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

BHARATH KUMAR MOHAN
Indian Institute of Science
BENJAMIN J. KELLER
Eastern Michigan University
and
NAREN RAMAKRISHNAN
Virginia Tech

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 characterization of nearest-neighbor collaborative filtering that allows us to disaggregate global recommender performance measures into contributions made by each individual rating. 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, respectively. 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

Additional Key Words and Phrases: Recommender systems, collaborative filtering, neighborhoods, user-based and item-based algorithms, scouts, promoters, connectors.

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

This paper is an expanded version of an article of the same name that appeared in *Proceedings of the Seventh ACM Conference on Electronic Commerce (EC'06)*, Ann Arbor, MI, pages 250–259, June 2006.

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 [Herlocker et al. 1999; McLaughlin and Herlocker 2004], designing interfaces for presenting recommendations to users [Cosley et al. 2001], and studying the robustness of these algorithms [O'Mahony et al. 2004]. However, with the exception of a few studies on the influence of users [Rashid et al. 2005], 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 direction on how to monitor and manage a recommender system, and how to engineer mechanisms to sustain the recommender, and thereby ensure its continued success.

Our motivation here is to disaggregate global recommender performance measures into contributions made by each individual rating, allowing us to characterize the roles played by ratings in nearest-neighbor collaborative filtering. We identify three possible roles: to connect the user into the system to receive recommendations (scout), to connect an item into the system to be recommended (promoter), and to connect ratings of these two kinds (connector). 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 involving newly arrived items [Schein et al. 2002]. 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 [Lam and Riedl 2004].

As we show, the characterization of rating roles presented here provides broad primitives to manage a recommender system and its community. The main contributions of this paper are to use the three roles of scouts, promoters, and connectors to help understand the way individual ratings affect global performance measures. We present examples of using these roles to improve recommendations for users, to improve targeting of items, and to track the evolution of the recommender system. While the tracking task is descriptive, improving user/item measures can be

viewed as selectively removing, 're-wiring,' or, in general, modifying the underlying ratings graph to improve some aspect of recommender system usage. Examples of such modifications are given here, for specific users and specific items; however, conducting such re-wirings to accommodate all users and all items in a live recommender is beyond the scope of the present work.

This paper improves upon a preliminary version of our study [Mohan, Keller, and Ramakrishnan 2006] by (i) expanding the scope to two classes of algorithms (item-based and user-based) and two evaluation measures (rank and predictive accuracy), and (ii) providing concrete examples of users for whom recommendations can be improved and items which can be better targeted (and how). It is important to note that we focus on not merely global measures (recommender systems are already heavily tuned to achieve high scores in this regard) but local, individual-based measures.

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 characterize ratings and in Section 6, we show how such characterizations help 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* [Sarwar et al. 2001]. 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 [Resnick et al. 1994] with variations from Herlocker *et al.* [Herlocker et al. 1999], and an item-based algorithm similar to that of Sarwar *et al.* [Sarwar et al. 2001].

The algorithm used by Resnick *et al.* [Resnick et al. 1994] defines the similarity of two users u and v as the Pearson correlation of their common ratings:

$$sim(u,v) = \frac{\sum_{i \in \mathcal{I}_u \cap \mathcal{I}_v} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in \mathcal{I}_u} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{i \in \mathcal{I}_v} (r_{v,i} - \bar{r}_v)^2}}, \quad (1)$$

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:

$$sim'(u,v) = \frac{max(|\mathcal{I}_u \cap \mathcal{I}_v|, \gamma)}{\gamma} \cdot sim(u,v), \tag{2}$$

where $\gamma \approx 5$ is a constant used as a lower limit in scaling [Herlocker et al. 1999]. 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

$$p_{u,i} = \bar{r}_u + \frac{\sum_{v \in V} sim'(u,v)(r_{v,i} - \bar{r}_v)}{\sum_{v \in V} sim'(u,v)} \tag{3}$$

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.* [Sarwar et al. 2001]. In this algorithm, similarity is defined as the adjusted cosine measure

$$sim(i,j) = \frac{\sum_{u \in \mathcal{U}_i \cap \mathcal{U}_j} (r_{u,i} - \bar{r}_u)(r_{u,j} - \bar{r}_u)}{\sqrt{\sum_{u \in \mathcal{U}_i} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{u \in \mathcal{U}_j} (r_{u,j} - \bar{r}_u)^2}} \tag{4}$$

where $\mathcal{U}_i$ is the set of users who have rated item i. (Although at first glance, this equation appears isomorphic to Eqn. 1, observe that the denominator still involves the deviation w.r.t. the mean user rating; it hence does not correspond to the Pearson's correlation over the vectors of users' ratings for the two items.) As for the user-based algorithm, the similarity weights are adjusted proportionally to the number of users that have rated the items in common

$$sim'(i,j) = \frac{max(|\mathcal{U}_i \cap \mathcal{U}_j|, \gamma)}{\gamma} \cdot sim(i,j). \tag{5}$$

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

$$p_{u,i} = \bar{r}_i + \frac{\sum_{j \in J} sim'(i,j)(r_{u,j} - \bar{r}_j)}{\sum_{j \in J} sim'(i,j)} \tag{6}$$

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 [Herlocker et al. 2004]. Studies on recommender algorithms, notably Herlocker *et al.* [Herlocker et al. 1999] and Sarwar *et al.* [Sarwar et al. 2001], 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,

$$MAE = \frac{\sum_{(u,i) \in \mathcal{T}} |p_{u,i} - r_{u,i}|}{|\mathcal{T}|}. \tag{7}$$

Coverage has a number of definitions, but generally refers to the proportion of items that can be predicted by the algorithm [Herlocker et al. 2004].

