Improving Item-Item Similarity Estimation during Collaborative Filtering

Bill Chickering and Jamie Irvine
CS 399 with Anand Rajaraman
Stanford University

March 25, 2014

Abstract

The problem of inaccurate item-item similarity estimates due to a lack of data is explored within the context of collaborative filtering. An off-line experiment that evaluates the performance of similarity estimators is described. Two novel item-item similarity estimators are described and analyzed. The first is based on a model of noisy similarity score generation and proves effective at estimating both future Pearson correlations and cosine similarities. The second models noisy ratings as generated by similarity score dependent distributions. Using the notion of user-predictivity, accurate estimates of future Pearson correlation are made.

Introduction

Determining the similarity of two users or items is valuable for a number of data-mining applications including recommender systems as well as product assortment and differentiation. Collaborative filtering is often used to exploit user-item rating data to make such estimates. This is typically done by computing the Pearson correlation or cosine similarity of vector representations of the users or items [1]. Given sufficient user-item rating data, this approach can be very effective. However, when the rating data is sparse for one or both users/items, these metrics can yield dramatic discrepancies compared to values computed with sufficient data at a later point in time. Moreover, these early, low-confidence measurements can be indistinguishable from later, high-confidence measurements. In this report, we describe two novel techniques for estimating the true similarity of two users or items given limited common rating data by incorporating confidence into the measurement.

To keep the present work focused, we confine ourselves to memory-based collaborative filtering (CF) in which we work with a logical user-item matrix. Each element of the matrix corresponds to a rating given by a particular user to a particular item. Further, we choose to limit our analysis to item-item similarity but note that these techniques should be equally effective at improving user-user similarity estimation. In this representation, an item is a vector of the ratings it has received from all users. We now formalize the notion of item-item similarity. Two alternative models for the generation of user-item ratings are useful in arriving at two different definitions of the *true* item-item similarity.

True Item-Item Similarity

In the first model, we imagine users are randomly presented items to rate. Given a sufficient period of time, all users will eventually rate all items, creating item vectors that contain provided ratings from all users. The true similarity in this model of *random rating*, denoted $TrueSimRand$, is then the vector product of two complete item vectors. That is,

$$TrueSimRand(A,B) = \frac{\sum\limits_{u \in U} r_{u,A} r_{u,B}}{\sqrt{\sum\limits_{u \in U} r_{u,A}^2} \sqrt{\sum\limits_{u \in U} r_{u,B}^2}}, \tag{1}$$

where A and B are items, U is the set of all users and $r_{u,I}$ is the rating given by user u to item I.

In the second model, we imagine an unlimited supply of users, each of whom chooses to rate a subset of items. The lack of a rating is now meaningful since it indicates a lack of interest of a particular user for a particular item. This is captured by the model through a default rating of zero[1]. At any point in time, we may estimate the true similarity in this model of *preferential rating*, by computing the cosine similarity of the two item vectors. Only in the limit of infinite time does this estimate converge to the true similarity, which we denote $TrueSimPref$. That is,

$$TrueSimPref(A,B) = lim_{t \to \infty} \frac{\sum\limits_{u \in U_{AB}} r_{u,A} r_{u,B}}{\sqrt{\sum\limits_{u \in U_A} r_{u,A}^2} \sqrt{\sum\limits_{u \in U_B} r_{u,B}^2}}, \tag{2}$$

where U_A is the set of users who have rated A, U_B is the set of users who have rated B, and U_{AB} is the set of users who have rated both A and B.

Estimating Item-Item Similarity

Of course, most systems will never acquire ratings from all users for all items. Nor are we able to work in the limit of infinite time. We must therefore make estimates of the true item-item similarity using only partial information. In the case of $TrueSimPref$, the cosine similarity computed at any point in time serves as a simple and obvious estimate. In using this estimate, we are treating all missing ratings as intentionally omitted. When estimating $TrueSimRand$, the choice of how to handle missing ratings is perhaps less obvious. Following the convention of Breese et al. [2], we choose to use the Pearson correlation[2] for this estimate, ignoring all ratings by users who have not rated both items. To summarize, our naive estimates are

$$PearSim(A,B) = \frac{\sum\limits_{u \in U_{AB}} r_{u,A} r_{u,B}}{\sqrt{\sum\limits_{u \in U_{AB}} r_{u,A}^2} \sqrt{\sum\limits_{u \in U_{AB}} r_{u,B}^2}} \stackrel{?}{\approx} TrueSimRand(A,B) \tag{3}$$

[1] We address the issue of rating bias and the precise meaning of a zero valued ratings later in this report.

[2] The Pearson correlation is traditionally characterized by subtracting an expectation value from each dimension. This is only approximately true in our case. We discuss how we subtract rating biases later in this report.

$$CosSim(A,B) = \frac{\sum\limits_{u \in U_{AB}} r_{u,A} r_{u,B}}{\sqrt{\sum\limits_{u \in U_A} r_{u,A}^2}\sqrt{\sum\limits_{u \in U_B} r_{u,B}^2}} \overset{?}{\approx} TrueSimPref(A,B). \tag{4}$$

Problem

Both similarity estimates primarily leverage information from common users, that is users who have rated both items. Because of this, they perform well with a large number of common users, but give unreliable, and dramatically varying, results when the items have few common users. In an extreme case, if only one user has rated item A and item B, $PearsSim(A,B) = 1$ or -1. Not only is this unlikely to be an accurate assessment of the true similarity of A and B, it also gives the most extreme result possible, without giving any indication that this is a low-confidence calculation.

In this paper, we present two approaches to better approximate the true similarity given a limited number of common reviewers. Both approaches are motivated by probabilistic models with distributions grounded in the data. The first method leverages the observed naive similarity estimate and the number of common users to better estimate the true similarity. The second method leverages the observed ratings themselves for the same ends.

Data

In the following section, we will describe an off-line experiment used to evaluate our similarity estimators. But first, we provide some details about the data used for this work. The data, originally from Amazon.com, was acquired from the Stanford Network Analysis Project[3]. It consists of 5.6M ratings by 1.2M users of 600k music items (e.g. albums and songs). Each rating is from 1 to 5 stars and is associated with a particular user and item at a particular time. For practical reasons, we focus our work on two subsets of this data. The first consists of the 13,752 item pairs having at least 40 common users (i.e. users that have provided a rating for both items). The second consists of the 1,312 item pairs having at least 80 common reviewers. As discussed below, we must confine ourselves to item pairs that have many common reviewers in order to determine a good approximation of their true similarity.

