CS224W: Machine Learning with Graphs **Fall 2019**

Homework 0

Due 11:59pm PT October 3, 2019

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

General Instructions

This homework is easy and will get you started on tools for network analysis. If you find these questions difficult, then you might not be ready for this course.

Submission instructions: 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. Use the submission template files included in the bundle to prepare your submission.

Fill out the information sheet located 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: Students also need to upload their code on Gradescope. Put all the code for a specific question into a single compressed file and upload it.

Questions

The purpose of these exercises is to get you started with network analysis and the SNAP software. For this homework, you need to install and try out the SNAP network analysis tools. We strongly encourage you to use Snap.py for Python (available from `http://snap.stanford.edu/snappy/`). However, you can also use SNAP for C++ (available from `http://snap.stanford.edu/snap/download.html`)

1 Analyzing the Wikipedia voters network [27 points]

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

Using one of the network analysis tools above, load the Wikipedia voting network. Note that Wikipedia is a directed network. Formally, we consider the Wikipedia network as a directed graph $G = (V, E)$, with node set V and edge set $E \subset V \times V$ where (edges are ordered pairs of nodes). An edge $(a, b) \in E$ means that user a voted on user b.

To make our questions clearer, we will use the following small graph as a running example: $G_{\text{small}} = (V_{\text{small}}, E_{\text{small}})$, where $V_{\text{small}} = \{1, 2, 3\}$ and $E_{\text{small}} = \{(1, 2), (2, 1), (1, 3), (1, 1)\}$.

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

1. *The number of nodes in the network.* (G_{small} has 3 nodes.)
2. *The number of nodes with a self-edge (self-loop), i.e., the number of nodes $a \in V$ where $(a, a) \in E$.* (G_{small} has 1 self-edge.)

3. *The number of directed edges in the network, i.e., the number of ordered pairs* $(a, b) \in E$ *for which* $a \neq b$. (G_{small} has 3 directed edges.)
4. *The number of undirected edges in the network, i.e., the number of unique* unordered *pairs* (a, b), $a \neq b$, *for which* $(a, b) \in E$ *or* $(b, a) \in E$ *(or both).* If both (a, b) and (b, a) are edges, this counts a single undirected edge. (G_{small} has 2 undirected edges.)
5. *The number of reciprocated edges in the network, i.e., the number of unique unordered pairs of nodes* (a, b), $a \neq b$, *for which* $(a, b) \in E$ *and* $(b, a) \in E$. (G_{small} has 1 reciprocated edge.)
6. *The number of nodes of zero out-degree.* (G_{small} has 1 node with zero out-degree.)
7. *The number of nodes of zero in-degree.* (G_{small} has 0 nodes with zero in-degree.)
8. *The number of nodes with more than 10 outgoing edges (out-degree* > 10*).*
9. *The number of nodes with fewer than 10 incoming edges (in-degree* < 10*).*

Each sub-question is worth 3 points.

2 Further Analyzing the Wikipedia voters network [33 points]

For this problem, we use the Wikipedia voters network. If you are using Python, you might want to use NumPy, SciPy, and/or Matplotlib libraries.

1. (18 points) Plot the distribution of out-degrees of nodes in the network on a log-log scale. Each data point is a pair (x, y) where x is a positive integer and y is the number of nodes in the network with out-degree equal to x. Restrict the range of x between the minimum and maximum out-degrees. You may filter out data points with a 0 entry. For the log-log scale, use base 10 for both x and y axes.
2. (15 points) Compute and plot the least-square regression line for the out-degree distribution in the log-log scale plot. Note we want to find coefficients a and b such that the function $\log_{10} y = a \cdot \log_{10} x + b$, equivalently, $y = 10^b \cdot x^a$, best fits the out-degree distribution. What are the coefficients a and b? For this part, you might want to use the method called `polyfit` in NumPy with `deg` parameter equal to 1.

3 Finding Experts on the Java Programming Language on StackOverflow [40 points]

Download the StackOverflow network `stackoverflow-Java.txt.gz`: `http://snap.stanford.edu/class/cs224w-data/hw0/stackoverflow-Java.txt.gz`. An edge (a, b) in the network means that person a endorsed an answer from person b on a Java-related question.

Using one of the network analysis tools above, load the StackOverflow network. Note that StackOverflow is a directed network.

Compute and print out the following statistics for the `stackoverflow-Java` network:

1. *The number of weakly connected components in the network.* This value can be calculated in Snap.py via function `GetWccs`.

2. *The number of edges and the number of nodes in the largest weakly connected component.* The largest weakly connected component is calculated in Snap.py with function `GetMxWcc`.
3. *IDs of the top 3 most central nodes in the network by PagePank scores.* PageRank scores are calculated in Snap.py with function `GetPageRank`.
4. *IDs of the top 3 hubs and top 3 authorities in the network by HITS scores.* HITS scores are calculated in Snap.py with function `GetHits`.

Each sub-question is worth 10 points.

You can find more details about this exercise on the Snap.py tutorial page: `http://snap.stanford.edu/proj/snap-icwsm/`. As an extra exercise, extend the tutorial to find experts in other programming languages or topics.