SYSC 3110 Project – Simulation of a file-sharing social network

The goal of this team project is to come up with a simulation environment for a document-sharing social network. In this environment, users will be able to search for documents, rank the search results according to various criteria, and "like" documents, as in Facebook. The social network is the result of users "following" each other, as in Twitter. We can represent a user's taste by a simple string (e.g., "jazz"), and a document will similarly have a "tag" string, unknown to the user until the document is evaluated by the user. Logically, a user should "like" a document when its tag matches the user's taste. The simulation should help us observe the evolution of the site over time, i.e. whether there is an incentive for "liking" documents, "following" other users, and whether it is possible for certain users to influence other users into using their own files.

Each simulation cycle, a user is chosen at random (this means a user may be picked multiple times) and is asked to "act". We will have two types of users: "consumers" and "producers". A consumer "acts" the following way:

1- search the simulation for the "top" *k* documents. There will be many criteria for ranking the top documents, such as the popularity of the documents (i.e., the number of "likes" it has), the popularity of the users ranking the documents (i.e., number of times he/she is "followed"), the distance of the user in the social network (i.e., I trust my immediate friends more than the friends of my friends), the "like" similarity (i.e., if we tend to "like" the same documents, I will trust you when you "like" documents I haven't seen yet) and the "follow" similarity.

2- calculate a *payoff*, as a way to calculate the performance of the user's searches based on the relevance of the documents that were returned (i.e., whether they matched the user's taste or not). I leave it up to you to decide on a good payoff function; one simple way is to add one point for each document in the top k that hasn't been seen by the user yet and that matches the user's taste.

3- decide whether to "like" any of the documents that were retrieved in the search, or to "follow" any of the users who happened to "like" any of those documents. The point is that by "liking" or "following", the user is hoping to improve the relevance of search results next time, especially depending on the ranking criteria used in step 1. One simple strategy, as mentioned earlier, is simply to "like" any document which "tag" matches the "taste" of the user, and also to "follow" users who also "liked" such documents. You are allowed to investigate other strategies as well!

As for the producer, it first produces a new document that matches its own taste, then follows steps 1 and 3 above, but for the purpose of influencing consumers by "liking" certain documents or "following" certain users that "like" them. There are a couple of "like"/"follow" strategies that you have to implement here for producers:

a) act like a consumer, by "liking" your own document and others that match your taste, and following "users" that like them. The idea here is that if there are enough producers who do this, the popularity of these documents and of the users

that produce them will get high enough that they will influence popularity rankings used by consumers.

b) pick a taste other than your own, and "like" documents that match them and users that "like" them. Here the idea is to influence the similarity rankings of the consumers.

The payoff for a producer should be updated each time a consumer "follows " him/her or "likes" a document he/she posted.

Note that there are many parameters one can play with: the number of producers and consumers, the strategies they follow, the number of different tastes and tags, the number of iterations for the simulator, the initial configuration of the network (e.g., how the network is "seeded"), etc. It's the variety of these parameters and the effects they can have that makes simulations interesting!

The user interface for the simulator should be allow for the user to set these parameters, to launch a simulation and step through it, and it should display and update the social network, and the relation between users and the documents they "like" at each step. It should also distinguish between producers and consumers. Finally, it should be able to graph the evolution of the accumulated payoff metric over time for a given user. We're not expecting anything fancy; just make sure the interface is functional!

The project is divided into 4 iterations, each ending with a milestone corresponding to deliverables that will be graded. You will be able to use the TA feedback from iteration i for iteration i+1.

Milestone 1: A simple console-based version. Only the document popularity ranking strategy for consumers and one for producers is required at this point. Use strategy a) for liking/following for both producers and consumers.

- Deliverables: code (source + executable in a jar file), documentation complete with UML diagrams (including with complete variable and method signatures), and a readme file (see explanation below), all in one zip file. The code and the design are allowed to "smell" at this point.
- Deadline: Monday Oct 19th. Weight: 15% of the overall project grade.

Milestone 2: GUI and unit tests.

- Deliverables: source (including tests) + updated documentation and diagrams + readme file, all in one zip file. In particular, document the changes you made to your design and explain why.
- Deadline: Monday Nov 9th. Weight: 15% of the overall project grade.

Milestone 3: all ranking strategies (step 1 of consumer and producer actions) and strategy b) for producers should be provided. It should be possible, through the GUI, to mix-and-match various consumer and producer profiles with different strategies.

- Deliverables: code + tests + documentation + readme file. Make sure that you document the changes since the last iteration and the reasons for those changes. Your code and design should be "smell-free" at this point.

- Deadline: Monday Nov 23rd. Weight: 20% of the overall project grade.

Milestone 4: Two more features! Ability to step back (like an "undo" function) and ability to save/restore a simulation at any point.

- Deliverables: code + design + documentation + readme file.
- Deadline: Monday Dec 7th. Weight: 25% of the overall project grade.

Milestones must contain all necessary files and documentation, even those items that are unchanged from previous milestones. Missing files cannot be submitted after the deadline, no exceptions. Verify your submission contains all necessary files (in particular, don't forget to include your source code!) before submitting on cuLearn.
The "readme" file, listed as a deliverable for each iteration, is typically a short text file that lists and describes: the rest of the deliverables, the authors, the changes that were made since the previous deliverable, the known issues (known issues are graded less severely than undocumented ones!) and the roadmap ahead.
"Documentation" includes up-to-date UML diagrams, detailed descriptions of design decisions made, complete user manuals, and javadoc documentation.

Note that nobody is stopping you from working ahead of schedule! In fact, iteration i+1 will very often give you good insight into iteration i.

This is a **team** project. Each team should have 4 members (or exceptionally 3). A project's success obviously depends on contributions from each member! So divide the work and cooperate. **Each contribution (source code, documentation, etc.) must contain the name of its author**: we will use this to determine whether there is any significant difference in the quality and quantity of the contributions of the team members. If any such difference is detected, the individual grades will be adjusted accordingly.
You are expected to use Github to manage your project (version control, issue-tracking, etc...), but the deliverables for each milestone should also be submitted for marking on cuLearn. Each team will be given a private Github team repository. A TA and/or the instructor will also be members of each of the Github teams, so that they can track your use of the tool, verify that all group members are contributing, and their feedback will be given by opening new issues.

The marking scheme for each iteration will be comprised of: completeness of the deliverables, the quality of the deliverables, and, for iterations 2 to 4, we also look at whether you took into account the feedback you received from the TA in the previous iteration.