Before any analysis of the data, we remove the biases from each rating, as in [4]:

$$\hat{r_{ui}} = r_{ui} - \mu_r - b_u - b_i \tag{5}$$

where r_{ui} is the raw rating user u gave item i, μ_r is the average rating over the entire dataset, b_u is the user bias, calculated as the average rating from user u after μ_r has been subtracted from each, and b_i is the item bias, calculated as the average rating of item i after μ_r and b_u have been subtracted from each.

Experiment

Ideally, we would know *a priori* the true similarity for a set of item pairs. We could then generate similarity estimates and readily determine the true error in these estimates. But both $TrueSimRand$ and $TrueSimPref$ are theoretical constructs and not metrics we have access

[3] http://snap.stanford.edu

to in the real world. Fortunately, we can achieve good approximations of either $TrueSim$ given sufficient data. We therefore define $GoldSimRand = PearSim_N$ and $GoldSimPref = CosSim_N$ where N is a large enough number of common users that $GoldSim \approx TrueSim$. With a measurable $GoldSim$ as our goal, we can implement a simple predictivity experiment.

To obtain good $GoldSim$ values, we work only with the item pairs that have at least either $N = 40$ or 80 common users. These choices are a compromise between having enough data for statistically significant results and having a sufficiently large N that $GoldSim \approx TrueSim$.

In order to evaluate estimates of $TrueSim$, we construct an off-line experiment to simulate observing sparse amounts of rating data. Our experiment reenacts the passing of time. Starting with all ratings withheld, we apply ratings to items in the order they actually occured. The similarity of item pairs is then estimated each time the number of common users n is incremented. In this way, we can quantitatively compare various estimators by calculating their mean squared error. Since different estimators may perform better for different degrees of sparsity, we compute the mean squared error for $n = 1, ..., N$ separately.

Score-based Method

Probabilistic Model

In our first approach, we model the probability distributions of true and observed similarity. The goal is to use the observed similarity score directly to approximate the true similarity. In this way, we can consider an abstact $TrueSim$ which ignores the details of $TrueSimRand$ or $TrueSimPref$. Similarly, we consider Sim_n to be a generic observed similarity score when the two items have n common reviewers. Let Y be a random variable representing the $TrueSim$ of a randomly chosen pair of items. Let X_n be a random variable representing the Sim_n of a pair of items when n common users are observed.

We model Y as a Normal distribution

$$Y \sim N(\mu_s, \sigma_s^2), \tag{6}$$

where μ_s and σ_s^2 are the average similarity score and the variance of similarity scores of all pairs of items, respectively. Figure 1 shows the distributions of $GoldSim$, which motivates the Gaussian model for $TrueSim$.

Since X_n represents a noisy reading of the true similarity Y, we model $X_n|Y = y$ as a Gaussian error around y:

$$(X_n|Y = y) \sim N(y, \sigma_{err,n}^2) \tag{7}$$

Note that $\sigma_{err,n}^2$ represents how noisy Sim_n is as an estimate of $TrueSim$ and therefore should decrease as n increases. The probability distribution of X_n is illustrated in Figure 2.

The problem of approximating $TrueSim$ given Sim_n can now be seen as finding the value of Y that most likely produced X_n. In other words we desire the Maximum Likelihood Estimate

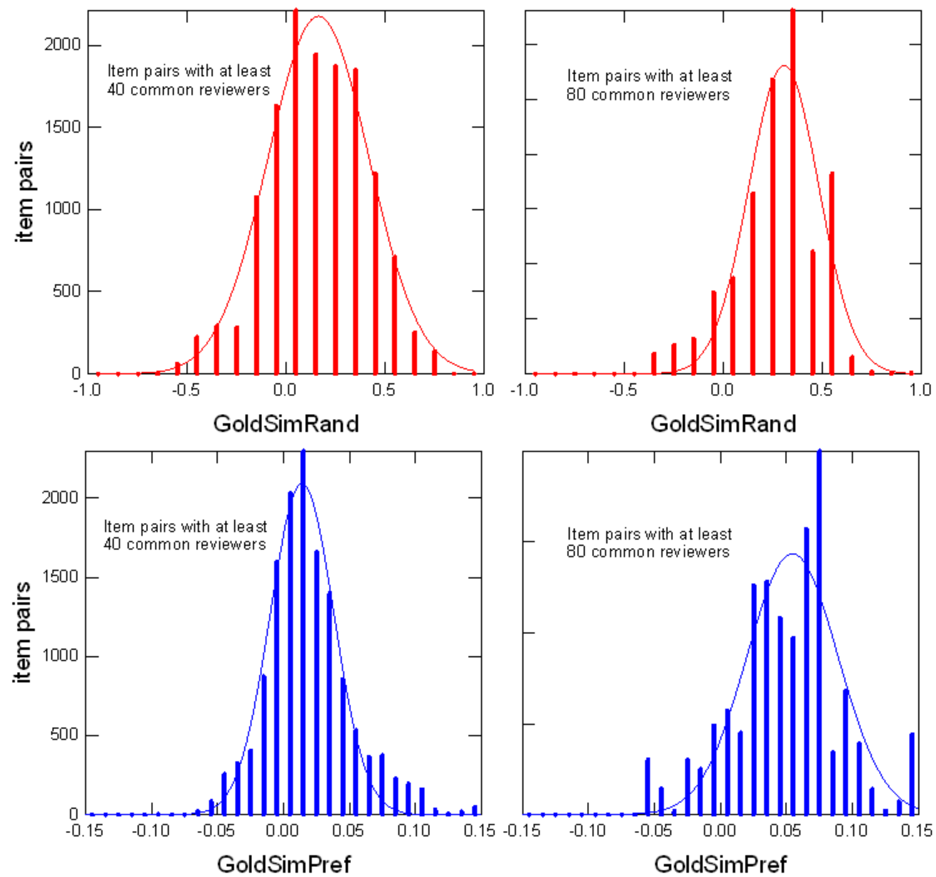

Figure 1: Histograms of *GoldSimRand* (top) and *GoldSimPref* (bottom) for training datasets containing at least 40 common reviewers (left) and 80 common reviewers (right).

