CS224W: Social and Information Network Analysis **Fall 2016**

Problem Set 1

Due 11:59pm PDT October 13, 2016

General Instructions

These questions require thought, but do not require long answers. Please be as concise as possible. You are allowed to take a maximum of 1 late period (see the information sheet for at the end of this document for the definition of a late period).

Submission instructions: You should submit your answers via GradeScope and your code via the SNAP submission site. Register for GradeScope at `http://gradescope.com` using your Stanford e-mail (if not SCPD) and include your student ID number with sign-up. Use the entry code **MJ2ZDM** to sign up for CS224W.

Submitting answers: Prepare answers to your homework in a single PDF file and submit it via GradeScope. Make sure that the answer to each sub-question is on a *separate, single page*. The number of the question should be at the top of each page. Please use the submission template file to prepare your submission: [tex | pdf].

Fill out the information sheet located at the end of this problem set or at the end of the submission template file and sign it in order to acknowledge the Honor Code (if typesetting the homework, you may type your name instead of signing). This should be the last page of your submission. Failure to fill out the information sheet will result in a reduction of 2 points from your homework score.

Submitting code: Upload your code at `http://snap.stanford.edu/submit`. Put all the code for a single question into a single file and upload it.

Questions

1 Network Characteristics [35 points – Austin]

One of the goals of network analysis is to find mathematical models that characterize real-world networks and that can then be used to generate new networks with similar properties. In this problem, we will explore two famous models—Erdős-Rényi and Small World—and compare them to real-world data from an academic collaboration network. Note that in this problem all networks are *undirected*.

- *Erdős-Rényi Random graph ($G(n, m)$ random network):* Generate a random instance of this model by using $n = 5242$ nodes and picking $m = 14484$ edges at random.
- *Small-World Random Network:* Generate an instance from this model as follows: begin with $n = 5242$ nodes arranged as a ring, i.e., imagine the nodes form a circle and each node is connected to its two direct neighbors (e.g., node 399 is connected to nodes 398 and 400), giving us 5242 edges. Next, connect each node to the neighbors of its neighbors (e.g., node 399 is also connected to nodes 397 and 401). This gives us another 5242 edges. Finally, randomly select 4000 pairs of nodes not yet connected and add an edge between them. In total, this will make $m = 5242 \cdot 2 + 4000 = 14484$ edges. (Write code to construct instances of this model, i.e., do not call a SNAP function as above)

- *Real-World Collaboration Network:* Download this undirected network from http://snap.stanford.edu/data/ca-GrQc.txt.gz. Nodes in this network represent authors of research papers on the arXiv in the General Relativity and Quantum Cosmology section. There is an edge between two authors if they have co-authored at least one paper together. Note that some edges may appear twice in the data, once for each direction Ignoring repeats and self-edges, there are 5242 nodes and 14484 edges. (Note: repeats are automatically ignored when loading an (un)directed graph with SNAP's `LoadEdgeList` function).

1.1 Degree Distribution [10 points]

Generate a random graph from both the Erdős-Rényi (i.e., $G(n, m)$) and Small-World models and read in the collaboration network. Delete all of the self-edges in the collaboration network (there should be 14,484 total edges remaining).

Plot the degree distribution of all three networks *in the same plot* on a log-log scale. In other words, generate a plot with the horizontal axis representing node degrees and the vertical axis representing the proportion of nodes with a given degree (by "log-log scale" we mean that both the horizontal and vertical axis must be in logarithmic scale). In one to two sentences, describe one key difference between the degree distribution of the collaboration network and the degree distributions of the random graph models.

Hint: Use SNAP's `GenRndGnm` function to generate from the Erdős-Rényi model. However, you need to write a routine to generate a random instance of the Small-World model as described above.

