CS224W: Social and Information Network Analysis **Fall 2013**

Problem Set 0

Due 9:30am October 3, 2013

No late days are allowed for this problem set. This problem set should be completed individually.

General Instructions

This homework is to be completed individually (no collaboration is allowed). Also, you are not allowed to use any late days for the homework.

You submit Question 1 on paper and Question 2 online. For paper submission (Question 1), please fill the cover sheet (http://cs224w.stanford.edu/cover.pdf) and submit it as a front page with your answers.

Regular (non-SCPD) students should submit hard copies of the answers either in class or in the submission box (see course website for location). You should print-out as well as upload your source code and any other files you used.

SCPD students should submit their answers through SCPD and also upload the code. The submission must include all the answers, the source code and the usual SCPD routing form (http://scpd.stanford.edu/generalInformation/pdf/SCPD_HomeworkRouteForm.pdf).

For Question 2, everyone should submit their answers at http://snap.stanford.edu/submit-cs224w/.

Questions

The purpose of this homework is to get you started with network analysis. You will need to install and try out graph analysis tools.

For Question 1, we strongly encourage you to use Snap.py for Python (peek forward to Q2 for installation and tutorial of Snap.py). However, for Question 1 you can use any other graph analysis tool or package you want (SNAP for C++, NetworkX for Python, JUNG for Java, etc.). See http://cs224w.stanford.edu/resources.html for more details.

For Question 2, your task is to help with the development of the Snap.py package. Snap.py is a Python interface for SNAP. It provides the performance benefits of SNAP, combined with the flexibility of Python.

1 Analyzing the Wikipedia voters network

1. Download the Wikipedia voting network `wiki-Vote.txt.gz`: http://snap.stanford.edu/data/wiki-Vote.html.

2. Using one of the network analysis tools above load the Wikipedia voting network. Note Wikipedia is a directed network.

3. Submit a table with the following statistics of the Wikipedia network. Just to be precise by what needs to be computed we consider the Wikipedia network as a directed graph $G(V, E)$, with node set V and edge set $E = \{(a, b); a \in V, b \in V\}$ where (a, b) are the edges, i.e., unique ordered pairs of nodes.

   As a running example, consider a directed network with the following set of edges $\{(1, 2), (2, 1), (1, 3), (1, 1)\}$.

   Compute and submit the following statistics for the `wiki-Vote` network:

   (a) *Number of nodes in the network:* Our example network above has 3 nodes.

   (b) *Number of nodes with a self-edge (i.e., a self-loop):* Number of nodes a where there exists an edge (a, a). Our example network above has 1 self-edge.

   (c) *Number of directed edges in the network:* Number of unique ordered (a, b) pairs, where a and b are nodes ($a \in V, b \in V$ and $a \neq b$). Our example network above has 3 directed edges.

   (d) *Number of undirected edges in the network:* Number of unique unordered (a, b) pairs ($a \in V, b \in V$ and $a \neq b$). This means that if both (a, b) and (b, a) exist in the data, you count this as a single undirected edge. Our example network above has 2 undirected edges.

   (e) *Number of reciprocated edges in the network:* Number of pairs of nodes (a, b) where there exists a directed edge in both directions (there exist edges $(a, b), (b, a)$ where $a \neq b$). Our example network above has 1 reciprocated edge

   (f) *Number of nodes of zero out-degree:* Our example network above has 1 nodes with zero out-degree.

   (g) *Number of nodes of zero in-degree:* Our example network above has 0 nodes with zero in-degree

   (h) *Number of nodes with more than 10 outgoing edges* (out-degree > 10)

   (i) *Number of nodes with less than 10 incoming edges* (in-degree < 10)

What to submit:

1. Print-out of your answers to questions (a)—(i)

2. Print-out of your code

3. Also submit your code at http://snap.stanford.edu/submit

2 Snap.py Reference Guide

Your task here is to help with documenting the functions in Snap.py. You will be assigned three functions for which you will perform a simple test of their functionality, and prepare a short example code.

1. Download and install Snap.py from http://snap.stanford.edu/snap/snap.py.html.
2. Complete the Snap.py tutorial at http://snap.stanford.edu/snappy/doc/index.html
3. Obtain the list of functions for your task by logging to http://snap.stanford.edu/submit-cs224w/. Note that each student will get a different set of Snap.py functions to work on.
4. Prepare a reference page for each of your functions. Each reference page is a separate text file that you will later upload.

   Your reference page should include:

   (a) the function name in the page title,
   (b) the function prototype: function type, name and parameters,
   (c) descriptions of functionality, parameters, and the return value,
   (d) any notes, and
   (e) a short code example demonstrating the function use. If applicable, use all of **TNGraph**, **TUNGraph** and **TNEANet** in your examples.

   Your task is to create one such reference page for each of the functions assigned to you.

   A sample reference page for a PageRank function is available at: http://snap.stanford.edu/snappy/doc/reference/GetPageRank.html. The *Show Source* link on the left side of the sample page includes source text for the page in the Sphinx markup language (http://sphinx-doc.org/). Use this sample source text as a starting point for your answer and modify it for each of your functions.

   A quick guide to the Sphinx markup language is available here http://snap.stanford.edu/snappy/guide.txt. If you wish to know more, see http://sphinx-doc.org/.

   Each Snap.py function has an equivalent SNAP C++ function. You can use SNAP C++ documentation to help you with the Snap.py reference page (see http://snap.stanford.edu/snap/doc/snapdev-ref/d3/d73/namespaceTSnap.html).

   Note that Snap.py is still in the Beta stage. There may be bugs and functions may not work as expected. If you think that a function does not work, report the problem on Piazza, and provide a description of the bug or unexpected behavior on your submission page.

5. Upload individual reference pages and any error reports at http://snap.stanford.edu/submit-cs224w/. Upload reference page of each function as a separate `.txt` file.