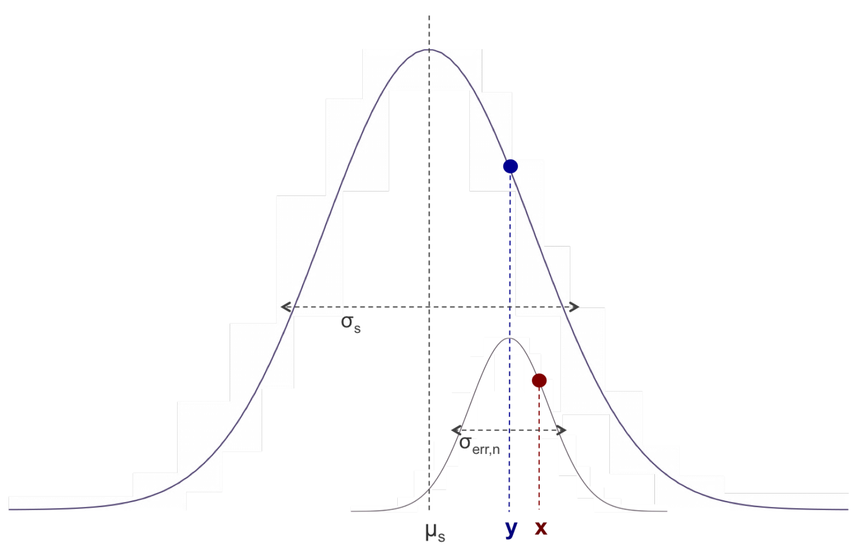

Figure 2: The blue curve represents Y, the distribution of true similarities between a random pair of items. The red curve represents $X|Y$, the observed similarity when a small number of common users exist.

of $Y|X_n$:

$$\hat{y} = \underset{y}{\operatorname{argmax}} P(Y = y|X_n = x) \tag{8}$$

$$= \underset{y}{\operatorname{argmax}} \left[\frac{P(X_n = x|Y = y)P(Y = y)}{P(X_n = x)}\right] \tag{9}$$

$$= \underset{y}{\operatorname{argmax}} \left[P(X_n = x|Y = y)P(Y = y)\right] \tag{10}$$

$$= \underset{y}{\operatorname{argmax}} \left[\frac{1}{\sigma_{err,n}\sqrt{2\pi}} \exp\left(\frac{-(x-y)^2}{2\sigma_{err,n}^2}\right) \frac{1}{\sigma_s\sqrt{2\pi}} \exp\left(\frac{-(y-\mu_s)^2}{2\sigma_s^2}\right)\right] \tag{11}$$

$$= \underset{y}{\operatorname{argmin}} \left[\frac{(x-y)^2}{2\sigma_{err,n}^2} + \frac{(y-\mu_s)^2}{2\sigma_s^2}\right] \tag{12}$$

$$= \underset{y}{\operatorname{argmin}} \left[\left(\sigma_s^2 + \sigma_{err,n}^2\right) y^2 - 2\left(\sigma_s^2 x + \sigma_{err,n}^2 \mu_s\right) y\right] \tag{13}$$

We find the exact minimum by taking the derivative with respect to y and setting it to zero:

$$\frac{d}{dy}\left[\left(\sigma_s^2 + \sigma_{err,n}^2\right) y^2 - 2\left(\sigma_s^2 x + \sigma_{err,n}^2 \mu_s\right) y\right] = 2\left(\sigma_s^2 + \sigma_{err,n}^2\right) y - 2\left(\sigma_s^2 x + \sigma_{err,n}^2 \mu_s\right) \tag{14}$$

$$= 0 \tag{15}$$

This produces the following linear equation for $\hat{y}$ in terms of x

$$\hat{y} = \frac{\sigma_s^2 x + \sigma_{err,n}^2 \mu_s}{\sigma_s^2 + \sigma_{err,n}^2}, \tag{16}$$

which can be rewritten as

$$(\hat{y} - \mu_s) = \frac{\sigma_s^2}{\sigma_s^2 + \sigma_{err,n}^2}(x - \mu_s). \tag{17}$$

The model suggests that there is a linear correlation between X_n (Sim_n) and Y ($TrueSim$).[4] Moreover, the best predicted $TrueSim$ is a linear combination of the average true similarity and the observed similiarity. Since $\sigma_{err,n}^2$ is nonnegative, the slope is always less than or equal to one. This means that, on average, the observed distance from μ_s overestimates the true distance from μ_s. This is reasonable, since the prior distribution suggests that the true similarity is likely to be near μ_s.

This model requires a different parameter $\sigma_{err,n}^2$ for each n. To gain more generality, we model $\sigma_{err,n}^2$ as a function of n instead of considering them independent of each other. Recall that $\sigma_{err,n}^2$ is the noisiness of the observed Sim_n about $TrueSim$. Sim_n converges to $TrueSim$ as the number of users n increases. Although Sim is not simply a sample mean, The Central Limit Theorem motivates the intuition that the variance of Sim would decrease as the square root of n. Thus we model $\sigma_{err,n}^2$ as

$$\sigma_{err,n}^2 = \frac{\alpha}{\sqrt{n}}, \tag{18}$$

yielding the single parameter model

$$\hat{y} = \frac{\sigma_s^2}{\sigma_s^2 + \frac{\alpha}{\sqrt{n}}}(x - \mu_s) + \mu_s. \tag{19}$$

[4]It is worth noting that despite the resemblance to Slope One Predictors [3], this is a fundamentally different technique. Here, we have a linear relationship between observed scores and true scores, whereas the Slope One model seeks a linear relationship between the ratings a user gives to different items.

Technique and Results

With a measurable *GoldSim* as our predictive goal, we can apply supervised learning techniques. First, we find pairs of items that have at least N common users and calculate their *GoldSim*. Next we withold some of the users to produce estimated Sim_n for $n = 1, ..., N$. By doing this over many pairs of items, we produce pairs of $(Sim_n, TrueSim)$ across different values of n that can be used for training. Now the parameters μ_s, σ_s^2, and $\sigma_{err,n}^2$ can be approximated using the method of moments over the training data. The values of $\sigma_{err,n}^2$ are fit to the function $\frac{\alpha}{\sqrt{n}}$ and the α that minimizes the sum of squared error is computed.

The graph of Fig. 3 compares the results of three linear predictors, with varying degrees of fidelity to the probabilistic model. The most faithful predictor uses μ_s, σ_s^2 and α as calculated above to predict the $TrueSim$ using Eqn (19). The next predictor uses the calculated values of $\sigma_{err,n}^2$ directly, following Eqn (16) for predictions. The third predictor simply uses a series of linear regressions, one for each n. This model only assumes that for each n, Sim_n and $TrueSim$ are linearly correlated. The linear regression technique trades the generality of the model for a better fit on the training data. As the results show, the trade-off is worth it.

