

## Research Note

## The Halo Effect in Multicomponent Ratings and Its Implications for Recommender Systems: The Case of Yahoo! Movies

Nachiketa Sahoo

Tepper School of Business, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213, nsahoo@cmu.edu

Ramayya Krishnan, George Duncan

Heinz College, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213 {rk2x@cmu.edu, gd17@andrew.cmu.edu}

Jamie Callan

Carnegie Mellon University, Pittsburgh, Pennsylvania 15213, callan@cs.cmu.edu

Collaborative filtering algorithms learn from the ratings of a group of users on a set of items to find personalized recommendations for each user. Traditionally they have been designed to work with one-dimensional ratings. With interest growing in recommendations based on multiple aspects of items, we present an algorithm for using multicomponent rating data. The presented mixture model-based algorithm uses the component rating dependency structure discovered by a structure learning algorithm. The structure is supported by the psychometric literature on the halo effect. This algorithm is compared with a set of model-based and instance-based algorithms for single-component ratings and their variations for multicomponent ratings. We evaluate the algorithms using data from Yahoo! Movies. Use of multiple components leads to significant improvements in recommendations. However, we find that the choice of algorithm depends on the sparsity of the training data. It also depends on whether the task of the algorithm is to accurately predict ratings or to retrieve relevant items. In our experiments a model-based multicomponent rating algorithm is able to better retrieve items when training data are sparse. However, if the training data are not sparse, or if we are trying to predict the rating values accurately, then the instance-based multicomponent rating collaborative filtering algorithms perform better. Beyond generating recommendations we show that the proposed model can fill in missing rating components. Theories in psychometric literature and the empirical evidence suggest that rating specific aspects of a subject is difficult. Hence, filling in the missing component values leads to the possibility of a rater support system to facilitate gathering of multicomponent ratings.

*Key words:* collaborative filtering; multicomponent rating; halo effect; Bayesian network; mixture model; expectation maximization; recommender system

*History:* Paul Goes, Senior Editor; Balaji Padmanabhan, Associate Editor. This paper was received on July 16, 2008, and was with the authors 10.75 months for 2 revisions. Published online in *Articles in Advance* April 8, 2011.

## 1. Introduction

Recommender systems are increasingly used in online communities, e.g., shopping sites, subscription service sites, and online meeting places (see Table 1). The recommendations are generated from the collection of user preferences, yet they are personalized to each user. Recommender systems are especially useful when the user has too many choices to explore—they assist the users in discovering items that will appeal to them.

From the retailer's perspective, recommender systems may be used to target advertise items to its customers. A merchant at an online marketplace can use a recommender system to induce demand for

less-known items in the system. By using its proprietary recommender system, Netflix is able to effectively merchandise its collection of more than 100,000 movies. It is able to create demand for older, and often less-known, movies by advertising them to users who might like those movies. Given the constraint on the number of movies a subscriber can rent at a time, increase in demand for older movies reduces the demand for the newer releases, which are more expensive to stock. As reported in the annual Securities and Exchange Commission filing of the company in 2006, the success of the Netflix business model depends, to a certain extent, on effective use of

**Table 1** Examples of Collaborative Filtering-Based Recommender Systems

Item type recommended	Commercial	Noncommercial
Music	iTunes, Last.fm, Yahoo! Music	iRATERadio.com, mystrands.com
Movies	Netflix.com, blockbuster.com	movielens.umn.edu, filmaffinity.com
Websites Aggregated	Amazon.com, half.ebay.com	StumbleUpon.com

and user satisfaction in relation to their recommender system (Netflix 2006).

Online storefronts are not the only places where recommender systems can be used. There are communities of users with common interests who use recommender systems to find new items that they might enjoy. Some examples of such communities are Last.fm and Pandora.com (Internet radio stations with music recommender systems) and StumbleUpon.com (a Web page recommender system). The development of such services suggests that recommender systems are important tools for mining collective user preferences to help users better navigate large choice spaces.

A key input to recommender systems is the ratings that users give to the items in the system. The ratings provide information about the quality of the item as well as about the taste of the user who gave the rating. Most recommender systems have been designed for single-valued ratings, i.e., for each pair (user, item) we have one rating indicating how much the user liked the item. However, sometimes there are multiple components to a rating. For instance, the popular Zagat survey (Zagat.com) rates restaurants on four criteria: food, decor, services, and cost. Similarly, a movie could be rated for its plot, acting, visual effects, and direction. When such ratings are available from users, it is plausible that a recommender system could be designed that makes use of these component ratings and produces better recommendations for the users.

### 1.1. Contributions of This Paper

In this paper we build on a successful collaborative filtering algorithm for single component ratings to design an algorithm that makes use of multicomponent ratings. We do this by discovering a dependency structure among multicomponent rating data using the Chow-Liu structure discovery algorithm. The discovered structure is validated by the literature on the halo effect. We embed this dependency structure in a flexible mixture model (FMM) (Si and Jin 2003). FMM has been shown to work better than the traditional mixture models for collaborative filtering. We

evaluate a set of model-based and instance-based one-component rating collaborative filtering algorithms and their extensions for the multicomponent rating data set. The algorithms were tested using multicomponent rating data collected from Yahoo! Movies. The test results show a significant improvement from the use of multiple component ratings. We identify which multicomponent rating algorithm performs better in which scenario and provide some insight into the behaviors of model-based and instance-based algorithms for collaborative filtering. We also show that the proposed model can be used to fill in the missing rating components for incomplete records. This allows us to generate better recommendations for more users when there are incomplete ratings in the data. This also raises the possibility of a rater support system for helping users rate specific aspects of an item.

It is important to distinguish the current work from collaborative filtering in the presence of multidimensional *context information*, such as finding a recommendation for a movie to watch on a *Sunday* in the *evening* with *children*. Studies exist in the literature to incorporate multidimensional context information into the recommendation generation process. Adomavicius et al. (2005) suggest a reduction-based approach where a context-specific recommendation is generated by using only the ratings collected in the context of interest. In the current work we address a different research question: *How can we use the information in various components of a rating to make better recommendations for the users?*

### 1.2. Background

Given a user's ratings on a subset of items and his peers' ratings on possibly different subsets of items, collaborative filtering algorithms predict which of the items the user would like among the items that he or she has not yet rated. Collaborative filtering algorithms recommend to each user items that are popular among the group of users who are similar to him. This can be thought of as automating the spread of information through word of mouth (Shardanand and Maes 1995). Because collaborative filtering algorithms do not use the content information of the items, they are not limited to recommending only the items with content that the user has rated before (see Adomavicius and Tuzhilin 2005 for a review of the literature in collaborative filtering).

The first group of collaborative filtering algorithms were primarily instance-based on the work of Resnick et al. (1994). In the training step of the algorithm a database of user ratings is built and is used to find similar users and/or items for generating recommendations. These algorithms became popular because they are simple, intuitive, and are sufficient

for many small data sets. However, they do not scale to large data sets without further approximations. Also, because they do not learn any user model from the available preferences, they are of limited use as data mining tools (Hofmann 2004).

A second group of collaborative filtering algorithms, known as model-based algorithms, surfaced later (Breese et al. 1998, Chien and George 1999, Getoor and Sahami 1999). They compile the available user preferences into compact statistical models from which the recommendations are generated. Notable model-based collaborative filtering approaches include singular value decomposition to identify latent structure in ratings (Billsus and Pazzani 1998); probabilistic clustering and Bayesian networks (Breese et al. 1998, Chien and George 1999); repeated clustering (Ungar and Foster 1998); dependency networks (Heckerman et al. 2001); latent class models (Hofmann and Puzicha 1999) and latent semantic models (Hofmann 2004) to cluster the ratings; and flexible mixture models to separately cluster users and items (Si and Jin 2003). Unlike the instance-based approach, the model-based algorithms are slow to train; once trained, however, they can generate recommendations quickly.

The model-based algorithms are often described with the help of probabilistic graphical models. Probabilistic graphic models provide a framework based on probability theory and graph theory to approximate complex distributions (Pearl 2000). They graphically express conditional independencies among variables. The variables are represented as nodes, and dependence among them is expressed as edges in a graph. The assumption they encode about the distribution is that each node is independent of the nondescendent nodes conditional on its parent nodes. This allows one to use the chain rule of probability to factor the joint distribution over all the variables into the product of small conditional distributions. These smaller distributions can be individually estimated. This simplifies the operation on the joint distribution during training and inference (Koller and Friedman 2009).

Product recommendation systems have been explored in marketing science as well. Often the goal is to predict the purchase outcome when a customer is targeted. Recently Moon and Russell developed an autologistic recommendation model based on tools from the spatial statistics literature (Moon and Russell 2008). This model uses the consumers' purchase history to estimate the probability of a future purchase.

Most of the algorithms in the literature are designed to use unidimensional ratings. In a recent work, Adomavicius and Kwon present approaches for multicriteria rating collaborative filtering (Adomavicius and Kwon 2007). Their work is instance-based in identifying similar users but model-based in aggregating

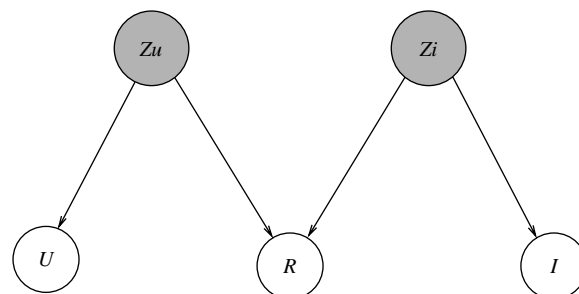
component ratings into one overall rating. Lee and Teng use skyline queries to generate multicriteria-based recommendations from individual component rating predictions, where each component rating is predicted using traditional collaborative filtering algorithms (Lee and Teng 2007).

