INFORMATION SYSTEMS RESEARCH
Vol. 00, No. 0, Xxxxx 0000, pp. 000–000
ISSN 1047-7047 | EISSN 1526-5536 | 00 | 0000 | 0001

INFORMS
DOI 10.1287/xxxx.0000.0000

The Halo Effect in Multi-component Ratings and its Implications for Recommender Systems: The Case of Yahoo! Movies

Nachiketa Sahoo
Tepper School of Business and iLab, Carnegie Mellon University

Ramayya Krishnan
Heinz College and iLab, Carnegie Mellon University

George Duncan
Heinz College, Carnegie Mellon University

Jamie Callan *
Language Technologies Institute and iLab, Carnegie Mellon University

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 recommending based on multiple aspects of items (Adomavicius and Kwon 2007, Adomavicius and Tuzhilin 2005) we present an algorithm for using multi-component 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 multi-component 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 retrieve relevant items. In our experiments a model based multi-component rating algorithm is able to better retrieve items when training data is sparse. However, if the training data is not sparse, or if we are trying to predict the rating values accurately then the instance based multi-component 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 multi-component ratings.

Key words: Collaborative Filtering, Multi-component Rating, Halo Effect, Bayesian Network, Mixture Model, Expectation Maximization, Recommender System

---

* Author names are in reverse alphabetical order

Item type recommended	Commercial	Non Commercial
Music	iTunes, Last.fm, Yahoo! Music	iRATEradio.com, mystrands.com
Movies	Netflix.com, blockbuster.com	movielens.umn.edu, filmaffinity.com
Books	StoryCode.com	gnooks.com
Dating	reciprodate.com	
Web Sites		GiveALink.org, StumbleUpon.com
Aggregated	Amazon.com, half.ebay.com	

Table 1 Examples of Collaborative Filtering based recommender systems

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 the 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. They are 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 SEC filing of the company in 2006, the success of the Netflix business model depends, to a certain extent, on effective use of 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). These developments suggest 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 given by the users on 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 (user, item) pair 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.

Contributions of this paper

In this paper we build upon a successful collaborative filtering algorithm for single component ratings to design an algorithm that makes use of multi-component ratings. We do this by discovering a dependency structure among multi-component 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 multi-component rating dataset. The algorithms were tested using multi-component rating data collected from Yahoo! Movies. The test results show a significant improvement from the use of multiple component ratings. We identify which multi-component 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 the users in rating specific aspects of an item.

It is important to distinguish the current work from collaborative filtering in the presence of multi-dimensional *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 multi-dimensional context information into the recommendation generation process. Adomavicius et al. (2005) suggest a reduction based approach where 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?*

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/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). Since, 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.

The first group of collaborative filtering algorithms were primarily instance based (Resnick et al. 1994). In the training step they build a database of user ratings that is used to find similar users and/or items while generating recommendations. These algorithms became popular because they are simple, intuitive, and sufficient for many small datasets. However, they do not scale to large datasets 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, but, once trained they can generate recommendations quickly.

The model based algorithms are often described with the help of probabilistic graphical models. Probabilistic graphical 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: each node is independent of the non-descendent 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 and advertised. Recently Moon

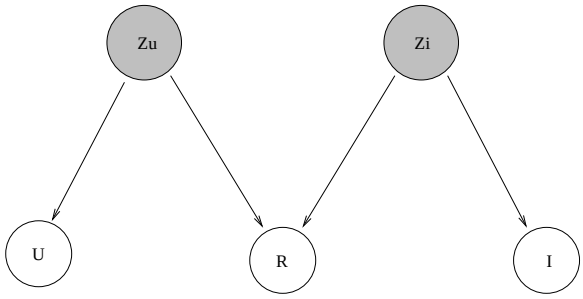

Figure 1 **Flexible Mixture model of Luo Si, Rong Jin, 2003. Latent variable nodes are shaded and observed variable nodes are not shaded.**

and Russell have 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 multi-criteria 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 multi-criteria based recommendations from individual component rating predictions where each component rating is predicted using traditional collaborative filtering algorithms (Lee and Teng 2007).

The multi-component 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 experience 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 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 upon the Flexible Mixture Model (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 multi-component ratings taking into account specific properties of these types of data.

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

Table 2 An example of multi-component rating. Ratings are on a scale of 0 to 4.

Multi-component rating data exhibits high correlation among the rating components. This is known as the halo effect (Thorndike 1920, Wells 1907). Halo occurs in part due to 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 multi-component 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 R. Murphy 1988). Although there is some debate whether halo is entirely bad (Cooper 1981, Fisicaro 1988), it is generally considered undesirable because correlated components provide less information than independent ones. At the rating time 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 (E. F. Borgatta 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 halo (Borman 1979, G. P. Latham 1980, Ivancevich 1979).

Some halo 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. (Steele 1980) and by Myers (Myers 1965) to arrive at more accurate component ratings. There are some similarities of our approach described in Section 2.4 to the approach proposed by Holzbach. To the best of our knowledge, the effect of halo has not been studied in the context of recommender systems despite recent advances in multi-component rating collaborative filtering algorithms.

