, degrees

average_degree, degrees = calc_average_number_of_friends(G)
print("Average number of friends (average degree) = {}".format(average_degree))
```

```
Average number of friends (average degree) = 25.641838351822503
```

2.2 1.2 Average number of "friends of friends" (FOF)

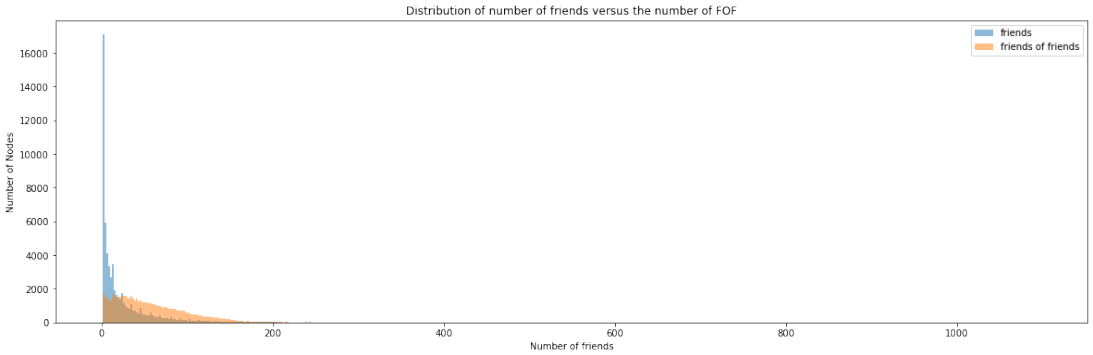
[6]:
```
def calc_average_number_of_friends_of_friends(G):
    number_friends_of_friend_per_node = []

    for node in G.nodes():
        friends = G.neighbors(node)
        friends_of_friend = []
        for friend in friends:
            friends_of_friend.append(G.degree[friend])
        average_number_friends_of_friend = np.sum(np.array(friends_of_friend)) /␣
 ↪len(friends_of_friend)
        number_friends_of_friend_per_node.
 ↪append(average_number_friends_of_friend)

    average_number_friends_of_friend = np.sum(np.
 ↪array(number_friends_of_friend_per_node)) /␣
 ↪len(number_friends_of_friend_per_node)
    return average_number_friends_of_friend, number_friends_of_friend_per_node

average_number_friends_of_friend, number_friends_of_friend_per_node =␣
 ↪calc_average_number_of_friends_of_friends(G)
print("Average number of friends of friends = {}".
 ↪format(average_number_friends_of_friend))
```

