Assignment 1

Jasper Schelling, Mediatechnology MSc. Programme - S1605976

Exercise 1: Neighborhoods (40p)

Note on exercise 1:

(Note to Frank) We discussed in our e-mail conversation in August, coming from a non-CS background it's especially the theoretical part where I need to acquire knowledge quickly, reading notation, grasping what is meant (or implicated) in the assignments. (Especially with regards to algorithms and computational complexity) Unfortunately at this point I'm only somewhat secure in my ability to answer Q1.1 and Q1.2. I have an intuitive notion of what the k-neighborhood function might do (grow a graph by adding neighbours, and neighbours of neighbours etc, creating a subgraph in which all nodes are connected), but I'm not sure I'm interpreting the notation correctly…

The past few weeks I just went along with the course, but wrestling with this exercise made me realize that I need to organize my study activities for this course a little differently. I'll use friday's lab session also to talk with Govert, and I'll keep updating this report.

Already in the three lectures I've seen (and learned) enough to see how valuable this course can be to my Mediatech graduation project.

Q1.1 Give a definition of the indegree and outdegree of a node using the notion of a (reversed) neighborhood.

The indegree of node v can be expressed as the number of edges that are in the revese neighborhood (edges of the set N'(v)). The outdegree of node v can be expressed as the number of edges that are in the neighborhood (edges of the set N(v)).

Q1.2 What do we know about nodes u, v V if for these nodes it holds that |N(u) N (v)| > 0 ?

If this is the case we know that nodes u and v are directly connected by at least one connected edge.

Q1.3 Explain how the k-neighborhood Nk(W) function can be used to determine if two nodes are in the same connected component of an undirected graph.

– See Note(1)

Q1.4 Write down a definition of the diameter of a connected undirected graph using the notion of a k-neighborhood.

– See Note(1)

Q1.5 The neighborhood of a node u V can easily be obtained if the underlying data structure of the network is an adjacency list: simply access node u's list of adjacent nodes. However, querying the existence of a link (u,v) between nodes u,v V will take O(k) time, where k is u's degree (list length). Indeed, each of the k nodes in the neighborhood of u should be checked to see if it is equal to v. Now, assume we want to answer many (say, more than n3) link queries (u,v), checking if a link exists between u and v. Explain a simple preprocessing step to speed up the O(k) time complexity of checking link existence, without using extra memory. Discuss the time complexity of your approach.

– See Note(1)

Q1.6 A square is a path (u,v,w,x,u) such that (u,v),(v,w), (w,x),(x,u) E. Use the (k-)neighborhood function to give, either in unambiguous and precise words or using simple pseudo-code, an algorithm for counting the number of squares in a connected undirected graph with n > 3 nodes. What is the time complexity of your algorithm?

– See Note(1)

Q1.7 Use the previous notions and algorithms to give, either in unambiguous and precise words or using simple pseudo-code, an algorithm for checking if a given undirected graph is a tree.

– See Note(1)

Exercise 2: Mining an Online Social Network (60p)

All sourcecode for these exercises is available from my Github Repository

Notes on elaborations for Exercise 2

All questions for exercise 2 were executed in Python. For each question an appropriate script was written. The scripts also take care of reading in the edgelist, and producing the appropriate (directed) network.

```
~ local$: python snacs_assignment_1_Q_2_#.py
```

Where # is the question number. Each script produces markdown output, that was used to write the answers to the questions above. For questions 2.3 and 2.5 there is also a snacs_assignment_1_Q_2_#_plot.py. I'm new to plotting data with matplotlib, I saved the results from my computation in a Python pickle so I could load that and plot my results. This was especially handy when computing the answers to question 2.5, since that required copious amounts of running time, so I didn't have to recompute the distance distribution every time I was tweaking my plot.

I also added an extra network; 'Tiny' it's based on the 'Toy Network' that is used in lecture presentations. It allowed me to manually verify that my scripts were producing the intended output.

