Scouts, Promoters, and Connectors: The Roles of Ratings in Nearest Neighbor Collaborative Filtering

Bharath Kumar Mohan
Dept. of CSA
Indian Institute of Science
Bangalore 560 012, India
mbk@csa.iisc.ernet.in

Benjamin J. Keller
Dept. of Computer Science
Eastern Michigan University
Ypsilanti, MI 48917, USA
bkeller@emich.edu

Naren Ramakrishnan
Dept. of Computer Science
Virginia Tech, Blacksburg
VA 24061, USA
naren@cs.vt.edu

ABSTRACT

Recommender systems aggregate individual user ratings into predictions of products or services that might interest visitors. The quality of this aggregation process crucially affects the user experience and hence the effectiveness of recommenders in e-commerce. We present a novel study that disaggregates global recommender performance metrics into contributions made by each individual rating, allowing us to characterize the many roles played by ratings in nearest-neighbor collaborative filtering. In particular, we formulate three roles—*scouts*, *promoters*, and *connectors*—that capture how users receive recommendations, how items get recommended, and how ratings of these two types are themselves connected (resp.). These roles find direct uses in improving recommendations for users, in better targeting of items and, most importantly, in helping monitor the health of the system as a whole. For instance, they can be used to track the evolution of neighborhoods, to identify rating subspaces that do not contribute (or contribute negatively) to system performance, to enumerate users who are in danger of leaving, and to assess the susceptibility of the system to attacks such as shilling. We argue that the three rating roles presented here provide broad primitives to manage a recommender system and its community.

Categories and Subject Descriptors

H.4.2 [**Information Systems Applications**]: Types of Systems—*Decision support*; J.4 [**Computer Applications**]: Social and Behavioral Sciences

General Terms

Algorithms, Human Factors

Keywords

Recommender systems, collaborative filtering, neighborhoods, user-based and item-based algorithms, scouts, promoters, connectors.

EC'06, June 11–15, 2006, Ann Arbor, Michigan, USA.

1. INTRODUCTION

Recommender systems have become integral to e-commerce, providing technology that suggests products to a visitor based on previous purchases or rating history. Collaborative filtering, a common form of recommendation, predicts a user's rating for an item by combining (other) ratings of that user with other users' ratings. Significant research has been conducted in implementing fast and accurate collaborative filtering algorithms [2, 7], designing interfaces for presenting recommendations to users [1], and studying the robustness of these algorithms [8]. However, with the exception of a few studies on the influence of users [10], little attention has been paid to unraveling the inner workings of a recommender in terms of the individual ratings and the roles they play in making (good) recommendations. Such an understanding will give an important handle to monitoring and managing a recommender system, to engineer mechanisms to sustain the recommender, and thereby ensure its continued success.

Our motivation here is to disaggregate global recommender performance metrics into contributions made by each individual rating, allowing us to characterize the many roles played by ratings in nearest-neighbor collaborative filtering. We identify three possible roles: *(scouts)* to connect the user into the system to receive recommendations, *(promoters)* to connect an item into the system to be recommended, and *(connectors)* to connect ratings of these two kinds. Viewing ratings in this way, we can define the contribution of a rating in each role, both in terms of allowing recommendations to occur, and in terms of influence on the quality of recommendations. In turn, this capability helps support scenarios such as:

1. *Situating users in better neighborhoods:* A user's ratings may inadvertently connect the user to a neighborhood for which the user's tastes may not be a perfect match. Identifying ratings responsible for such bad recommendations and suggesting new items to rate can help situate the user in a better neighborhood.

2. *Targeting items*: Recommender systems suffer from lack of user participation, especially in cold-start scenarios [13] involving newly arrived items. Identifying users who can be encouraged to rate specific items helps ensure coverage of the recommender system.

3. *Monitoring the evolution of the recommender system and its stakeholders:* A recommender system is constantly under change: growing with new users and

items, shrinking with users leaving the system, items becoming irrelevant, and parts of the system under attack. Tracking the roles of a rating and its evolution over time provides many insights into the health of the system, and how it could be managed and improved. These include being able to identify rating subspaces that do not contribute (or contribute negatively) to system performance, and could be removed; to enumerate users who are in danger of leaving, or have left the system; and to assess the susceptibility of the system to attacks such as shilling [5].

As we show, the characterization of rating roles presented here provides broad primitives to manage a recommender system and its community. The rest of the paper is organized as follows. Background on nearest-neighbor collaborative filtering and algorithm evaluation is discussed in Section 2. Section 3 defines and discusses the roles of a rating, and Section 4 defines measures of the contribution of a rating in each of these roles. In Section 5, we illustrate the use of these roles to address the goals outlined above.

2. BACKGROUND

2.1 Algorithms

Nearest-neighbor collaborative filtering algorithms either use neighborhoods of users or neighborhoods of items to compute a prediction. An algorithm of the first kind is called *user-based*, and one of the second kind is called *item-based* [12]. In both families of algorithms, neighborhoods are formed by first computing the similarity between all pairs of users (for user-based) or items (for item-based). Predictions are then computed by aggregating ratings, which in a user-based algorithm involves aggregating the ratings of the target item by the user's neighbors and, in an item-based algorithm, involves aggregating the user's ratings of items that are neighbors of the target item. Algorithms within these families differ in the definition of similarity, formation of neighborhoods, and the computation of predictions. We consider a user-based algorithm based on that defined for GroupLens [11] with variations from Herlocker *et al.* [2], and an item-based algorithm similar to that of Sarwar *et al.* [12].

The algorithm used by Resnick *et al.* [11] defines the similarity of two users u and v as the Pearson correlation of their common ratings:

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

where $\mathcal{I}_u$ is the set of items rated by user u, $r_{u,i}$ is user u's rating for item i, and $\bar{r}_u$ is the average rating of user u (similarly for v). Similarity computed in this manner is typically scaled by a factor proportional to the number of common ratings, to reduce the chance of making a recommendation made on weak connections:

$$sim'(u,v) = \frac{max(|\mathcal{I}_u \cap \mathcal{I}_v|, \gamma)}{\gamma} \cdot sim(u,v),$$

where $\gamma \approx 5$ is a constant used as a lower limit in scaling [2]. These new similarities are then used to define a static neighborhood $\mathcal{N}_u$ for each user u consisting of the top K users most similar to user u. A prediction for user u and item i is computed by a weighted average of the ratings by the neighbors

$$p_{u,i} = \bar{r}_u + \frac{\sum_{v \in V} sim'(u,v)(r_{v,i} - \bar{r}_v)}{\sum_{v \in V} sim'(u,v)} \quad (1)$$

where $V = \mathcal{N}_u \cap \mathcal{U}_i$ is the set of users most similar to u who have rated i.