2. Multi-component 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. The user u_1 likes the plot of the movie, while the user u_2 likes the direction of the movie. Without the component ratings we would have concluded that the users

would not particularly like any movie similar to m_1. But the component ratings tell us more. They suggest that the user u_1 might like other movies that have a story similar to m_1, while 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 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 multi-component rating data from the Yahoo! Movies Web site. 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! dataset. A description of the dataset follows.

2.1. Data description and preliminary analysis

The rating data was collected from the Yahoo! movies Web site using a custom written program written in the 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 thirteen point scale ($A+, A, A-, B+, B, B-, C+, C, C-, D+, D, D-, F$). We recoded them to a scale of 0 to 4 ($\{A+, A, A-\} \longrightarrow 4, \{B+, B, B-\} \longrightarrow 3, \{C+, C, C-\} \longrightarrow 2, \{D+, D, D-\} \longrightarrow 1, \{F\} \longrightarrow 0$), 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 a 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 dataset, 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, since 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 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 dataset for halo effect, we find that the components are highly correlated (Table 3). One way to detect halo 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 halo (D. Kafry 1979). Principal Component Analysis of the ratings show that there is one component that explains 84.5% variance. 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.

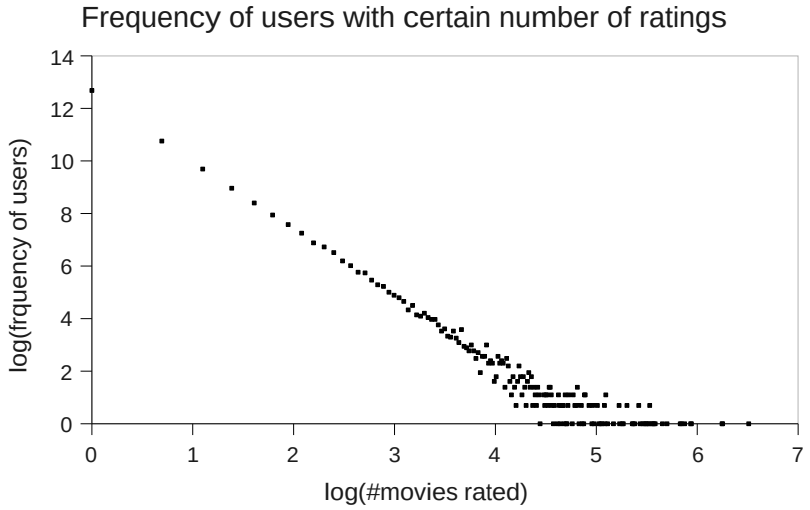

Figure 2 log_e-log_e **plot of frequency of users who have rated a certain number of movies.**

	S	A	D	V	O
S	1.00	0.79	0.82	0.74	0.87
A	0.79	1.00	0.81	0.73	0.83
D	0.82	0.81	1.00	0.79	0.88
V	0.74	0.73	0.79	1.00	0.80
O	0.87	0.83	0.88	0.80	1.00

Table 3 **Correlation among components of rating**

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

Table 4 **Principal components**

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

Table 5 **Factor loadings after quartimax rotation**

2.2. Modeling component ratings for collaborative filtering

The information contained in the multi-component rating has two parts: the overall component captures the overall impression of the user 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.

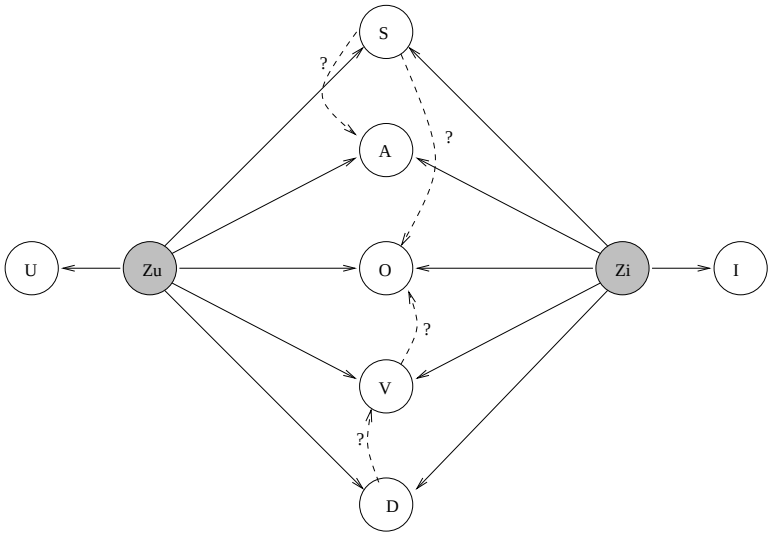

Figure 3 Flexible Mixture Model for component rating collaborative filtering

We use a mixture model similar to the Flexible Mixture Model (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 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 graphical model in place of the only overall component used in FMM. However, rating *components* are highly correlated as shown in Table 3. An incorrect independence among the component ratings in the model would mislead us to believe 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, the structure that maximizes the probability in this exercise 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 ten 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

Mutual Information

	S	A	V	D	O
S	–	0.61	0.72	0.51	0.88
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	–

MST $\Longrightarrow$

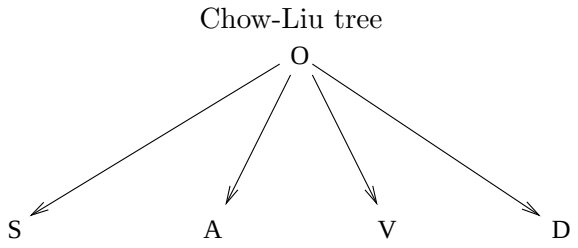

Figure 4 Discovered structure in the sub ratings

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 have 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[1] over each edge in the graph. Hence the tree can be found by a maximum weight spanning tree (MST) 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, 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, Steele 1980) with the discovered Chow-Liu tree dependency structure among the movie rating components.