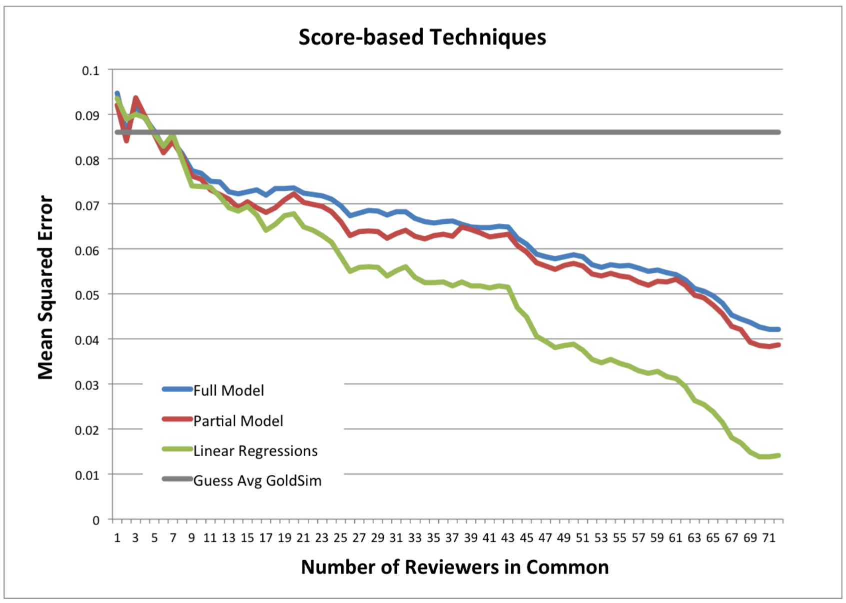

Figure 3: Comparison of different linear predictors. For each number of reviewers in common, n, the graphs represent the average squared error of predicted *GoldSim* calculated from the observed Sim_n. As a baseline, the technique of guessing the average *GoldSim* is included. For these experiments, we use $Sim = PearSim$ and $N \geq 80$ for determining *GoldSim*.

Ratings-based Method

Probabilistic Model

We now construct a different probabilistic model based on the pairs of ratings themselves. This model is readily exploited for making estimates of $TrueSimRand$ and, to a lesser extent, $TrueSimPref$. In addition, this approach allows us to infer implied predictions of $TrueSimRand$ from individual users each time they rate two items. This latter feature allows us to define a new metric for users that we call *user-predictivity*. User-predictivity allows us to ascribe varying levels of confidence to a user's ratings and thus modify the contribution a user makes to a similarity estimate accordingly.

Let Y be a random variable representing the *TrueSimRand* of a randomly chosen pair of items. As before, we model Y as a Normal distribution

$$Y \sim N(\mu_s, \sigma_s^2), \tag{20}$$

where μ_s and σ_s^2 are the average and variance of all true similarity scores, respectively. Let $\vec{R} = (R_A \;\; R_B)$ be a random two element vector representing the ratings given to items A and B. We model the conditional distribution of $\vec{R}|Y$ as a multivariate Normal distribution

$$(\vec{R}|Y = y) \sim N(\vec{\mu_r}, \mathbf{\Sigma_y}), \tag{21}$$

where $\vec{\mu_r} = (\mu_r \;\; \mu_r)$ are the average ratings and $\mathbf{\Sigma_y}$ is the y-dependent 2×2 covariance matrix. Since y is the *TrueSimRand* score, it can be thought of as the Pearson correlation over the distribution of ratings $R_A|Y$ and $R_B|Y$. Assuming that the variance of individual ratings σ_r^2 is independent of y, we have

$$y = \frac{\mathrm{Cov}[R_A|Y = y, R_B|Y = y]}{\sigma_r^2}, \tag{22}$$

It therefore follows that if the ratings are indeed normally distributed then the covariance matrix is related to the true similarity y by

$$\mathbf{\Sigma_y} = \sigma_r^2 \cdot \begin{bmatrix} 1 & y \\ y & 1 \end{bmatrix}. \tag{23}$$

Using this model, we may view the problem of estimating the similarity Y of items A and B that most likely produced ratings R_A and R_B as that of finding the Maximum Likelihood Estimate of $Y|\vec{R}$:

$$\begin{aligned}
\hat{y} &= \operatorname*{argmax}_y P(Y = y|\vec{R} = \vec{r}) && (24)\\
&= \operatorname*{argmax}_y \left[\frac{P(\vec{R} = \vec{r}|Y = y)P(Y = y)}{P(\vec{R} = \vec{r})}\right] && (25)\\
&= \operatorname*{argmax}_y \left[P(\vec{R} = \vec{r}|Y = y)P(Y = y)\right] && (26)\\
&= \operatorname*{argmax}_y \left[\frac{1}{2\pi\sqrt{|\mathbf{\Sigma_y}|}} \exp\left(-\frac{1}{2}(\vec{r} - \vec{\mu_r})^{\mathrm{T}}\mathbf{\Sigma_y}^{-1}(\vec{r} - \vec{\mu_r})\right) \frac{1}{\sigma_s\sqrt{2\pi}} \exp\left(\frac{-(y - \mu_s)^2}{2\sigma_s^2}\right)\right] && (27)\\
&= \operatorname*{argmin}_y \left[(\vec{r} - \vec{\mu_r})^{\mathrm{T}}\mathbf{\Sigma_y}^{-1}(\vec{r} - \vec{\mu_r}) + \frac{(y - \mu_s)^2}{\sigma_s^2} + \ln|\mathbf{\Sigma_y}|\right] && (28)\\
&= \operatorname*{argmin}_y \left[\vec{r'}^{\mathrm{T}}\mathbf{\Sigma_y}^{-1}\vec{r'} + \frac{(y - \mu_s)^2}{\sigma_s^2} + \ln|\mathbf{\Sigma_y}|\right] && (29)\\
&= \operatorname*{argmin}_y \left[\frac{1}{\sigma_r^2}\frac{1}{1 - y^2}\left[r'_A(r'_A - yr'_B) + r'_B(r'_B - yr'_A)\right] + \frac{(y - \mu_s)^2}{\sigma_s^2} + \ln\left[\sigma_r^2(1 - y^2)\right]\right], && (30)
\end{aligned}$$