The multicomponent rating collaborative filtering has some apparent similarity with the conjoint analysis in marketing science (Green and Srinivasan 1978). In conjoint analysis the objective is to estimate a consumer's preference function in terms of the weight the consumer assigns to the attributes of a product. However, collaborative filtering is effective for goods for which attributes can not be readily determined. For instance, directorial style of a movie is hard to express using an attribute-value system. It is worth noting that a high rating for directorial style of a particular movie does not mean the user puts more weight on the directorial quality. Rather, it means that the user perceives a better match of the directorial style, which is not part of the data, with her preference. In this regard, content-based filtering strategies have more similarity with the conjoint analysis than do the collaborative filtering strategies.

The current work is built on the FMM for collaborative filtering (Si and Jin 2003). FMM models the user and item distribution separately by using two latent variables (Figure 1). It has been shown to work better than other latent class models for collaborative filtering. We extend it for multicomponent ratings, taking into account specific properties of these types of data.

Multicomponent rating data exhibit high correlation among the rating components. This is known as the halo effect (Thorndike 1920, Wells 1907). The halo occurs in part because of the failure of the raters to evaluate each component independent of the others (Cooper 1981, Shweder 1975). In some cases this could be a result of the design of the questionnaire that collects multicomponent ratings. If the definitions of the components are ambiguous or not sufficiently different from each other, the collected ratings are likely to be correlated (Kevin and Reynolds 1988). Although

Figure 1 Flexible Mixture Model of Si and Jin (2003)



Note. Latent variable nodes are shaded; observed variable nodes are not shaded. Source: Si and Jin (2003).

there is some debate about whether halo is entirely bad (Cooper 1981, Fiscaro 1988), it is generally considered undesirable because correlated components provide less information than independent ones. At the rating time the halo can be reduced by increasing the familiarity between the rater and the subject (Heneman 1974, Koltuv 1962, Landy and Farr 1980), reducing the time between the observation and the rating (Borgatta et al. 1958, Shweder and D'Andrade 1980), clever questionnaire designs that make the rater aware of the differences among the components (Rizzo and Frank 1977), and sensitizing the raters by training them to observe and avoid the halo effect (Borman 1979, Latham et al. 1980, Ivancevich 1979).

Some halo effect is almost always present despite precautions. Holzbach suggests using a global component in the rating to collect each rater's overall impression of the subject and statistically remove its effect from each component rating (Holzbach 1978). Similar approaches are taken by Landy et al. (Landy et al. 1980) and by Myers (1965) to arrive at more accurate component ratings. Some elements of our approach described in §2.4 are similar to the approach proposed by Holzbach. To the best of our knowledge, the halo effect has not been studied in the context of recommender systems despite recent advances in multicomponent rating collaborative filtering algorithms.

## 2. Multicomponent Rating Recommender System

By rating multiple aspects of an item, users provide more information about their preferences. The variation in different users' component ratings while they seemingly agree on their overall impression of the item can be informative. For instance, consider two users  $u_1$  and  $u_2$  who have given the same overall ratings to the movie  $m_1$  (Table 2). But, they differ in how they rate the components of the movie. User  $u_1$  likes the plot of the movie, whereas user  $u_2$  likes the direction of the movie. Without the component ratings we would have concluded that users would not particularly like any movie similar to  $m_1$ . But the component ratings tell us more. They suggest that user  $u_1$  might like other movies that have a story similar to  $m_1$ , and user  $u_2$  might like a movie that has been directed by the same director or a director with a similar style. Therefore, if we can effectively use the information

**Table 2** An Example of Multicomponent Rating

User	Movie	Story	Acting	Visuals	Direction	Overall
$u_1$	$m_1$	4	1	2	1	2
$u_2$	$m_1$	2	1	1	4	2

Note. Ratings are on a scale of 0–4.

in the component ratings, we should be able to find more relevant items for the users.

Our empirical work has been motivated by the availability of extensive multicomponent rating data from the Yahoo! Movies website. Although the general approach taken in this work is applicable for any data with component ratings, for clarity we shall describe the methods of this work with the help of the Yahoo! data set.

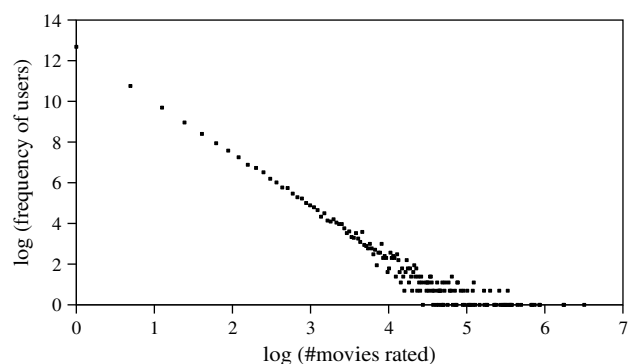
### 2.1. Data Description and Preliminary Analysis

The rating data were collected from the Yahoo! Movies website using a custom program written in Java programming language. Each record of the rating data has seven variables: item or movie ID ( $I$ ), user ID ( $U$ ), ratings on story ( $S$ ), acting ( $A$ ), visuals ( $V$ ), direction ( $D$ ), and overall ( $O$ ) quality of the movie. The ratings are on a 13-point scale ( $A+$ ,  $A$ ,  $A-$ ,  $B+$ ,  $B$ ,  $B-$ ,  $C+$ ,  $C$ ,  $C-$ ,  $D+$ ,  $D$ ,  $D-$ ,  $F$ ). We recoded them to a scale of 0–4:

$$\begin{aligned} (\{A+, A, A-\} \rightarrow 4, \{B+, B, B-\} \rightarrow 3, \\ \{C+, C, C-\} \rightarrow 2, \{D+, D, D-\} \rightarrow 1, \{F\} \rightarrow 0), \end{aligned}$$

so that there are enough data points in each rating bucket. This is especially important for the conditional probability tables estimated in this paper. The models are estimating the probability of observing certain rating values when a user and an item probabilistically belong to some latent classes. If there are not enough data points for a rating value, the probability estimates will be unreliable. Although there were 691,496 (user, item) pairs in the original data set, the user frequency in the data turns out to be skewed (Figure 2). Ratings from users who have rated very few movies are not useful for collaborative filtering, because we cannot reliably know the preferences of a user from only a few of his ratings. Also, we need enough ratings per individual to both train and test the model. Therefore, we have retained only those

**Figure 2**  $\log_e - \log_e$  Plot of Frequency of Users Who Have Rated a Certain Number of Movies



**Table 3 Correlation Among Components of Rating**

	<i>S</i>	<i>A</i>	<i>D</i>	<i>V</i>	<i>O</i>
<i>S</i>	1.00	0.79	0.82	0.74	0.87
<i>A</i>	0.79	1.00	0.81	0.73	0.83
<i>D</i>	0.82	0.81	1.00	0.79	0.88
<i>V</i>	0.74	0.73	0.79	1.00	0.80
<i>O</i>	0.87	0.83	0.88	0.80	1.00

**Table 4 Principal Components**

Component	One	Two	Three	Four	Five
% variance explained	84.5	5.7	4.4	3.2	2.2

**Table 5 Factor Loadings After Quartimax Rotation**

	Factor 1	Factor 2	Uniquenesses
<i>S</i>	0.91	0.23	0.11
<i>A</i>	0.87	−0.02	0.21
<i>V</i>	0.93	−0.08	0.10
<i>D</i>	0.84	−0.12	0.25
<i>O</i>	0.95	0.03	0.07

records that contain users who have at least 20 ratings. After this filtering there were 45,892 records, 1,058 unique users, and 3,430 unique movies.

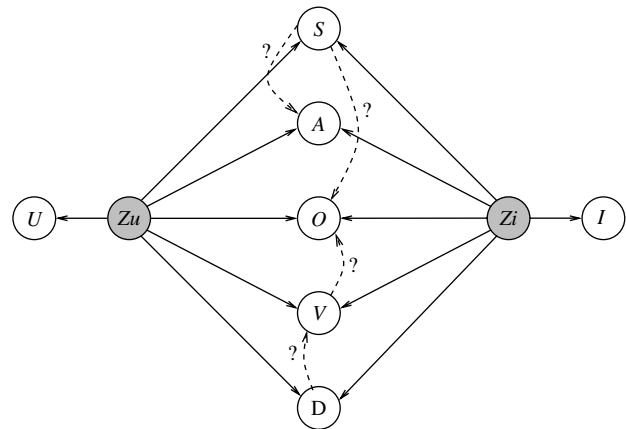
Examining the data set for the halo effect, we find that the components are highly correlated (Table 3). One way to detect halo effect is by principal component analysis (PCA) and factor analysis (Morrison 1967). If most of the variance in the components can be explained by one principal component or one factor, then it suggests the presence of the halo effect (Kafry et al. 1979). PCA of the ratings shows that there is one component that explains 84.5% variance (Table 4). Factor analysis of the components produced a factor structure dominated by one factor (Table 5). These indicate that there is halo error in the collected ratings.

## 2.2. Modeling Component Ratings for Collaborative Filtering

The information contained in the multicomponent rating has two parts: the overall component captures the user’s overall impression about the item and the variation among the components after partialling out the effect of the overall component tells us how the user evaluates aspects of the item. Traditionally, only the overall component has been used to carry out collaborative filtering (e.g., in Si and Jin 2003). In this section we show how to use the additional information in components *along with* the overall component.