[1] Mutual information is a metric to measure the strength of the dependence between two variables (MacKay 2003)

$r_{R_iR_j.O}$	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

Table 6 Partial correlations controlling for Overall rating

Following a procedure like Holzbach's when we statistically remove the halo effect of the Overall component on the other components through partial correlation, the average inter-component correlation among variables S, A, D, 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, 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 (Steele 1980) using a much larger dataset. 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 effect. We believe this agreement adds to the validity of both the 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. They are:

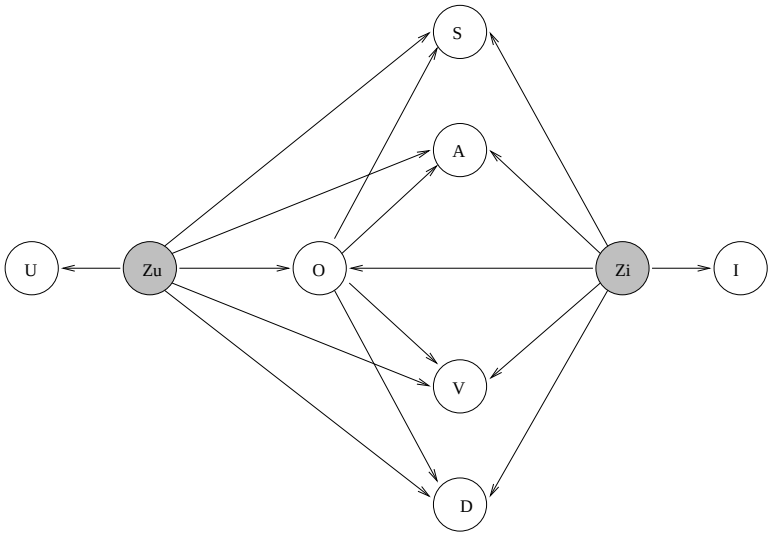

Figure 5 Flexible Mixture Model with component rating dependency structure

E (expectation) step where one computes the distribution of the unobserved variable given all the observed variables. This is 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.

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

$$P(Z_u, Z_i|\vec{X}) = \frac{P(Z_u)P(Z_i)P(I|Z_i)P(U|Z_u)\prod_{j=1}^{5} P\left(R_j|Z_u, Z_i, \mathrm{Pa}_{R_j}\right)}{\sum_{Z_u}\sum_{Z_i} P(Z_u)P(Z_i)P(I|Z_i)P(U|Z_u)\prod_{j=1}^{5} P\left(R_j|Z_u, Z_i, \mathrm{Pa}_{R_j}\right)} \tag{1}$$

M-step

$$P(Z_u) = \frac{1}{L}\sum_{l}\sum_{Z_i} P(Z_u, Z_i|\overrightarrow{X_{(l)}}) \tag{2}$$

$$P(Z_i) = \frac{1}{L}\sum_{l}\sum_{Z_u} P(Z_u, Z_i|\overrightarrow{X_{(l)}}) \tag{3}$$

$$P(U|Z_u) = \frac{\sum_{l:U_{(l)}=U}\sum_{Z_i} P(Z_u, Z_i|\overrightarrow{X_{(l)}})}{\sum_{l}\sum_{Z_i} P(Z_u, Z_i|X_l)} \tag{4}$$

$$P(I|Z_i) = \frac{\sum_{l:I_{(l)}=I} \sum_{Z_u} P\left(Z_u, Z_i|\overrightarrow{X_{(l)}}\right)}{\sum_l \sum_{Z_u} P\left(Z_u, Z_i|\overrightarrow{X_{(l)}}\right)} \tag{5}$$

$$P\left(R_j|Z_u, Z_i, \mathrm{Pa}_{R_j}\right) = \frac{\sum_{l:R_{j(l)}=R_j \,\&\, \mathrm{Pa}_{\mathrm{Rj}(l)}=\mathrm{Pa}_{\mathrm{Rj}}} P\left(Z_u, Z_i|\overrightarrow{X_{(l)}}\right)}{\sum_{l:\mathrm{Pa}_{\mathrm{Rj}(l)}=\mathrm{Pa}_{\mathrm{Rj}}} P\left(Z_u, Z_i|\overrightarrow{X_{(l)}}\right)} \tag{6}$$

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 dataset

l = record index

$\overrightarrow{X_{(l)}}$ = record numbered l. It consists of observations for U, I, S, A, V, D, 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

$\mathrm{Pa}_{R_j(l)}$ = 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 (MLE) of the parameters using both the observed and the unobserved variables. If we could observe all variables, we could find the MLE 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. But 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. Since this conditional distribution is multinomial the expected number of times a hidden variable takes a certain value in one record is 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.