The item-based algorithm we use is the one defined by Sarwar *et al.* [12]. In this algorithm, similarity is defined as the adjusted cosine measure

$$sim(i,j) = \frac{\sum_{u \in \mathcal{U}_i \cap \mathcal{U}_j} (r_{u,i} - \bar{r}_u)(r_{u,j} - \bar{r}_u)}{\sqrt{\sum_{u \in \mathcal{U}_i} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{u \in \mathcal{U}_j} (r_{u,j} - \bar{r}_u)^2}} \quad (2)$$

where $\mathcal{U}_i$ is the set of users who have rated item i. As for the user-based algorithm, the similarity weights are adjusted proportionally to the number of users that have rated the items in common

$$sim'(i,j) = \frac{max(|\mathcal{U}_i \cap \mathcal{U}_j|, \gamma)}{\gamma} \cdot sim(i,j). \quad (3)$$

Given the similarities, the neighborhood $\mathcal{N}_i$ of an item i is defined as the top K most similar items for that item. A prediction for user u and item i is computed as the weighted average

$$p_{u,i} = \bar{r}_i + \frac{\sum_{j \in J} sim'(i,j)(r_{u,j} - \bar{r}_j)}{\sum_{j \in J} sim'(i,j)} \quad (4)$$

where $J = \mathcal{N}_i \cap \mathcal{I}_u$ is the set of items rated by u that are most similar to i.

2.2 Evaluation

Recommender algorithms have typically been evaluated using measures of predictive accuracy and coverage [3]. Studies on recommender algorithms, notably Herlocker *et al.* [2] and Sarwar *et al.* [12], typically compute predictive accuracy by dividing a set of ratings into training and test sets, and compute the prediction for an item in the test set using the ratings in the training set. A standard measure of predictive accuracy is mean absolute error (MAE), which for a test set $\mathcal{T} = \{(u,i)\}$ is defined as,

$$MAE = \frac{\sum_{(u,i) \in \mathcal{T}} |p_{u,i} - r_{u,i}|}{|\mathcal{T}|}. \quad (5)$$

Coverage has a number of definitions, but generally refers to the proportion of items that can be predicted by the algorithm [3].

A practical issue with predictive accuracy is that users typically are presented with recommendation lists, and not individual numeric predictions. Recommendation lists are lists of items in decreasing order of prediction (sometimes stated in terms of star-ratings), and so predictive accuracy may not be reflective of the accuracy of the list. So, instead we can measure *recommendation* or *rank* accuracy, which indicates the extent to which the list is in the correct order. Herlocker *et al.* [3] discuss a number of rank accuracy measures, which range from Kendall's Tau to measures that consider the fact that users tend to only look at a prefix of the list [5]. Kendall's Tau measures the number of inversions when comparing ordered pairs in the true user ordering of

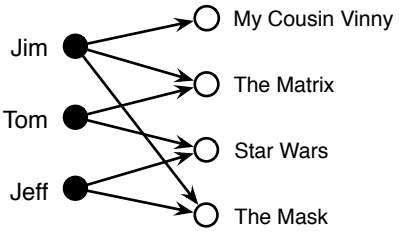

Figure 1: Ratings in simple movie recommender.

items and the recommended order, and is defined as

$$\tau = \frac{C - D}{\sqrt{(C + D + TR)(C + D + TP)}} \quad (6)$$

where C is the number of pairs that the system predicts in the correct order, D the number of pairs the system predicts in the wrong order, TR the number of pairs in the true ordering that have the same ratings, and TP is the number of pairs in the predicted ordering that have the same ratings [3]. A shortcoming of the Tau metric is that it is oblivious to the position in the ordered list where the inversion occurs [3]. For instance, an inversion toward the end of the list is given the same weight as one in the beginning. One solution is to consider inversions only in the top few items in the recommended list or to weight inversions based on their position in the list.

3. ROLES OF A RATING

Our basic observation is that each rating plays a different role in each prediction in which it is used. Consider a simplified movie recommender system with three users Jim, Jeff, and Tom and their ratings for a few movies, as shown in Fig. 1. (For this initial discussion we will not consider the rating values involved.) The recommender predicts whether Tom will like `The Mask` using the other already available ratings. How this is done depends on the algorithm:

1. An item-based collaborative filtering algorithm constructs a neighborhood of movies around `The Mask` by using the ratings of users who rated `The Mask` and other movies similarly (e.g., Jim's ratings of `The Matrix` and `The Mask`; and Jeff's ratings of `Star Wars` and `The Mask`). Tom's ratings of those movies are then used to make a prediction for `The Mask`.

2. A user-based collaborative filtering algorithm would construct a neighborhood around Tom by tracking other users whose rating behaviors are similar to Tom's (e.g., Tom and Jeff have rated `Star Wars`; Tom and Jim have rated `The Matrix`). The prediction of Tom's rating for `The Mask` is then based on the ratings of Jeff and Tim.

Although the nearest-neighbor algorithms aggregate the ratings to form neighborhoods used to compute predictions, we can disaggregate the similarities to view the computation of a prediction as simultaneously following parallel paths of ratings. So, irrespective of the collaborative filtering algorithm used, we can visualize the prediction of Tom's rating of `The Mask` as walking through a sequence of ratings. In

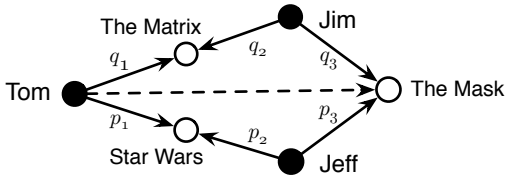

Figure 2: Ratings used to predict `The Mask` for Tom.

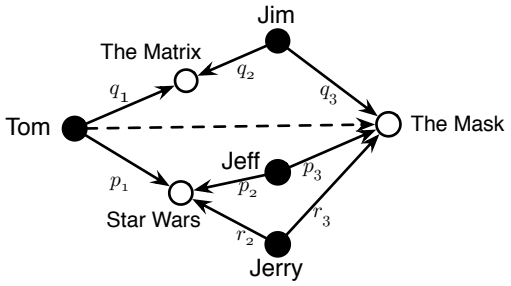

Figure 3: Prediction of `The Mask` for Tom in which a rating is used more than once.

this example, two paths were used for this prediction as depicted in Fig. 2: (p_1, p_2, p_3) and (q_1, q_2, q_3). Note that these paths are undirected, and are all of length 3. Only the order in which the ratings are traversed is different between the item-based algorithm (e.g., (p_3, p_2, p_1), (q_3, q_2, q_1)) and the user-based algorithm (e.g., (p_1, p_2, p_3), (q_1, q_2, q_3).) A rating can be part of many paths for a single prediction as shown in Fig. 3, where three paths are used for a prediction, two of which follow p_1: (p_1, p_2, p_3) and (p_1, r_2, r_3).