A practical issue with predictive accuracy is that users typically are presented with recommendation lists, rather than 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.* [Herlocker et al. 2004] 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 [Lam and Riedl 2004]. Kendall's Tau measures the number of inversions when comparing ordered pairs in the true user ordering of items and the recommended order, and is defined as

$$\tau = \frac{C - D}{\sqrt{(C + D + TR)(C + D + TP)}} \tag{8}$$

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 [Herlocker et al. 2004]. A shortcoming of the Tau metric is that it is oblivious to the position in the ordered list where the inversion occurs [Herlocker et al. 2004]. 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 may play one of three different roles in each prediction in which it is used. Consider a simplified recommender system with four users and their ratings for six items, as shown in Fig. 1. (For this initial discussion we will not consider the rating values involved.) The recommender predicts whether user A will like item 6 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 items around item 6 by using the ratings of users who rated item 6 as well as other items. For instance, Fig. 2a highlights the set of ratings involved in computing the similarity between items 4 and 6. Similarly, we use other sets of ratings to compute similarities between other items and item 6. In this manner, we identify the nearest neighbors of item 6, whose ratings by user A are then used to make a prediction for item 6.

(2) A user-based collaborative filtering algorithm, on the other hand, constructs a neighborhood of users around user A by tracking other users whose rating behaviors are similar to that of user A. For instance, Fig. 2b highlights the set of ratings involved in computing the similarity between user A and user C. The prediction of A's rating for item 6 is then based on the ratings of item 6 by neighbors such as C.

As Fig. 2a and Fig. 2b suggest, irrespective of the collaborative filtering algorithm used, we can visualize the prediction of A's rating of item 6 as walking

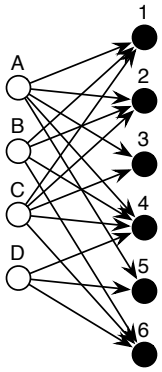


Fig. 1. Ratings in simple movie recommender.

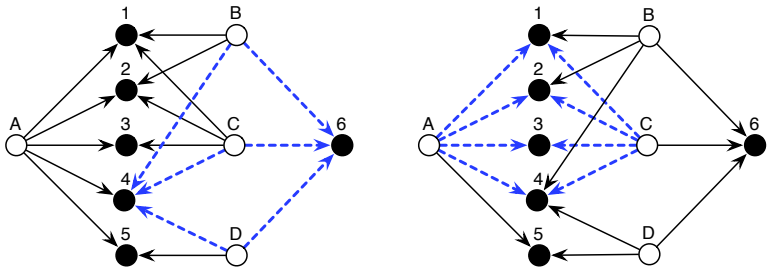


Fig. 2. Ratings involved in computing (a) an item-based prediction, and (b) a user-based prediction for user A and item 6.

sequences of ratings we call *recommendation paths*. In fact, both algorithms use the same set of nine paths for this prediction. One such path is $[r_{A,4}, r_{C,4}, r_{C,6}]$, or simply $((A,4),(C,4),(C,6))$. Note that these paths are undirected, and are all of length 3. The difference in the algorithms is essentially the order in which the ratings are visited. In the item-based algorithm the above path is effectively visited from item to user $((C,6),(C,4),(A,4))$, and in the user-based algorithm the path is from user to item $((A,4),(C,4),(C,6))$. A rating can be part of many paths for a single prediction as shown in Fig. 2b, where the rating $(C,6)$ is used in four paths $(((A,1),(C,1),(C,6))$, $((A,2),(C,2),(C,6))$, $((A,3),(C,3),(C,6))$, and $((A,4),(C,4),(C,6))$.

Predictions in a collaborative filtering algorithms may involve thousands of such simultaneous walks, each playing a part in influencing the predicted value. To understand at a fine-grain level the influences played by the ratings, we observe the three roles called *scouts*, *promoters*, and *connectors*. To illustrate these roles, consider the rating $r_{A,1}$ from Fig. 1 in three scenarios:

(1) In Fig. 3a, the rating $r_{A,1}$ is a scout, the first rating in the recommendation

path, and helps user A reach other ratings that can be used to predict A's rating for item 6.

(2) In Fig. 3b, the rating $r_{A,1}$ is a connector, the second rating in the recommendation path, and performs an altruistic role: it helps connect user B to item 3, and this role referred to as a *connector*.

(3) In Fig. 3c, the rating $r_{A,1}$ is a promoter, the third rating in the recommendation path, allowing predictions to be made for item 1.

It is important to note, as shown in the above example, that the *same* rating can serve different purposes in different predictions. Formally, we define these roles as follows.

Definition 3.1. 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 $p_{u,a}$ is a rating $r_{v,i}$ by some user v and rating i, such that there exists ratings $r_{u,i}$ and $r_{v,a}$.

We can also extend these role definitions to roles of a rating for a set of predictions by requiring that a rating play that role for each individual prediction in the set. Thus, the scouts for the prediction of A's rating of item 6 are:

$$\mathcal{S} = \{(A,1),(A,2),(A,3),(A,4),(A,5)\};$$

the connectors for the prediction of A's rating of item 6 are:

$$\mathcal{C} = \{(B,1),(B,2),(B,4),(C,1),(C,2),(C,3),(C,4),(D,4),(D,5)\};$$

and the promoters for the prediction of A's rating of item 6 are:

$$\mathcal{P} = \{(B,6),(C,6),(D,6)\}.$$

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.

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. Consider again the three different views from Fig. 3, but this time from the perspective of studying the scouts for users A (Fig. 3a), B (Fig. 3b), and D (Fig. 3c). The ratings of user A help the system to recommend only item 6 to user A. But the ratings of user B help the system to recommend more items to user B (similarly for user D). This is because the ratings of users B and D help make connections to users A and C, who are more prolific users of the system. From the perspective of coverage, a user's rating of a particular item will be more valuable as a scout if that item is rated by many relatively prolific users who have rated many more items than those in common with the target user. So, in general, the better scouts allow a user to build similarity with prolific users, and thereby ensure they get more recommendations from the system.