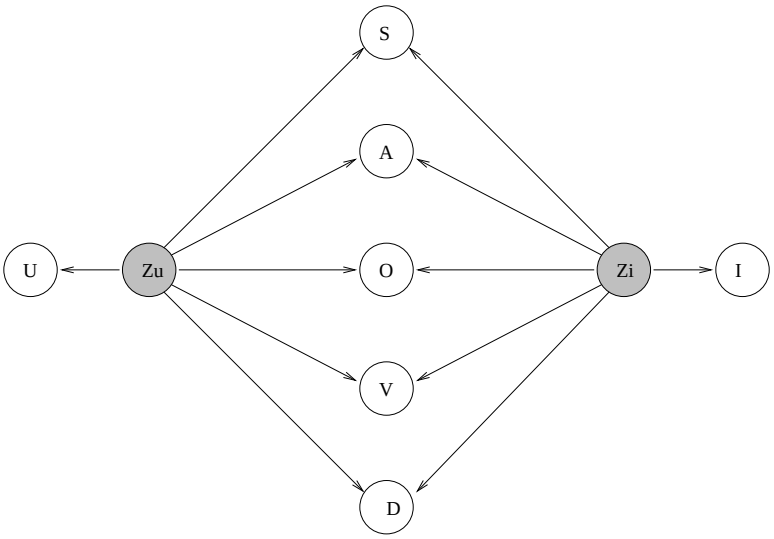

Figure 6 Flexible mixture model with independent component ratings

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

$$P(Z_u, Z_i|\vec{X}) = \frac{P(Z_u)P(Z_i)P(I|Z_i)P(U|Z_u)\prod_{j=1}^{5} P\left(R_j|Z_u, Z_i\right)}{\sum_{Z_u}\sum_{Z_i} P(Z_u)P(Z_i)P(I|Z_i)P(U|Z_u)\prod_{j=1}^{5} P\left(R_j|Z_u, Z_i\right)} \tag{7}$$

M-step

$$P(Z_u) = \frac{1}{L}\sum_{l}\sum_{Z_i} P(Z_u, Z_i|\overrightarrow{X_{(l)}}) \tag{8}$$

$$P(Z_i) = \frac{1}{L}\sum_{l}\sum_{Z_u} P(Z_u, Z_i|\overrightarrow{X_{(l)}}) \tag{9}$$

$$P(U|Z_u) = \frac{\sum_{l:U_{(l)}=U}\sum_{Z_i} P(Z_u, Z_i|\overrightarrow{X_{(l)}})}{\sum_{l}\sum_{Z_i} P(Z_u, Z_i|X_l)} \tag{10}$$

$$P(I|Z_i) = \frac{\sum_{l:I_{(l)}=I}\sum_{Z_u} P\left(Z_u, Z_i|\overrightarrow{X_{(l)}}\right)}{\sum_{l}\sum_{Z_u} P\left(Z_u, Z_i|\overrightarrow{X_{(l)}}\right)} \tag{11}$$

$$P\left(R_j|Z_u, Z_i\right) = \frac{\sum_{l:R_{j(l)}=R_j} P\left(Z_u, Z_i|\overrightarrow{X_{(l)}}\right)}{\sum_{l} P\left(Z_u, Z_i|\overrightarrow{X_{(l)}}\right)} \tag{12}$$

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 Equation 1–12.

E-step

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

M-step

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

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

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

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

$$P\left(O|Z_u, Z_i\right) = \frac{\sum_{l:O_{(l)}=O}\sum_{Z_u} P\left(Z_u, Z_i|U_{(l)}, I_{(l)}, O_{(l)}\right)}{\sum_{l} P\left(Z_u, Z_i|U_{(l)}, I_{(l)}, O_{(l)}\right)} \tag{18}$$

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 will be the fit of the model 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 times taken to complete with higher levels of classes has been the bottleneck. We have experimented with 4, 8, 16, and 32 classes. The time taken to complete the experiments were approximately 2.5 hours, 17 hours, 92 hours, and 670 hours respectively on a 3.0GHz 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 perform 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 dataset at hand would be most appropriate in a real life setting. However, we leave that for a future study.

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 $(\theta_1, \ldots, \theta_K)$ following Dirichlet distribution with the hyper-parameters $(\alpha_1, \ldots, \alpha_K)$ has the probability distribution

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

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

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

Therefore the α's can be thought of as pseudocounts and the sum of α'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 is 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 Section 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:

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

Although the expression for all three models is the same the parameters estimated are different due to different conditional independence assumptions.

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

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

The mode of this distribution of O is predicted as the output[2].