1.2 Excess Degree Distribution [15 points]

An important concept in network analysis is the *excess degree distribution*, denoted as q_k, for $k \geq 0$. Intuitively, q_k gives the probability that a randomly chosen edge goes to a node of degree $k+1$. Excess degree can be calculated as follows:

$$q_k = \frac{q'_k}{\sum_i q'_i}, \qquad q'_k = \sum_{i \in V} \sum_{(i,j) \in E} I_{[k_j = k+1]},$$

where $I_{\text{condition}} = 1$ when condition is true and 0 otherwise. V denotes the set of nodes, E the set of edges and k_j the number of neighbors of node j (equivalently, the degree of node j). Additionally, the *expected excess degree* is $\sum_{k \geq 0} k \cdot q_k$, and the *expected degree* is $\sum_{k \geq 0} k \cdot p_k$, where p_k is the proportion of nodes having degree exactly k.

1.2 (a) [10 points] Plot the excess degree distributions of all three networks in the same plot on a log-log scale. In one to two sentences, describe one key difference between the degree distribution and the excess degree distribution of the collaboration network. Then compute and report the expected degree and the expected excess degree for each network.

1.2 (b) [5 points] Show how to compute the excess degree distribution $\{q_k\}$ given only the degree distribution $\{p_k\}$.

1.3 Clustering Coefficient [10 points]

Recall that the local clustering coefficient for a node v_i was defined in class as

$$C_i = \begin{cases} \frac{2|e_i|}{k_i \cdot (k_i - 1)} & k_i \geq 2 \\ 0 & \text{otherwise,} \end{cases}$$

where k_i is the degree of node v_i and e_i is the number of edges between the neighbors of v_i. The *average clustering coefficient* is defined as

$$C = \frac{1}{|V|} \sum_{i \in V} C_i.$$

Compute and report the average clustering coefficient of the three networks. For this question, write your own implementation to compute the clustering coefficient, instead of using a built-in library function.

Which network has the largest clustering coefficient? In one to two sentences, explain.

What to submit

Page 1:
- Log-log degree distribution plot for all three networks (in same plot)
- One to two sentence description of a difference between the collaboration network's degree distribution and the degree distributions from the random graph model.

Page 2:
- (part a) Log-log excess degree distribution plot for all three networks (in same plot)
- (part a) One to two sentence description of the difference in the distribution of the degree and excess degree distributions for the collaboration network.
- (part a) Expected degree and expected excess degree for each network.
- (part b) Short proof showing how to calculate $\{q_k\}$ in terms of $\{p_k\}$.

Page 3:
- Average clustering coefficient for each network.
- Network that has the largest average clustering coefficient.
- One to two sentences explaining why this network has the largest average clustering coefficient.

2 Who is the most central actor [30 points – Leon]

NOTE: The actual computation and run time for the code you will produce for the following question could take many minutes to run so please start early.

For the 2005 Graph Drawing conference a data set was provided of the IMDB movie database. We will use a reduced version of this dataset, which derived all actor-actor collaboration edges where the actors co-starred in at least 2 movies together between 1995 and 2004. Our task is to identify the most central actor, but we're not sure which centrality measure to use, so we'll try three of them: degree, betweenness, closeness. Please download the required files from:
http://cs224w.stanford.edu/homeworks/hw1/imdb_actor_edges.tsv
http://cs224w.stanford.edu/homeworks/hw1/imdb_actors_key.tsv

Compute the following just for actors within the largest weakly connected component (LWCC) of the actor network:

2.1 Degree centrality [10 points]

Compute the degree centrality for all actors in the LWCC and list the 20 actors with the highest degree. Considering the number of movies made by these actors vs. others, briefly (2–4 sentences) explain their high degree.

2.2 Betweenness centrality [10 points]

Compute the betweenness centrality for all actors in the LWCC and list the 20 actors with the highest betweenness. Only a couple of actors with top degree centrality remain on this list. Considering the distribution of genres these actors acted in, briefly (2–4 sentences) explain the discrepancy between degree and betweenness centrality for these actors.