While scouts represent beneficial ratings from the perspective of a user, promoters are their duals, and benefit items. Thus consider again Fig. 3b, originally used to illustrate connectors, but this time to study the promoters for item 5. As shown, item 5 benefits from the ratings by users A and D, which allow recommendations

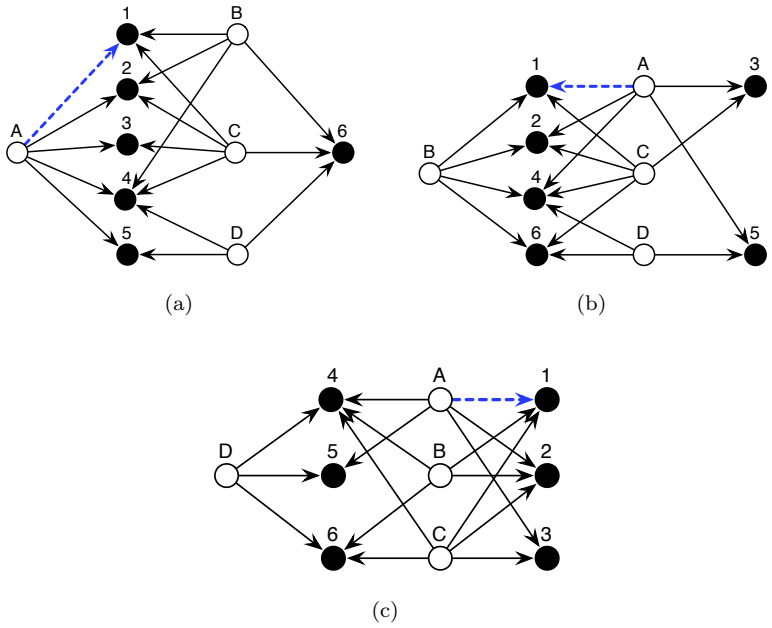


Fig. 3. Different roles of rating $r_{A,1}$: (a) as a scout for the prediction of item 6 for user A; (b) as a connector for the prediction of items 3 and 5 for user B; and (c) as a promoter for the prediction of item 1 for user D.

to user B (and user C as well). Items benefit from ratings by prolific users that rate a wide variety of items, so a good promoter is a rating that helps define a broader neighborhood of items, ensuring that an item can be recommended to more users.

Connectors serve a crucial role in a recommender system that is not as obvious. Both good scouts and promoters depend on other ratings to allow recommendations to be made. So a rating that is a good connector for a set of predictions would be by a user who has rated a number of items in that set (e.g., is responsible for several promoters), and the rating rates an item that has many ratings itself. A connector is qualified by the number of (user,item) pairs that it helps relate. In Fig. 3c, the possible connectors for making predictions for user D are: (A,4), (A,5), (B,4), (B,6), (C,4), and (C,6). Rating (A,4) helps connect 3 (user,item) pairs whereas rating (B,6) helps connect only 2 (user,item) pairs. Hence (A,4) is a better connector than (B,6). In fact, connecting through A's ratings or C's ratings gives access to all three items on the right of Fig. 3c (items 1, 2, and 3) whereas connecting through B's ratings gives access to only item 1 and 2. Hence, from the standpoint of coverage, the non-promoter ratings of users A and C are the better connectors for the set of predictions for user D than are the non-promoter ratings of user B. It is likely that good promoters have been or are also good scouts, but since a connector serves to improve the system's ability to make recommendations, there is no explicit gain for the user or item.

Ratings evolve in the roles that they play. Ratings of rarely rated items may start as promoters, and transition to scouts as more people rate the item. Scouts

and connectors around a common item are interchangeable for neighbors, in a recommender algorithm for which neighborhoods are symmetric. For instance, we see in Fig. 3a and Fig. 3b that ratings $r_{A,1}$ and $r_{B,1}$ exchange roles. Similarly, connectors and promoters around a common user can exchange roles; observe $r_{A,5}$ $r_{A,2}$ in Fig. 3b and Fig. 3c. Such role exchanges happen only if we utilize absolute levels of similarity (e.g., where we connect users or items whose ratings have a similarity greater than a threshold). However, the definition of neighborhoods in the algorithms we consider are not symmetric, since only the top K nearest neighbors are included. Therefore, whether a rating ever serves as a connector depends on the rating values and how neighborhoods are formed.

As we have discussed, 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 by itself 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 our 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 performance, 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, for both item-based and user-based algorithms. Then we instantiate the approaches further, for predictive accuracy and rank accuracy. We also demonstrate how these contributions can be aggregated to study the neighborhood of ratings involved in computing a user's recommendations.

4.1 Influence of Ratings: Item-Based Algorithms

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 (4) and (5)). 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 (6). 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 $C_{ij} = \mathcal{U}_i \cap \mathcal{U}_j$ is the set of coraters of items i and j (users who rate both i and j), 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 $[r_{u,j}, r_{v,j}, r_{v,i}]$ 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 Influence of Ratings: User-Based Algorithms

A user-based approach to collaborative filtering relies on building user neighborhoods using the similarity of ratings for the same item by different users. As described earlier, similarity is defined by the Pearson's correlation measure (Equations (1) and (2)). 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 user's mean rating as shown in Equation (3). The list of recommendations for a user is then the list of items sorted in descending order of their predicted values.

We first define $impact(i,a,b)$, the impact an item i has in determining the similarity between two users a and b. This is the change in the similarity between a and b when i's ratings by the two are removed, and is defined as

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

where $C_{ab} = \mathcal{I}_a \cap \mathcal{I}_b$ is the set of items corated by users a and b, and $sim'_{\bar{i}}(a,b)$ is the similarity between a and b when the users' ratings for i are removed

$$sim_{\bar{i}}(a,b) = \frac{\sum_{v \in \mathcal{I} \setminus \{i\}} (r_{a,v} - \bar{r}_a)(r_{b,v} - \bar{r}_b)}{\sqrt{\sum_{v \in \mathcal{I} \setminus \{i\}} (r_{a,v} - \bar{r}_a)^2} \sqrt{\sum_{v \in \mathcal{I} \setminus \{i\}} (r_{b,v} - \bar{r}_b)^2}},$$

and adjusted for the number of co-rated items

$$sim'_i(a,b) = \frac{max(|\mathcal{I}_a \cap \mathcal{I}_b| - 1, \gamma)}{\gamma} \cdot sim(a,b).$$

As before, if a and b rate their common items identically, we define the impact as

$$impact(i,a,b) = \frac{1}{|C_{ab}|}$$

since $\sum_{w \in C_{ab}} |sim'(a,b) - sim'_{\bar{w}}(a,b)| = 0$.