2.6. Instance based approaches

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

$$\widehat{R_{tj}} = \bar{R}_t + \frac{1}{\sum_i^{N_u} abs(sim(t,i))} \sum_i^{N_u} sim(t,i)(R_{ij} - \bar{R}_i) \tag{22}$$

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

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

$$sim(t,i) = \frac{1}{1 + dist_u(t,i)} \tag{23}$$

$$dist_u(t,i) = \frac{1}{|U_t \cap U_i|} \sum_{l \in U_t \cap U_i} dist_r(R_{tl}, R_{il}) \tag{24}$$

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

Several distance metrics are explored for $dist_r$ between two multi-component ratings, such as :

1. Manhattan distance
$$\sum_k |R_i(k) - R_j(k)|$$
2. Euclidean distance
$$\sqrt{\sum_k (R_i(k) - R_j(k))^2}$$

[2] Use of expectation of O to predict did not change the conclusions of the experiments.

	Model based	Instance based
Multi-component	1. 5 dependent subratings 2. 5 independent subratings	3. Chebyshev 4. Mahalanobis 5. PCA
One component	6. Flexible Mixture Model	7. User-User correlation

Table 7 Set of algorithms compared

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 multi-component rating collaborative filtering do not take into account correlation between rating components. One multi-dimensional 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:

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

where S is the covariance of the distribution. In one set of experiments we use Mahalanobis distance to compute recommendations. This is done by using Equation 25 in Equations 23 and 24 to compute recommendations. 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 if 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 multi-component 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 are 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 use the remaining 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 is 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.

In an application where the user is recommended the items along with the predicted ratings for the user it is important to be able to predict the numerical values of these 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 Mean Absolute Error (MAE) of the predictions.

$$\text{MAE} = \frac{1}{L_{\text{test}}} \sum_{l=1}^{L_{\text{test}}} |o_l - \widehat{o_l}|$$

where, L_{test} = the number of records in the test data

o_l = the true rating

$\widehat{o_l}$ = the predicted rating

However, in many other applications the user is only presented 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 can be 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 as the user would, i.e., assign 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 such ordering of items for users can be evaluated using Precision-Recall plots, precision at top-5, or Mean Reciprocal Rank.

To describe them briefly, let's 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 given rating 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

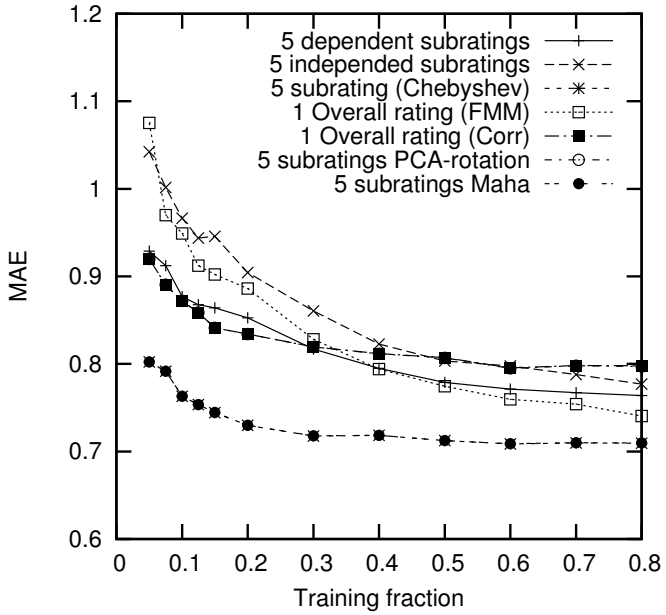

Figure 7 Plot of errors by amount of training data used, for different models.

levels $(0\%, 10\%, 20\%, \ldots, 100\%)$ is commonly used to compare the performance of different systems (Baeza-Yates and Ribeiro-Neto 1999).

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

1. Mean precision of the top-5 items recommended
2. Mean reciprocal rank of the first relevant item recommended (Voorhees 2000)

$$MRR = \frac{1}{\#users} \sum_{i}^{\#users} \frac{1}{rank_{firstrelevant}}$$

Mean reciprocal rank measures how soon a person gets a relevant recommendation from the algorithm.

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 multi-component framework does best in predicting the rating values. Using the principal component after a PCA rotation of the 5 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, they 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 Flexible Mixture Model 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 Section 2.2, components of the ratings are correlated and assuming independence among them given latent class leads to over counting 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 5 components with dependency, show that there is an advantage of using multi-component ratings when the training set is small—up to about 30% of the available dataset for training in this case. But, when there is 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.

Accuracy in retrieving top-N items The seven algorithms were trained as described in Section 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 rating 4 in the test set as relevant. The precision vs recall curve is given in the Figure 8. A separate experiment that treats movies with rating 3 or higher in the test set as relevant returns similar results albeit with a higher precision value at each recall level due to 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 multi-component rating algorithms using Chebyshev or Mahalanobis distance do better. The instance