2.3 Closeness centrality [10 points]

Compute the closeness centrality for all actors in the LWCC and list the 20 actors with the highest closeness. Using Figure 1 below, briefly (2–4 sentences) explain why the actors with top closeness tend to overlap so little with the two previous lists. Note that the figure shows a large pink "documentary" clump where many of the high closeness actors are located. It's a bit puzzling, but many of these high-closeness actors do seem to have been involved in documentary serials or "the making-of" documentaries. We've excluded "Documentary" and "Short" from being candidates for an actor's main genre, but have used them to color the figure (e.g. Tom Hanks and Whoopi Goldberg are somehow in over 30 documentaries during the time span of the dataset).

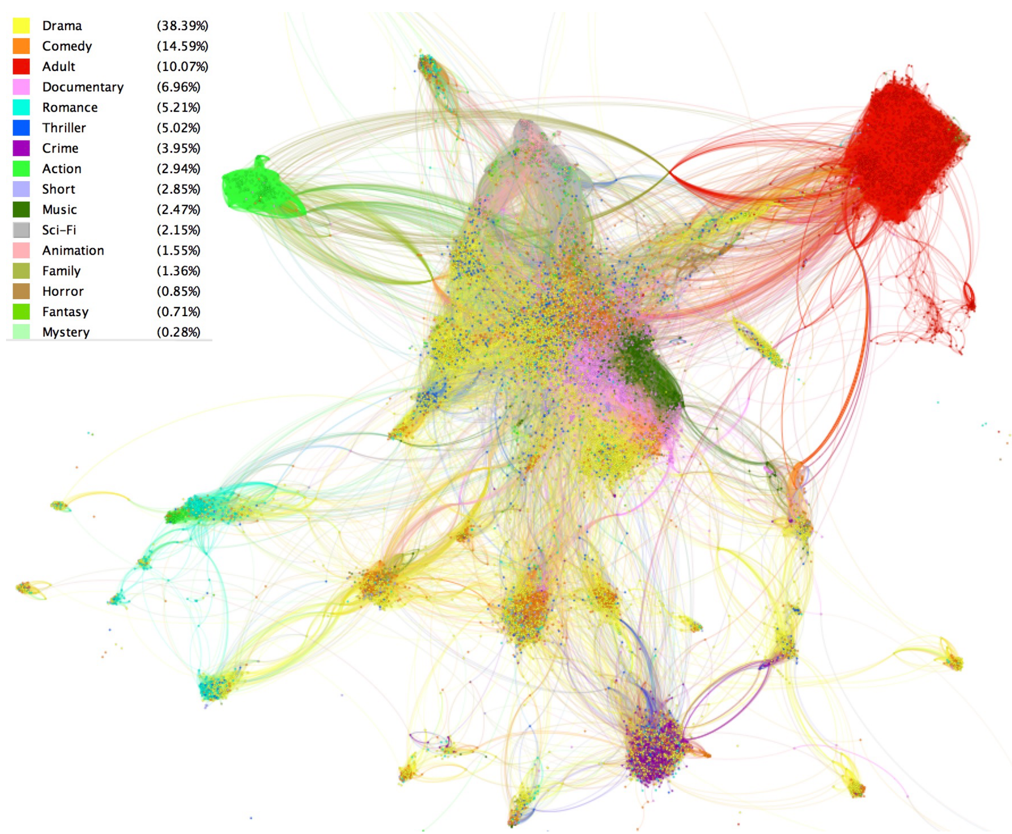

Figure 1: Visualization of the actor network.

What to submit

Page 4:
- The 20 actors with the highest degree centrality in the LWCC.
- Brief explanation of their high degree (2–4 sentences).

Page 5:
- The 20 actors with the highest betweenness in the LWCC.
- Brief explanation of the discrepancy between the actors with high degree and high betweenness (2–4 sentences).

Page 6:
- The 20 actors with the highest closeness in the LWCC.
- Brief explanation of why these actors tend to overlap so little with the actors of high degree and high betweenness (2–4 sentences).