Predictions in a collaborative filtering algorithms may involve thousands of such walks in parallel, each playing a part in influencing the predicted value. Each prediction path consists of three ratings, playing roles that we call *scouts*, *promoters*, and *connectors*. To illustrate these roles, consider the path (p_1, p_2, p_3) in Fig. 2 used to make a prediction of `The Mask` for Tom:

1. The rating p_1 (Tom $\mapsto$ `Star Wars`) makes a connection from Tom to other ratings that can be used to predict Tom's rating for `The Mask`. This rating serves as a scout in the bipartite graph of ratings to find a path that leads to `The Mask`.

2. The rating p_2 (Jeff $\mapsto$ `Star Wars`) helps the system recommend `The Mask` to Tom by connecting the scout to the promoter.

3. The rating p_3 (Jeff $\mapsto$ `The Mask`) allows connections to `The Mask`, and, therefore, promotes this movie to Tom.

Formally, given a prediction $p_{u,a}$ of a target item a for user u, a *scout* for $p_{u,a}$ is a rating $r_{u,i}$ such that there exists a user v with ratings $r_{v,a}$ and $r_{v,i}$ for some item i; a *promoter* for $p_{u,a}$ is a rating $r_{v,a}$ for some user v, such that there exist ratings $r_{v,i}$ and $r_{u,i}$ for an item i, and; a *connector* for $p_{u,a}$

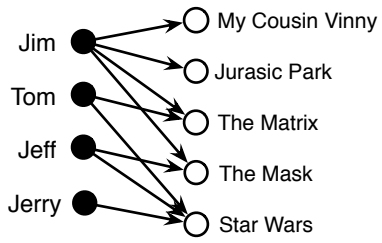

Figure 4: Scouts, promoters, and connectors.

is a rating $r_{v,i}$ by some user v and rating i, such that there exists ratings $r_{u,i}$ and $r_{v,a}$. The scouts, connectors, and promoters for the prediction of Tom's rating of `The Mask` are p_1 and q_1, p_2 and q_2, and p_3 and q_3 (respectively). Each of these roles has a value in the recommender to the user, the user's neighborhood, and the system in terms of allowing recommendations to be made.

3.1 Roles in Detail

Ratings that act as scouts tend to help the recommender system suggest more movies to the user, though the extent to which this is true depends on the rating behavior of other users. For example, in Fig. 4 the rating Tom $\mapsto$ `Star Wars` helps the system recommend only `The Mask` to him, while Tom $\mapsto$ `The Matrix` helps recommend `The Mask`, `Jurassic Park`, and `My Cousin Vinny`. Tom makes a connection to Jim who is a prolific user of the system, by rating `The Matrix`. However, this does not make `The Matrix` the best movie to rate for everyone. For example, Jim is benefited equally by both `The Mask` and `The Matrix`, which allow the system to recommend `Star Wars` to him. His rating of `The Mask` is the best scout for Jeff, and Jerry's only scout is his rating of `Star Wars`. This suggests that good scouts allow a user to build similarity with prolific users, and thereby ensure they get more from the system.

While scouts represent beneficial ratings from the perspective of a user, promoters are their duals, and are of benefit to items. In Fig. 4, `My Cousin Vinny` benefits from Jim's rating, since it allows recommendations to Jeff and Tom. `The Mask` is not so dependent on just one rating, since the ratings by Jim and Jeff help it. On the other hand, Jerry's rating of `Star Wars` does not help promote it to any other user. We conclude that a good promoter connects an item to a broader neighborhood of other items, and thereby ensures that it is recommended to more users.

Connectors serve a crucial role in a recommender system that is not as obvious. The movies `My Cousin Vinny` and `Jurassic Park` have the highest recommendation potential since they can be recommended to Jeff, Jerry and Tom based on the linkage structure illustrated in Fig. 4. Beside the fact that Jim rated these movies, these recommendations are possible only because of the ratings Jim $\mapsto$ `The Matrix` and Jim $\mapsto$ `The Mask`, which are the best connectors. A connector improves the system's ability to make recommendations with no explicit gain for the user.

Note that every rating can be of varied benefit in each of these roles. The rating Jim $\mapsto$ `My Cousin Vinny` is a poor scout and connector, but is a very good promoter. The rating Jim $\mapsto$ `The Mask` is a reasonably good scout, a very good connector, and a good promoter. Finally, the rating Jerry $\mapsto$ `Star Wars` is a very good scout, but is of no value as a connector or promoter. As illustrated here, a rating can have different value in each of the three roles in terms of whether a recommendation can be made or not. We could measure this value by simply counting the number of times a rating is used in each role, which alone would be helpful in characterizing the behavior of a system. But we can also measure the contribution of each rating to the quality of recommendations or health of the system. Since every prediction is a combined effort of several recommendation paths, we are interested in discerning the influence of each rating (and, hence, each path) in the system towards the system's overall error. We can understand the dynamics of the system at a finer granularity by tracking the influence of a rating according to the role played. The next section describes the approach to measuring the values of a rating in each role.

4. CONTRIBUTIONS OF RATINGS

As we've seen, a rating may play different roles in different predictions and, in each prediction, contribute to the quality of a prediction in different ways. Our approach can use any numeric measure of a property of system health, and assigns credit (or blame) to each rating proportional to its influence in the prediction. By tracking the role of each rating in a prediction, we can accumulate the credit for a rating in each of the three roles, and also track the evolution of the roles of rating over time in the system.

This section defines the methodology for computing the contribution of ratings by first defining the influence of a rating, and then instantiating the approach for predictive accuracy, and then rank accuracy. We also demonstrate how these contributions can be aggregated to study the neighborhood of ratings involved in computing a user's recommendations. Note that although our general formulation for rating influence is algorithm independent, due to space considerations, we present the approach for only item-based collaborative filtering. The definition for user-based algorithms is similar and will be presented in an expanded version of this paper.

4.1 Influence of Ratings

Recall that an item-based approach to collaborative filtering relies on building item neighborhoods using the similarity of ratings by the same user. As described earlier, similarity is defined by the adjusted cosine measure (Equations (2) and (3)). A set of the top K neighbors is maintained for all items for space and computational efficiency. A prediction of item i for a user u is computed as the weighted deviation from the item's mean rating as shown in Equation (4). The list of recommendations for a user is then the list of items sorted in descending order of their predicted values.