We use a mixture model similar to the FMM (Si and Jin 2003) (Figure 1). FMM proposes that the rating an item receives from a user is governed by a small number of latent classes for the users and a small number of latent classes for the items. Given

**Figure 3 Flexible Mixture Model for Component Rating Collaborative Filtering**



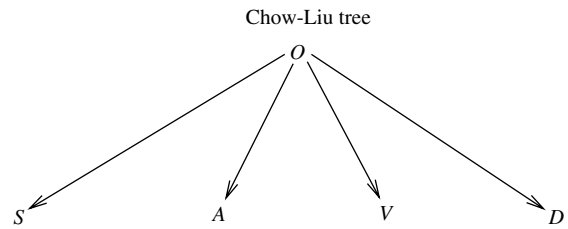
the latent classes, the rating is independent of the particular user and the particular item. We can start by embedding all five rating components in the graphic model in place of the only overall component used in the FMM. However, rating *components* are highly correlated, as shown in Table 3. An incorrect independence among the component ratings in the model would lead us to believe—incorrectly—that each rating component provides completely new additional information about the user preferences.

Therefore, one would do well to find the dependency structure among the five components of the rating (Figure 3). This can be posed as a search problem, where the goal is to find the dependency graph that maximizes the probability of the data. Just as during maximum likelihood estimation, the parameter that maximizes the probability of data is deemed to be closest to the true parameter, in the structure search exercise, the structure that maximizes the probability is deemed the best approximation of the true dependency (Koller and Friedman 2009). To estimate the number of different subgraphs among the five components, note that there are 10 unique pairs, each of which can have an edge between them in either direction or have no edge at all. Discarding structures containing cycles, we have 29,281 candidate structures to search through. This is time consuming. Another problem with a complete model search is that the configurations with multiple parents will lead to large conditional probability tables for which we do not have enough data to estimate the probabilities reliably.

We strike a balance between having completely independent rating variables with no edge between them and accommodating fully connected rating variables that account for all possible dependencies. We restrict ourselves to a category of structures that can capture much of the dependency among the rating variables while being amenable to fast and reliable parameter estimation. Chow and Liu (1968) have

Figure 4 Discovered Structure in the Subratings

	Mutual information					
	S	A	V	D	O	
S	—	0.61	0.72	0.51	0.88	MST ⇒
A	0.61	—	0.68	0.47	0.73	
V	0.72	0.68	—	0.64	0.92	
D	0.51	0.47	0.64	—	0.66	
O	0.88	0.73	0.92	0.66	—	



shown that if we have only discrete variables and we restrict ourselves to only those structures in which there is at most one parent node for each node, i.e., trees, we can efficiently search through them to find the tree that maximizes the probability of data. When the probability distribution is factored according to a tree dependency structure, the graph that maximizes the log probability of the data is the one that maximizes the sum of pairwise mutual information<sup>1</sup> over each edge in the graph. Hence the tree can be found by a maximum weight spanning tree algorithm (Chow and Liu 1968).

Such an exercise over the five component ratings leads to the structure shown in Figure 4. The structure states that the strongest of the dependencies among the components of the ratings is between the Overall rating and the components, as can be verified from the pairwise mutual information table in Figure 4. This shows the strong influence of a user’s Overall impression of a movie on the perception of the other aspects of the movie. In the data we would find evidence of dependence between any pair of variables, but changing the parent for any of the components from O to any other variable (under the tree structure restriction one variable can have at most one parent) would lead to a lower probability of the data. Another way of reading this discovered structure is: given the Overall rating, the components are independent. Note that this dependency structure says that if we do not condition on the Overall rating, then the S, A, D, and V variables are dependent or correlated, which is consistent with the expectation we started with.

2.3. Parallels

It is interesting to compare the approach taken in the psychometric literature (Holzbach 1978, Myers 1965, Landy et al. 1980) with the discovered Chow-Liu tree dependency structure among the movie rating components.

Following a procedure like Holzbach’s, when we statistically remove the halo effect of the Overall

component on the other components through partial correlation (Table 6), the average intercomponent correlation among variables S, A, D, and V reduces from 0.78 to 0.26. As all correlations are positive, some reduction in correlation is expected when computing partial correlations. However, the average partial correlation among the variables is the least when we control for the variable O among the possible five variables. The average partial correlations when we control for S, A, D, and V are 0.47, 0.53, 0.35, and 0.60, respectively. These observations are in accordance with Holzbach’s proposition that by controlling for Overall rating we can peer beyond the halo effect at the more accurate component ratings.

The dependency structure given in Figure 4 says that if we condition on the Overall rating (O), then the components should be independent of each other. Strictly speaking, this assertion of the Chow-Liu tree is correct only if the assumption that the dependency among the rating components can be described by a tree structure is correct. However, a weaker assertion that states that among all possible variables that we might have conditioned on, conditioning on O leads to least residual dependency among the remaining components, is still true. We found that the discovered structure persists over different random subsets of the data, suggesting that it is robust.

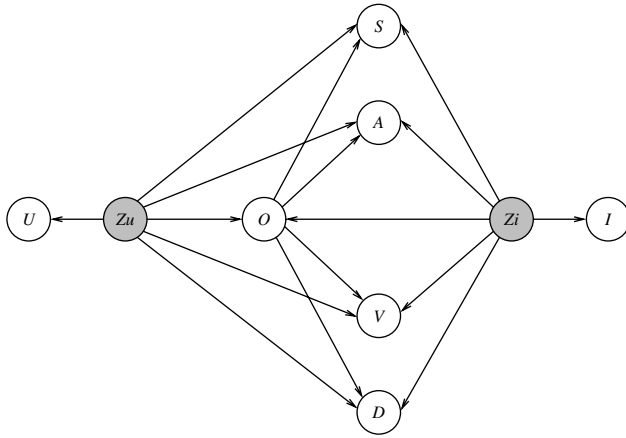
This result empirically validates the approach taken by Holzbach (1978), Myers (1965), and Landy et al. (1980) using a much larger data set. It is interesting to note that the Chow-Liu tree structure discovery method, which is agnostic to the meaning of the rating components, arrives at a conclusion based on the empirical distribution of the data that agrees with the what researchers in psychometric literature arrived at based on the general impression theory of the halo

Table 6 Partial Correlations Controlling for Overall Rating

$r_{R_i R_j, O}$	S	A	D	V
S	1.00	0.25	0.26	0.15
A	0.25	1.00	0.32	0.22
D	0.26	0.32	1.00	0.33
V	0.15	0.22	0.33	1.00

<sup>1</sup> Mutual information is a metric to measure the strength of the dependence between two variables (MacKay 2003).

**Figure 5 Flexible Mixture Model with Component Rating Dependency Structure**



effect. We believe this agreement adds to the validity of both approaches.

## 2.4. Model Estimation Using the Expectation-Maximization (EM) Algorithm

Using the discovered structure between the components of the ratings, we construct the model shown in Figure 5 for collaborative filtering. As we have hidden variables, the parameters need to be estimated using an indirect method. We propose an algorithm based on the EM framework (Dempster et al. 1977). The algorithms based on the EM framework have two alternating steps that monotonically increase probability of the data or the likelihood of the parameters. First is the *E (expectation) step*, where one computes the distribution of the unobserved variable given all the observed variables. This is the same as doing an inference on the graphical model for the hidden variables. It can be shown that among all distributions over the hidden variables, the posterior distribution given the observed data maximizes the expected log probability. Intuitively, the posterior distribution over the hidden variables given the observation is our best guess about the values of the hidden variables.

Second is the *M (maximization) step*, where one estimates the parameters of the complete distribution (consists of observed and unobserved variables) using the standard maximum likelihood estimation. This operation maximizes the expected log probability given the posterior distribution of the unobserved variables.

The *E* and the *M* steps in our case are:

*E step*

$$P(Z_u, Z_i | \vec{X}),$$

$$= \frac{P(Z_u)P(Z_i)P(I|Z_i)P(U|Z_u)\prod_{j=1}^5 P(R_j|Z_u, Z_i, Pa_{R_j})}{\sum_{Z_u} \sum_{Z_i} P(Z_u)P(Z_i)P(I|Z_i)P(U|Z_u)\prod_{j=1}^5 P(R_j|Z_u, Z_i, Pa_{R_j})}, \quad (1)$$