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

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

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

4.3 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 measure.

In general, the quality of a prediction determines 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) [Konstan 2003], 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 a triple $[SF, CF, PF]$.

4.4 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 $(|p_{u,i} - p_{u,j}| \leq \epsilon)$.

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

$$c(i,j) = \begin{cases} \frac{t}{T} \cdot \frac{1}{C+D} & \text{if } (i,j) \text{ are concordant} \\ -\frac{t}{T} \cdot \frac{1}{C+D} & \text{if } (i,j) \text{ are discordant} \end{cases} \quad (9)$$

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.

4.5 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. To compute these recommendations, the collaborative filtering algorithm traverses many paths initiated by the user's scouts and using other users' ratings as connectors and promoters. We call these connector and promoter 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.

The aim here is to 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 of neighborhood factors $[SFN(u), CFN(u), PFN(u)]$, where

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

and $CFN(u)$ 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.

Note that in defining the connector and promoter factors of the rating neighborhood for a user, we use the connector and promoter factors of the ratings of other users. These factors depend on the influence that these ratings have had in predictions for other users, and as such we assume that the influence of a rating as a connector or promoter will be uniform for all predictions in which they are used.

5. EXPERIMENTATION

As we have illustrated, we can assign role values to each rating when evaluating a collaborative filtering system. In this section, we demonstrate the use of this approach through experiments done on the MovieLens million rating dataset. This section relates to gross statistics about scouts, promoters, and connectors; the next provides specific examples of characterizing neighborhoods and studying the evolution of roles.

5.1 Design

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. We consider both item- and user-based algorithms using both predictive and rank accuracy measures. For both algorithms we form top K neighborhoods using $K = 30$.

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, after 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.

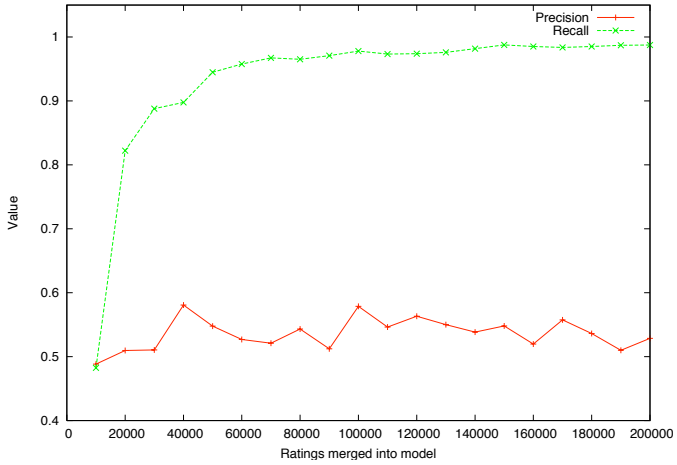


Fig. 4. Precision and recall for the item-based collaborative filtering algorithm.

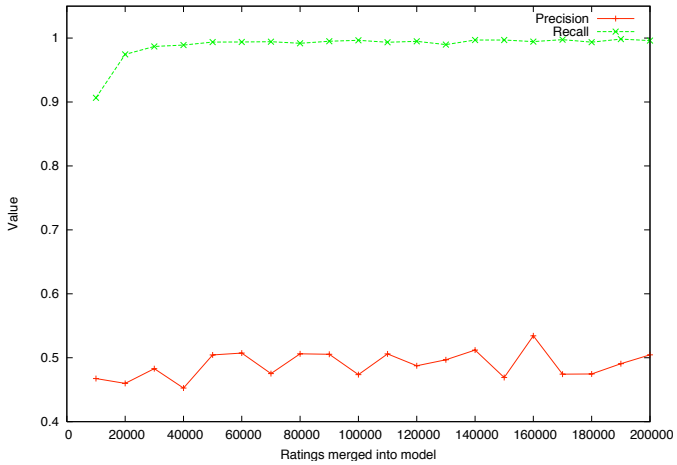


Fig. 5. Precision and recall for the user-based collaborative filtering algorithm.

The item-based collaborative filtering algorithm's performance was ordinary with respect to rank accuracy. Fig. 4 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%. The user-based algorithm performed similarly (Fig. 5), but with an average precision about 4% lower.

5.2 Characterizing 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 val-

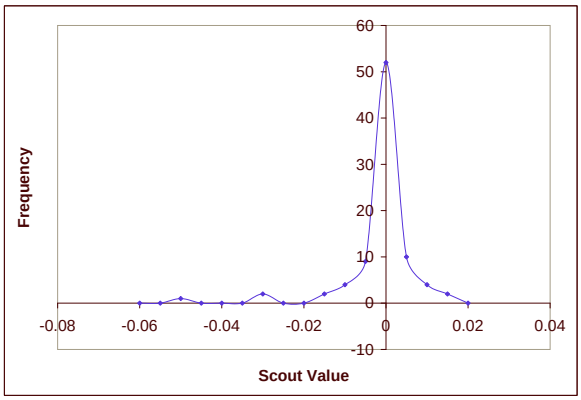
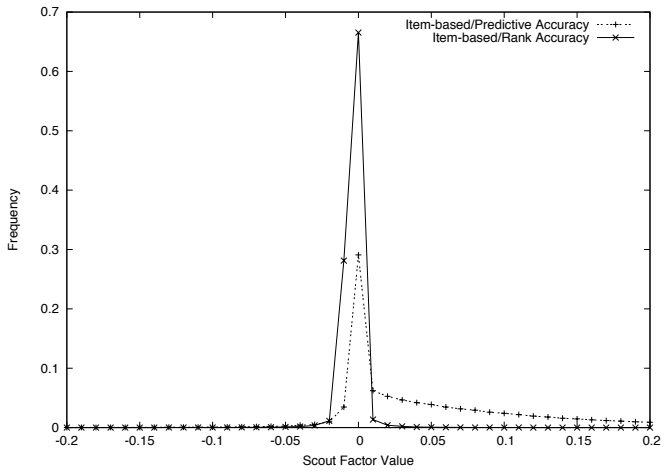


Fig. 6. Distribution of scout values for a sample user (item-based algorithm with rank accuracy).