where we used the fact that

$$\mathbf{\Sigma_y}^{-1} = \frac{1}{\sigma_r^2}\begin{bmatrix} 1 & y \\ y & 1 \end{bmatrix}^{-1} = \frac{1}{\sigma_r^2}\frac{1}{1 - y^2}\begin{bmatrix} 1 & -y \\ -y & 1 \end{bmatrix}. \tag{31}$$

and that

$$|\mathbf{\Sigma}| = \sigma_r^2(1-y^2) \tag{32}$$

along with the substitution $\vec{r'} = (r'_A \;\; r'_B) = \vec{r} - \vec{\mu_r}$. Next, we find the minimum by taking the derivative with respect to y and setting it to zero:

$$\frac{d}{dy}\left[\frac{1}{\sigma_r^2}\frac{1}{1-y^2}\left[r'_A(r'_A - yr'_B) + r'_B(r'_B - yr'_A)\right] + \frac{(y-\mu_s)^2}{\sigma_s^2} + \ln\left[\sigma_r^2(1-y^2)\right]\right] = 0 \tag{33}$$

$$-r'_A r'_B \hat{y}^2 + ({r'_A}^2 + {r'_B}^2)\hat{y} - r'_A r'_B + \frac{\sigma_r^2}{\sigma_s^2}(1-\hat{y}^2)^2(\hat{y}-\mu_s) - \sigma_r^2(1-\hat{y}^2)\hat{y} = 0. \tag{34}$$

This final expression implicitly yields the MLE of y as a function of the ratings r_A and r_B. That is, by constraining the parameters μ_r, σ_r, μ_s, and σ_s to realistic values (e.g. $\sigma_r, \sigma_s > 0$), this expression defines a surface that can be used to efficiently estimate y given a pair of ratings[5].

User Predictivity

As already mentioned, the estimator $PearSim$ is unable to say much about the similarity of two items when only provided ratings from a single common user. In this case, $PearSim$ yields ± 1 (or the indeterminate value $0/0$ in the case that both ratings are zero) depending on whether the ratings have the same sign or not. The value of $\hat{y}$ provided by the above model rectifies this by generating similarity estimates in the range $[-1, +1]$ given a single pair of ratings. We therefore consider each user who rates a pair of items as implicitly predicting the similarity of the items. A key discovery of the present work is that the accuracy of users' predictions in a training set is correlated with the accuracy in a test set. We exploit this finding by ascribing a quantity we call *user-predictivity* to each user defined as

$$p_u = e^{-\epsilon_u/\lambda}, \tag{35}$$

where λ is a parameter to be learned and

$$\epsilon_u = \frac{1}{|I_u|}\sum_{A,B \in I_u} |\hat{y}(r_{u,A}, r_{u,B}) - GoldSim(A,B)| \tag{36}$$

with I_u the set of item pairs rated by user u. We are once again using $GoldSim$ as a proxy for $TrueSim$ (See **Experiment** section) such that ϵ_u is the average absolute error of user u's similarity predictions.

Technique and Results

The parameter values μ_s and σ_s are obtained by fitting the distribution of $GoldSim$ values in the training set (see Fig. 1). For example, in the $N \geq 80$ dataset we have $\mu_s = 0.27$ and $\sigma_s = 0.25$. The choice of μ_r and σ_r is less clear. Some insight can be gained by examining the prior distribution of ratings for our datasets, which are shown in Fig. 4. From there, we experiment with different values of these parameters to minimize the resulting error over the training set. We find $\mu_r = 0.25$ and $\sigma_r = 0.3$ to be good parameters and use them when estimating $TrueSimRand$.

[5] Note that this equation neglects the constraint that $-1 \leq y \leq 1$. As a consequence, some combinations of ratings and parameters can yield MLEs for y that exceed 1 in magnitude. This is easily fixed, however, by applying this additional constraint.