```
Average number of friends of friends = 58.3634028072132
```

3 2. Create a plot showing the distribution of number of friends versus the number of FOF.

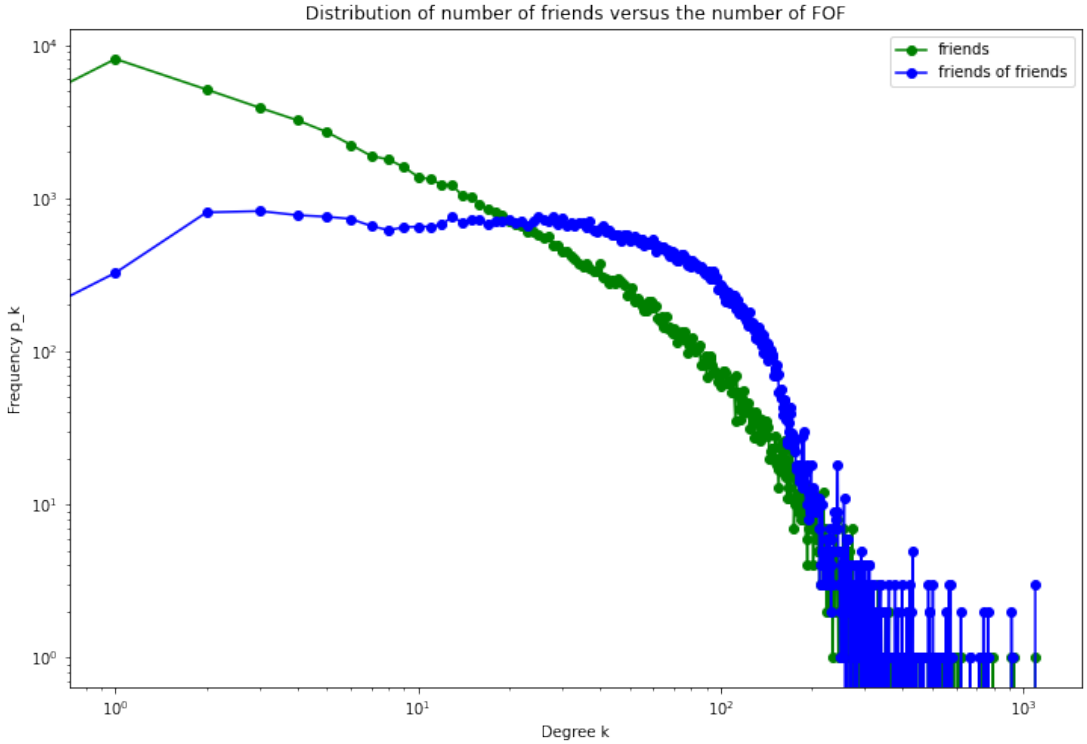
We created two plots for this. The first is a normal histogram without loglog binning. As expected it is not very meaningful. Secondly we created a plot with loglog binning and received a better plot.

[7]:
```
def distribution_friend_fof_normal(degrees, number_friends_of_friend_per_node):
    plt.figure(figsize=(20, 6))
    plt.hist(degrees, 500, alpha=0.5, label='friends')#, density=True)
    plt.hist(number_friends_of_friend_per_node, 500, alpha=0.5, label='friends␣
 ↪of friends')#, density=True)
    plt.legend(loc='upper right')

    plt.title("Distribution of number of friends versus the number of FOF")
    plt.xlabel("Number of friends")
    plt.ylabel("Number of Nodes")
    plt.show()

distribution_friend_fof_normal(degrees, number_friends_of_friend_per_node)
```



[9]:
```
def distribution_friend_fof_loglog(G, number_friends_of_friend_per_node):
    # degree_freq = how many nodes have this (index) number of links
    # number_degrees = degree number from 0 up to the maximum degree
    degree_freq = nx.degree_histogram(G)
    number_friends_of_friend_per_node_freq = np.
 ↪bincount(number_friends_of_friend_per_node, minlength=len(degree_freq))

    number_degrees = range(max(len(degree_freq),␣
 ↪max(number_friends_of_friend_per_node)))

    plt.figure(figsize=(12, 8))
    plt.loglog(number_degrees, degree_freq,'go-', label="friends")
```

```
    plt.loglog(number_degrees, number_friends_of_friend_per_node_freq,'bo-',␣
 ↪label="friends of friends")
    plt.title("Distribution of number of friends versus the number of FOF")
    plt.xlabel('Degree k')
    plt.ylabel('Frequency p_k')
    plt.legend()
    plt.show()

distribution_friend_fof_loglog(G, number_friends_of_friend_per_node)
```



4 3. Repeat the above tasks for a network based on the Barabási-Albert Model with n = 10,000 and m = 7. Discuss and compare the results with that for the Facebook network.

[38]:
```
G_barabasi_albert = nx.barabasi_albert_graph(10000,7) #63731, 13)
```

[39]:
```
print("Real network: Average number of friends (average degree) = {}".
 ↪format(average_degree))
print("Real network: Average number of friends of friends = {}".
 ↪format(average_number_friends_of_friend))
print("Real network: Number of nodes = {}".format(G.number_of_nodes()))
```

