TIETS43 Recommender Systems Fall 2017

Lucas Machado
Student Number 413875

Assignment 3

Problem 2 (friends2.py)

Instructions to run the code

Install Python in your computer if you don't already have it. The code was developed and tested with Python 3.6.3. The packages being used are `pandas` and `networkx`. A default installation of anaconda should already contain these packages.

First, we need to create a Graph object from the txt file. For that, execute the following command *(Assuming that* `facebook-links.txt` *is in the same folder as* `friends2.py`*. Otherwise, pass the path as argument)*:

```
python friends2.py facebook-links.txt
```

Then, we can get recommendations of friends for a determined user, **based on influence score algorithm** and represented by *<user_id>*, with the following command:

```
python friends2.py graph.pickle <user_id>
```

Example

```
$ python friends2.py graph.pickle 42

The recommended friendships, in order, are:
62756, 18811, 85, 14546

Out of curiosity, their scores were respectively:
0.5, 0.423246536747, 0.397782562381, 0.386751819511
```

Assumptions and explanations

A recommendation system like this probably have a noSQL graph database for storing and computing the relations of friendship between users efficiently. To emulate that, I used `networkx` Python package to create a Graph object after reading all of the text file. That Graph object is saved in binary format in the `graph.pickle` file to avoid reprocessing the txt file every time we need to calculate a recommendation. Because of that, the first instruction is to generate that Graph object.

After that, it is fairly easy and fast to use the package functions to find neighbours and neighbours of neighbours from a particular node. Then, we can fetch the common neighbours between two nodes to calculate the desired score.

The information is organised and sorted in a `pandas` DataFrame since it is easy to do data manipulations within it.

Top four recommendations to particular users using *number of common friends* (i) algorithm:

User	Recommendation	Respective scores
20341	14209, 14273, 14275, 14276	47, 46, 46, 43
33722	32620, 34691, 11168, 13817	43, 42, 41, 41
35571	24197, 7959, 35573, 35747	8, 7, 7, 7
25017	17387, 27089, 26799, 22349	26, 22, 21, 19

*Note: These results might differ from Assignment 2 because I found a syntax bug in my code. Where it was "**second_degree != user_id & second_degree not in friends**" I corrected to "**second_degree != user_id and second_degree not in friends**". Sorry!

Top four recommendations to particular users using *influence* (ii) algorithm:

User	Recommendation	Respective scores
20341	37590, 44260, 14275, 14245	0.560274278634, 0.5, 0.482580463474, 0.444541262888
33722	27500, 38234, 34691, 32620	0.531244851481, 0.500713198385, 0.493341048584, 0.476205825405
35571	24197, 35573, 41061, 41062	0.238705126226, 0.200243587765, 0.17619047619, 0.17619047619
25017	27100, 24947, 27089, 17387	0.465723881813, 0.4415491347, 0.44126071851, 0.405478124567

Problem 3 (group.py)

Instructions to run the code

Again, we will need the same environment as in Problem 2.

First, we need to create a Graph object from the txt file (if not already created). For that, execute the following command *(Assuming that* `facebook-links.txt` *is in the same folder as* `group.py`*. Otherwise, pass the path as argument)*:

```
python group.py facebook-links.txt
```

Then, we can get recommendations of friends for a determined group of users, represented by *<user_id>* parameters, with the following command and the chosen aggregation method (*<method>* can be either 'borda' or 'average'). **The aggregation is based on individual recommendations with the influence algorithm**. There can be many users as wanted:

```
python group.py graph.pickle <method> <user_id> <user_id> <user_id> …
```

Top five recommendations to a particular user group using *borda count* (a) algorithm:

Users in group	20341 33722 35571 25017
Recommendation	24197, 27100, 27500, 37590, 24947
Respective scores	4, 4, 4, 4, 3

Top five recommendations to a particular user group using *average* (b) algorithm:

Users in group	20341 33722 35571 25017
Recommendation	37590, 27500, 38234, 44260, 34691
Respective scores	0.560274278634, 0.531244851481, 0.500713198385, 0.5, 0.493341048584