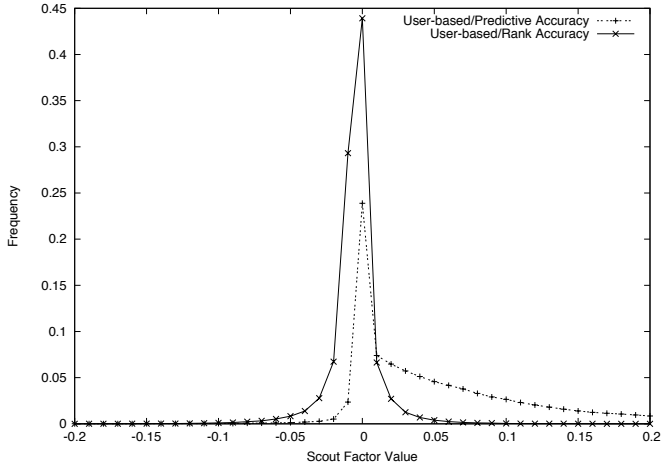
ues. Note that both item-based and user-based algorithms discussed here produce recommendation lists with nearly half of the pairs in the list discordant from the user's preference. Whether all of these discordant pairs are observable by the user is unclear, however, certainly this suggests that there 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 mean-centered at zero. Fig. 6 shows the frequency distribution of scout values for a sample user at a given snapshot of the item-based algorithm, and Fig. 7 shows the frequency distribution of scout values for all four combinations of algorithms and accuracy measures. We observe that a large number of ratings never serve as scouts for their users. A related 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. As we will show later, removing bad scouts is worthwhile for individual users (see Section 6.1) 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.* [Rashid et al. 2002]. The popularity of a movie is the number of times an item is rated. 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. For the item-based algorithm using rank accuracy, we measured the Pearson correlation coefficient between aggregated scout values for a movie (across ratings) and its popularity (number of times it is rated), at different snapshots of the system. Similarly, we track the correlation between aggregated scout value and the popularity*variance measure for a movie. Here, variance is calculated by considering the ratings for



(a)



(b)

Fig. 7. Distribution of scout values for 200,000 ratings using (a) item-based algorithm, and (b) user-based algorithms, with both rank and predictive accuracy.

a movie as a random variable and characterizing the expected squared deviation from the mean rating. Note that the scout values were initially anti-correlated with both notions of popularity (Fig. 8), but became moderately correlated as the system evolved. A possible explanation is that there has been insufficient time for the popular movies to accumulate ratings.

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. Using the observations from the item-based algorithm with rank accuracy, 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

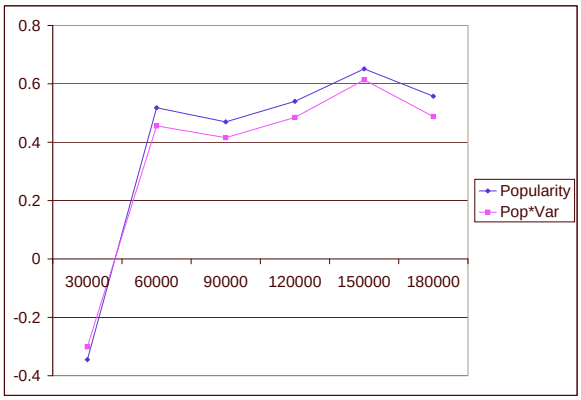


Fig. 8. Correlation between aggregated scout value and item popularity (computed at different intervals).

list. Movies appearing consistently high over time are expected to remain there in the future. The effective confidence in a movie m is

$$C_m = \frac{T_m - B_m}{N} \tag{10}$$

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 in the item-based algorithm (with rank accuracy) are shown in Table I. Movies that would be bad scout choices are shown in Table II 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—involving '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.

Table I. Movies forming the best scouts (item-based algorithm with rank accuracy).

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 II. Movies forming the worst scouts (item-based algorithm with rank accuracy).

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

5.3 Characterizing 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 zero-centered (similar to Fig. 6), as is the promoter distribution in general (Fig. 9). 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. In the item-based algorithm, we find a strong correlation between a user's number of ratings and his aggregated promoter value. Fig. 10 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.

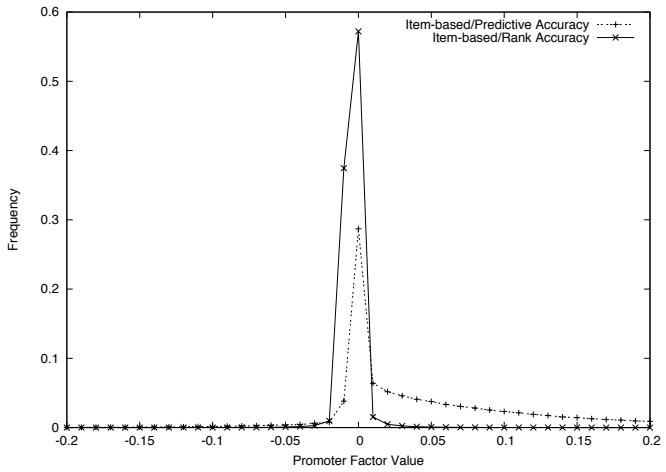
Given this observation, the obvious rule to follow when introducing a new movie is to have it rated directly by prolific users who posses high aggregated promoter values. A new movie is thus cast into the neighborhood of many other movies improving its visibility. Note, though, that a user may have long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