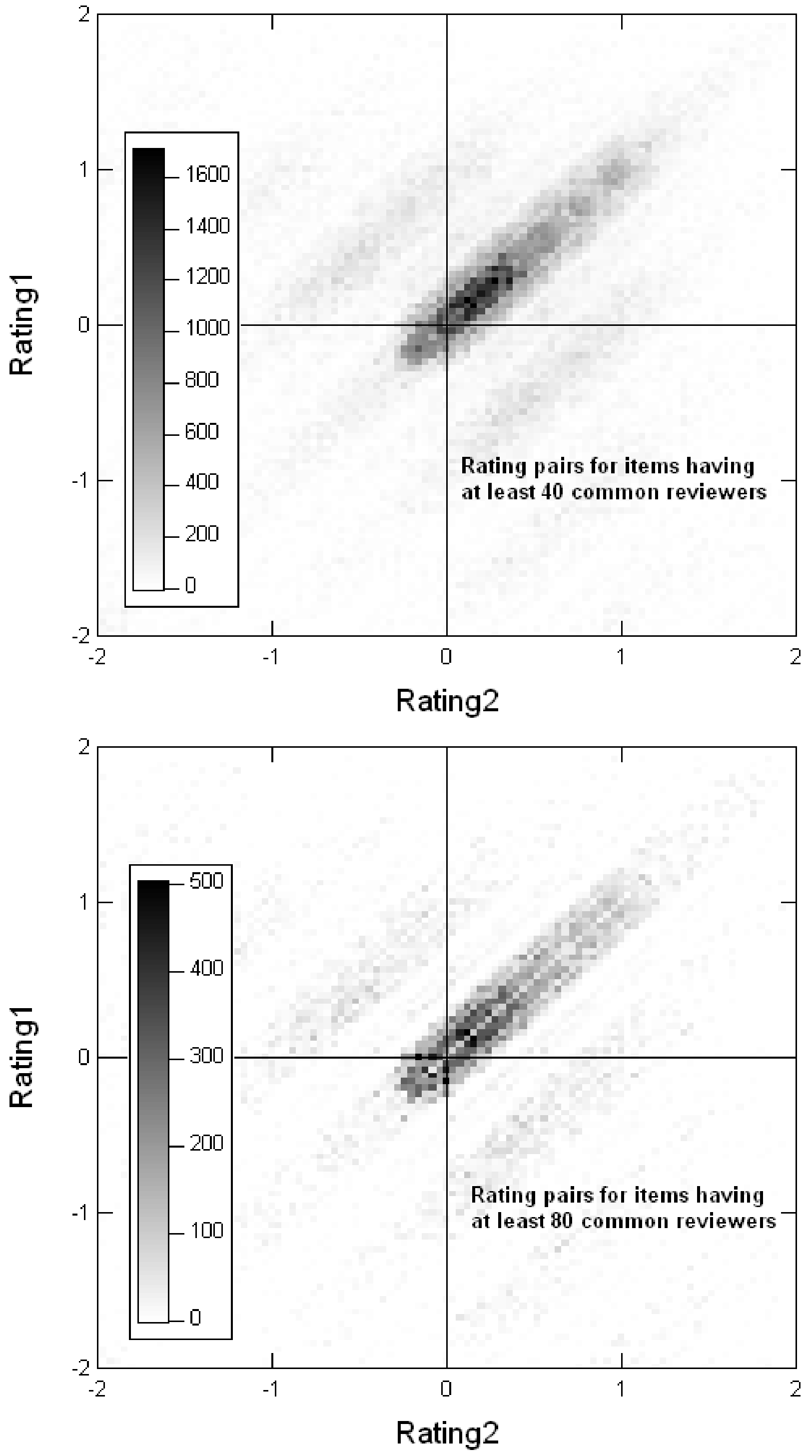

Figure 4: Distribution of rating pairs by common users for item pairs. Top figure shows distribution for item pairs having at least 40 common users. Bottom figure shows distribution for item pairs having at least 80 common users. Rating1 and Rating2 are chosen in arbitrary order.

Figure 4 shows density plots of rating pairs, for $N \geq 40$ and $N \geq 80$. These data, together with the similarity score histograms of Fig. 1, suggest that the data for our two data sets are similar and that large biases are not introduced by varying the number of common users, N. At the same time, both rating distributions reveal significant correlations between Rating1 and Rating2 as evidenced by the elongation in the 45° direction. This is consistent with the data of Fig. 1, which shows the mean $PearSim > 0$ for both sets. Further, these data indicate a bias toward positive values among rating pairs compared to those of ratings in general, which have an average value of zero by construction[6]. This latter feature indicates that items with large numbers of common users tend to be rated higher by their common users than by their other users since we have already subtracted the item's average rating thereby avoiding any simple popularity effects.

We observe that while the data of Fig. 4 do not conform precisely to a multivariate Gaussian distribution, this approximation is not an unreasonable one. In passing, we remark on the faint clusters that flank the central cluster in the figures. We understand these to be artifacts of the bias subtraction. Moreover, a more sophisticated Rating-based model could account for these by considering a $P(\vec{R}|Y)$ that is a mixture of Gaussians.

Having chosen values for μ_s, σ_s, μ_r, and σ_r we are now able to estimate the similarity of item pairs. To do this, we must solve Eq. (34) for $\hat{y}$ given a pair of ratings. This can be easily done in practice using a root finding algorithm[7]. The implied predictions of individual users can then be aggregated to form a new estimate of the similarity of items A and B as

$$UserPredSim(A, B) = \frac{\sum_{u \in U_{AB}} p_u \cdot \hat{y}(r_{u,A}, r_{u,B})}{\sum_{u \in U_{AB}} p_u}, \tag{37}$$

where, as before, U_{AB} is the set of common users for items A and B and p_u is the *user-predictivity* defined in Eq. (35). We leverage the training set to find p_u for each user u. This is done by recording the implied prediction $\hat{y}(r_{u,A}, r_{u,B})$ for each common user u for each item pair (A, B) in the training set and then computing each user's average absolute error ϵ_u as per Eq. (36). Finally, we apply supervised learning techniques to optimize the value of the parameter λ that appears in the definition of p_u. This is achieved by minimizing the error in Eq. 37 made against the training set. We find that $\lambda = 0.01$, which yields a strong preference for high *user-predictivity*, works well for the data studied here.

Figure 5 shows the results of our experiment using $UserPredSim$ to estimate $TrueSimRand$. To be clear, these plots are showing the mean squared error (MSE) given by

$$\text{MSE} = \frac{1}{|S|} \sum_{(A,B) \in S} (Estimator(A, B) - GoldSim(A, B))^2, \tag{38}$$

where S is the test set of item pairs and $Estimator(A, B)$ represents an arbitrary estimator of the similarity between items A and B. In the upper panel, we show the MSE for the $N \geq 40$ data. The black dashed line represents the estimates of $PearSim$. For example, with $n = 20$ common users, the MSE of $PearSim$ is about 0.07, which corresponds to an average absolute error of 0.26 in the similarity score—roughly the magnitude of σ_s (see Fig. 1). These data clearly show that the reliability of $PearSim$ in estimating $TrueSimRand$ deteriorates rapidly as n decreases. Below about $n = 20$, the horizontal red line, which represents the MSE obtained from always guessing the average $GoldSimRand$ value from the training set, outperforms $PearSim$. These two estimators serve as simple benchmarks.

[6] See discussion of bias subtraction in **Data** section.

[7] To solve Eq. (34) for $\hat{y}$ given ratings r_A and r_B, we use the `brentq` root finding function of the widely available Python module `scipy.optimize`.