*M step*

$$P(Z_u) = \frac{1}{L} \sum_l \sum_{Z_i} P(Z_u, Z_i | \vec{X}_{(l)}), \quad (2)$$

$$P(Z_i) = \frac{1}{L} \sum_l \sum_{Z_u} P(Z_u, Z_i | \vec{X}_{(l)}), \quad (3)$$

$$P(U | Z_u) = \frac{\sum_{l: U_{(l)}=U} \sum_{Z_i} P(Z_u, Z_i | \vec{X}_{(l)})}{\sum_l \sum_{Z_i} P(Z_u, Z_i | \vec{X}_{(l)})}, \quad (4)$$

$$P(I | Z_i) = \frac{\sum_{l: I_{(l)}=I} \sum_{Z_u} P(Z_u, Z_i | \vec{X}_{(l)})}{\sum_l \sum_{Z_u} P(Z_u, Z_i | \vec{X}_{(l)})}, \quad (5)$$

$$P(R_j | Z_u, Z_i, Pa_{R_j})$$

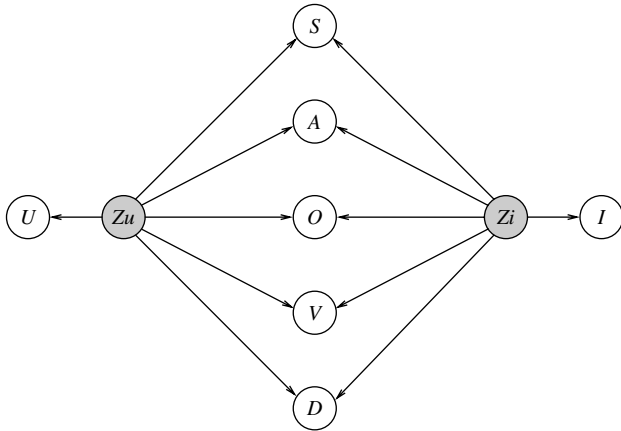
$$= \frac{\sum_{l: R_{j(l)}=R_j \& Pa_{R_{j(l)}}=Pa_{R_j}} P(Z_u, Z_i | \vec{X}_{(l)})}{\sum_{l: Pa_{R_{j(l)}}=Pa_{R_j}} P(Z_u, Z_i | \vec{X}_{(l)})}, \quad (6)$$

where

- $Z_u$  = Latent class variable to cluster the users,
- $Z_i$  = Latent class variable to cluster the items,
- $R_j$  =  $j$ th rating node;  $R_j \in \{S, A, V, D, O\}$ ,
- $Pa_{R_j}$  = parent rating node of  $R_j$ ,
- $L$  = number of records in the data set,
- $l$  = record index,
- $\vec{X}_{(l)}$  = record numbered  $l$ ; it consists of observations for  $U, I, S, A, V, D$ , and  $O$ ,
- $U_{(l)}$  = variable  $U$  in the record numbered  $l$ ,
- $I_{(l)}$  = variable  $I$  in the record numbered  $l$ ,
- $R_{j(l)}$  = rating variable  $R_j$  in the record numbered  $l$ ,
- $Pa_{R_{j(l)}}$  = rating variable  $Pa_{R_j}$  in the record numbered  $l$ .

The *E* step shown above is the conditional distribution computed by dividing joint distribution of all variables, factored using the conditional independencies, by the joint distribution of only the observed variables, obtained by marginalizing out the hidden variables. The *M* step in the EM algorithm estimates the maximum likelihood estimate of the parameters using both the observed and the unobserved variables. If we could observe all variables, we could find the maximum likelihood estimate of parameters of each conditional probability table by dividing the number of records with matching values for all the variables in the conditional probability table by the total number of records with matching values of the conditioning variables. However, we do not observe the hidden variables. Therefore, we have to use our best guess about their number of occurrences or their expected occurrence counts at a record given the observations of other variables in the same record. This is obtained from the posterior distribution of the hidden variable. Because this conditional distribution is multinomial, the expected number of times a hidden variable takes a certain value in one record is

Figure 6 Flexible Mixture Model with Independent Component Ratings



same as the probability of the hidden variable taking that value given the value of the observed variables. All equations of the *M* step can be obtained by tabulating the values of the observed variables and, for the hidden variables, using the expected number of times the hidden variables take a certain value.

We compare this model with discovered dependency structure with the model that assumes independence among the component ratings conditional on the latent classes (Figure 6). The *E* and the *M* steps for the case when all components are assumed independent are derived in a similar manner.

*E* step

$$P(Z_u, Z_i | \vec{X}),$$

$$= \frac{P(Z_u)P(Z_i)P(I | Z_i)P(U | Z_u)\prod_{j=1}^5 P(R_j | Z_u, Z_i)}{\sum_{Z_u} \sum_{Z_i} P(Z_u)P(Z_i)P(I | Z_i)P(U | Z_u)\prod_{j=1}^5 P(R_j | Z_u, Z_i)} \quad (7)$$

*M* step

$$P(Z_u) = \frac{1}{L} \sum_l \sum_{Z_i} P(Z_u, Z_i | \vec{X}_{(l)}), \quad (8)$$

$$P(Z_i) = \frac{1}{L} \sum_l \sum_{Z_u} P(Z_u, Z_i | \vec{X}_{(l)}), \quad (9)$$

$$P(U | Z_u) = \frac{\sum_{l: U_{(l)}=U} \sum_{Z_i} P(Z_u, Z_i | \vec{X}_{(l)})}{\sum_l \sum_{Z_i} P(Z_u, Z_i | \vec{X}_{(l)})}, \quad (10)$$

$$P(I | Z_i) = \frac{\sum_{l: I_{(l)}=I} \sum_{Z_u} P(Z_u, Z_i | \vec{X}_{(l)})}{\sum_l \sum_{Z_u} P(Z_u, Z_i | \vec{X}_{(l)})}, \quad (11)$$

$$P(R_j | Z_u, Z_i) = \frac{\sum_{l: R_{j(l)}=R_j} P(Z_u, Z_i | \vec{X}_{(l)})}{\sum_l P(Z_u, Z_i | \vec{X}_{(l)})}. \quad (12)$$

Note that the key difference between these two sets of expressions is the absence of any parent node in the conditioning part of the conditional probability of the component ratings (Expressions (6) and (12)). The

intuitions behind these equation are similar to those described for the previous set.

We also compare these approaches with the baseline case where there is only one rating: the Overall rating on the movie. The *E* and the *M* steps can be borrowed from Si and Jin (2003) or derived following the approach used to arrive at Equations (1)–(12).

*E* step

$$P(Z_u, Z_i | U, I, O)$$

$$= \frac{P(Z_u)P(Z_i)P(I | Z_i)P(U | Z_u)P(O | Z_u, Z_i)}{\sum_{Z_u} \sum_{Z_i} P(Z_u)P(Z_i)P(I | Z_i)P(U | Z_u)P(O | Z_u, Z_i)} \quad (13)$$

*M* step

$$P(Z_u) = \frac{1}{L} \sum_l \sum_{Z_i} P(Z_u, Z_i | U_{(l)}, I_{(l)}, O_{(l)}), \quad (14)$$

$$P(Z_i) = \frac{1}{L} \sum_l \sum_{Z_u} P(Z_u, Z_i | U_{(l)}, I_{(l)}, O_{(l)}), \quad (15)$$

$$P(U | Z_u) = \frac{\sum_{l: U_{(l)}=U} \sum_{Z_i} P(Z_u, Z_i | U_{(l)}, I_{(l)}, O_{(l)})}{L \times P(Z_u)}, \quad (16)$$

$$P(I | Z_i) = \frac{\sum_{l: I_{(l)}=I} \sum_{Z_u} P(Z_u, Z_i | U_{(l)}, I_{(l)}, O_{(l)})}{L \times P(Z_i)}, \quad (17)$$

$$P(O | Z_u, Z_i) = \frac{\sum_{l: O_{(l)}=O} \sum_{Z_u} P(Z_u, Z_i | U_{(l)}, I_{(l)}, O_{(l)})}{\sum_l P(Z_u, Z_i | U_{(l)}, I_{(l)}, O_{(l)})}. \quad (18)$$

In each of these approaches the number of the classes (levels of  $Z_u$  and  $Z_i$ ) is a user-specified parameter. The greater the number of classes, the better the fit of the model will be with to the training data, resulting in larger probability of data, but that will increase the risk of overfitting the model to the training data. Another factor to keep in mind is that increasing the number of classes by  $n$  times leads to  $n^2$  times more multiplications in the *E* step and  $n$  times more additions in the *M* step. In our experiments the time taken to complete with higher levels of classes has been the bottleneck. We have experimented with 4, 8, 16, and 32 classes. The times taken to complete the experiments were approximately 2.5 hours, 17 hours, 92 hours, and 670 hours, respectively, on a 3.0-GHz pentium processor computer. This leads to some interesting findings. The model with only an Overall rating performs best, with 4 hidden classes. The more complex model with 5 independent components performs best at 8 hidden classes and almost as well with 16. The model with 5 dependent components performs best with 16 hidden classes. The reported results contain the best results for each model.

All models performed poorly when we used 32 classes because of too many parameters to estimate. It causes at least two problems:

1. Small number of data points for each parameter leads to unreliable estimations;



2. Overfitting of the model to the training data.

A method that adaptively determines the number of classes given the size of the data set at hand would be most appropriate in a real-life setting. However, we leave that for a future study.

**2.4.1. Smoothing of Parameter Estimates Using BDe Prior (Koller and Friedman 2009).** To guard against over-fitting to the training data we smooth the parameter estimates in the  $M$  step using a Dirichlet prior, which is the multivariate generalization of the beta distribution and conjugate prior for the multinomial distribution. A parameter  $(\theta_1, \dots, \theta_K)$  following Dirichlet distribution with the hyperparameters  $(\alpha_1, \dots, \alpha_K)$  has the probability distribution

$$P(\theta_1, \dots, \theta_K) \sim \prod_k \theta_k^{\alpha_k - 1}.$$

The expected value of  $\theta_k = \alpha_k / (\sum_{k=1}^K \alpha_k)$ . If we update this prior using multinomial data with counts  $M_1, \dots, M_K$ , then we obtain the posterior that is another Dirichlet distribution with hyper parameters  $(\alpha_1 + M_1, \dots, \alpha_K + M_K)$ . Thus the expected values of the components can be obtained by adding the counts to the numerator and denominator of the original formula:

$$E(\theta_k) = \frac{M_k + \alpha_k}{(\sum_k M_k) + (\sum_k \alpha_k)}. \quad (19)$$

Therefore, the  $\alpha$ s can be thought of as pseudocounts and the sum of  $\alpha$ 's is a measure of the weight of the prior. Note that for each combination of values of the parent nodes there is a different set of parameters and priors. If the same Dirichlet prior were used for each conditional distribution, the nodes with more parent configurations would have larger weights of the priors. BDe prior is a Dirichlet prior constructed so that the weight of the prior would be the same for each node, irrespective of how many parent configurations, and thus conditional probability tables, there are for each of them (Nir Friedman 1998). We use a BDe prior to smooth the parameter estimates during the model training phase.