For Questions 2.1 - 2.4 I used NetworkX to answer the questions. In the case of question 2.5, computing the distance distribution for the 'Large' network with NetworkX managed to turn my Macbook Pro into a functioning remote drone (read, fans were spinning so fast they managed to lift the laptop off my desk). Joking aside, Python / NetworkX quit after a total running time of an hour with an out of memory error. So, for answering question 2.5 I switched to using Graph-Tool. It produced the shortest distance distribution after 15 mins of computation on a single core. I wasn't able to install the OpenMP enabled version of graph-tool. I'm curious how much that would speed up the computation time…

Q2.1 How many Directed links does this network have? | snacs_assignment_1_Q_2_1.py

- **Medium Network:** 12864
- **Large Network:** 1731653

Q2.2 How many users does this network have? | snacs_assignment_1_Q_2_2.py

- **Medium Network:** 2239
- **Large Network:** 279630

Q2.3 Give the indegree and outdegree distribution of this graph

Medium Network

Indegree Distribution

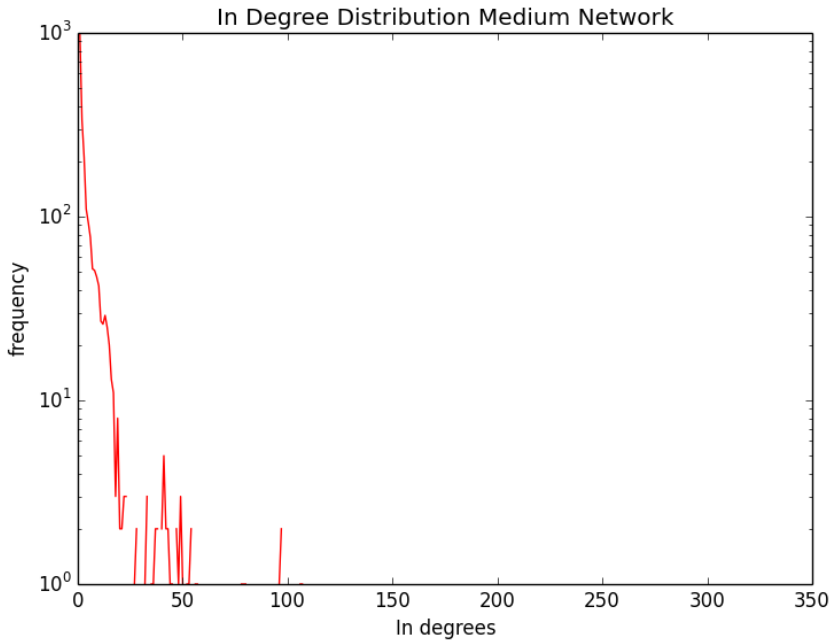

Figure 1: Medium In Degree Distribution

Outdegree Distribution

Large Network

Indegree Distribution

Outdegree Distribution

Q2.4 How many weakly connected components and how many strongly connected components does this network have? How many nodes and links are in the largest strongly connected component of this graph? | snacs_assignment_1_Q_2_4.py

- **Number of weakly connected components Medium Network:** 9
- **Number of weakly connected components Large Network:** 6863