3 Decentralized Search [45 points – Will, Ben]

In class, we saw a decentralized search algorithm based on geography that seeks a path between two nodes on a grid by successively taking the edge towards the node closest to the destination.

Here we will examine an example of a decentralized search algorithm on a network where the edges are created according to an underlying hierarchical tree structure. In particular, the nodes of our network will be defined as the leaves of a tree, and the edge probabilities between two nodes will depend on their proximity in this underlying tree structure. The tree may, for instance, be interpreted as representing the hierarchical organization of a university where one is more likely to have friends inside the same department, a bit less likely in the same school, and the least likely across schools.

Let us organize students at Stanford into a tree hierarchy, where the root is Stanford University and the second level contains the different schools (engineering, humanities, etc.). The third level represents the departments and the final level (i.e., the leaves) are the Stanford students. Tom, a student from the computer science department, wants to hang out with Mary, who is in sociology. If Tom does not know Mary, he could ask a friend in the sociology department to introduce them. If Tom does not know anybody in the sociology department, he may seek a friend in the Stanford humanities school instead. In general, he will try to find a friend who is "close" to Mary in the tree.

There are three parts in this problem. The first two parts explore an effective decentralized search algorithm on the hierarchical model in a specific setting. The third part involves simulation experiments on the model under a more general setting.

One important thing to keep in mind: In this problem, the network nodes are only the leaf nodes in the tree. Other nodes in the tree are virtual and only there to determine the edge creation probabilities between the nodes of the network. So there are two networks: one is the observed network (i.e., "edges between students") and the other is the underlying tree structure that is used to generate the edges in the observed network ("the hierarchy of Stanford").

3.1 Basic Tree Properties [15 points]

This part covers some basic facts of the setting of the hierarchical model and defines the notion of tree based "distance" between the nodes.

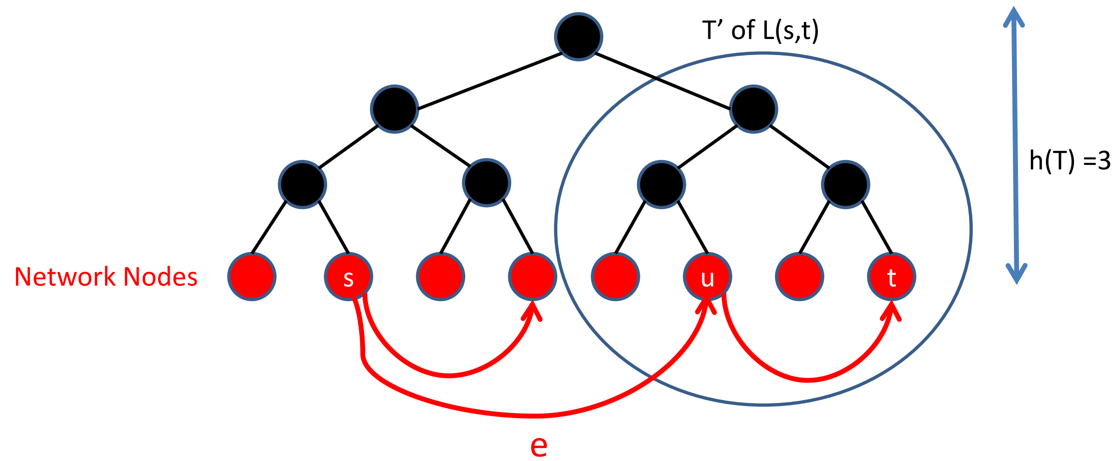

Figure 2: Illustration of the graph in Question 3. Black nodes and edges are used to illustrate the hierarchy and structure, but are not part of our network. Red nodes (leaf nodes) and red edges are the ones in our network. The lowest common ancestor of s and t is the root of the tree. The decentralized search proceeds as follows. Denote the starting node by s and the destination by t. At each step, the algorithm looks at the neighbors of the current node s and moves to the one "closest" to t, that is, the algorithm moves to the node with the lowest common ancestor with t. In this graph, from s we move to u.