**2.5. Predicting the Overall Rating**

The goal is to predict the rating a user would give to an item he has not yet rated. Hence, the partial observation consists of the *user* and the *item* from which we are trying to predict the Overall rating.

The joint distribution over all variables (observed and latent) is the product of the conditional probability table estimated in §2.4, from which we need to marginalize away those variables that we are not interested in. In this section we focus on our ability to predict the overall rating. Therefore, we need to marginalize all variables except  $U, I,$  and  $O$ . For the

three models discussed in the previous section, the distribution over the variables  $U, I,$  and  $O$  is

$$P(U, I, O) = \sum_{Z_u} P(Z_u)P(U | Z_u) \cdot \sum_{Z_i} P(O | Z_u, Z_i)P(Z_i)P(I | Z_i). \quad (20)$$

Although the expression for all three models is the same, the parameters estimated are different because of different conditional independence assumptions.

For user  $u_a$  and item  $i$  we are interested in the conditional distribution of the overall rating, namely,

$$P(O | u_a, i) = \frac{P(u_a, i, O)}{\sum_O P(u_a, i, O)}. \quad (21)$$

The mode of this distribution of  $O$  is predicted as the output.<sup>2</sup>

**2.6. Instance-Based Approaches**

We compare the performance of these proposed algorithms with several of the existing one-component and multicomponent instance-based methods. As a baseline we use the framework proposed by Breese et al. (1998):

$$\hat{R}_{ij} = \bar{R}_i + \frac{1}{\sum_i^{N_u} \text{abs}(\text{sim}(t, i))} \sum_i^{N_u} \text{sim}(t, i)(R_{ij} - \bar{R}_i), \quad (22)$$

where the expected rating a user  $t$  would give to item  $j$  is computed by ratings of other users weighted by similarity of those users to the target user  $t$ . One of several metrics can be used to compute the similarity between two users who are represented by the vectors of ratings they have given. Some choices are cosine, correlation, inverse Euclidian distance, etc. We use the correlation coefficient to obtain baseline results, because it has often been used in the literature.

Adomavicius and Kwon (2007) have generalized the similarity measure in Equation (22) to the case of ratings with multiple components. Their predicted rating can be summarized as

$$\text{sim}(t, i) = \frac{1}{1 + \text{dist}_u(t, i)}, \quad (23)$$

$$\text{dist}_u(t, i) = \frac{1}{|U_t \cap U_i|} \sum_{l \in U_t \cap U_i} \text{dist}_r(R_{tl}, R_{il}), \quad (24)$$

where  $U_i$  is the set of ratings given by the user  $i$ .

Several distance metrics are explored for  $\text{dist}_r$ , between two multicomponent ratings, such as

- 1. Manhattan distance

$$\sum_k |R_i(k) - R_j(k)|;$$

<sup>2</sup>Use of expectation of  $O$  to predict did not change the conclusions of the experiments.

## 2. Euclidean distance

$$\sqrt{\sum_k (R_i(k) - R_j(k))^2};$$

## 3. Chebyshev distance

$$\max_k |R_i(k) - R_j(k)|.$$

They found that the Chebyshev distance-based approach performs best among the distance-based approaches considered. The distance metrics proposed in Adomavicius and Kwon (2007) for multicomponent rating collaborative filtering do not take into account the correlation between rating components. One multidimensional distance measure that takes into account correlation between dimensions is Mahalanobis distance (Mahalanobis 1936). Mahalanobis distance between two random vectors  $\vec{x}$  and  $\vec{y}$  that are assumed to be drawn from one common distribution is

$$dist_{maha}(\vec{x}, \vec{y}) = \sqrt{(\vec{x} - \vec{y})^T S^{-1} (\vec{x} - \vec{y})}, \quad (25)$$

where  $S$  is the covariance of the distribution. In one set of experiments we use Mahalanobis distance. This is done using Equation (25) in Equations (23) and (24). We compute the covariance matrix from the ratings used in the training data.

Because of the correlation among the component ratings, it is worth considering whether it would suffice to isolate the principal rating component through a principal component analysis and use that in a single component rating collaborative filtering algorithm. We obtain the principal component of the multicomponent rating through a PCA rotation and compute the correlation between pairs of users based on their thus identified principal component ratings on movies. After computing the similarity between the users, we use original ratings in Equation (22) to compute the predicted value of a user's rating on a movie.

The set of collaborative filtering methods compared is summarized in Table 7.

### 3. Results and Discussion

#### 3.1. Experiments with Random Training Samples

To compare the effectiveness of the three models we use a fraction of the user ratings to train our models (training set) and the remaining ratings to test

the prediction (test set). A certain fraction of each user's records was randomly selected to include in the training data to make sure that there were some training data for each user in the test set. For each user-item pair in the test data we predict their overall rating ( $O$ ) using each model. We calculate the mean absolute error (MAE) of the predictions with the help of the known ratings. We also evaluate the algorithms' ability to retrieve the highest rated items. These two results need not be correlated (Herlocker et al. 2004). Appropriateness of each depends on the application environment.

For an application where the recommended items are presented to the user along with a score for each item, it is important to predict the numerical values of the ratings accurately. One example of such an application environment can be found in the movie recommender system of Netflix. The rating prediction accuracy can be measured using the MAE of the predictions:

$$MAE = \frac{1}{L_{test}} \sum_{l=1}^{L_{test}} |o_l - \hat{o}_l|,$$

where  $L_{test}$  = the number of records in the test data;  $o_l$  = the true rating; and  $\hat{o}_l$  = the predicted rating.

However, in many other applications the user is only presented with the top few items that he is likely to rate highly. The numerical values of the items are deemed of no interest. One example of such an application environment can be found at Amazon.com. Here, if the recommender system can identify the top few items for the user with little noise, then it is considered to have fulfilled the requirement. It does not matter if all the predicted ratings are biased up or down, as long as the predicted ratings order the items in the same way the user would, i.e., a relatively higher rating for items that the user would rate A followed by lower ratings to items that the user would rate B, and so on. This correctness of ordering of items for users can be evaluated using precision-recall plots, precision at top five, or mean reciprocal rank (MRR).

To describe them briefly, assume for a moment that the user would be only interested in the items rated  $A$ . Consider the top- $N$  item predictions. *Precision* is the fraction of the  $N$  items that the user would have rated  $A$ . *Recall* is the fraction of items that the user would have rated  $A$  that are in the top  $N$ . With increasing  $N$ , more  $A$ -rated items are fetched, improving recall. But, at the same time, more of those items that are rated less than  $A$  are also retrieved, damaging the *precision* score. A good recommender system should be able to retrieve many of the  $A$ -rated items while maintaining high precision. Hence, a plot of precision at 11 standard recall levels (0%, 10%, 20%, ..., 100%) is commonly used to compare the performance of different systems (Baeza-Yates and Ribeiro-Neto 1999).

**Table 7** Set of Algorithms Compared

	Model-based	Instance-based
Multicomponent	1. 5 Dependent subratings 2. 5 Independent subratings	3. Chebyshev 4. Mahalanobis 5. PCA
One component	6. Flexible mixture model	7. User-user correlation

Often in retrieval and recommendation tasks the quality of only the top few recommendations is of interest, because users typically do not look beyond the first few items. We measure the quality of top items recommended using two metrics:

1. Mean precision of the top five items recommended;
2. MRR of the first relevant item recommended (Voorhees 2000):

$$\text{MRR} = \frac{1}{\#users} \sum_i^{\#users} \frac{1}{\text{rank}_{\text{firstrelevant}}}$$

MRR measures how soon a person gets a relevant recommendation from the algorithm.

**3.1.1. Accuracy in Rating Prediction.** The average MAE scores using 30 different random train/test partitions are shown in Figure 7. As expected, the overall error for each model decreases with increasing amounts of training data. However, we see that in this set of results using Chebyshev and Mahalanobis distance metrics in an instance-based multicomponent framework best predicts the rating values. Using the principal component after a PCA rotation of the five component ratings does not do any better or worse than simply using the overall rating in a correlation-based single-component rating algorithm.

However, one of the advantages of model-based approaches is that after training the models, the rating prediction step is quick. Therefore, these approaches are more suitable for online recommendation generations. If one is interested in using a model-based recommender system, we have the following insights. Naively extending the existing FMM for collaborative

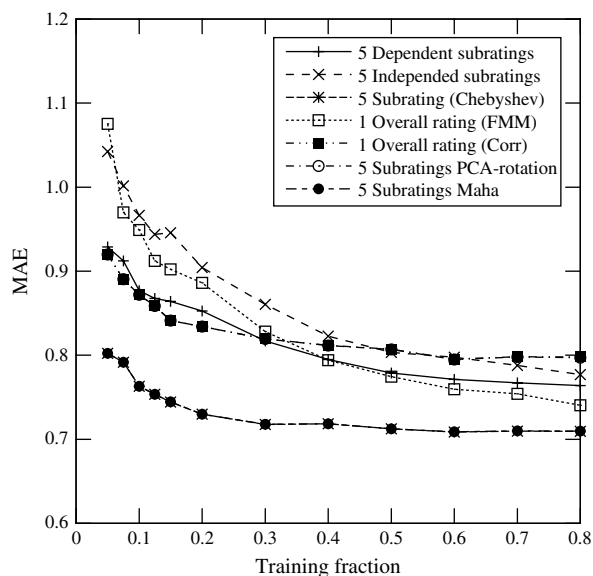
filtering with component ratings without considering the existing correlation among the component ratings does not lead to any improvement in the prediction of overall ratings over using only one component rating. As discussed in §2.2, components of the ratings are correlated and assuming independence among them given latent class leads to overcounting of evidence. When we capture the dependence among the rating components through the Overall rating and explicitly model for it, the prediction accuracy improves. Error plots of the two model-based algorithms, one using only the overall rating and the other using five components with dependency, show that there is an advantage of using multicomponent ratings when the training set is small—up to about 30% of the available data set for training in this case. But when there are enough training data, using only the Overall rating leads to more accurate prediction of Overall rating. This suggests that when we have a user’s Overall rating over a large number of items, adding component ratings does not lead to any further improvement in the ability to predict the Overall rating the user might place on a new item.