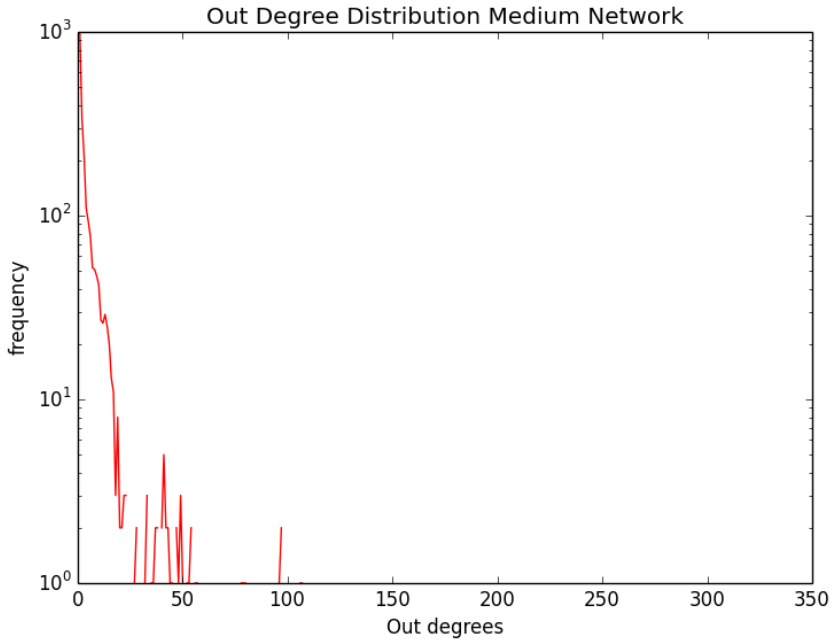

Figure 2: Medium Out Degree Distribution

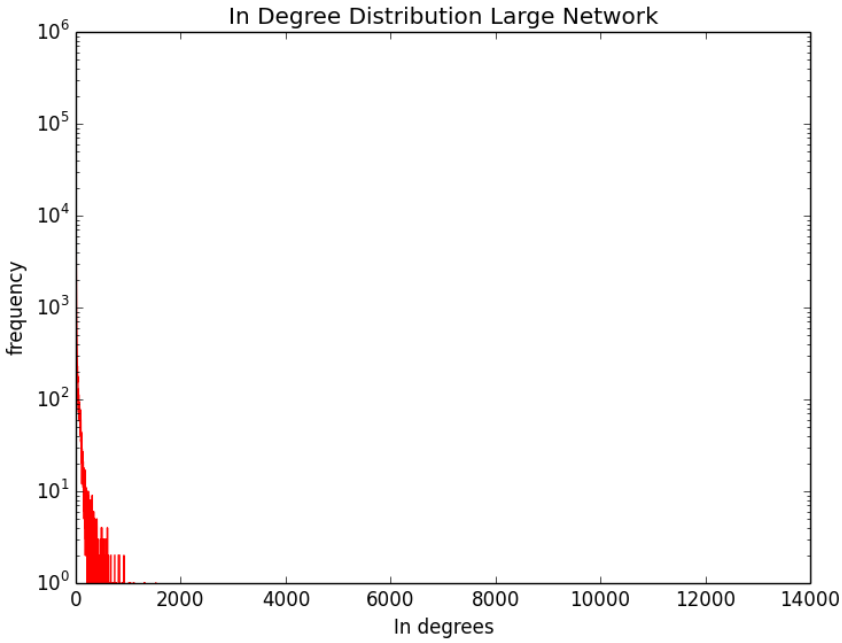

Figure 3: Medium In Degree Distribution

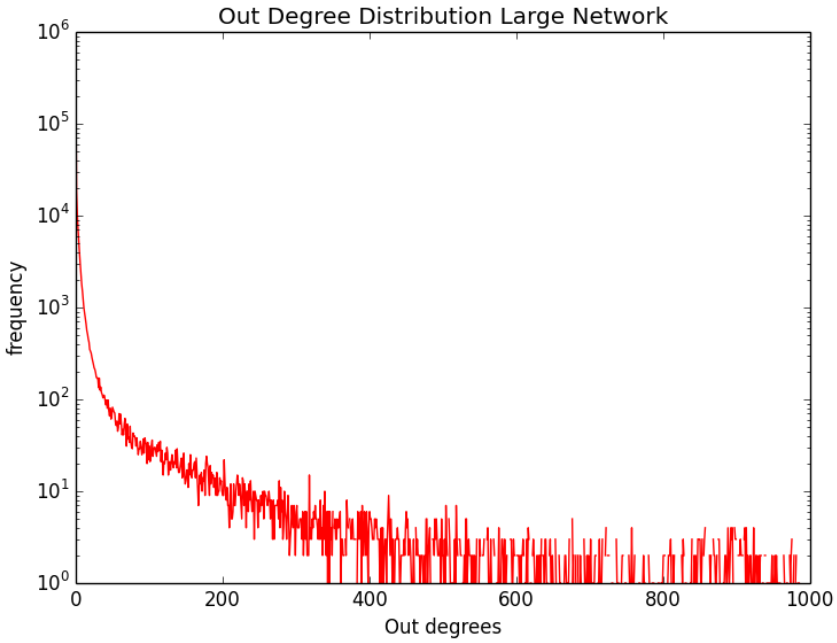

Figure 4: Medium Out Degree Distribution

- **Number of strongly connected components Medium Network:** 9
- **Number of strongly connected components Large Network:** 240113

How many nodes are in the largest strongly connected component?

- **Medium Network:** 2217
- **Large Network:** 34826

How many links are in the largest strongly connected component?

- **Medium Network:** 12836
- **Large Network:** 799102

Q2.5 Give the exact or approximated distance distribution of the largest strongly connected component of these graphs as a diagram.

