CS224W: Social and Information Network Analysis **Fall 2011**

Assignment 3: Competition

Due 9:30am Monday November 14, 2011

General Instructions

This assignment is for extra credit and is not mandatory. The assignment must be done individually and submitted by the date specified above. No late days can be used.

The five students with the highest scores will get extra credit: 10% for first place, 8% for the second place, 6% for the third, and 4% for fourth and fifth. Depending on the number of solutions we receive, we will make the extra credit policy more generous (for example, top 10 solutions get extra credit). In the case of ties, the extra credit from the given positions will be evenly distributed among the students in the tie: if two students tie for first place, each of them will get 9% and the third best score will get 6%.

You are not allowed to use any graph alignment software, but can use packages for general network analysis like NetworkX, JUNG, or SNAP.

Aligning networks

In this exercise your goal is to find correspondences between the nodes of two different networks. You are given two networks F and G and your goal will be to find a *one to one* mapping between the nodes in F and those in G. The basic idea is that if a node $u \in F$ maps to a node $u' \in G$, then the neighbors of u should also map to the neighbors of u'.

You can think of this problem as social network de-anonimization. For example, assume F and G are two online social networks both with exactly the same set of users but not necessarily having the same set of edges. Assume you know the identities of users in F. Your goal is to match the users of F to those in G based on the structure of the two networks. Your intuition is that every user has a similar connectivity structure in both networks. This means that graphs F and G are "similar" and thus the same user x has a similar set of friends in both F and G. An example of one possible matching is shown in Figure 1.

The problem you are trying to solve here is of great interest. You can think of it as de-anonymizing social networks or finding pairs of nodes that have similar structural role. In biology it is common to compare protein or gene interaction networks across two different species F and G, where for one species F we know the labels of the proteins (nodes) and we would like to discover the labels of the proteins in G. Again the intuition is that the same protein interacts with a similar set of proteins in each network.

Task

You will be given two directed graphs $F(V_F, E_F)$, $G(V_G, E_G)$. Both graphs have the same number of nodes ($|V_F| = |V_G|$) but a different number of edges ($|E_F| \neq |E_G|$). We will also give you a small set M of pairs (u_i, v_i) ($u_i \in V_F, v_i \in V_G$) of true node correspondences. That is, we know that node u_i in V_F corresponds to node v_i in V_G. Your task now is to discover the remaining *one-to-one* node correspondences. This means that each node in F corresponds to exactly one node in G (and vice versa).

Given that the course staff knows the correct correspondences between the nodes of F and G, you will be evaluated on how many of the correspondences you found are indeed correct.

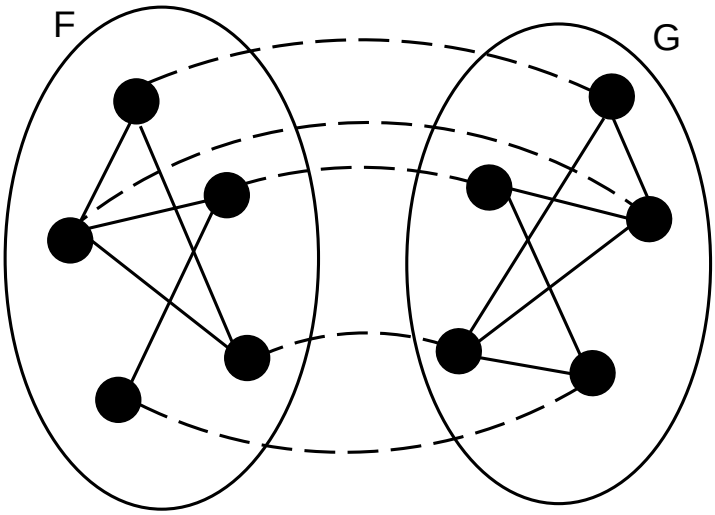

Figure 1: An example of an alignment of networks F and G.

Data

You will be finding correspondences between two graphs derived from the French and German versions of Wikipedia. Each node is a Wikipedia page and directed edges are hyperlinks between Wikipedia articles. And your goal is to identify corresponding articles — for example, an article about Paris in German Wikipedia probably links to a similar set of articles as a Paris article in French Wikipedia.

The dataset contains three files: `F.txt`, `G.txt` and `M.txt`. All files are TAB separated and each line contains a pair of node ids.

```
<node_1> \t <node_2>
```

In `F.txt` (and `G.txt`) each line gives a directed edge between the corresponding nodes `<node_1>` and `<node_2>`. In `M.txt` each line describes that `<node_1>` in F corresponds to `<node_2>` in G.

You can download the files from the course website.

What to submit

Use the submission website https://www.stanford.edu/class/cs224w/submit to submit your solution. Your submission should consist of the following three files (replace `SUNetID` with your SUNetID):

1. `matching_SUNetID.txt` : a text file with your final answers (see below)
2. `writeup_SUNetID.pdf` : Writeup describing your solution/algorithm
3. `code_SUNetID.zip` : Zip file with all the code.

The `matching_SUNetID.txt` should have $N = |V_G|$ lines (note that $|V_G| = |V_F|$) of the form:

```
<g_1>  <f(g_1)>
<g_2>  <f(g_2)>
.
.
<g_N>  <f(g_N)>
```

Where each line contains a pair of corresponding nodes. This means that for every node $g_i \in V_G$ we output the corresponding node $f(g_i) \in V_F$.

Ideas about possible approaches

Interestingly it is not known whether the general category of graph matching problems is P or NP-complete. Generally, there are no exact and efficient methods for solving it. However, there are several interesting heuristics that provide reasonably good results. Your job is to find your own heuristic and report its performance on the given data.

First useful strategy is to use the provided set M for a training and evaluation. You can split M into two sets M_t and M_e. You can then use correspondences in M_t to "seed" your algorithm and discover the remaining correspondences. Then you can check the discovered correspondences against those in M_e and get an idea about how well your method is doing.

Second, here are some ideas for simple heuristics you can try out and refine:

- **Similarity based:** For every node u (in G and F) you can construct a long feature vector that describes the connectivity structure around node u. For example, degree and clustering coefficient of u could be such features. Then for every node $u \in F$ you find node $u' \in G$ with the most similar feature vector, while making sure that you create a one-to-one matching.

- **Greedy:** Pick two nodes u and u' that are already matches (remember you have matches in M to start from). Among their neighbors, match the two which seem most similar. You can evaluate this similarity as the number of overlapping edges with the current matching, largest degree, etc. You should try to find a set of features that makes a specific pair stand out from the others.

- **Quality based:** You can define the score of a given matching to be the number of "squares" you discovered. A "square" is a cycle on 4 nodes (u, u', v', v) where (u, u') are connected by an undirected match edge, (u', v') are connected by a directed edge in G, (v', v) is a match edge and (u, v) are connected by a directed edge in F. Now you can think of heuristic optimization methods that try to increase the score of the matching. Note, that you can also generalize the method beyond counting squares.

- **Partitioning:** Split the nodes into subsets according to their degree, distance from the ground truth set, using a clustering algorithm. Collapse the subsets into supernodes and attempt a matching between these supersets. Then, try to find a matching for each subset independently (if they are small, you might be able to run a brute force algorithm).