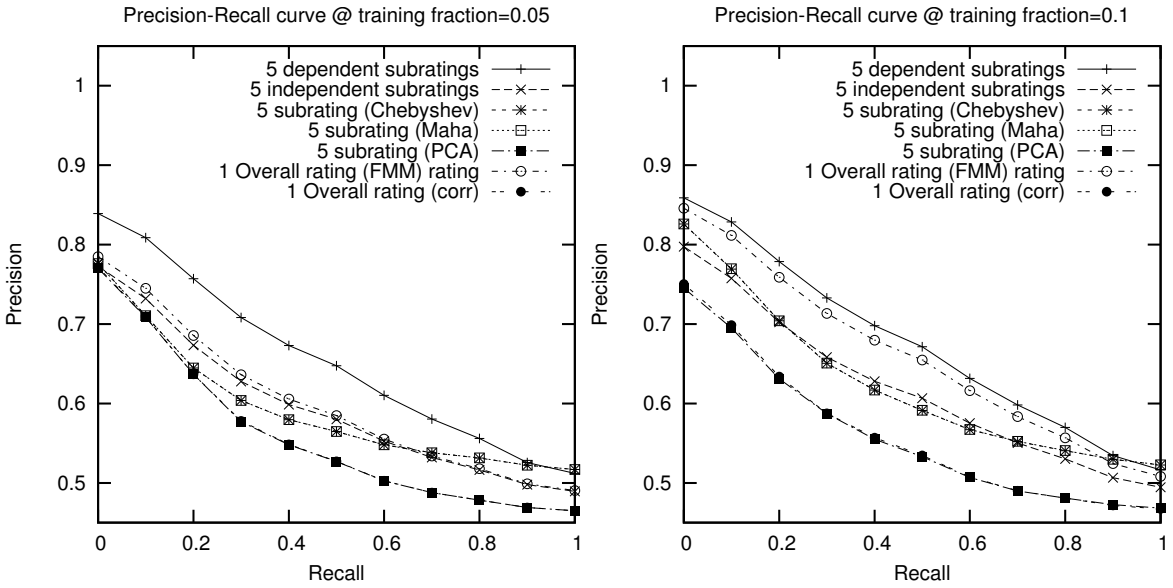

Figure 8 **Typical Precision-Recall curves with sparse training data**

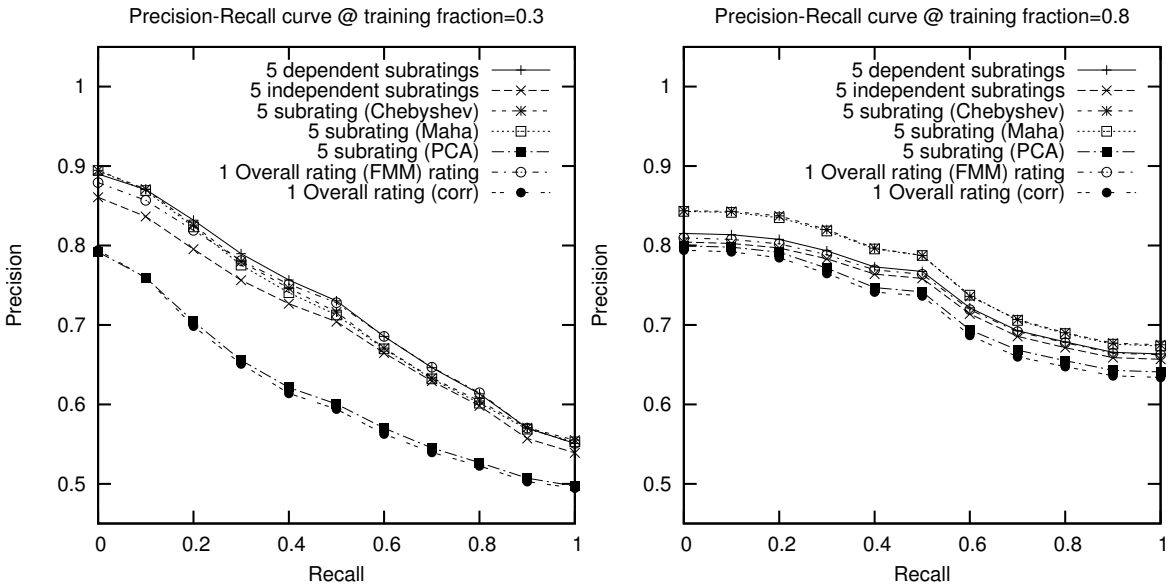

Figure 9 **Typical Precision-Recall curves with less sparse training data. Instance based algorithms using Chebyshev distance and Mahalanobis distance do best when we use 80% of the data for training.**

based approaches using only the overall component or the principal 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-5 retrievals and mean reciprocal ranks 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