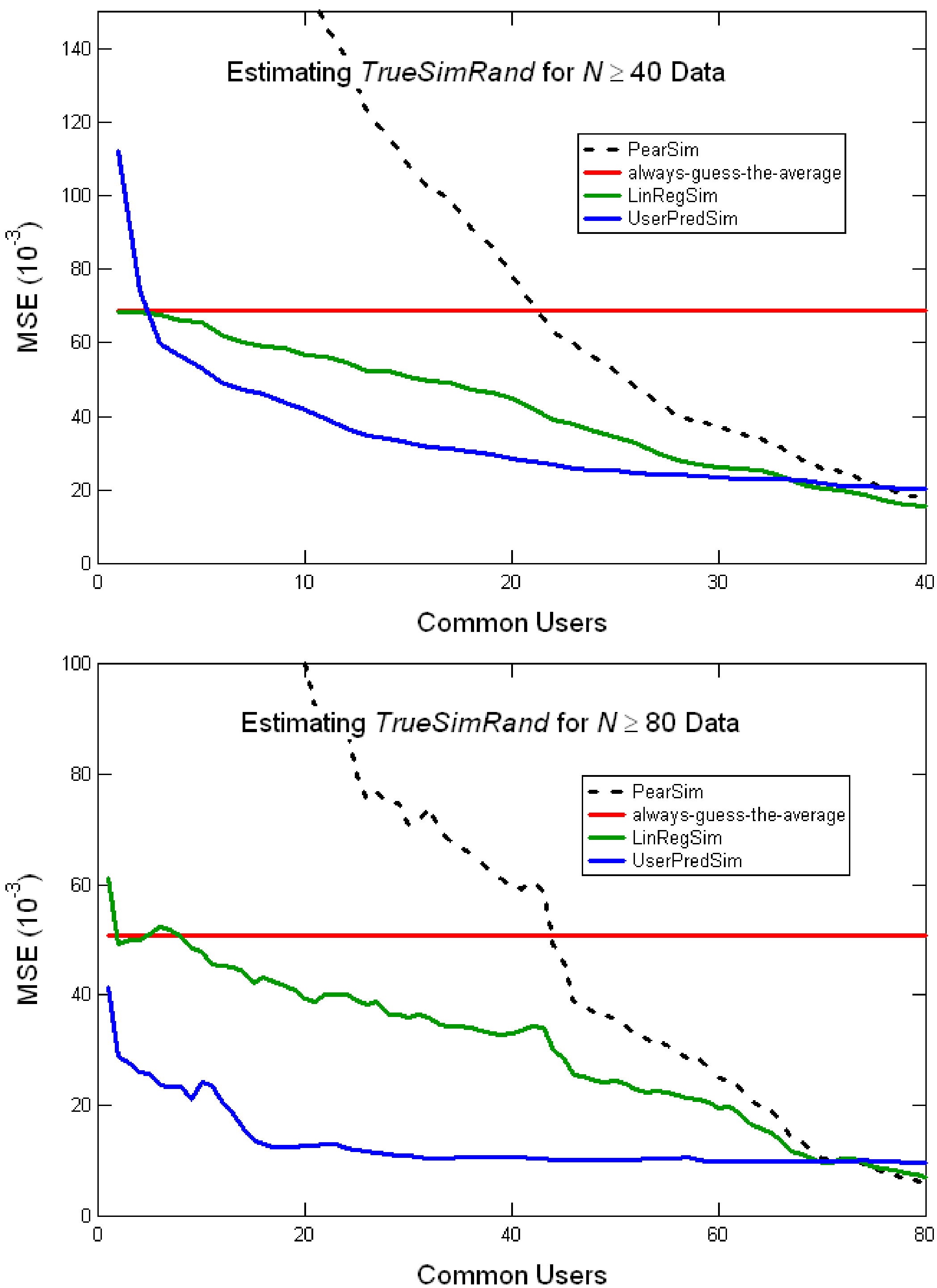

Figure 5: Mean square error *vs* number of common users, n, for estimates of *TrueSimRand* for item pairs ultimately having at least 40 common users (top) and 80 common users (bottom). The dashed curve represents *PearSim* computed using only the ratings from the first n users. The flat red line indicate the MSE that results from always guessing the average *TrueSimRand* from the training set. The green curve is *LinRegSim* and the blue curve is *UserPredSim*.

The green curve in the upper panel of Fig. 5 is the MSE of $LinRegSim$, which was already shown in Fig. 3, and is included here for comparison. $LinRegSim$ outperforms always-guessing-the-average for all $n > 2$ and does better than $PearSim$ over the entire scope of the plot. $UserPredSim$ is shown in blue and for $3 < n < 33$ achieves a lower MSE than $LinRegSim$.

The lower panel of Fig. 5 shows the MSE of estimators in the experiment carried out with item pairs with $N \geq 80$. The curves for $PearSim$, always guessing the average, and $LinRegSim$ exhibit behavior analogous to that observed in the $N \geq 40$ experiment, with addition of increased noise. The performance of $UserPredSim$, meanwhile, is significantly improved, achieving the MSE that $PearSim$ reaches at $n \approx 70$ by $n \approx 20$. This result demonstrates the efficacy, at least in some datasets, of measuring and exploiting *user-predictivity* when estimating item-item similarity.

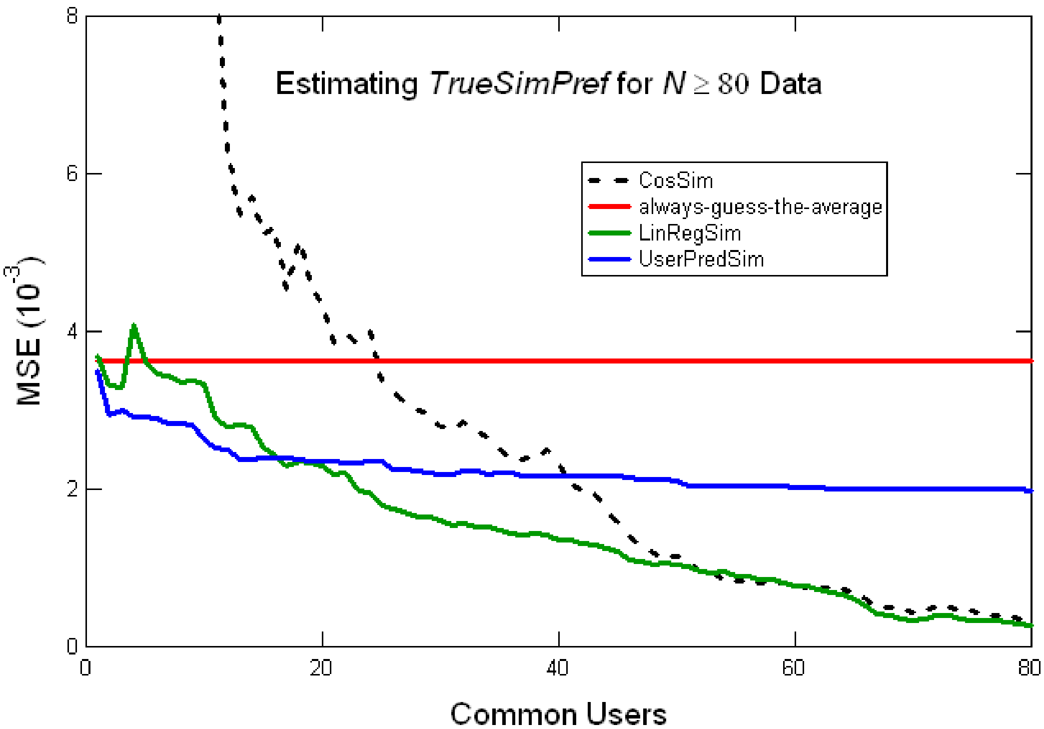

Figure 6: Mean square error *vs* number of common reviewers, n, for estimates of *TrueSimPref* for item pairs ultimately having at least 80 common reviewers. The dashed curve represents *CosSim* computed using only the ratings from the first n reviewers. The flat red line indicate the MSE that results from always guessing the average *TrueSimPref* from the training set. The green curve is *LinRegSim* and the blue curve is *UserPredSim*.