5.4 Characterizing connectors

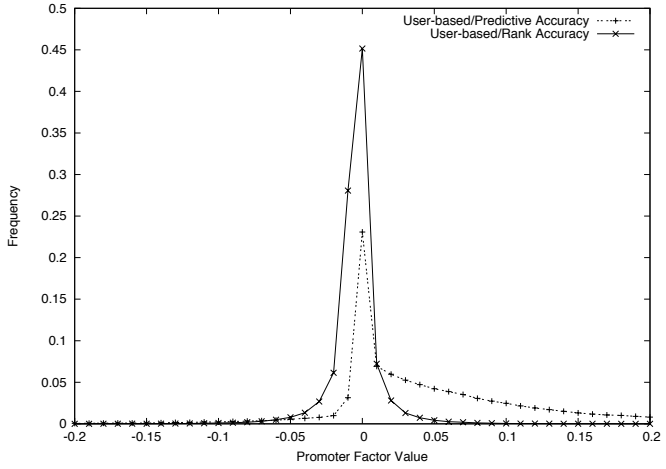
Given the way scouts, connectors, and promoters are characterized, it follows that the movies that are part of the best scouts are also part of the best connectors. Similarly, the users that constitute the best promoters are also part of the best connectors. Good connectors are induced by ensuring a 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.

6. USING ROLES TO IMPROVE AND UNDERSTAND RECOMMENDER SYSTEMS

Thus far, we have characterized the roles played by ratings and tried to situate them in the three-folded distinction of scouts, promoters, and connectors. We now turn to using such characterizations to actively improve recommendations for users, improve targeting of items, and providing tools to monitor the recommender system as a whole. The first two tasks are typically realized by 're-wiring' possibilities, i.e., adjusting the rating value for an already rated movie, transferring the rating value to a different movie, or even removing the existing rating from consideration altogether. We illustrate using examples of removing ratings, but other forms of re-wiring—perhaps with the involvement of the cognizant user—will readily suggest themselves to the reader.



(a)



(b)

Fig. 9. Distribution of promoter values for 200,000 ratings using (a) item-based algorithm, and (b) user-based algorithms, with both rank and predictive accuracy.

6.1 Improving recommendations for users

As noted earlier, manipulating the scout ratings of individuals can yield a major improvement in accuracy for that individual. Here we consider an example from experiments with the item-based algorithm using rank accuracy. Note that it is easier to "rewire" scouts in the item-based algorithm since they are not aggregated into the similarity between items.

In our experiments, we found that user 5863 has a poorly ordered recommendation list. The random selection chose four of his ratings as test cases, one of which was unpredictable. The remaining items ranked by the user's ratings are listed in the first column of Table III. The second column of the table shows the recommendation list based on the predictions, which is completely discordant with

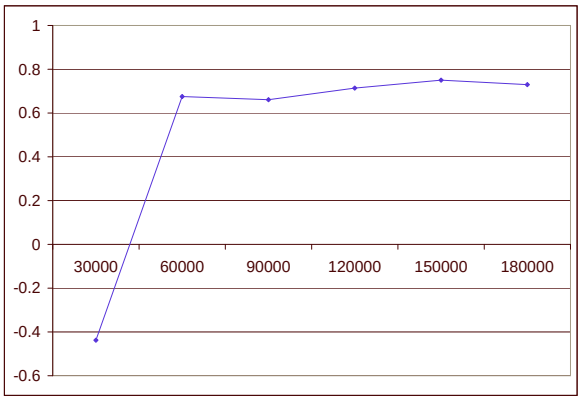


Fig. 10. Correlation between aggregated promoter value and user prolificity (computed at different intervals).

Table III. Removing the rating of *The Sixth Sense* by user 5863 improves concordances between his preferred list and the recommended list.

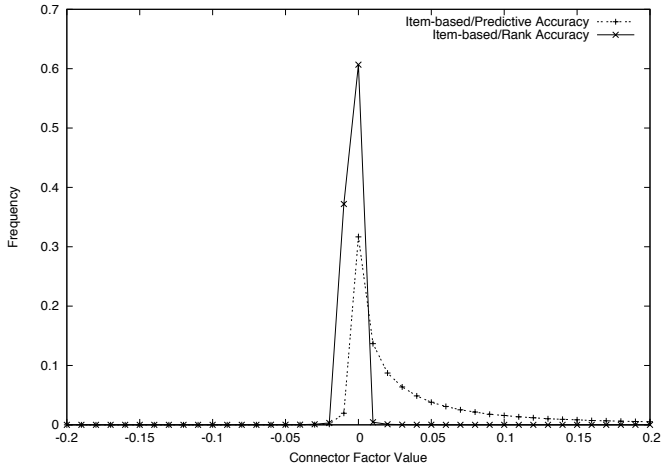
User Preference	With Bad Scout	Without Bad Scout
Double Jeopardy (5)	*The Silence of the Lambs* (4.688)	*Men in Black* (4.145)
Men in Black (4)	*Men in Black* (4.146)	*Double Jeopardy* (3.874)
Silence of the Lambs (3)	*Double Jeopardy* (3.874)	*Silence of the Lambs* (3.473)

the user – rank accuracy is zero!