We first define $impact(a, i, j)$, the impact a user a has in determining the similarity between two items i and j. This is the change in the similarity between i and j when a's rating is removed, and is defined as

$$impact(a, i, j) = \frac{|sim'(i, j) - sim'_{\bar{a}}(i, j)|}{\sum_{w \in C_{ij}} |sim'(i, j) - sim'_{\bar{w}}(i, j)|}$$

where $C_{ij} = \{u \in \mathcal{U} \mid \exists\, r_{u,i}, r_{u,j} \in R(u)\}$ is the set of coraters

of items i and j (users who rate both i and j), $R(u)$ is the set of ratings provided by user u, and $sim'_{\bar{a}}(i,j)$ is the similarity of i and j when the ratings of user a are removed

$$sim_{\bar{a}}(i,j) = \frac{\sum_{v \in \mathcal{U} \setminus \{a\}} (r_{u,i} - \bar{r}_u)(r_{u,j} - \bar{r}_u)}{\sqrt{\sum_{u \in \mathcal{U} \setminus \{a\}} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{u \in \mathcal{U} \setminus \{a\}} (r_{u,j} - \bar{r}_u)^2}},$$

and adjusted for the number of raters

$$sim'_{\bar{a}}(i,j) = \frac{max(|\mathcal{U}_i \cap \mathcal{U}_j| - 1, \gamma)}{\gamma} \cdot sim(i,j).$$

If all coraters of i and j rate them identically, we define the impact as

$$impact(a,i,j) = \frac{1}{|C_{ij}|}$$

since $\sum_{w \in C_{ij}} |sim'(i,j) - sim'_{\bar{w}}(i,j)| = 0$.

The influence of each path $(u,j,v,i) = [r_{u,j}, r_{v,j}, r_{v,i}]$ in the prediction of $p_{u,i}$ is given by

$$influence(u,j,v,i) = \frac{sim'(i,j)}{\sum_{l \in \mathcal{N}_i \cap \mathcal{I}_u} sim'(i,l)} \cdot impact(v,i,j)$$

It follows that the sum of influences over all such paths, for a given set of endpoints, is 1.

4.2 Role Values for Predictive Accuracy

The value of a rating in each role is computed from the influence depending on the evaluation measure employed. Here we illustrate the approach using predictive accuracy as the evaluation metric.

In general, the goodness of a prediction decides whether the ratings involved must be credited or discredited for their role. For predictive accuracy, the error in prediction $e = |p_{u,i} - r_{u,i}|$ is mapped to a comfort level using a mapping function $M(e)$. Anecdotal evidence suggests that users are unable to discern errors less than 1.0 (for a rating scale of 1 to 5) [4], and so an error less than 1.0 is considered acceptable, but anything larger is not. We hence define $M(e)$ as $(1-e)$ binned to an appropriate value in $[-1, -0.5, 0.5, 1]$. For each prediction $p_{u,i}$, $M(e)$ is attributed to all the paths that assisted the computation of $p_{u,i}$, proportional to their influences. This tribute, $M(e) * influence(u,j,v,i)$, is in turn inherited by each of the ratings in the path $[r_{u,j}, r_{v,j}, r_{v,i}]$, with the credit/blame accumulating to the respective roles of $r_{u,j}$ as a scout, $r_{v,j}$ as a connector, and $r_{v,i}$ as a promoter. In other words, the scout value $SF(r_{u,j})$, the connector value $CF(r_{v,j})$ and the promoter value $PF(r_{v,i})$ are all incremented by the tribute amount. Over a large number of predictions, scouts that have repeatedly resulted in big error rates have a big negative scout value, and vice versa (similarly with the other roles). Every rating is thus summarized by its triple $[SF, CF, PF]$.

4.3 Role Values for Rank Accuracy

We now define the computation of the contribution of ratings to observed rank accuracy. For this computation, we must know the user's preference order for a set of items for which predictions can be computed. We assume that we have a test set of the users' ratings of the items presented in the recommendation list. For every pair of items rated by a user in the test data, we check whether the predicted order is concordant with his preference. We say a pair (i,j) is *concordant* (with error ϵ) whenever one of the following holds:

- if $(r_{u,i} < r_{u,j})$ then $(p_{u,i} - p_{u,j} < \epsilon)$;
- if $(r_{u,i} > r_{u,j})$ then $(p_{u,i} - p_{u,j} > \epsilon)$; or
- if $(r_{u,i} = r_{u,j})$ then $(|p_{u,i} - p_{u,j}| \leq \epsilon)$.

Similarly, a pair (i,j) is *discordant* (with error ϵ) if it is not concordant. Our experiments described below use an error tolerance of $\epsilon = 0.1$.

All paths involved in the prediction of the two items in a concordant pair are credited, and the paths involved in a discordant pair are discredited. The credit assigned to a pair of items (i,j) in the recommendation list for user u is computed as

$$c(i,j) = \begin{cases} \frac{t}{T} \cdot \frac{1}{C+D} & \text{if } (i,j) \text{ are concordant} \\ -\frac{t}{T} \cdot \frac{1}{C+D} & \text{if } (i,j) \text{ are discordant} \end{cases} \quad (7)$$

where t is the number of items in the user's test set whose ratings could be predicted, T is the number of items rated by user u in the test set, C is the number of concordances and D is the number of discordances. The credit c is then divided among all paths responsible for predicting $p_{u,i}$ and $p_{u,j}$ proportional to their influences. We again add the role values obtained from all the experiments to form a triple $[SF, CF, PF]$ for each rating.

4.4 Aggregating rating roles

After calculating the role values for individual ratings, we can also use these values to study neighborhoods and the system. Here we consider how we can use the role values to characterize the health of a neighborhood. Consider the list of top recommendations presented to a user at a specific point in time. The collaborative filtering algorithm traversed many paths in his neighborhood through his scouts and other connectors and promoters to make these recommendations. We call these ratings the *recommender neighborhood* of the user. The user implicitly chooses this neighborhood of ratings through the items he rates. Apart from the collaborative filtering algorithm, the health of this neighborhood completely influences a user's satisfaction with the system. We can characterize a user's recommender neighborhood by aggregating the individual role values of the ratings involved, weighted by the influence of individual ratings in determining his recommended list. Different sections of the user's neighborhood wield varied influence on his recommendation list. For instance, ratings reachable through highly rated items have a bigger say in the recommended items.

Our aim is to study the system and classify users with respect to their positioning in a healthy or unhealthy neighborhood. A user can have a good set of scouts, but may be exposed to a neighborhood with bad connectors and promoters. He can have a good neighborhood, but his bad scouts may ensure the neighborhood's potential is rendered useless. We expect that users with good scouts and good neighborhoods will be most satisfied with the system in the future.

A user's neighborhood is characterized by a triple that represents the weighted sum of the role values of individual ratings involved in making recommendations. Consider a user u and his ordered list of recommendations L. An item i

in the list is weighted inversely, as $K(i)$, depending on its position in the list. In our studies we use $K(i) = \sqrt{position(i)}$. Several paths of ratings $[r_{u,j}, r_{v,j}, r_{v,i}]$ are involved in predicting $p_{u,i}$ which ultimately decides its position in L, each with $influence(u, j, v, i)$.