Consider a complete, perfectly balanced b-ary tree T (each non-leaf node has b children and $b \geq 2$), and a network whose nodes are the leaves of T (the red nodes in Figure 2). Let the number of network nodes (equivalently, the number of leaves in the tree) be N and let $h(T)$ denote the height of T (see Figure 2 for an example). Recall that a tree with one node has a height of zero.

Note: When answering questions in this section, feel free to cite well-known properties of trees (e.g., number of nodes in a complete binary tree of a certain height), but please provide evidence of reasoning as well. In other words, we don't expect you to prove basic tree properties by induction; just provide some sound reasoning.

3.1(a) [5 points] Write $h(T)$ in terms of N.

3.1(b) [5 points] Next we want to define the notion of "tree distance." The intuition we want to capture is that students that share the same department are closer than for example students sharing schools. For instance, in the tree in Figure 2 nodes u and t are "closer" than nodes u and s. We formalize the notion of "distance" as follows:

Given two network nodes (leaf nodes) v and w, let $L(v, w)$ denote the subtree of T rooted at the lowest common ancestor of v and w, and $h(v, w)$ denote its height (that is, $h(L(v, w))$). In Figure 2, $L(u, t)$ is the tree in the circle and $h(u, t) = 2$. Note that we can think of $h(v, w)$ as a "distance" between nodes v and w.

For a given node v, what is the maximum possible value of $h(v, w)$?

3.1(c) [5 points] Given a value d and a network node v, show that there are $b^d - b^{d-1}$ nodes satisfying $h(v, w) = d$.

3.2 Network Path Properties [20 points]

This part helps you design a decentralized search algorithm in the network.

We will generate a random network on the leaf nodes in a way that models the observation that a node is more likely to know "close" nodes than "distant" nodes according to our university organizational hierarchy captured by the tree T. For a node v, we define a probability distribution of node v creating an edge to any other node w:

$$p_v(w) = \frac{1}{Z} b^{-h(v,w)}$$

where $Z = \sum_{w \neq v} b^{-h(v,w)}$ is a normalizing constant. By symmetry, all nodes v have the same normalizing constant.

Next, we set some parameter k and ensure that every node v has exactly k outgoing edges. We do this with the following procedure. For each node v, we repeatedly sample a random node w according to p_v and create edge (v, w) in the network. We continue this until v has exactly k neighbors.Equivalently, after we add an edge from v to w, we can set $p_v(w)$ to 0 and renormalize with a new Z to ensure that $\sum_w p(w) = 1$. This results in a k-regular directed network.

3.2(a) [5 points] Show that $Z \leq \log_b N$. (Hint: use the results in parts 3.1(a) and 3.1(b)).

3.2(b) [5 points] For two leaf nodes v and t, let T' be the subtree of $L(v, t)$ satisfying:

- T' is of height $h(v, t) - 1$,
- T' contains t,
- T' does not contain v.

For instance, in Figure 2, T' of $L(s, t)$ is the tree in the circle.

Let us consider node v and an edge e from v to a random node u sampled from p_v. We say that e points to T' if u is a leaf node of T'. Show that the probability of e pointing to T' is no less than $\frac{1}{b \log_b N}$.

3.2(c) [5 points] Let the out-degree k for each node be $c \cdot (\log_b N)^2$, where c and b are constants. Show that when N grows very large, the probability that v has no edge pointing to T' is asymptotically no more than $N^{-\theta}$, where θ is a positive constant which you need to compute.

(Hints: Use the result in part 3.2(b) and recall that each of the k edges is independently created. Also, use $\lim_{x \to \infty}(1 - \frac{1}{x})^x = \frac{1}{e}$.)

Argue why the above result indicates that for any node v, we can, with high probability, find an edge to a (leaf) node u satisfying $h(u, t) < h(v, t)$.

