Project 3: Friend Recommendation

Note: You may work in a group of up to three students for this project, but there must be someone who you have not worked with in the group. Every group (triples, pairs, or individuals) needs to send me an email saying who they are working with.

Introduction: Collaborative Filtering

Every web company nowadays seems to want their own social network. In addition to use peer pressure and FOMO (the Fear Of Missing Out) to entice people to stay on the website, many companies also use the social information to collect additional information about their users. From recommending new songs and artists to predicting future buying patterns, these *collaborative filtering* algorithms operate on the premise the preferences of other users can be used to make inferences about individuals.

For example, consider how Netflix-like system might recommend films. Let's say that the system has six users and six movies, and each user has watched several films:

User	Howl's Moving Castle	Good Will Hunting	Lost in Translation	The Dark Knight	Wonder Woman	Zootopia
Alex		✔	✔	✔	✔	
Bailey	✔			✔	✔	
Cameron	✔	✔		✔	✔	✔
Darcy			✔			
Erin				✔		✔
Francis		✔			✔	

How might the system decide what film to recommend to Francis next? Netflix does not have to know anything about the films themselves - that Good Will Hunting and Lost in Translation are both dramas, that Christian Bale was in both The Dark Knight and Howl's Moving Castle. Instead, Netflix can rely entirely on what other users have watched. (In practice, Netflix likely uses both kinds of information, as well as any other information it can get, such as box office draw, critical response, etc.) One possible algorithm is to identify the user most like Francis, then recommend something else that they watched that Francis has not seen.

1. To find the user most like Francis, we might use the *Jaccard index*, a way to measure the "distance" between two collections of things. Given two collections, the Jaccard index divides the number of things in common by the total number of things. The larger the Jaccard index, the more "similar" the two collections are. Walking through this calculation for each user:
    - Alex has two films in common with Francis, and together they have seen a total of four films. This gives a Jaccard index is 2/4 = 0.50.
    - Bailey has one film in common with Francis, and together they have seen a total of four films. This gives a Jaccard index is 1/4 = 0.25.
    - Cameron has two films in common with Francis, and together they have seen a total of five films. This gives a Jaccard index of 2/5 = 0.40.

    - Darcy and Erin has no films in common with Francis, which gives a Jaccard index of 0.

  Since Alex's film-watching history has the highest Jaccard index with Francis', Alex is the most similar user.

2. Once the most similar user has been identified, we look at the other films that they have seen. In this case, Alex has seen two other films Francis has yet to see: Lost in Translation and The Dark Knight. Of these two films, four users have seen The Dark Knight, while only two users have seen Lost in Translation. The system will therefore recommend The Dark Knight to Francis.

In this project, you will be writing a basic friend recommendation system, similar to the ones you see on Facebook and Twitter. Unlike the Netflix example above, the thing you are recommending are other *users*.

Code Walkthrough

The majority of the code you will be writing is in the `SocialNetwork` class, which stores the users in the social network and which other users they follow. This is represented by the `users` dictionary member variable, where the keys are the usernames and the values are lists of usernames that a user follows. Although we call these "friends", these relationships are one-way; Alex will be Bailey's friend if Bailey follows Alex, but that does not mean that Alex is following Bailey.

There are several basic methods you should complete first:

- `list_users`, which returns a list of all users in the network
- `add_user`, which adds a new user to the network (with no initial friends)
- `add_friend`, which allows a user to follow a friend
- `get_friends`, which returns a list of all the friends of a user

The main method you must complete is `suggest_friend`, which suggests a new friend for a user. The algorithm we will use is analogous to the Netflix example above:

1. Find another user that is the most similar to the specified user, based on the Jaccard index of their friends.
2. Out of the most-similar-user's friends, find the one with the most followers that the specified user does not already follow.

Two additional functions/methods have been written for you:

- `create_network_from_file` reads an example network file and returns a `SocialNetwork` instance. Two such files are provided for you: `simple.network` contains a network of six users analogous to the Netflix example above, while `intermediate.network` contains a more complicated network of 26 users.
- `to_dot` returns a string in the Graphviz format. You can copy this string into the visualizer at http://viz-js.com/ to visually inspect the social network.

Finally, the example `main` function loads the simple network, prints out a visualizable version, and asks for a friend recommendation for Francis.

Submission

In addition to attaching the completed code, your email submission should also answer each of the following questions:

1. How good is the algorithm you implemented? Without using additional information, how else might we recommend new friends?
2. How might a similar system have been in use by Target to figure out that the 16 year old girl was pregnant?
3. What might you do to avoid being (for lack of better word) "profiled" in this way?

When you are done, please also fill out the project evaluation.