CIT 594 – Homework 4

Milestones

This homework has three milestones with the following points and deadlines:

Milestone 1 – Design (30 points)
Due Monday, Mar. 21, at 12pm

Milestone 2 – Implementation (70 points)
Due Monday, Mar. 28, at 12pm

Milestone 3 – Changing Requirements (30 points)
Due Friday, Apr. 1, at 12pm

Recommender Systems

Recommender Systems have become very popular in recent times. Broadly, the goal of a recommender system is to recommend new movies, friends, things to buy, etc. to a specific user based on their interests, background, history, etc. An excellent resource on recommender systems that talks about the history, techniques, etc. is:

Michael D. Ekstrand, John T. Riedl, and Joseph A. Konstan. 2011. Collaborative Filtering Recommender Systems. Found. Trends Hum.-Comput. Interact. 4, 2 (February 2011), 81-173. DOI=http://dx.doi.org/10.1561/1100000009

This article is available at http://files.grouplens.org/papers/FnT%20CF%20Recsys%20Survey.pdf

A lot of the text below is taken and adapted from the above article.

Introduction and Background

The information domain for a collaborative filtering system consists of users which have expressed preferences for various items. A preference expressed by a user for an item is called a rating and is frequently represented as a (User, Item, Rating) triple. These ratings can take many forms, depending on the system in question. Some systems use real- or integer-valued rating scales such as 0–5 stars, while others use binary or ternary (like/dislike) scales. Unary ratings, such as “has purchased”, are particularly common in e-commerce deployments as they express well the user’s purchasing history absent ratings data.

The set of all rating triples forms a sparse matrix referred to as the ratings matrix. (User, Item) pairs where the user has not expressed a preference for (rated) the item are unknown values in this matrix. The table below shows an example ratings matrix for three users and four movies in a movie recommender system; cells marked '?' indicate unknown values (the user has not rated that movie).

	Batman Begins	Alice in Wonderland	Dumb and Dumber	Equilibrium
User A	4	?	3	5
User B	?	5	4	?
User C	5	4	2	?

Table 1: Sample ratings matrix (on a 5-star scale).

In describing use and evaluation of recommender systems, we typically focus on two tasks. The first is the *predict* task: given a user and an item, what is the user's likely preference for the item. The second task is the *recommend* task: given a user, produce the best ranked list of n items for the user's need. An n-item recommendation list is not guaranteed to contain the n items with the highest predicted preferences, as predicted preference may not be the only criteria used to produce the recommendation list.

We will use a consistent mathematical notation for referencing various elements of the recommender system model. The universe consists of a set U of users and a set I of items. I_u is the set of items rated or purchased by user u, and U_i is the set of users who have rated or purchased i. The rating matrix is denoted by R, with $r_{u,i}$ being the rating user u provided for item i, r_u being the vector of all ratings provided by user u, and r_i being the vector of all ratings provided for item i (the distinction will be apparent from context). $\bar{r}_u$ and $\bar{r}_i$ are the average of a user u or an item i's ratings, respectively.

Collaborative Filtering

Collaborative filtering (CF) is a popular recommendation algorithm that bases its predictions and recommendations on the ratings or behavior of other users in the system. The fundamental assumption behind this method is that other users' opinions can be selected and aggregated in such a way as to provide a reasonable prediction of the active user's preference. Intuitively, they assume that, if users agree about the quality or relevance of some items, then they will likely agree about other items — if a group of users likes the same things as Mary, then Mary is likely to like the things they like which she hasn't yet seen.

User–user collaborative filtering, also known as k-Nearest Neighbor collaborative filtering, was the first of the automated CF methods. User–user CF is a straightforward algorithmic interpretation of the core premise of collaborative filtering: find other users whose past rating behavior is similar to that of the current user and use their ratings on other items to predict what the current user will like. To predict Mary's preference for an item she has not rated, user–user CF looks for other users who have high agreement with Mary on the items they have both rated. These users' ratings for the item in question are then weighted by their level of agreement with Mary's ratings to predict Mary's preference.

Besides the rating matrix R, a user–user CF system requires:

1. A similarity function s computing the similarity between two users.
2. A method for using similarities and ratings to generate predictions.