The recommender neighborhood of a user u is characterized by the triple, $[SFN(u), CFN(u), PFN(u)]$ where

$$SFN(u) = \sum_{i \in L} \left(\frac{\sum_{[r_{u,j}, r_{v,j}, r_{v,i}]} SF(r_{u,j}) influence(u, j, v, i)}{K(i)} \right)$$

$CFN(u)$ and $PFN(u)$ are defined similarly. This triple estimates the quality of u's recommendations based on the past track record of the ratings involved in their respective roles.

5. EXPERIMENTATION

As we have seen, we can assign role values to each rating when evaluating a collaborative filtering system. In this section, we demonstrate the use of this approach to our overall goal of defining an approach to monitor and manage the health of a recommender system through experiments done on the MovieLens million rating dataset. In particular, we discuss results relating to identifying good scouts, promoters, and connectors; the evolution of rating roles; and the characterization of user neighborhoods.

5.1 Methodology

Our experiments use the MovieLens million rating dataset, which consists of ratings by 6040 users of 3952 movies. The ratings are in the range 1 to 5, and are labeled with the time the rating was given. As discussed before, we consider only the item-based algorithm here (with item neighborhoods of size 30) and, due to space considerations, only present role value results for rank accuracy.

Since we are interested in the evolution of the rating role values over time, the model of the recommender system is built by processing ratings in their arrival order. The timestamping provided by MovieLens is hence crucial for the analyses presented here. We make assessments of rating roles at intervals of 10,000 ratings and processed the first 200,000 ratings in the dataset (giving rise to 20 snapshots). We incrementally update the role values as the time ordered ratings are merged into the model. To keep the experiment computationally manageable, we define a test dataset for each user. As the time ordered ratings are merged into the model, we label a small randomly selected percentage (20%) as test data. At discrete epochs, i.e., after processing every 10,000 ratings, we compute the predictions for the ratings in the test data, and then compute the role values for the ratings used in the predictions. One potential criticism of this methodology is that the ratings in the test set are never evaluated for their roles. We overcome this concern by repeating the experiment, using different random seeds. The probability that every rating is considered for evaluation is then considerably high: $1 - 0.2^n$, where n is the number of times the experiment is repeated with different random seeds. The results here are based on $n = 4$ repetitions.

The item-based collaborative filtering algorithm's performance was ordinary with respect to rank accuracy. Fig. 5 shows a plot of the precision and recall as ratings were merged in time order into the model. The recall was always high, but the average precision was just about 53%.

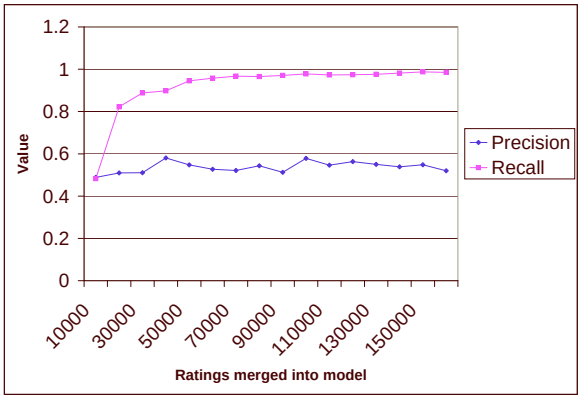

Figure 5: Precision and recall for the item-based collaborative filtering algorithm.

5.2 Inducing good scouts

The ratings of a user that serve as scouts are those that allow the user to receive recommendations. We claim that users with ratings that have respectable scout values will be happier with the system than those with ratings with low scout values. Note that the item-based algorithm discussed here produces recommendation lists with nearly half of the pairs in the list discordant from the user's preference. Whether all of these discordant pairs are observable by the user is unclear, however, certainly this suggests that there is a need to be able to direct users to items whose ratings would improve the lists.

The distribution of the scout values for most users' ratings are Gaussian with mean zero. Fig. 6 shows the frequency distribution of scout values for a sample user at a given snapshot. We observe that a large number of ratings never serve as scouts for their users. A relatable scenario is when Amazon's recommender makes suggestions of books or items based on other items that were purchased as gifts. With simple relevance feedback from the user, such ratings can be isolated as bad scouts and discounted from future predictions. Removing bad scouts was found to be worthwhile for individual users but the overall performance improvement was only marginal.

An obvious question is whether good scouts can be formed by merely rating popular movies as suggested by Rashid *et al.* [9]. They show that a mix of popularity and rating entropy identifies the best items to suggest to new users when evaluated using *MAE*. Following their intuition, we would expect to see a higher correlation between popularity-entropy and good scouts. We measured the Pearson correlation coefficient between aggregated scout values for a movie with the popularity of a movie (number of times it is rated); and with its popularity*variance measure at different snapshots of the system. Note that the scout values were initially anti-correlated with popularity (Fig. 7), but became moderately correlated as the system evolved. Both popularity and popularity*variance performed similarly. A possible explanation is that there has been insufficient time for the popular movies to accumulate ratings.

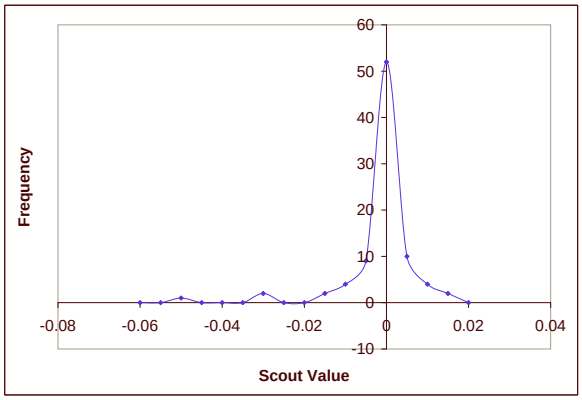

Figure 6: Distribution of scout values for a sample user.

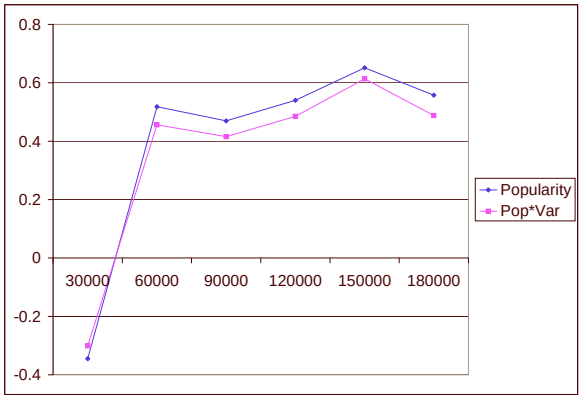

Figure 7: Correlation between aggregated scout value and item popularity (computed at different intervals).