```
print("Real network: Number of edges = {}".format(G.number_of_edges()))
#print("Real network: Average clustering = {}".format(nx.average_clustering(G)))
print()
ba_average_degree, ba_degrees = calc_average_number_of_friends(G_barabasi_albert)
print("BA model: Average number of friends (average degree) = {}".
 ↪format(ba_average_degree))
ba_average_number_friends_of_friend, ba_number_friends_of_friend_per_node =␣
 ↪calc_average_number_of_friends_of_friends(G_barabasi_albert)
print("BA model: Average number of friends of friends = {}".
 ↪format(ba_average_number_friends_of_friend))
print("BA model: Number of nodes = {}".format(G_barabasi_albert.
 ↪number_of_nodes()))
print("BA model: Number of edges = {}".format(G_barabasi_albert.
 ↪number_of_edges()))
#print("BA model: Average clustering = {}".format(nx.
 ↪average_clustering(G_barabasi_albert)))
```

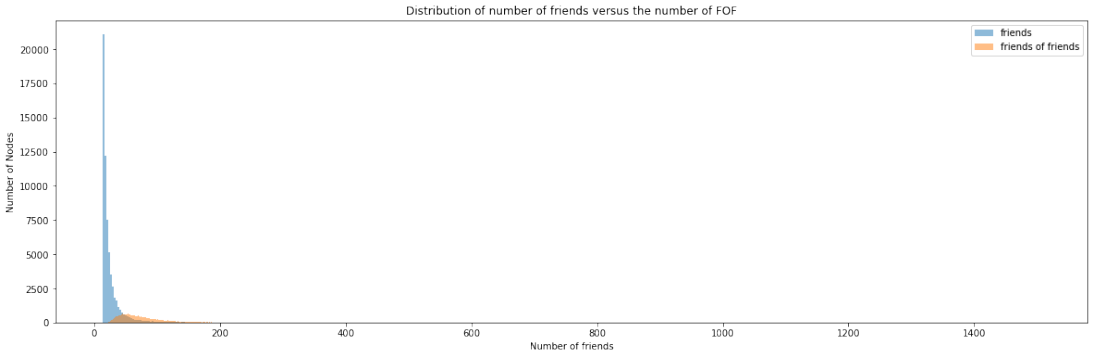
```
Real network: Average number of friends (average degree) = 25.641838351822503
Real network: Average number of friends of friends = 58.3634028072132
Real network: Number of nodes = 63731
Real network: Number of edges = 817090