Distance Distribution Medium Network

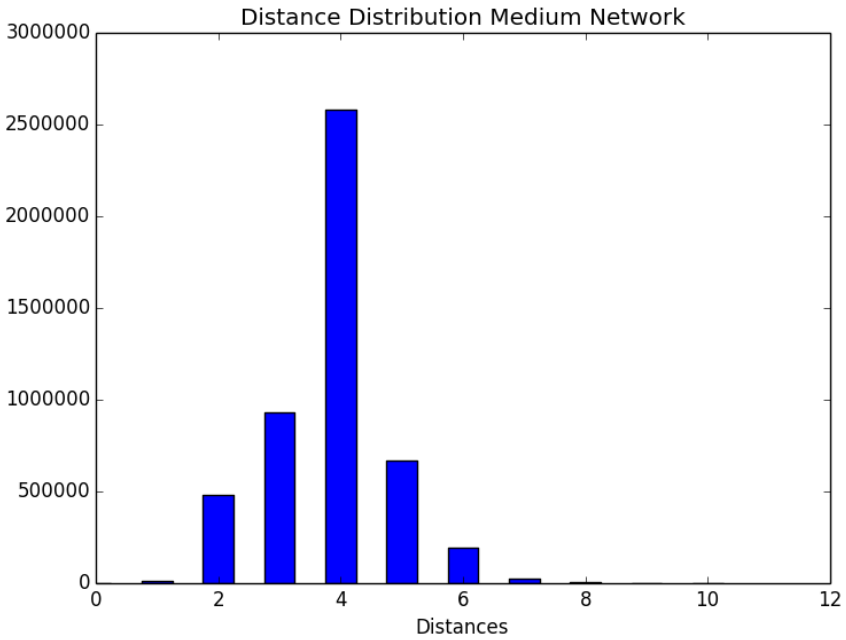

Figure 5: Medium Distance Distribution

Distance Distribution Large Network

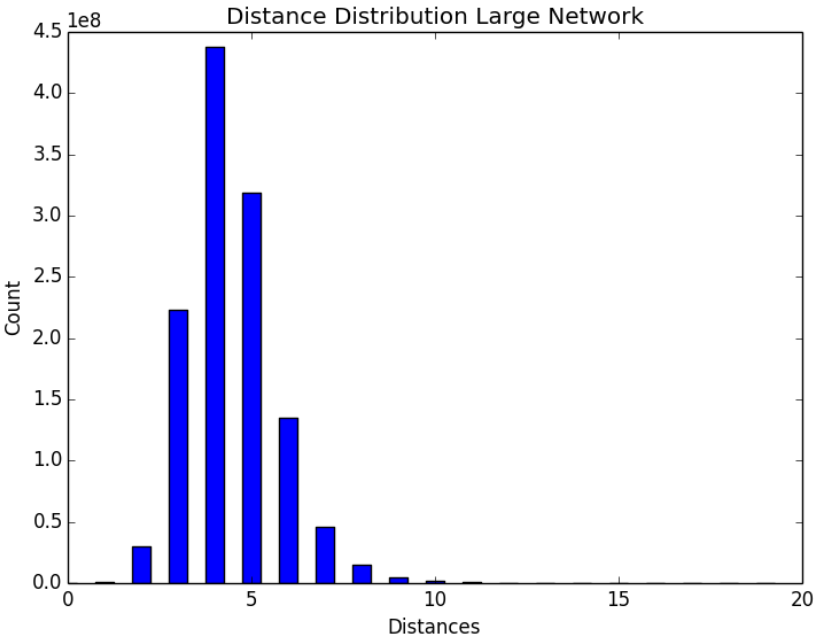

Figure 6: Large Distance Distribution

Q.2.6 Visualize medium.in

For this visualization, I generated a gephiready file which added a 'Directed' property to every edge. After importing I applied the Force Atlas 2 algorithm to the graph, after it stabilized I ran the distance metric. Using the betweenness centrality for both node size and color, I