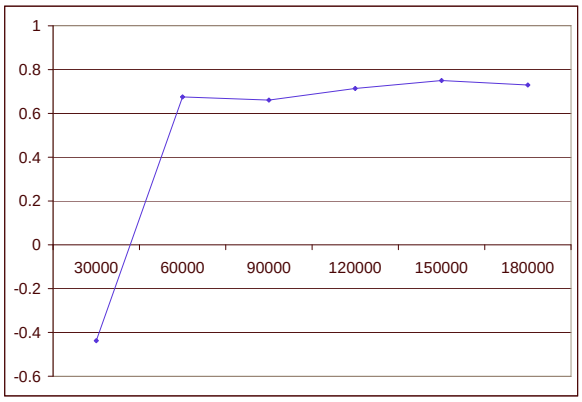

Figure 8: Correlation between aggregated promoter value and user prolificity (computed at different intervals).

Table 1: Movies forming the best scouts.

Best Scouts	Conf.	Pop.
Being John Malkovich (1999)	1.00	445
Star Wars: Episode IV - A New Hope (1977)	0.92	623
Princess Bride, The (1987)	0.85	477
Sixth Sense, The (1999)	0.85	617
Matrix, The (1999)	0.77	522
Ghostbusters (1984)	0.77	441
Casablanca (1942)	0.77	384
Insider, The (1999)	0.77	235
American Beauty (1999)	0.69	624
Terminator 2: Judgment Day (1991)	0.69	503
Fight Club (1999)	0.69	235
Shawshank Redemption, The (1994)	0.69	445
Run Lola Run (Lola rennt) (1998)	0.69	220
Terminator, The (1984)	0.62	450
Usual Suspects, The (1995)	0.62	326
Aliens (1986)	0.62	385
North by Northwest (1959)	0.62	245
Fugitive, The (1993)	0.62	402
End of Days (1999)	0.62	132
Raiders of the Lost Ark (1981)	0.54	540
Schindler's List (1993)	0.54	453
Back to the Future (1985)	0.54	543
Toy Story (1995)	0.54	419
Alien (1979)	0.54	415
Abyss, The (1989)	0.54	345
2001: A Space Odyssey (1968)	0.54	358
Dogma (1999)	0.54	228
Little Mermaid, The (1989)	0.54	203

Table 2: Movies forming the worst scouts.

Worst scouts	Conf.	Pop.
Harold and Maude (1971)	0.46	141
Grifters, The (1990)	0.46	180
Sting, The (1973)	0.38	244
Godfather: Part III, The (1990)	0.38	154
Lawrence of Arabia (1962)	0.38	167
High Noon (1952)	0.38	84
Women on the Verge of a... (1988)	0.38	113
Grapes of Wrath, The (1940)	0.38	115
Duck Soup (1933)	0.38	131
Arsenic and Old Lace (1944)	0.38	138
Midnight Cowboy (1969)	0.38	137
To Kill a Mockingbird (1962)	0.31	195
Four Weddings and a Funeral (1994)	0.31	271
Good, The Bad and The Ugly, The (1966)	0.31	156
It's a Wonderful Life (1946)	0.31	146
Player, The (1992)	0.31	220
Jackie Brown (1997)	0.31	118
Boat, The (Das Boot) (1981)	0.31	210
Manhattan (1979)	0.31	158
Truth About Cats & Dogs, The (1996)	0.31	143
Ghost (1990)	0.31	227
Lone Star (1996)	0.31	125
Big Chill, The (1983)	0.31	184

By studying the evolution of scout values, we can identify movies that consistently feature in good scouts over time. We claim these movies will make viable scouts for other users. We found the aggregated scout values for all movies in intervals of 10,000 ratings each. A movie is said to induce a good scout if the movie was in the top 100 of the sorted list, and to induce a bad scout if it was in bottom 100 of the same list. Movies appearing consistently high over time are expected to remain up there in the future. The effective confidence in a movie m is

$$C_m = \frac{T_m - B_m}{N} \tag{8}$$

where T_m is the number of times it appeared in the top 100, B_m the number of times it appeared in the bottom 100, and N is the number of intervals considered. Using this measure, the top few movies expected to induce the best scouts are shown in Table 1. Movies that would be bad scout choices are shown in Table 2 with their associated confidences. The popularities of the movies are also displayed. Although more popular movies appear in the list of good scouts, these tables show that a blind choice of scout based on popularity alone can be potentially dangerous. Interestingly, the best scout—'Being John Malkovich'—is about a puppeteer who discovers a portal into a movie star, a movie that has been described variously on amazon.com as 'makes you feel giddy,' 'seriously weird,' 'comedy with depth,' 'silly,' 'strange,' and 'inventive.' Indicating whether someone likes this movie or not goes a long way toward situating the user in a suitable neighborhood, with similar preferences.

On the other hand, several factors may have made a movie a bad scout, like the sharp variance in user preferences in the neighborhood of a movie. Two users may have the same opinion about Lawrence of Arabia, but they may differ sharply about how they felt about the other movies they saw. Bad scouts ensue when there is deviation in behavior around a common synchronization point.

5.3 Inducing good promoters

Ratings that serve to promote items in a collaborative filtering system are critical to allowing a new item be recommended to users. So, inducing good promoters is important for cold-start recommendation. We note that the frequency distribution of promoter values for a sample movie's ratings is also Gaussian (similar to Fig. 6). This indicates that the promotion of a movie is benefited most by the ratings of a few users, and are unaffected by the ratings of most users. We find a strong correlation between a user's number of ratings and his aggregated promoter value. Fig. 8 depicts the evolution of the Pearson correlation co-efficient between the prolificity of a user (number of ratings) versus his aggregated promoter value. We expect that conspicuous shills, by recommending wrong movies to users, will be discredited with negative aggregate promoter values and should be identifiable easily.

Given this observation, the obvious rule to follow when introducing a new movie is to have it rated directly by prolific users who posses high aggregated promoter values. A new movie is thus cast into the neighborhood of many other movies improving its visibility. Note, though, that a user may have long stopped using the system. Tracking promoter values consistently allows only the most active recent users to be considered.

5.4 Inducing good connectors

Given the way scouts, connectors, and promoters are characterized, it follows that the movies that are part of the best scouts are also part of the best connectors. Similarly, the users that constitute the best promoters are also part of the best connectors. Good connectors are induced by ensuring a user with a high promoter value rates a movie with a high scout value. In our experiments, we find that a rating's longest standing role is often as a connector. A rating with a poor connector value is often seen due to its user being a bad promoter, or its movie being a bad scout. Such ratings can be removed from the prediction process to bring marginal improvements to recommendations. In some selected experiments, we observed that removing a set of badly behaving connectors helped improve the system's overall performance by 1.5%. The effect was even higher on a few select users who observed an improvement of above 10% in precision without much loss in recall.