BA model: Average number of friends (average degree) = 13.9902
BA model: Average number of friends of friends = 37.598432486950315
BA model: Number of nodes = 10000
BA model: Number of edges = 69951
```

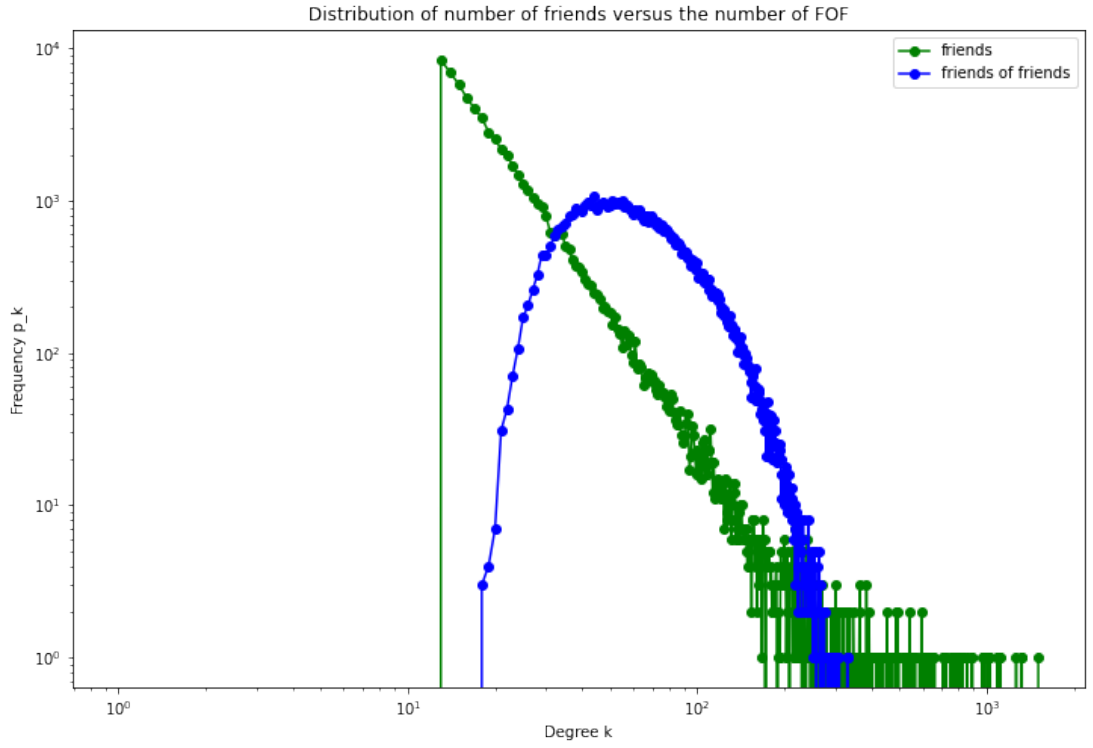
4.1 Comparison and Comment

Both, the average number of friends and the average number of friends of friends is higher in the real network compared to the network generated with the BA model. This follows the fact, that the number of nodes and edges differs between the real network and the BA model (see values above). The network generated with the BA model is much smaller then the real network (the BA model network consists of about $1/6$ of the nodes of the real network and about $7/80$ of the edges of the real network) .

When choosing the BA model parameters $N = 63731$ and $m = 13$, the calculated metrices above get very similar (this does not necessarily imply, that the network topology is similar).

[31]:
```
distribution_friend_fof_normal(ba_degrees, ba_number_friends_of_friend_per_node)
distribution_friend_fof_loglog(G_barabasi_albert,␣
 ↪ba_number_friends_of_friend_per_node)
```





4.2 Comparison and Comment

4.2.1 Real network

- In the real network we observe approximately a powerlaw distribution $p_k \sim k^{-\gamma}$ for the friends curve. The loglog plot shows, that lots of nodes have a small degree (left upper part) and few nodes are hubs and have a high degree (right part at the bottom).
- The FOF curve of the real network is similar to the loglog plot of model B on slide 5-19. This model eliminates growth and keeps preferential attachment.

4.2.2 BA model network

- For the generated network with the BA model, the friends curve follows the typical distibution with the clearly visible powerlaw $p_k \sim k^{-\gamma}$. In comparison to the real network the powerlaw distribution is more accurate. Furthermore the real network contains nodes with a very little degree. The network from the BA model has no nodes with very little degree (which is caused by the parameter m).
- The FOF curve shows that the majority of nodes has a medium degree. Moreover some nodes are hubs (tail of the distribution)

	0	**1**	**2**	**3**
0	0	0	0	0
1	0	0	0	0.125
2	0	0	0.25	0.25
3	0	0.125	0.25	0

Table 1: Degree Correlation Matrix

k	q_k
0	0.000
1	0.125
2	0.500
3	0.375

Table 2

3 Problem 6-3 Degree Correlation Coefficient

The computations for this solutions can be found in problem_ 3.ipynb

1. Compute the degree correlation matrix We computed the degree correlation matrix using networkx using inbuild functions (since it is repetitive). The manual process would be to calculate for each cell of the matrix the fraction of edges which connect nodes with the correct degree of i and j. For example, in the graph only one edge connects nodes which have the degree 1 and 3 with each other. There are a total of 4 edges in the graph. Normalized, so that E sums up to 1 (and since e_{ij} as well as e_{ji} are always taken into account) the cells $e_{1,3}$ and $e_{3,1}$ have a value of 0.125. The same process goes for the remaining cells. The degree correlation matrix is shown in Table 1.

2. Compute the probabilities q_k We used formula 7.3 from the lecture slides. The results are shown in Table 2

3. Compute the degree correlation coefficient r Our solution is $r = -0.714$. We used formula 7.11 from the lecture slides.

   Since $r < 0$, the network is disassortative.