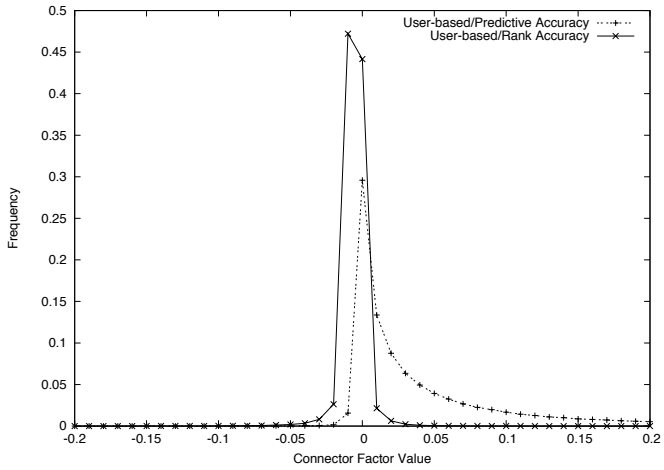
Upon closer inspection, we observe that user 5863 had also rated *The Sixth Sense* (rating 5), but this rating had a large negative scout factor of -0.3497 based on the four runs (where different sets of movies were chosen). We therefore decide to remove *The Sixth Sense* from this user's ratings, and recompute the recommendation list. The new recommendation list indeed moves the user's lowest rated item to the bottom of the list, improving rank accuracy to 66%.

6.2 Improving targeting of items

Items can similarly be affected by manipulations of promoters. We consider an example from experiments with the user-based algorithm using predictive accuracy. Rewiring promoters is easier in this algorithm, because promoters are not aggregated into the similarity of users. In these experiments, we found that the movie *Bad Girls* (ID 416) was being incorrectly recommended to a few users as shown in Table IV. The mean absolute error for these predictions is 1.62. On inspecting the promoter values for ratings involving the movie *Bad Girls*, we found that one rating from user 6016 had a very strong negative promoter factor (-1.7). We decided to remove this rating from the prediction computation. Removing this rating reduced recall, but the accuracy of predictions improved. In particular, removal of the rating



(a)



(b)

Fig. 11. Distribution of connector values for 200,000 ratings using (a) item-based algorithm, and (b) user-based algorithms, with both rank and predictive accuracy.

Table IV. Comparison of user ratings and predictions for *Bad Girls* (ID 416).

User	Rating	Prediction
6003	4	2.34943
5916	4	1.90082
5894	4	2.34684
5848	1	2.21317
5831	3	1.63221

by 6016 disallowed predictions of rating for users 6003, 5916 and 5894. However, the prediction for 5848 changed to 1.6404, while for 5831 it became 2.3853. Hence, there was a 60% loss in coverage, but an improvement in precision to a MAE of 0.63.

As another example, predictions for the the movie *Magnolia* (movie id 3120) trained from 17 users were tested on 6 users and found to have an average absolute error of 1.34. This means that every rating prediction was off by at least 1 level. Closer inspection of the promoter factors revealed that the rating of *Magnolia* by user 6036 stood out with a high negative factor (-0.244). User 6036 also turns out to be a movie buff with 888 ratings in total (recall that, for the MovieLens dataset, every user was required to rate at least 20 movies). We decided to remove this rating of user 6036 from the predictions and observed the average absolute error to improve to 0.63. We also noted that we could only predict ratings for 5 users, out of the 6 tested; specifically we were unable to predict a rating of *Magnolia* for user 5840. This user actually rated the movie with a score of 1, whereas the predicted rating if we had used the bad promoter was 4.3! This example hence shows a huge improvement in precision with negligible loss in recall.

The above examples only reveal a preview of the possibilities; we can explore more sophisticated schemes involving removals/rewirings of ratings for specific classes of users, and for targeted groups of items.

6.3 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. For ease of presentation, we consider only the item-based algorithm with rank accuracy but the same qualitative observations apply to the other algorithms and evaluation metrics as well. 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 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. For simplicity, we will represent these triples as just, for instance, [+, 0, −]. To indicate time stamps, we will use subscripts, e.g., $[+, 0, -]_t$ indicates a time stamp of t for the above triple.

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. Conversely, 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 [Laxman et al. 2005] to count such transitions. In this formulation, two occurrences of a transition (episode) are said to be *non-overlapping* if no event associated with one appears in between the events associated with the other. We define $a \rightarrow b$ to indicate that the event b follows a in the next time frame, and $a \rightsquigarrow b$ to indicate that b occurs after a (not necessarily in the immediate next time frame). Consider the sequence $[+,0,0]_1 \rightarrow [+,0,0]_2 \rightarrow [+,0,+]_3 \rightarrow [+,0,0]_4 \rightarrow [0,0,0]_5$ and let us count the number of occurrences of $[+,0,0] \rightsquigarrow [0,0,0]$. According to the non-overlapping principle, if $[+,0,0]_4 \rightsquigarrow [0,0,0]_5$ counts as one transition, $[+,0,0]_1 \rightsquigarrow [0,0,0]_5$ cannot count as another. The frequency of a transition is defined as the maximum number of non-overlapping occurrences of the transition in the event sequence. Clearly, the set of the non-overlapping occurrences is a subset of the collection of all possible occurrences. Non-overlapping episodes makes detection and tracking of frequency more efficient, and has the desirable property of not overcounting due to repeated states [Laxman et al. 2005].

As another example, the sequence

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

is interpreted as the episodes

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

Table V. Illustration of episode counting.

Time	Current State	States Already Seen	Transitions incremented
t1	[+, 0, 0]	{}	
t2	[+, 0, 0]	$\{([+,0,0],t1)\}$	$([+,0,0] \rightsquigarrow [+,0,0])$
t3	[0, +, 0]	$\{([+,0,0],t2)\}$	$([+,0,0] \rightsquigarrow [0,+,0])$
t4	[0, 0, 0]	$\{([+,0,0],t2),([0,+,0],t3)\}$	$([+,0,0] \rightsquigarrow [0,0,0])$ $([0,+,0] \rightsquigarrow [0,0,0])$
t5	[0, 0, 0]	$\{([+,0,0],t2),([0,+,0],t3),([0,0,0],t4\}$	$([0,0,0] \rightsquigarrow [0,0,0])$

instead of

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

See [Laxman et al. 2005] for further details about the probabilistic basis underlying the counting of non-overlapping episodes.