Next, we consider the results of using $UserPredSim$ to estimate $TrueSimPref$. Recall that $UserSimPref$ corresponds to $CosSim$ as time goes to infinity. As such, the relationship in our model between the covariance of the conditional probability distributions that generate ratings given a similarity does not hold for $TrueSimPref$. That is, in our model

$$\mathbf{\Sigma_z} \neq \sigma_r^2 \cdot \begin{bmatrix} 1 & z \\ z & 1 \end{bmatrix}, \tag{39}$$

where $\mathbf{\Sigma_z}$ is the covariance matrix of $P(\vec{R}|Z = z)$ with Z a random variable representing the $TrueSimPref$ of two items. Nonetheless, we can choose parameters and naively use

$UserPredSim$ to estimate $TrueSimPref$. For this experiment, we choose $\mu_s = 0.054$, $\sigma_s = 0.049$, $\mu_r = 0.25$, $\sigma_r = 0.15$, and $\lambda = 0.002$ using the same methods as before.

In Fig. 6 we show the results of our experiments estimating $TrueSimPref$ using the $N \geq 80$ data set. The MSE of $CosSim$, like that of $PearSim$ in the case of $TrueSimRand$, rapidly diverges as the number of common users n goes to zero. Interestingly, $CosSim$ outperforms always-guess-the-average for all $n > 25$, in contrast with $PearSim$, which does the same only after $n > 45$. The speaks to the greater stability of $CosSim$, which takes into consideration all ratings, not only those from common users. Meanwhile, the $LinRegSim$ performs surprisingly similar for estimations of $TrueSimPref$ and $TrueSimRand$. In both cases, the MSE begins at approximately that of always-guess-the-average for $n = 1$ and then decreases approximately linearly with increasing n. Indeed, this was also true for the $N \geq 40$ experiment. We attribute this robustness to the fact that the model from which $LinRegSim$ is derived abstracts away the underlying mechanism from which $TrueSim$ is computed. Finally, $UserPredSim$, perhaps surprisingly, has the lowest MSE for $n < 15$, although not by much. For all $n > 15$, $LinRegSim$ outperforms $UserPredSim$ in estimating $TrueSimPref$. And for all $n > 40$, $CosSim$ achieves lower MSE than that of $UserPredSim$. This lackluster performance by $UserPredSim$ might be expected, however, given the model on which it is based is a dubious fit for $TrueSimPref$.

Conclusion

In this report, we explored the issue of item-item similarity in the context of collaborative filtering. Estimates of item-item similarity such as the Pearson correlation and cosine similarity can have very large variances as data is acquired, yielding large discrepancies between early results and those computed later after sufficient rating data is acquired. We described an offline experiment that simulates the accumulation of rating data over time and the impact that this data has on the performance of various similarity estimators. Two new similarity estimators were described and analyzed.

The first is based on a model of noisy score observations. Using a series of linear regressions, this estimator consistently performs as well or better than both the Pearson correlation or cosine similarity at estimating future values of these same functions. The second new estimator is based on a model of noisy rating pairs generated by an item pair having a true, but unknown, similarity. This method uses a parametrized surface to efficiently estimate item-item similarity given only a single pair of ratings. Combining these implied similarity "predictions" with the notion of *user-predictivity*, we constructed a novel item-item similarity estimator that performs very well, but whose effectiveness appears limited to estimations of future Pearson correlation.

Future Work

The results achieved over this dataset were significant. In order to show that the techniques are general, though, we would like to run the same experiments on other datasets, with different types of items and user behavior. The assumptions made for this model were not specific to the data used, so we hope to achieve similar results. It would also be interesting to see how different the learned parameters would be for other datasets and how well the techniques could perform using the same parameters learned here.

We would also like to incorporate latent factors into our model. Latent factor techniques, such as those presented in [4], have become very popular since the Netflix Challenge and have proved to be better for predicting user-item rating than classical CF. One experiment would be to construct a new $TrueSimLatent$ based on the similarity of two items in latent factor space. Then we could apply both of the approaches in this paper analagously in latent factor space. Another idea is to use latent factors as a new input to a model for predicting $TrueSimRand$ or $TrueSimPref$.

One difficulty with each of these experiments is that it is time consuming to compute the latent factors at different points in simulated time. Since we could not recompute the matrix factorization for every new user rating, we would instead compute the latent factorization of the whole matrix at a few "snap-shots" in simulated time. We then calculate the latent factors of many different pairs at this time, each varying in their degrees of certainty. Hopefully, this should give enough pairs with dfferent degrees of certainty to train and test over.

In a different direction, we believe there are a number of possible applications of the user-predictivity score presented in this paper. Based on the results found here, it is reasonable to believe that adding weights to users by their predictivity would increase the performance of any user-item rating predictor. The predictivity could also have applications in the problem of Robust Collaborative Filtering [5], where rating data is particularly noisy or riddled with spam.

References

[1] Xiaoyuan Su and Taghi M. Khoshgoftaar. "A Survey of Collaborative Filtering Techniques." *Advances in Artificial Intelligence.* vol. 2009, Article ID 421425, 19 pages, 2009. doi:10.1155/2009/421425

[2] John S. Breese, David Heckerman, and Carl Kadie, "Empirical analysis of predictive algorithms for collaborative filtering." *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence.* Morgan Kaufmann Publishers Inc., 1998.

[3] Daniel Lemire and Anna Maclachlan. "Slope One Predictors for Online Rating-Based Collaborative Filtering." *SDM.* Vol. 5, 2005.

[4] Yehuda Koren, Robert Bell, and Chris Volinsky. "Matrix factorization techniques for recommender systems." *Computer* 42.8, 2009. 30-37.

[5] Bhaskar Mehta, Thomas Hofmann, and Wolfgang Nejdl. "Robust collaborative filtering." *Proceedings of the 2007 ACM conference on Recommender systems.* ACM, 2007.