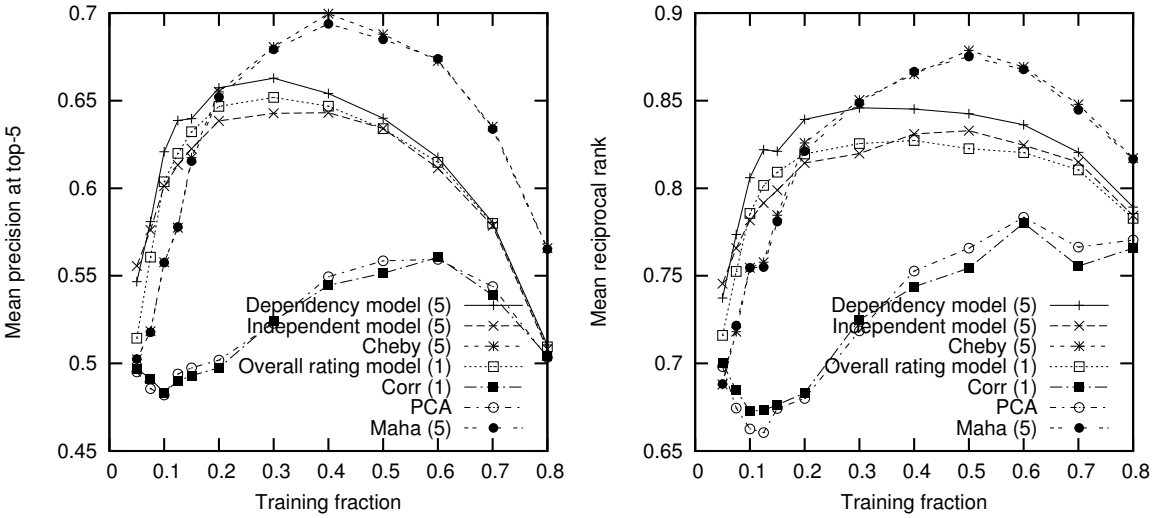

Figure 10 Mean precision at top-5 and Mean reciprocal rank

with dependency structure performs best. However, as we use more training data instance based multi-component rating approaches that use Chebyshev or Mahalanobis distances out-perform the model based approaches.

One possible reason for better performance of model based approaches over the instance based approaches when the training fractions are low is that instance based approaches are based on pairwise similarity computations. When the training data is 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 is more training data the pairwise user-to-user comparisons can be done more reliably. This leads to improved performance of the instance based multi-component approaches.

Hence, the takeaway from these tests is that we can improve the rating prediction accuracy by using multi-component 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 while collecting multi-component rating data:

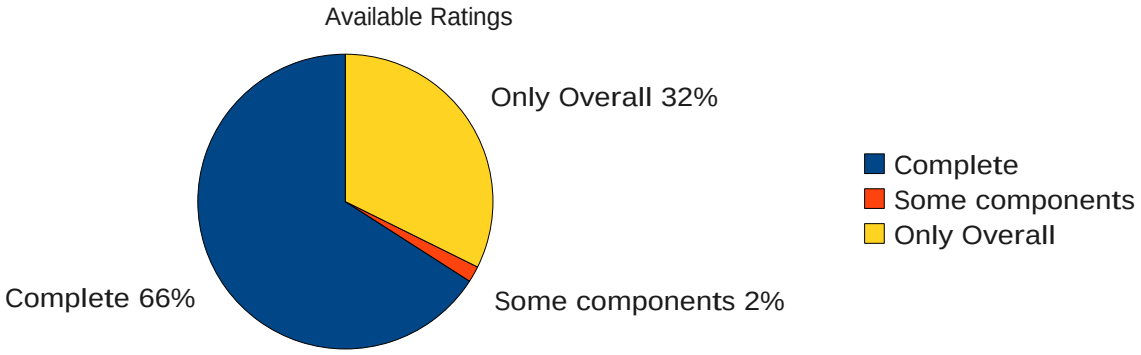

Figure 11 **Records with partial information**

	With unfilled components	Filled-in components	% Increase
Number of User	1058	1680	59%
Number of Item	3430	3926	14%
Number of Records	45892	74110	61%

Table 8 **Increase in dataset size after filling in missing components**

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 Sections 1, 2.1, and 2.3.

2. **Missing Values** If the raters choose to skip rating the components, we have a missing data problem for rating components. In our dataset 34% of the records (235,659 out 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. 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 pre-fills 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 dataset we can generate recommendations for more users. Since we need a minimum number of ratings per user in the dataset, 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 dataset when we discard the incomplete records and when we fill-in the missing values using the method described in this section.

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

Method	MAE
Multi-component FMM	0.353
SPSS EM	0.368
SPSS Regression	0.569
CF predicting Components	0.701

Table 9 Comparison of different methods filling in missing rating components.

Figure 12 Error distributions while filling-in missing rating components.

$$P(U,\vec{R},I) = \sum_{Z_u,Z_i} P(Z_u)P(Z_i)P(I|Z_i)P(U|Z_u)\prod_{j=1}^{5} P\left(R_j|Z_u,Z_i,\mathrm{Pa}_{R_j}\right) \tag{26}$$

Since 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

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

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.

Experiments and Results First we use only the complete records in this experiment to be able 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 Mean Absolute Error. 10-fold [3] 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 done with traditional collaborative filtering the MAE values are between $0.6 - 0.7$. 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 co-efficients,

2. Predict missing variable values of the incomplete records using a linear regression model with the help of the co-efficients 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?

To answer this question we collected multi-component movie rating data from Yahoo! Movies. Since 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

[3] Experiments with 2, 3 and 5 fold cross validations yield similar results

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 multi-component ratings were evaluated on the collected dataset. The multi-component 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 is sparse. However, when more training data is available, using instance based multi-component rating approaches, that use Chebyshev or Mahalanobis distance to measure distance between two ratings, perform better. In addition these two instance based multi-component 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 Web sites where real time recommendation generation is important. When the training data is sparse, a common problem faced by real world scenarios, there is an advantage of using multi-component ratings in a model that accounts for the halo effect among the components. 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 multi-component model can be used to predict values of the missing component ratings. This is useful because in the current dataset 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 recommend 14% more items.

