Social Media Analysis (SS2016) Dr. Krishna P. Gummadi

Exercise Sheet 1 – Basics of Network Theory

Max Planck Institute for Software Systems / Saarland University

Instructions (Please read them carefully)

1. You have to submit this assignment by 14:00 on 6th May 2016.
2. To submit your assignments, email your solutions to sma-ss16-instructors@mpi-sws.org, with the subject as "[SMA] HW 1 : <last name 1> <matriculation number 1>, <last name 2> <matriculation number 2> solutions".
3. Add your full name(s) and the matriculation number(s) in the beginning of all types of assignment solution submissions (in digital copy, inside the code as comment etc).
4. You are free to use any programming language you are comfortable with. Here is one library that handles large scale graphs efficiently and provides a large number of functions for analyzing graphs: `http://snap.stanford.edu/`. Its is called SNAP and has two versions : one for C++ and another for Python.
5. We should be able to run your submitted code in a computer with a reasonable configuration (for instance 2GB or more RAM) by following your submitted instructions.
6. For any random sampling (e.g. picking random nodes, edges etc.) use a pseudo random number generator with one of the matriculation numbers from your group as a seed. Clearly mention in the instruction file and in the code file (as comment) the random seed you used.
7. Please submit your code and instructions on how to run your code in a text file via email to the instructors. The name of the code file for problem 1 should be: **gen.structure.py** (or cpp or some other programming language). Similarly the name of the running instruction file should be: **gen.structure.howtorun.txt**. If there are any additional resources (eg. library) that are required to run your code, please mention those in the how to run file.
8. For the bonus problem the name of the code file should be: **gen.path.py** (or cpp or some other programming language). Similarly the name of the running instruction file should be: **gen.path.howtorun.txt**. If there are any additional resources (eg. library) that are required to run your code, please mention those in the how to run file.
9. We should be able to run the submitted code on our machine and get results. If it fails to run on our machine, you will not get any marks for the coding part of the assignment.
10. If any part of your code takes a long time to run (e.g., more than 10 minutes) report that in the instruction file with an estimate of time required.
11. References : Chapters 6 - 7 and 9 - 10, Networks - An Introduction, Mark Newman

Problem 1 (20 marks)

In this exercise, you will have to programmatically compute structural properties of 5 real world networks.

Datasets

Download the networks from the pages in Table 1 (the gzipped files in the Files section). Each graph is represented as edge list, for this exercise assume that the networks are undirected.

Network	Description	Network download location
soc-Epinions1	Epinions network	`https://snap.stanford.edu/data/soc-Epinions1.html`
cit-HepPh	Physics citation network	`https://snap.stanford.edu/data/cit-HepPh.html`
email-Enron	Enron email	`https://snap.stanford.edu/data/email-Enron.html`
p2p-Gnutella04	Gnutella p2p network	`https://snap.stanford.edu/data/p2p-Gnutella04.html`
complete-graph-1000-nodes	Complete graph with 1000 nodes	`http://courses.mpi-sws.org/sma-ss16/assignmentData/network-structure/complete-graph-1000-nodes.html`

Table 1: Name, description and download location for networks for this task

Task 1.1 (5 marks)

Once you have downloaded the above 5 networks, *generate the sub graphs* of each using the rules in Table 2. The table also contains the files to be submitted for this task. These subgraphs will be used throughout this assignment :

Subgraph name	Rule to extract the subgraph	Submission file name containing the subgraph as edgelist
soc-Epinions1-subgraph	Consider only the nodes with odd numbered node ids in soc-Epinions1 network.	**soc-Epinions1-subgraph.elist** with one edge per line (each edge is two node ids separated by white space/tab)
cit-HepPh-subgraph	Consider only the nodes with even numbered node ids in cit-HepPh network.	**cit-HepPh-subgraph.elist** with one edge per line (each edge is two node ids separated by white space/tab)
email-Enron-subgraph	Consider only the nodes with node ids divisible by 3 in email-Enron network.	**email-Enron-subgraph.elist** with one edge per line (each edge is two node ids separated by white space/tab)
p2p-Gnutella04-subgraph	Consider the full graph of p2p-Gnutella04 network.	**p2p-Gnutella04-subgraph.elist** with one edge per line (each edge is two node ids separated by white space/tab)
complete-graph-1000-nodes-subgraph	Consider only the nodes with node ids divisible by both 2 and 3.	**complete-graph-1000-nodes-subgraph.elist** with one edge per line (each edge is two node ids separated by white space/tab)

Table 2: Rules for extracting subgraph for each of the networks and submission instructions

[1+1+1+1+1]

Task 1.2 (15 marks)

For each of the 5 subgraphs given in Table 2, write a program **gen.structure.py** (or cpp or some other programming language) to generate following structural metrics below. Your code should take the name of the subgraph (specified in Table 2) as its argument, also note what your code should do when we run your code with a <subgraph name> as argument :

