Social Computing	Homework 2: Link Prediction
CS5750/IS4700 Fall 2013	August 31, 2013

This project is due at 11:59:59pm on October 16, 2013 and is worth 5% of your grade. You must complete it with a partner. You may only complete it alone or in a group of three if you have the instructor's explicit permission to do so for this project.

1 Description

In this assignment, we will be exploring different techniques for link prediction. Your job is to write a program that, given a network, predictions the next few links that will be established. Your program, called `5750linkpred` will read input from a file given as a command-line argument, open the file and read in the network, and then output the next links that it thinks will be formed. There are very specific requirements for the format of the input and the output; these are included below.

2 Requirements

Your program will accept two arguments: a filename and the number of links to predict. Your program will read in the graph file in the format described below, and will output the predicted links in the output described below. The instructors have a suite of test networks, and your program's output will be compared against the actual links that are created (i.e., the networks that you are being tested on are real networks). The test script is same code that your project will be graded with, so you should ensure that your program's output is accepted by the test script in order to achieve a good score.

2.1 Starter code

Very basic starter code and sample data for the assignment is available on the CCIS Linux network at `/course/cs5750f13/code/homework2`. You are free to implement this project in any language you choose (assuming the CCIS Linux machines have the necessary compiler and library support) and to use any techniques that you choose. You may also use any existing graph libraries to *store and query* the graph; you *must not* use these libraries to actually implement the prediction. All prediction code must be written by you. If you have any questions about specific situations or libraries, please contact the instructors.

To get started, you should copy down this directory into your own local directory (i.e., `cp -r /course/cs5750f13/code/homework2 ~/cs5750`). The `Makefile` is configured to test your code on the reference input (via the `test` target). You should feel free to edit the `Makefile` so that it will compile your code (or do other things). However, you should not change the `test` target. Your executable program must be named `5750linkpred`, and must be executable by running `./5750linkpred`.

2.2 Input format

Your `5750linkpred` program will accept exactly two arguments: a filename and a number of links to predict (in that order). The file will represent a graph, which will be a number of lines of the

form:

```
node1 [tab] node2 [\n]
```

The node1 and node2 labels can be of the form (in regular expression syntax):

```
[A-Za-z0-9-_]+
```

The graph files can be arbitrarily large, and the node labels can be of any length (greater than 0 characters). The graphs will be *directed*, meaning a link from A to B does not imply a link in the reverse direction.

2.3 Requirements

Your program must output the specified number of links that it predicts will be formed. It should output the links it thinks are most likely first; the scoring function (described below) scores earlier links better than later links. Your program's output should printed to standard out, and should be in the format

```
node1 [tab] node2 [\n]
```

using the same node labels that were given in the input file. In the output, the order of the nodes matters (e.g., the link above represents a link from node1 to node2).

2.4 Scoring

Your program's predicted links will be compared against the ground truth that was observed in the network. Specifically, suppose that we asked your program to predict the next n links; we call this list $A = \{a_1, a_2, ...a_n\}$. We would compare this to the list of the next $10 \times n$ links that were actually formed (recall that link prediction is hard, so we are using a much larger list of ground truth to compare against); we call this list $T = \{t_1, t_2, ...t_{10n}\}$. For each of your predicted links a_i, if $a_i \in T$, you would be awarded $\frac{1}{\log i+1}$ points. Your total score is the sum of all of your points, normalized by the highest possible score. Formally, your score is

$$\frac{\sum_{a_i} \frac{1}{\log i+1} \text{ if } a_i \in T}{\sum_{a_i} \frac{1}{\log i+1}}$$

For example, suppose $n = 5$ and your program returned $A = \{a, b, c, d, e\}$. If b and e appeared in T (i.e., b and e were among the next 50 links that were actually created), your score would be

$$\frac{\frac{1}{\log 2+1} + \frac{1}{\log 5+1}}{\sum_{a_i} \frac{1}{\log i+1}} = \frac{0.91023 + 0.55811}{4.25372} = 0.34519$$

2.5 Extra credit

The three teams with the top overall scores on the grading tests will receive 10, 5, and 2 extra credit points, respectively.

2.6 Error handling

If your program is unable to open the file, or if the file does not exist, you should output

```
Error: Unable to open graph file.
```

and exit. If you encounter a graph file which does not meet the specification listed above, you should output

```
Error: Malformed graph file.
```

and exit. If you encounter any other error while running the program, you should output

```
Error: [meaningful description of the error]
```

3 Implementation hints

You should develop your client program on the CCIS Linux machines, as these have the necessary compiler and library support. You are welcome to use your own Linux/OS X machines, but you are responsible for getting your code working, and your code *must* work when graded on the CCIS Linux machines. If you do not have a CCIS account, you should get one ASAP in order to complete the project. If you are using C, your code must be `-Wall` clean on `gcc`. Do not ask the TA for help on (or post to the forum) code that is not `-Wall` clean unless getting rid of the warning is what the problem is in the first place.

You should be careful to avoid *overfitting* on the training data that we give you. Your program will be graded on networks beyond just the ones that you are tested on; thus, any network-specific rules would likely result in poor performance on the testing graphs.

4 Submitting your project

4.1 Registering your team

You and your partner should first register as a team by running the `/course/cs5750f13/bin/register` script. You should pick out a team name (no spaces or non-alphanumeric characters, please) and run

```
/course/cs5750f13/bin/register homework2 <teamname>
```

This will either report back success or will give you an error message. If you have trouble registering, please contact the course staff.

You must register your team by 11:59:59pm on September 30, 2013.

4.2 Final submission

For the final submission, you should submit you (thoroughly documented) code along with a plain-text (no Word or PDF) README file. In this file, you should describe your high-level approach, the challenges you faced, a list of properties/features of your design that you think is

good, and an overview of how you tested your code. In your README, you should also point out any extra features of your project that you have implemented.

You should submit your project by running the `/course/cs5750f13/bin/turnin` script. Specifically, you should create a `homework2` directory, and place all of your code and README files in it. Then, run

```
/course/cs5750f13/bin/turnin homework2 <dir>
```

Where `<dir>` is the name of the directory with your submission. Again, the script will print out every file that you are submitting, make sure that it prints out all of the files you wish to submit!

You must submit your project by 11:59:59pm on October 16, 2013.

5 Grading

The grading in this project will consist of

75% Program functionality

15% Performance

10% Style and documentation

6 Advice

A few pointers that you may find useful while working on this project:

- Check the Piazza forum for question and clarifications. You should post project-specific questions there first, before emailing the course staff.
- Finally, get started early!