5.5 Monitoring the evolution of rating roles

One of the more significant contributions of our work is the ability to model the evolution of recommender systems, by studying the changing roles of ratings over time. The role and value of a rating can change depending on many factors like user behavior, redundancy, shilling effects or properties of the collaborative filtering algorithm used. Studying the dynamics of rating roles in terms of transitions between good, bad, and negligible values can provide insights into the functioning of the recommender system. We believe that a continuous visualization of these transitions will improve the ability to manage a recommender system.

We classify different rating states as good, bad, or negligible. Consider a user who has rated 100 movies in a particular interval, of which 20 are part of the test set. If a scout has a value greater than 0.005, it indicates that it is uniquely involved in at least 2 concordant predictions, which we will say is *good*. Thus, a threshold of 0.005 is chosen to bin a rating as good, bad or negligible in terms of its scout, connector and promoter value. For instance, a rating r, at time t with role value triple $[0.1, 0.001, -0.01]$ is classified as [scout +, connector 0, promoter −], where + indicates good, 0 indicates negligible, and − indicates bad.

The positive credit held by a rating is a measure of its contribution to the betterment of the system, and the discredit is a measure of its contribution to the detriment of the system. Even though the positive roles (and the negative roles) make up a very small percentage of all ratings, their contribution supersedes their size. For example, even though only 1.7% of all ratings were classified as good scouts, they hold 79% of all positive credit in the system! Similarly, the bad scouts were just 1.4% of all ratings but hold 82% of all discredit. Note that good and bad scouts, together, comprise only $1.4\% + 1.7\% = 3.1\%$ of the ratings, implying that the majority of the ratings are negligible role players as scouts (more on this later). Likewise, good connectors were 1.2% of the system, and hold 30% of all positive credit. The bad connectors (0.8% of the system) hold 36% of all discredit. Good promoters (3% of the system) hold 46% of all credit, while bad promoters (2%) hold 50% of all discredit. This reiterates that a few ratings influence most of the system's performance. Hence it is important to track transitions between them regardless of their small numbers.

Across different snapshots, a rating can remain in the same state or change. A good scout can become a bad scout, a good promoter can become a good connector, good and bad scouts can become vestigial, and so on. It is not practical to expect a recommender system to have no ratings in bad roles. However, it suffices to see ratings in bad roles either convert to good or vestigial roles. Similarly, observing a large number of good roles become bad ones is a sign of imminent failure of the system.

We employ the principle of non-overlapping episodes [6] to count such transitions. A sequence such as:

$$[+,0,0] \rightarrow [+,0,0] \rightarrow [0,+,0] \rightarrow [0,0,0]$$

is interpreted as the transitions

$$[+,0,0] \rightsquigarrow [0,+,0] : 1$$
$$[+,0,0] \rightsquigarrow [0,0,0] : 1$$
$$[0,+,0] \rightsquigarrow [0,0,0] : 1$$

instead of

$$[+,0,0] \rightsquigarrow [0,+,0] : 2$$
$$[+,0,0] \rightsquigarrow [0,0,0] : 2$$
$$[0,+,0] \rightsquigarrow [0,0,0] : 1.$$

See [6] for further details about this counting procedure. Thus, a rating can be in one of 27 possible states, and there are about 27^2 possible transitions. We make a further simplification and utilize only 9 states, indicating whether the rating is a scout, promoter, or connector, and whether it has a positive, negative, or negligible role. Ratings that serve multiple purposes are counted using multiple episode instantiations but the states themselves are not duplicated beyond the 9 restricted states. In this model, a transition such as $[+,0,+] \rightsquigarrow [0,+,0] : 1$ is counted as

$$[scout+] \rightsquigarrow [scout0] : 1$$
$$[scout+] \rightsquigarrow [connector+] : 1$$
$$[scout+] \rightsquigarrow [promoter0] : 1$$
$$[connector0] \rightsquigarrow [scout0] : 1$$
$$[connector0] \rightsquigarrow [scout+] : 1$$
$$[connector0] \rightsquigarrow [promoter0] : 1$$
$$[promoter+] \rightsquigarrow [scout0] : 1$$
$$[promoter+] \rightsquigarrow [connector+] : 1$$
$$[promoter+] \rightsquigarrow [promoter0] : 1$$

Of these, transitions like $[pX] \rightsquigarrow [q0]$ where $p \neq q, X \in \{+,0,-\}$ are considered uninteresting, and only the rest are counted.

Fig. 9 depicts the major transitions counted while processing the first 200,000 ratings from the MovieLens dataset. Only transitions with frequency greater than or equal to 3% are shown. The percentages for each state indicates the number of ratings that were found to be in those states. We consider transitions from any state to a good state as healthy, from any state to a bad state as unhealthy, and from any state to a vestigial state as decaying.

From Fig. 9, we can observe:

- The bulk of the ratings have negligible values, irrespective of their role. The majority of the transitions involve both good and bad ratings becoming negligible.

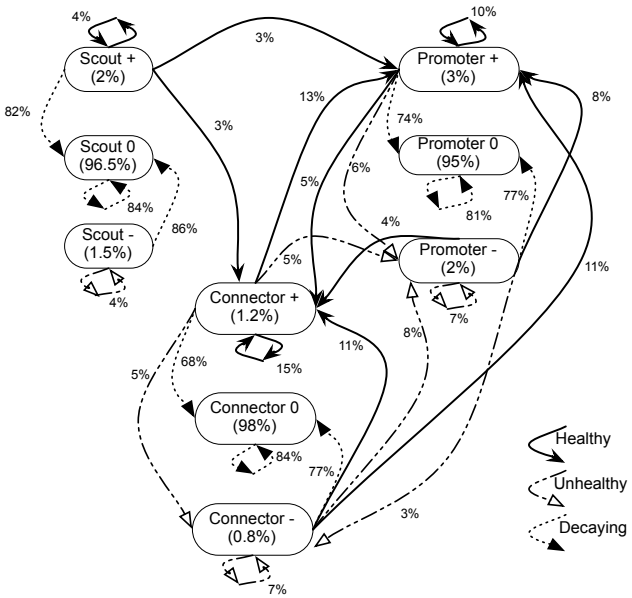

Figure 9: Transitions among rating roles.