To verify that the differences in the average MAE seen in Figure 7 are significant and not a result of chance, we performed a pairwise *t*-test using MAE obtained at the 30 different train/test splits. We found that the differences are indeed significant except where the error lines in Figure 7 cross.

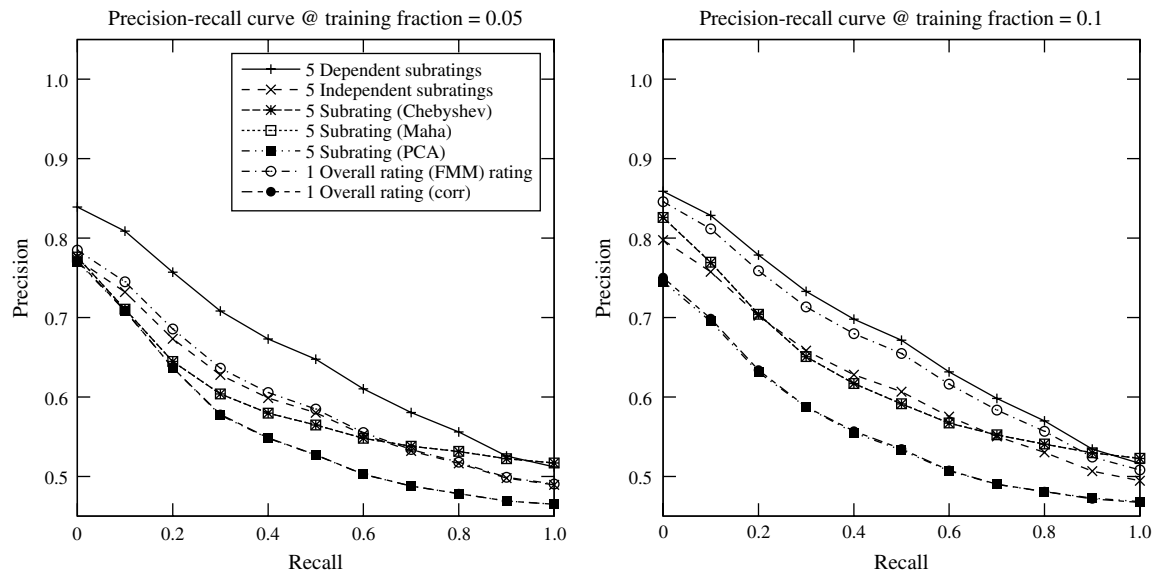
**3.1.2. Accuracy in Retrieving Top *N* Items.** The seven algorithms were trained as described in §2.4 at different training set sizes. Then the ratings for each user-item pair in the test set were computed using the prediction function of each method. This creates an ordering over the test item set for each user. A recommender system would recommend items from this list in decreasing order of predicted rating. The goal of the precision-recall curve is to find out how the precision of the recommendation is affected as more items are included from this list in the pursuit of retrieving all the relevant items. In this set of experiments we treat movies with a rating of 4 in the test set as relevant. The precision versus recall curve is given in Figure 8. A separate experiment that treats movies with a rating of 3 or higher in the test set as relevant returns similar results, albeit with a higher precision value at each recall level because of the presence of more relevant items.

When the training fraction is low the model with discovered structure gives the highest precision at each recall level. However, when we have more training data, the instance-based multicomponent rating algorithms using Chebyshev or Mahalanobis distance do better. The instance-based approaches using only the overall component or the principal

**Figure 7** Plot of Errors by Amount of Training Data Used, for Different Models



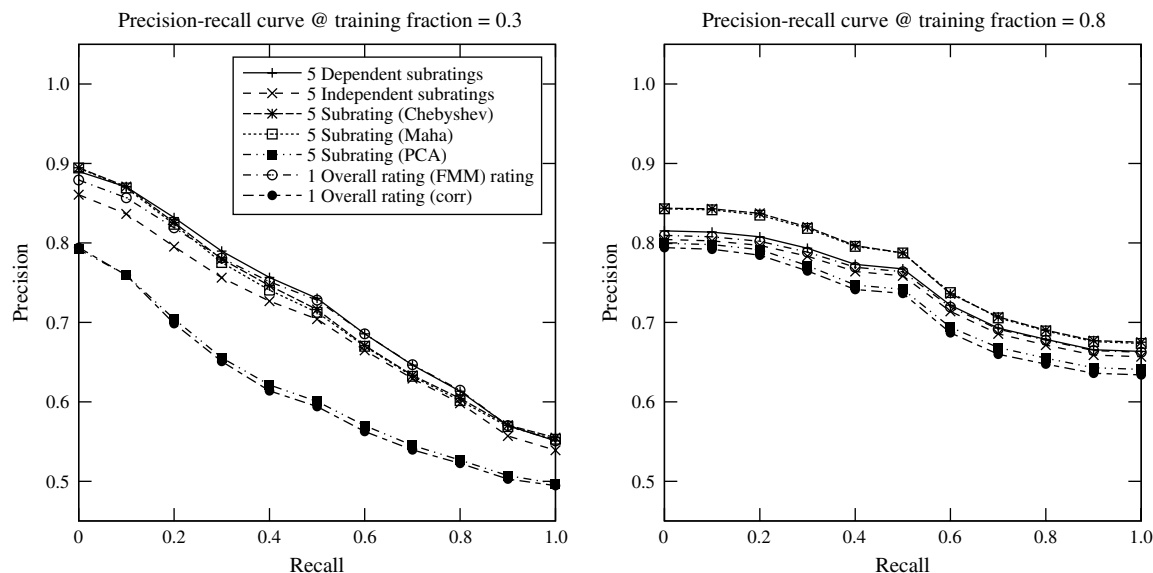
INFORMS holds copyright to this article and distributed this copy as a courtesy to the author(s). Additional information, including rights and permission policies, is available at http://journals.informs.org/.

**Figure 8** Typical Precision-Recall Curves with Sparse Training Data

component have the worst precision. The model with independence assumption among the component ratings returns the lowest precision among the model-based approaches.

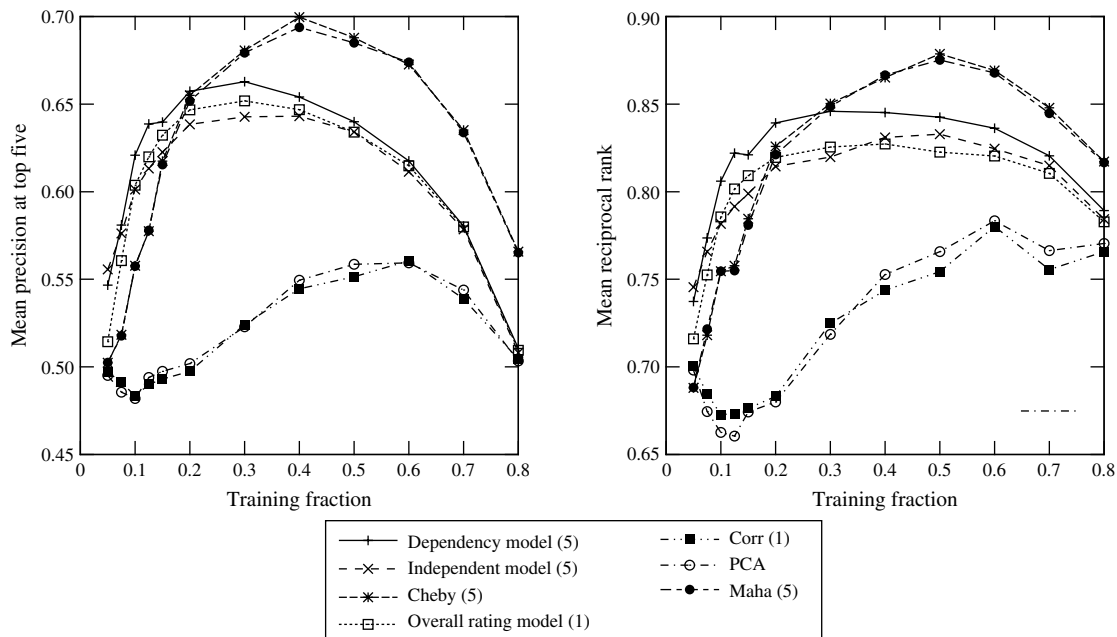
We also compute the mean precision after top-five retrievals and MRRs of each algorithm at these training fractions. The results shown in Figure 10 agree with the general observation made in the precision-recall curves in Figure 8. Among the model-based approaches, the model with dependency structure performs best. However, as we use more training data, instance-based multicomponent rating approaches that use Chebyshev or Mahalanobis distances outperform the model-based approaches (Figure 9).