re-ran the Force Atlas 2 algorithm to create more space around the largest nodes. After this, FA2 was applied again with the 'Prevent Overlap' option which created nicely clustered nodes. Finally when rendering the final PDF / PNG I rendered the edges as curves, and lowered their opacity to 50% that way the edge density in different places of the graph was a bit more obvious to spot. However the lower opacity is only visible in the rendered PNG.

- PDF
- PNG

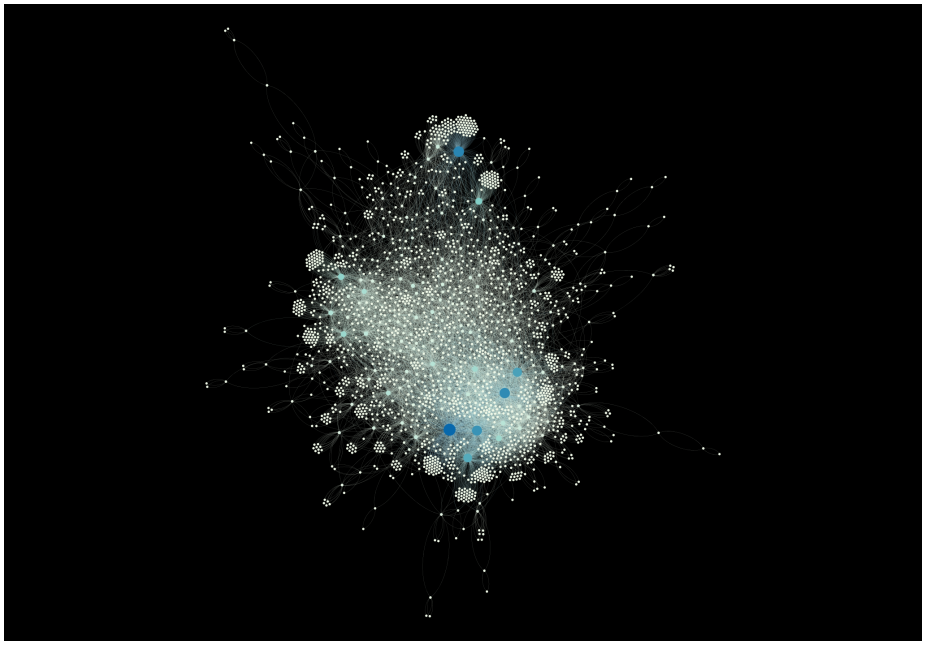

Figure 7: Snacs Assignment 1 Q 2 6