3.2(d) [5 points] Show that starting from any (leaf) node s, within $O(\log_b N)$ steps, we can reach any (leaf) node t. You do not need to prove it in a strict probabilistic argument. You can just assume that for any (leaf) node v, you can always get to a (leaf) node u satisfying $h(u, t) < h(v, t)$ and argue why you can reach t in $O(\log_b N)$ steps.

3.3 Simulation [10 points]

So far, we have set the theory to find an efficient decentralized search algorithm, assuming that for each edge of v, the probability of it going to w is proportional to $b^{-h(v,w)}$. Now we experimentally investigate a more general case where the edge probability is proportional to $b^{-\alpha h(v,w)}$. Here $\alpha > 0$ is a parameter in our experiments.

In the experiments below, we consider a network with the setting $h(T) = 10$, $b = 2$, $k = 5$, and a given α. That is, the network consists of all the leaves in a binary tree of height 10; the out degree of each node is 5. Given α, we create edges according to the distribution described above.

Create random networks for $\alpha = 0.1, 0.2, \ldots, 10$. For each of these networks, sample 1,000 unique random (s, t) pairs $(s \neq t)$. Then do a decentralized search starting from s as follows. Assuming that we are currently at (leaf) node s, we pick its neighbor u (also a leaf node) with smallest $h(u, t)$ (break ties arbitrarily). If $u = t$, the search succeeds. If $h(s, t) > h(u, t)$, we set s to u and repeat. If $h(s, t) \leq h(u, t)$, the search fails.

For each α, pick 1,000 pairs of nodes and compute the average path length for the searches that succeeded. Then draw a plot of the average path length as a function of α. Also, plot the search success probability as a function of α.

Hint: Be smart with how you do the normalization when you construct the network (i.e., when you sample a node's 5 neighbors). If you simply try to do "rejection sampling" then you will keep sampling the same node over and over (especially for certain values of α), and your code will be very very slow.

Briefly comment on the plots and explain the shape of the curve.

What to submit

Page 7:
- (part a) The expression and a short explanation (2–3 sentences).
- (part b) The expression and a short explanation (2–3 sentences).
- (part c) A short proof.

Page 8:
- (part a) A short proof.
- (part b) A short proof.
- (part c) A short proof with an expression for θ
- (part c) Brief argument for why we can find an edge to a (leaf) node u satisfying $h(u, t) < h(v, t)$ (1–2 sentences).
- (part d) A short proof.

Page 9:
- Both plots
- A brief comment (1–3 sentences) on each plot.

Information sheet
CS224W: Social and Information Network Analysis

Assignment Submission Fill in and include this information sheet with each of your assignments. This page should be the last page of your submission. Assignments are due at 11:59pm and are always due on a Thursday. All students (SCPD and non-SCPD) must submit their homeworks via GradeScope (http://www.gradescope.com). Students can typeset or scan their homeworks. Make sure that you answer each (sub-)question on a separate page. That is, one answer per page regardless of the answer length. Students also need to upload their code at http://snap.stanford.edu/submit. Put all the code for a single question into a single file and upload it. Please do not put any code in your GradeScope submissions.

Late Homework Policy Each student will have a total of *two* free late periods. *Homeworks are due on Thursdays at 11:59pm PDT and one late period expires on the following Monday at 11:59pm PDT*. Only one late period may be used for an assignment. Any homework received after 11:59pm PDT on the Monday following the homework due date will receive no credit. Once these late periods are exhausted, any assignments turned in late will receive no credit.

Honor Code We strongly encourage students to form study groups. Students may discuss and work on homework problems in groups. However, each student must write down their solutions independently i.e., each student must understand the solution well enough in order to reconstruct it by him/herself. Students should clearly mention the names of all the other students who were part of their discussion group. Using code or solutions obtained from the web (github/google/previous year solutions etc.) is considered an honor code violation. We check all the submissions for plagiarism. We take the honor code very seriously and expect students to do the same.

Your name: ____________________

Email: ____________________ **SUID:** ____________________

Discussion Group: ____________________

I acknowledge and accept the Honor Code.

(Signed) ____________________