Computing Predictions

To generate predictions or recommendations for a user u, user–user CF first uses s to compute a neighborhood N $\subseteq$ U of neighbors of u. Once N has been computed, the system combines the ratings of users in N to generate predictions for user u's preference for an item i. This is typically done by computing the weighted average of the neighboring users' ratings i using similarity as the weights:

$$p_{u,i} = \bar{r}_u + \frac{\sum_{u' \in N} s(u,u')(r_{u',i} - \bar{r}_{u'})}{\sum_{u' \in N} |s(u,u')|}$$

Subtracting the user's mean rating compensates for differences in users' use of the rating scale (some users will tend to give higher ratings than others). There remains the question of how many neighbors to select. In some systems, all users are considered as neighbors; in other systems, neighborhoods are selected for each item based on a similarity threshold or neighborhood size such that Ni is the k users most similar to u who have rated the target item i. Limiting neighborhood size can result in more accurate predictions, as the neighbors with low correlation introduce more noise than signal into the process. The particular threshold to use is domain- and system-specific, so analysis of a relevant data set is needed to choose the neighborhood size for a particular deployment. In offline analysis of available movie ratings data, Herlocker et al. found k = 20 to be a good value; values in the 20–50 range are a reasonable starting point in many domains.

Computing User Similarity

A critical design decision in implementing user–user CF is the choice of similarity function. Several different similarity functions have been proposed and evaluated in the literature.

Pearson correlation

This method computes the statistical correlation (Pearson's r) between two users' common ratings to determine their similarity. The correlation is computed by the following:

$$s(u,v) = \frac{\sum_{i \in I_u \cap I_v} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I_u \cap I_v} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{i \in I_u \cap I_v} (r_{v,i} - \bar{r}_v)^2}}$$

The limitation of Pearson correlation is that it suffers from computing high similarity between users with few ratings in common.

Example

	Batman Begins	Alice in Wonderland	Dumb and Dumber	Equilibrium
User A	4	?	3	5
User B	?	5	4	?
User C	5	4	2	?
User D	2	4	?	3
User E	3	4	5	?

Consider the ratings matrix shown above. We want to find User C's prediction for *Equilibrium* ($p_{C,e}$) with the following configuration:

- Pearson correlation.
- Neighborhood size of 2.
- Weighted average with mean offset.

C's mean rating is 3.667. There are only two users who have rated Equilibrium, and therefore only two candidate users for the neighborhood: A and D.

$s(C, A)$ = 0.781 and $s(C, D)$ = −0.515.

The prediction $p_{C,e}$ is therefore computed as follows:

$$p_{C,e} = \bar{r}_C + \frac{s(C,A)\left(r_{A,e} - \bar{r}_A\right) + s(C,D)\left(r_{D,e} - \bar{r}_D\right)}{|s(C,A)| + |s(C,D)|}$$

$$p_{C,e} = 3.667 + \frac{0.781 * (5-4) + -0.515 * (3-3)}{0.781 + 0.515}$$

$$p_{C,e} = 4.269$$

The Assignment

Your assignment is to build a system that will provide recommendations for a given user. In particular, your system should do the following:

1. We will provide a real-world data set containing users, items, ratings.
2. You should read in the file and process the data as you see fit.
3. The main functionality that your system should support is the following:
    a. Given a user u and item i, what is the system's prediction for the user's likely preference of that item?
    b. Given a user u and a threshold n, what are the n-highest predicted preferences for that user?

Milestone 1 – Design

For milestone 1, the goal is to do an initial design for your recommender system.

Please follow the process described in class, which is summarized below:

1. Conduct an analysis of the potential system and think about how potential users will use it. This will give you a better understanding of the problem domain. Don't focus on the design or implementation at this stage. Feel free to look at recommender system examples on websites like Amazon, Netflix, and Facebook. Please describe in simple English how a user will interact with your system (no more than 2 paragraphs) in your readme.txt file.
2. Next, start thinking about design. We encourage you to come up with CRCs, but you're not required to. Keeping in mind good design principles, create the following:
    a. A UML Class Diagram (15 points)
    b. A UML Sequence Diagram for the following task: Given a user u and item i, generate a prediction using the techniques described above (Pearson Correlation, k-Nearest Neighbors, Weighted Average with Mean Offset) (10 points)
    c. A UML Sequence Diagram for any other task of your choosing (5 points)
3. Finally, in your readme.txt file, explain your design decisions.

Grading Criteria for Milestone 1

50% for UML – Do your diagrams follow the UML conventions? Are there sufficient details? …
50% for good design – Is your system well designed? …

Submission Instructions

Please submit the readme.txt file and your UML diagrams (exported as .png files).

Please create a folder called YOUR_PENNKEY. Places all your files inside this. Zip up this folder. It will thus be called YOUR_PENNKEY.zip. So, e.g., my homework submission would be swapneel.zip. Please submit this zip file via canvas.