- The number of good ratings is comparable to the bad ratings, and ratings are seen to switch states often, except in the case of scouts (see below).
- The negative and positive scout states are not reachable through any transition, indicating that these ratings must begin as such, and cannot be coerced into these roles.
- Good promoters and good connectors have a much longer survival period than scouts. Transitions that recur to these states have frequencies of 10% and 15% when compared to just 4% for scouts. Good connectors are the slowest to decay whereas (good) scouts decay the fastest.
- Healthy percentages are seen on transitions between promoters and connectors. As indicated earlier, there are hardly any transitions from promoters/connectors to scouts. This indicates that, over the long run, a user's rating is more useful to others (movies or other users) than to the user himself.
- The percentages of healthy transitions outweigh the unhealthy ones — this hints that the system is healthy, albeit only marginally.

Note that these results are conditioned by the static nature of the dataset, which is a set of ratings over a fixed window of time. However a diagram such as Fig. 9 is clearly useful for monitoring the health of a recommender system. For instance, acceptable limits can be imposed on different types of transitions and, if a transition fails to meet the threshold, the recommender system or a part of it can be brought under closer scrutiny. Furthermore, the role state transition diagram would also be the ideal place to study the effects of shilling, a topic we will consider in future research.

5.6 Characterizing neighborhoods

Earlier we saw that we can characterize the neighborhood of ratings involved in creating a recommendation list L for

a user. In our experiment, we consider lists of length 30, and sample the lists of about 5% of users through the evolution of the model (at intervals of 10,000 ratings each) and compute their neighborhood characteristics. To simplify our presentation, we consider the percentage of the sample that fall into one of the following categories:

1. Inactive user: $(SFN(u) = 0)$
2. Good scouts, Good neighborhood:
$$(SFN(u) > 0) \wedge (CFN(u) > 0 \wedge PFN(u) > 0)$$
3. Good scouts, Bad neighborhood:
$$(SFN(u) > 0) \wedge (CFN(u) < 0 \vee PFN(u) < 0)$$
4. Bad scouts, Good neighborhood:
$$(SFN(u) < 0) \wedge (CFN(u) > 0 \wedge PFN(u) > 0)$$
5. Bad scouts, Bad neighborhood:
$$(SFN(u) < 0) \wedge (CFN(u) < 0 \vee PFN(u) < 0)$$

From our sample set of 561 users, we found that 476 users were inactive. Of the remaining 85 users, we found 26 users had good scouts and a good neighborhood, 6 had bad scouts and a good neighborhood, 29 had good scouts and a bad neighborhood, and 24 had bad scouts and a bad neighborhood. Thus, we conjecture that 59 users (29+24+6) are in danger of leaving the system.

As a remedy, users with bad scouts and a good neighborhood can be asked to reconsider rating of some movies hoping to improve the system's recommendations. The system can be expected to deliver more if they engineer some good scouts. Users with good scouts and a bad neighborhood are harder to address; this situation might entail selectively removing some connector-promoter pairs that are causing the damage. Handling users with bad scouts and bad neighborhoods is a more difficult challenge.

Such a classification allows the use of different strategies to better a user's experience with the system depending on his context. In future work, we intend to conduct field studies and study the improvement in performance of different strategies for different contexts.

6. CONCLUSIONS

To further recommender system acceptance and deployment, we require new tools and methodologies to manage an installed recommender and develop insights into the roles played by ratings. A fine-grained characterization in terms of rating roles such as scouts, promoters, and connectors, as done here, helps such an endeavor. Although we have presented results on only the item-based algorithm with list rank accuracy as the metric, the same approach outlined here applies to user-based algorithms and other metrics.

In future research, we plan to systematically study the many algorithmic parameters, tolerances, and cutoff thresholds employed here and reason about their effects on the downstream conclusions. We also aim to extend our formulation to other collaborative filtering algorithms, study the effect of shilling in altering rating roles, conduct field studies, and evaluate improvements in user experience by tweaking ratings based on their role values. Finally, we plan to develop the idea of mining the evolution of rating role patterns into a reporting and tracking system for all aspects of recommender system health.

7. REFERENCES

[1] Cosley, D., Lam, S., Albert, I., Konstan, J., and Riedl, J. Is Seeing Believing?: How Recommender System Interfaces Affect User's Opinions. In *Proc. CHI* (2001), pp. 585–592.

[2] Herlocker, J. L., Konstan, J. A., Borchers, A., and Riedl, J. An Algorithmic Framework for Performing Collaborative Filtering. In *Proc. SIGIR* (1999), pp. 230–237.

[3] Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T. Evaluating Collaborative Filtering Recommender Systems. *ACM Transactions on Information Systems Vol. 22*, 1 (2004), pp. 5–53.

[4] Konstan, J. A. Personal communication. 2003.

[5] Lam, S. K., and Riedl, J. Shilling Recommender Systems for Fun and Profit. In *Proceedings of the 13th International World Wide Web Conference* (2004), ACM Press, pp. 393–402.

[6] Laxman, S., Sastry, P. S., and Unnikrishnan, K. P. Discovering Frequent Episodes and Learning Hidden Markov Models: A Formal Connection. *IEEE Transactions on Knowledge and Data Engineering Vol. 17*, 11 (2005), 1505–1517.

[7] McLaughlin, M. R., and Herlocker, J. L. A Collaborative Filtering Algorithm and Evaluation Metric that Accurately Model the User Experience. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (2004), pp. 329 – 336.

[8] O'Mahony, M., Hurley, N. J., Kushmerick, N., and Silvestre, G. Collaborative Recommendation: A Robustness Analysis. *ACM Transactions on Internet Technology Vol. 4*, 4 (Nov 2004), pp. 344–377.

[9] Rashid, A. M., Albert, I., Cosley, D., Lam, S., McNee, S., Konstan, J. A., and Riedl, J. Getting to Know You: Learning New User Preferences in Recommender Systems. In *Proceedings of the 2002 Conference on Intelligent User Interfaces (IUI 2002)* (2002), pp. 127–134.

[10] Rashid, A. M., Karypis, G., and Riedl, J. Influence in Ratings-Based Recommender Systems: An Algorithm-Independent Approach. In *Proc. of the SIAM International Conference on Data Mining* (2005).

[11] Resnick, P., Iacovou, N., Sushak, M., Bergstrom, P., and Riedl, J. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of the Conference on Computer Supported Collaborative Work (CSCW'94)* (1994), ACM Press, pp. 175–186.

[12] Sarwar, B., Karypis, G., Konstan, J., and Reidl, J. Item-Based Collaborative Filtering Recommendation Algorithms. In *Proceedings of the Tenth International World Wide Web Conference (WWW'10)* (2001), pp. 285–295.

[13] Schein, A., Popescu, A., Ungar, L., and Pennock, D. Methods and Metrics for Cold-Start Recommendation. In *Proc. SIGIR* (2002), pp. 253–260.