In our implementation, at any given point in time, we track the different states a rating has been through, and their last observed time stamps. Let S be the set of time-stamped states (a, t_a) a rating has been through, where a is the state, and t_a is its last seen time stamp. When we encounter a state b at some new time instant (say t), we must look into S to see if there is already an encounter of b, and suitably modify the time stamp. Specifically, for each $(a, t_a) \in S$, we compute

$$frequency([a \rightsquigarrow b]) \leftarrow frequency([a \rightsquigarrow b]) + \begin{cases} 1 \text{ if } (b, t_b) \notin S \\ 1 \text{ if } (b, t_b) \in S \text{ and } t_b \leq t_a \\ 0 \text{ if } (b, t_b) \in S \text{ and } t_b > t_a \end{cases}.$$

The first clause considers the possibility that b has not been seen before, i.e., there is no occurrence of some (b, t_b) in S. In this case, the frequency is incremented (to 1). The second and third clauses cover the cases where b has been seen before but could either lead to a non-overlapping occurrence (second clause) or overlapping occurrence (third clause). Finally, we update the timestamp associated with b to the new time stamp (i.e., t) in S.

Consider an example: Let $[+,0,0]_{t1} \rightarrow [+,0,0]_{t2} \rightarrow [0,+,0]_{t3} \rightarrow [0,0,0]_{t4} \rightarrow [0,0,0]_{t5}$ be the sequence of states a single rating goes through, and t_i the timestamps when the rating reached that state. Table V illustrates the counting method at different time instants. At $t3$, $([+,0,0] \rightsquigarrow [0,+,0])$ is incremented only once. At $t5$, we do not increment the frequency for the transitions $([+,0,0] \rightsquigarrow [0,0,0])$ and $([0,+,0] \rightsquigarrow [0,0,0])$, since the last time $[0,0,0]$ was seen was at $t4$ which is after $t2$ and $t3$. (We know that we have already accounted for those transitions.) We increment only $([0,0,0] \rightsquigarrow [0,0,0])$. Hence we account only for non-overlapping transitions.

Since each rating can be in 27 possible states (3^3, where there are three rating values roles be tracked, each of which could be in three possible states), there are 27^2 possible transitions (including transitions where nothing changes). This would make

our graph of possible transitions too unwieldly. Typically we are interested in seeing if a good scout becomes a bad connector, a good promoter becomes a good scout, and so forth. We hence introduce a new, simpler, set of transitions to track, where both ratings on the left and right of the transition are modeled as scouts, promoters, *or* connectors (rather than *and* as done originally). Hence our transitions are of the form [scout/connector/promoter X] $\rightsquigarrow$ [scout/connector/promoter Y] where X, Y can be {+, 0, -}. This still provides 9×9 possibilities. For instance, a transition such as $[+, 0, +] \rightsquigarrow [0, +, 0] : 1$ is counted as

$$\begin{aligned}
[scout+] &\rightsquigarrow [scout0] : 1 \\
[scout+] &\rightsquigarrow [connector+] : 1 \\
[scout+] &\rightsquigarrow [promoter0] : 1 \\
[connector0] &\rightsquigarrow [scout0] : 1 \\
[connector0] &\rightsquigarrow [scout+] : 1 \\
[connector0] &\rightsquigarrow [promoter0] : 1 \\
[promoter+] &\rightsquigarrow [scout0] : 1 \\
[promoter+] &\rightsquigarrow [connector+] : 1 \\
[promoter+] &\rightsquigarrow [promoter0] : 1
\end{aligned}$$

Of these, transitions like $[p\ X] \rightsquigarrow [q\ 0]$ where $p \neq q, X \in \{+, 0, -\}$ are considered uninteresting, and only the rest are counted. This hence requires only $9^2 - 18$, or 63 transitions.

Fig. 12 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. 12, 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.

—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.

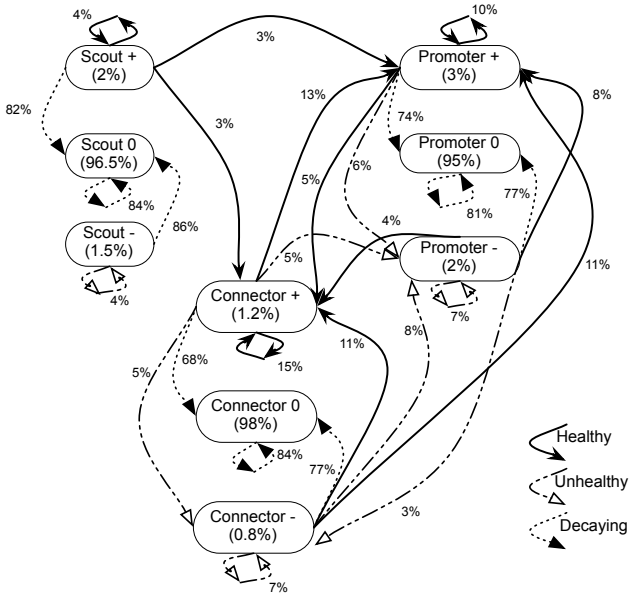


Fig. 12. Transitions among rating roles.

—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. 12 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.

6.4 Characterizing neighborhoods

Earlier we saw that we can characterize the neighborhood of ratings involved in creating a recommendation list L for a user. In the next experiment, we consider the item-based algorithm with rank accuracy, use lists of length 30, 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.

7. 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. In future research, we plan to conduct a live field study involving the use of rating roles to rewire ratings and evaluate improvements in user experience and other metrics as the changes are implemented. 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.

REFERENCES

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

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

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

Konstan, J. A. 2003. Personal communication.

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

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

McLaughlin, M. R. and Herlocker, J. L. 2004. 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*. 329 – 336.

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

Mohan, B. K., Keller, B. K., and Ramakrishnan, N. 2006. Scouts, Promoters, and Connectors: The Roles of Ratings in Nearest-Neighbor Collaborative Filtering, In *Proceedings of the Seventh ACM Conference on Electronic Commerce (EC'06)*, (June), pp. 250–259.

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

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

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

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

Schein, A., Popescu, A., Ungar, L., and Pennock, D. 2002. Methods and Metrics for Cold-Start Recommendation. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 253–260.