One possible reason for better performance of model-based approaches over instance-based approaches when the training fractions are low is that the latter are based on pairwise similarity computations. When the training data are sparse, the pairwise comparisons of users are unreliable. However, the model-based approaches suffer less from these problems because each user is effectively compared with a model of a group of users that is less sparse. Thus the users are classified into the correct class more accurately. However, when there are more training data, the pairwise user-to-user comparisons can be done more reliably. This leads to improved performance of the instance-based multicomponent approaches.

**Figure 9** Typical Precision-Recall Curves with Less Sparse Training Data

*Note.* Instance-based algorithms using Chebyshev distance and Mahalanobis distance do best when we use 80% of the data for training.

Figure 10 Mean Precision at Top Five and Mean Reciprocal Rank



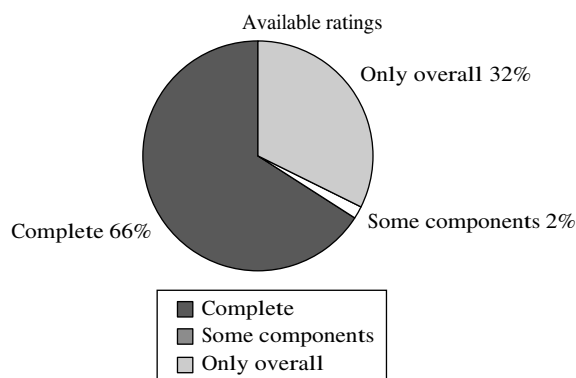
Hence, the takeaway from these tests is that we can improve the rating prediction accuracy by using a multicomponent rating, although the right algorithm to use depends on the sparsity of the training data. It also depends on whether the recommender system is being used to predict the ratings accurately or to retrieve the most relevant items quickly.

### 3.2. Filling in Missing Component Ratings

Raters find it easier to form an overall impression about their subject than to objectively evaluate specific aspects of it (Feeley 2002). This leads to two kinds of problems in collecting multicomponent rating data:

1. **Halo effect.** If they choose to rate the components without deliberating enough to evaluate them objectively, rating values get biased by their overall impression of the subject. This, known as the halo error, is treated in §§1.2, 2.1, and 2.3.

Figure 11 Records with Partial Information



2. **Missing values.** If the raters choose to skip rating the components, we have a missing data problem for rating components. In our data set, 34% of the records (235,659 of 691,495) had incomplete rating information and thus needed to be discarded for the experiments described in the previous sections. Of those 235,695 incomplete records 225,515 (95%) have only the Overall rating (Figure 11). This indicates the difficulty in obtaining component ratings from the users.

There are two opportunities for contribution here:

1. If we can predict the harder aspect ratings for a user for an item taking the user's Overall rating into account, then we can design a rating support system. One use case is: the user gives his Overall rating on the item and the system prefills the component ratings. Then the user confirms them or modifies them if he feels they are different from how he would rate.

2. If we can fill in the missing values in the data set, we can generate recommendations for more users. Because we need a minimum number of ratings per user in the data set, discarding incomplete records eliminates many users, and consequently many of their records even with complete ratings. Table 8 shows the difference between sizes of the data set

Table 8 Increase in Data Set Size After Filling in Missing Components

	With unfilled components	Filled-in components	Percentage of increase
Number of users	1,058	1,680	59
Number of items	3,430	3,926	14
Number of records	45,892	74,110	61

when we discard the incomplete records and when we fill in the missing values using the method described in this section.

We showed in §2 that the probability distribution over all the variables can be factored as

$$P(U, \vec{R}, I) = \sum_{Z_u, Z_i} P(Z_u)P(Z_i)P(I | Z_i)P(U | Z_u) \cdot \prod_{j=1}^5 P(R_j | Z_u, Z_i, Pa_{R_j}). \quad (26)$$

Because we always have Overall ratings in our data, we focus on predicting missing component ratings. To make an inference about one of the component ratings such as  $S$  using the values of  $U$ ,  $I$ , and  $O$  variables, we need to carry out two operations on distribution given in Equation (26):

1. Marginalize away the variables we do not need, i.e.,  $R_j \in A, D, V$
2. Plug in the values of the variable we have. Let's denote them as  $u, i$ , and  $o$ .

The operations result in the following:

$$\begin{aligned} P(U, I, S, O) &= \sum_{Z_u} P(Z_u) \sum_{Z_i} P(Z_i) P(O | Z_u, Z_i) P(I | Z_i) \\ &\quad \cdot P(U | Z_u) P(S | Z_u, Z_i, O), \\ \Rightarrow P(u, i, S, o) &= \sum_{Z_u} P(Z_u) \sum_{Z_i} P(Z_i) P(o | Z_u, Z_i) \\ &\quad \cdot P(i | Z_i) P(u | Z_u) P(S | Z_u, Z_i, o), \\ &\propto P(S | u, i, o). \end{aligned}$$

The result is a function of  $S$  that is proportional to its posterior distribution given the variable values  $u, i$ , and  $o$ . The mode of this distribution is output as the predicted value of  $S$ .

**3.2.1. Experiments and Results.** First we use only the complete records in this experiment to predict the missing components and verify the predictions. The component ratings are hidden in the test data. Only the  $U, I$ , and  $O$  variable values were used from the test data to predict the hidden component ratings. We predicted each of the component ratings ( $S, A, V, D$ ) for every record in the test set and computed the MAE. Tenfold<sup>3</sup> cross validation was used to generate the training and testing samples (Mitchell 1997). We compared our results with the performance of the missing value analysis (MVA) routines of SPSS. The Error values are given in Table 9.

MAEs in predicted missing values are close to 0.36 on a scale of length 4. When we predict the component ratings using only the  $U$  and  $I$  values, as is done

<sup>3</sup> Experiments with two-, three-, and fivefold cross-validations yield similar results.

**Table 9 Comparison of Different Methods Filling in Missing Rating Components**

Method	MAE
Multicomponent FMM	0.353
SPSS EM	0.368
SPSS regression	0.569
CF predicting components	0.701

with traditional collaborative filtering, the MAE values are between 0.6 and 0.7. This suggests that our method is able to extract considerable benefit from the additional available information in the Overall rating. The regression approach to predict missing values, part of the SPSS MVA module, was not very successful at an error of about 0.6. However, the EM algorithm used in the SPSS MVA module produced results almost as good as ours. The algorithm takes an iterative approach that alternates between the following two steps until convergence. Starting with random initialization of the missing values,

1. Regress each variable in turn against all other remaining variables and estimate the coefficients
2. Predict missing variable values of the incomplete records using a linear regression model with the help of the coefficients estimated so far and the other variables in the record.

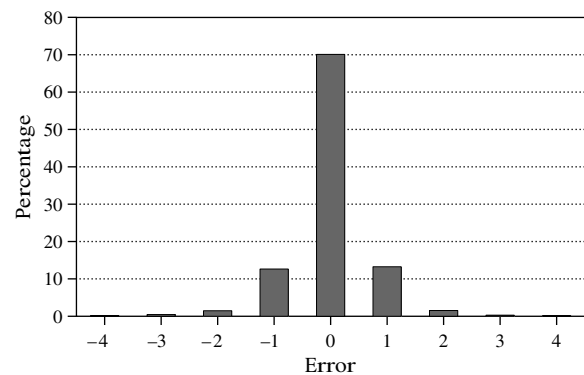
Examining the error distribution of our method, we found that the errors are well behaved, with very few predictions off by a large margin (Figure 12). In about 70% of the cases we were accurate in our prediction of the missing value and in about 96% of the cases the prediction was within one rating of the true value.

## 4. Conclusions

We started this study with the following question:

Can the recommendations by collaborative filtering algorithms be improved by using multiple component ratings?

**Figure 12 Error Distributions While Filling in Missing Rating Components**



To answer this question we collected multicomponent movie rating data from Yahoo! Movies. Because component ratings are correlated due to halo effect, a structure discovery exercise was carried out to find the dependency tree that captures most of the dependencies among the components. The discovered structure is interesting in itself. It says that the component ratings provided by the users are more correlated to the Overall ratings than they are to other component ratings. This suggests the possible relation between the Overall impression of the user and the ratings given to the components. In this context we draw a connection to the work on the halo effect in the psychometric literature. The body of work on halo effect indicates that component ratings are influenced by the presence of other strong factors and by the Overall impression.

We develop a mixture model-based collaborative filtering algorithm incorporating the discovered dependency structure. In addition, several one-component algorithms and their variations for multicomponent ratings were evaluated on the collected data set. The multicomponent rating algorithms lead to better performance than the one-component rating algorithms in both predicting the rating values accurately and in retrieving the most relevant movies quickly. The model-based algorithm using dependency structure leads to better *retrieval* performance when the training data are sparse. However, when more training data are available, using instance-based multicomponent rating approaches that use Chebyshev or Mahalanobis distance to measure distance between two ratings perform better. In addition, these two instance-based multicomponent rating algorithms are able to predict the ratings more accurately than other algorithms that we tested.

One of the advantages of the model-based approaches is that after the model is calibrated, it can be used to quickly generate recommendations. This makes them suitable for scenarios like shopping websites where real-time recommendation generation is important. When the training data are sparse—a common problem faced in real-world scenarios—there is an advantage of using multicomponent ratings in a model that accounts for the halo effect. However, if we have more training data, one component rating flexible mixture model is able to better predict the ratings than other model-based approaches.