1. **Size of the network**
   (a) Number of nodes
   Your code should print a line in stdout "Number of nodes in <subgraph name>: value"
   (b) Number of edges
   Your code should print a line in stdout "Number of edges in <subgraph name>: value"
2. **Degree of nodes in the network**
   (a) Number of nodes which have degree = 7
   Your code should print a line in stdout "Number of nodes with degree=7 in <subgraph name>: value"
   (b) Node id(s) for the node with the highest degree. Note that there might be multiple nodes with highest degree
   Your code should print a line in stdout "Node id (s) with highest degree in <subgraph name>: comma (i.e. ",") separated id(s) of nodes"
   (c) Plot of the Degree distribution
   Your code should create the plotted image in the same directory as your code and print on stdout "Degree distribution of <subgraph name> is in: <filename> "
3. **Paths in the network**
   (a) Approximate full diameter (maximum shortest path length) computed starting from 10, 100, 1000 random test nodes. Also calculate the average and variance across these 3 estimates of the diameter.
   Your code should print lines in stdout "Approximate full diameter in <subgraph name> with sampling <number of test nodes > nodes: diameter value" and "Approximate full diameter in <subgraph name> with sampling nodes (mean and variance): mean diameter value, variance value"
   (b) Approximate effective diameter computed starting from 10, 100, 1000 random test nodes. Also calculate the average and variance across these 3 estimates of the diameter.
   Your code should print lines in stdout "Approximate effective diameter in <subgraph name> with sampling <number of test nodes > nodes: diameter" and in the next line "Approximate effective diameter in <subgraph name> with sampling nodes (mean and variance): mean diameter value, variance value"
   (c) Plot of the distribution of the shortest path lengths in the network.
   Your code should create the plotted image in the same directory as your code and print on stdout "Shortest path distribution of <subgraph name> is in: <filename> "
4. **Components of the network**
   (a) Fraction of nodes in the largest connected component
   Your code should print a line in stdout "Fraction of nodes in largest connected component in <subgraph name>: value"

(b) Number of edge bridges. An edge is a bridge if, when removed, increases the number of connected components.
Your code should print a line in stdout "Number of edge bridges in <subgraph name>: value"

(c) Number of articulation points. A node is a articulation point if, when removed, increases the number of connected components.
Your code should print a line in stdout "Number of articulation points in <subgraph name>: value"

(d) Plot of the distribution of sizes of connected components
Your code should create the plotted image in the same directory as your code and print on stdout "Component size distribution of <subgraph name> is in: <filename> "

5. **Connectivity and clustering in the network**

(a) Average clustering coefficient of the network (briefly explained here `https://en.wikipedia.org/wiki/Clustering_coefficient#Network_average_clustering_coefficient`). For the average clustering coefficient round to 4 decimal places while reporting.
Your code should print a line in stdout "Average clustering coefficient in <subgraph name>: value"

(b) Number of triads
Your code should print a line in stdout "Number of triads in <subgraph name>: value"

(c) Clustering coefficient of a randomly selected node. Also report the selected node id.
Your code should print a line in stdout "Clustering coefficient of random node < node id > in <subgraph name>: value"

(d) Number of triads a randomly selected node participates in. Also report the selected node id.
Your code should print a line in stdout "Number of triads random node < node id > participates in <subgraph name>: value"

(e) Number of edges that participate in at least one triad
Your code should print a line in stdout "Number of edges that participate in at least one triad in <subgraph name>: value"

(f) Plot of the distribution of clustering coefficient
Your code should create the plotted image in the same directory as your code and print on stdout "Clustering coefficient distribution of <subgraph name> is in: <filename> "

(g) Plot of the k-core edge-size distribution: core k vs. number of edges in k-core.
Your code should create the plotted image in the same directory as your code and print on stdout "k-core edge-size distribution of <subgraph name> is in: <filename> "

(h) Plot of the k-core node-size distribution: core k vs. number of nodes in k-core.
Your code should create the plotted image in the same directory as your code and print on stdout "k-core node-size distribution of <subgraph name> is in: <filename> "

$[0.75 \times 20 = 15]$

Problem 2 (Bonus question - 5 marks)

Submit a program **gen.path.py** (or .cpp or any other language) which does as stated below (also remember to submit the running instruction file **gen.path.howtorun.txt**):

1. Picks a pair of random nodes from the largest connected component of any one of the following 4 subgraphs – soc-Epinions1-subgraph, cit-HepPh-subgraph, email-Enron-subgraph, p2p-Gnutella04-subgraph (mentioned in Table 2)

2. For this pair of random nodes, generate all the edge independent and vertex independent paths. **Your code should print on stdout** "Picked random node pair < node id 1 >, < node id 2 > from the subgraph < subgraph name >." Then it should print the number of edge independent paths between the two nodes and the actual paths (as list of nodes in the path, one path each line). The code should do the same for vertex independent paths

[1 + [2 + 2]]