The proposed multicomponent model can be used to predict values of the missing component ratings. This is useful because in the current data set approximately one-third of the records have one or more of the component ratings missing. We show that the missing values can be filled in reliably. This allows us to generate recommendations for 59% more users and to recommend 14% more items.

Multicomponent rating collaborative filtering algorithms can suffer because of poor data. One can argue that objectively rating aspects of an item requires deliberation that can only be expected from professional critiques. This can cause the halo effect and reduce the information in the components. In this work we provide an approach to use the limited but important information in the components to make better recommendations. However, with increased emphasis on user-generated content and on valuable services using them, we expect the quality of such data to improve. The proposed algorithm will be even more valuable in such a scenario.

Our work suggests several future research directions. One of foremost importance is the evaluation of the proposed method in other multicomponent rating data sets. Also, in the presence of adequate numbers of ratings, a more complete dependency graph among the ratings might be discovered and used, as it will better capture the dependency among the rating components. We have shown that the proposed model can be used to fill in the missing rating components. Such an approach can be used to design a rater support system that predicts a user's component ratings using his overall rating. But such a system might bias the rater. The implementation and evaluation of such a rater support system is an interesting topic to explore.

## References

- Adomavicius, G., Y. O. Kwon. 2007. New recommendation techniques for multicriteria rating systems. *IEEE Intelligent Systems* 22(3) 48–55.
- Adomavicius, G., A. Tuzhilin. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowledge Data Engrg.* 17(6) 734–749.
- Adomavicius, G., R. Sankaranarayanan, S. Sen, A. Tuzhilin. 2005. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Trans. Information Systems* 23(1) 103–145.
- Baeza-Yates, R. A., B. A. Ribeiro-Neto. 1999. *Modern Information Retrieval*. ACM Press, Addison-Wesley, New York.
- Billsus, D., M. J. Pazzani. 1998. *Learning Collaborative Information Filters*. Morgan Kaufmann Publishers, San Francisco.
- Borgatta, E. F., J. H. Mann, L. S. Cottrell. 1958. The spectrum of individual interaction characteristics: An interdimensional analysis. *Psych. Rep.* 4 279–319.
- Borman, W. C. 1979. Format and training effects on rating accuracy. *J. Appl. Psych.* 64(4) 410–421.
- Breese, J. S., D. Heckerman, C. Kadie. 1998. Empirical analysis of predictive algorithms for collaborative filtering. Technical report, Microsoft Research, Redmond, WA.
- Chien, Y. H., E. I. George. 1999. A Bayesian model for collaborative filtering. *Proc. 7th Internat. Workshop on Artificial Intelligence Statist., FT Lauderdale, FL*. <http://uncertainty99.microsoft.com/proceedings.com>.
- Chow, C. K., C. N. Liu. 1968. Approximating discrete probability distributions with dependence trees. *IEEE Trans. Information Theory* 14(3) 462–467.
- Cooper, W. H. 1981. Ubiquitous halo. *Psych. Bull.* 90(2) 218–244.

- Dempster, A. P., N. M. Laird, D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Statist. Soc.* **39**(1) 1–38.
- Feeley, T. H. 2002. Comment on halo effects in rating and evaluation research. *Human Comm. Res.* **28**(4) 578–586.
- Fiscaro, S. A. 1988. A reexamination of the relation between halo error and accuracy. *J. Appl. Psych.* **73**(2) 239–244.
- Getoor, L., M. Sahami. 1999. Using probabilistic relational models for collaborative filtering. *Workshop on Web Usage Analysis User Profiling (WEBKDD'99), San Diego.*
- Green, P. E., V. Srinivasan. 1978. Conjoint analysis in consumer research: Issues and outlook. *J. Consumer Res.* **5**(2) 103–123.
- Heckerman, D., D. M. Chickering, C. Meek, R. Rounthwaite, C. Kadie. 2001. Dependency networks for inference, collaborative filtering, and data visualization. *J. Machine Learn. Res.* **1** 49–75.
- Heneman, H. G. 1974. Comparison of self and superior ratings of managerial performance. *J. Appl. Psych.* **59**(5) 638–642.
- Herlocker, J. L., J. A. Konstan, L. G. Terveen, J. T. Riedl. 2004. Evaluating collaborative filtering recommender systems. *ACM Trans. Inform. Systems* **22**(1) 5–53.
- Hofmann, T. 2004. Latent semantic models for collaborative filtering. *ACM Trans. Inform. Systems (TOIS)* **22**(1) 89–115.
- Hofmann, T., J. Puzicha. 1999. Latent class models for collaborative filtering. D. Thomas, ed. *Proc. 16th Internat. Joint Conf. Artificial Intelligence (IJCAI-99-Vol2)*, Morgan Kaufmann Publishers, San Francisco, 688–693.
- Holzbach, R. L. 1978. Rater bias in performance ratings: Superior, self, and peer ratings. *J. Appl. Psych.* **63**(5) 579–588.
- Ivancevich, J. M. 1979. Longitudinal study of the effects of rater training on psychometric error in ratings. *J. Appl. Psych.* **64**(5) 502–508.
- Kafry, D., S. Zedeck, R. Jacobs. 1979. Discriminability in multidimensional performance evaluations. *Appl. Psych. Measurement* **3**(2) 187–192.
- Kevin, R. M., D. H. Reynolds. 1988. Does true halo affect observed halo? *J. Appl. Psych.* **73**(2) 235–238.
- Koller, D., N. Friedman. 2009. *Structured Probabilistic Models: Principles and Techniques*. MIT Press, Cambridge, MA.
- Koltuv, B. B. 1962. Some characteristics of intrajudge trait intercorrelations. *Psych. Monograph* **76** 1–24.
- Landy, F. J., J. L. Farr. 1980. Performance rating. *Psych. Bull.* **87**(1) 72–107.
- Landy, F. J., R. J. Vance, J. L. Barnes-Farrell, J. W. Steele. 1980. Statistical control of halo error in performance ratings. *J. Appl. Psych.* **65**(5) 501–506.
- Latham, G. P., E. D. Pursell, K. N. Wexley. 1980. Training managers to minimize rating errors in observation of behavior. *J. Appl. Psych.* **60**(5) 550–555.
- Lee, H. H., W. G. Teng. 2007. Incorporating multi-criteria ratings in recommendation systems. *IEEE Internat. Conf. Inform. Reuse and Integration, Las Vegas, NV*, 273–278.
- MacKay, D. J. C. 2003. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, New York.
- Mahalanobis, P. C. 1936. On the generalised distance in statistics. *Proc. Natl. Inst. Sci.* **2**(1) 49–55.
- Mitchell, T. 1997. *Machine Learning*. McGraw-Hill Book Company, New York.
- Moon, S., G. J. Russell. 2008. Predicting product purchase from inferred customer similarity: An autologistic model approach. *Management Sci.* **54**(1) 71–82.
- Morrison, D. F. 1967. *Multivariate Statistical Methods*. McGraw-Hill Book Company, New York.
- Myers, J. H. 1965. Removing halo from job evaluation factor structure. *J. Appl. Psych.* **49**(3) 217–221.
- Netflix, Inc. 2006. Form 10-K annual report pursuant to Section 13 or 15(d) of the Securities Exchange Act of 1934. United States Securities and Exchange Commission, Washington, DC.
- Friedman, N., M. Goldszmidt. 1998. *Learning in Graphical Models*, Chapter 15. Kluwer Academic Publishers, Dordrecht, The Netherlands, 431–432.
- Pearl, J. 2000. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, New York.
- Resnick, P., N. Iacovou, M. Suchak, P. Bergstrom, J. Riedl. 1994. GroupLens: An open architecture for collaborative filtering of netnews. *Proc. Conf. Comput.-Supported Cooperative Work, CSCW'94, Chapel Hill, NC*, 175–186.
- Rizzo, W. A., F. D. Frank. 1977. Influence of irrelevant cues and alternate forms of graphic rating scales on the halo effect. *Personnel Psych.* **30**(3) 405–417.
- Shardanand, U., P. Maes. 1995. Social information filtering: Algorithms for automating “word of mouth.” In *CHI: Proc. SIGCHI Conf. Human Factors Computing Systems*, ACM Press/Addison-Wesley Publishing, New York, 210–217.
- Shweder, R. A. 1975. How relevant is an individual difference in personality? *J. Personality* **43**(3) 455–484.
- Shweder, R. A., R. G. D’Andrade. 1980. The systematic distortion hypothesis. R. A. Schweder, D. W. Fiske, eds. *New Directions for Methodology of Behavioral Science: Fallible Judgment Behavioral Res.* Jossey Bass, San Francisco, 37–58.
- Si, L., R. Jin. 2003. Flexible mixture model for collaborative filtering. *Proc. Twentieth Internat. Conf. Machine Learning (ICML)*, AAAI Press, San Francisco, 704–711.
- Thorndike, E. L. 1920. A constant error in psychological ratings. *J. Appl. Psych.* **4**(1) 25–29.
- Ungar, L. H., D. P. Foster. 1998. Clustering methods for collaborative filtering. *Proc. Workshop Recommendation Systems*, AAAI, Menlo Park, CA, 112–125.
- Voorhees, E. M. 2000. The TREC-8 question answering track report. *Proc. Eighth Text Retrieval Conf., Gaithersburg, MD*. [http://trec.nist.gov/pubs/trec8/t8\\_proceedings.html](http://trec.nist.gov/pubs/trec8/t8_proceedings.html), 77–82.
- Wells, F. L. 1907. *A Statistical Study of Literary Merit*. R. S. Woodworth, ed. Science Press, New York.