Multi-component 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 halo 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 contents 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 multi-component rating datasets. 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, Gediminas, Ramesh Sankaranarayanan, Shahana Sen, Alexander Tuzhilin. 2005. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Trans. Inf. Syst.* **23**(1) 103–145. doi:http://doi.acm.org/10.1145/1055709.1055714.

Adomavicius, Gediminas, Alexander Tuzhilin. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng* **17**(6) 734–749. URL `http://doi.ieeecomputersociety.org/10.1109/TKDE.2005.99`.

Baeza-Yates, Ricardo A., Berthier A. Ribeiro-Neto. 1999. *Modern Information Retrieval*. ACM Press / Addison-Wesley. URL `citeseer.ist.psu.edu/baeza-yates99modern.html`.

Billsus, D., M.J. Pazzani. 1998. Learning collaborative information filters. San Francisco, CA, USA: Morgan Kaufmann Publishers.

Borman, W. C. 1979. Format and training effects on rating accuracy. *Journal of Applied Psychology* **64** 410–421.

Breese, J.S., D. Heckerman, C. Kadie. 1998. Empirical analysis of predictive algorithms for collaborative filtering. Tech. rep., Microsoft Research.

Chien, Y.H., E.I. George. 1999. A bayesian model for collaborative filtering. *Proceedings of the 7 thInternational Workshop on Artificial Intelligence and Statistics* .

Chow, C. K., C. N. Liu. 1968. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory* **14**(3) 462–467.

Cooper, William H. 1981. Ubiquitous halo. *Psychological Bulletin* **90** 218–244.

D. Kafry, S. Zedeck, R. Jacobs. 1979. Discriminability in multidimensional performance evaluations. *Applied psychological measurement* **3** 187–192.

Dempster, A. P., N. M. Laird, D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society* **39** 1–38.

E. F. Borgatta, J. H. Mann, L. S. Cottrell. 1958. The spectrum of individual interaction characteristics: An interdimensional analysis. *Psychological Reports* **4** 279–319.

Feeley, Thomas Hugh. 2002. Comment on halo effects in rating and evaluation research. *Human Communication Research* **28**(4) 578–586.

Fisicaro, Sebastiano A. 1988. A reexamination of the relation between halo error and accuracy. *Journal of Applied Psychology* **73** 239–244.

G. P. Latham, E. D. Pursell, K. N. Wexley. 1980. Training managers to minimize rating errors in observation of behavior. *Journal of Applied Psychology* **60** 550–555.

Getoor, L., M. Sahami. 1999. Using probabilistic relational models for collaborative filtering. *Workshop on Web Usage Analysis and User Profiling (WEBKDD'99)* .

Green, P.E., V. Srinivasan. 1978. Conjoint analysis in consumer research: issues and outlook. *Journal of consumer research* **5**(2) 103.

Heckerman, D., D.M. Chickering, C. Meek, R. Rounthwaite, C. Kadie. 2001. Dependency networks for inference, collaborative filtering, and data visualization. *The Journal of Machine Learning Research* **1** 49–75.

Heneman, H. G. 1974. Comparision of self and superior ratings of managerial performance. *Journal of Applied Psychology* **59** 638–642.

Herlocker, Jonathan L., Joseph A. Konstan, Loren G. Terveen, John T. Riedl. 2004. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.* **22**(1) 5–53. doi:http://doi.acm.org/10.1145/963770.963772.

Hofmann, T. 2004. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems (TOIS)* **22**(1) 89–115.

Hofmann, Thomas, Jan Puzicha. 1999. Latent class models for collaborative filtering. Dean Thomas, ed., *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99-Vol2)*. Morgan Kaufmann Publishers, S.F., 688–693.

Holzbach, R. L. 1978. Rater bias in performance ratings: Superior, self, and peer ratings. *Journal of Applied Psychology* **63** 579–588.

Ivancevich, J. M. 1979. Longtudinal study of the effects of rater training on psychometric error in ratings. *Journal of Applied Psychology* **64** 502–508.

Kevin R. Murphy, Douglas H. Reynolds. 1988. Does true halo affect observed halo? *Journal of Applied Psychology* **73** 235–238.

Koller, Daphne, Nir Friedman. 2009. *Structured Probabilistic Models: Principles and Techniques*. MIT Press. To appear.

Koltuv, B. B. 1962. Some characteristics of intrajudge trait intercorrelations. *Psychological Monograph* **76**.

Landy, F. J., J. L. Farr. 1980. Performance rating. *Psychological Bulletin* **87** 72–107.

Lee, H.H., W.G. Teng. 2007. Incorporating Multi-Criteria Ratings in Recommendation Systems. *IEEE International Conference on Information Reuse and Integration, 2007.* 273–278.

MacKay, D.J.C. 2003. *Information theory, inference, and learning algorithms*. Cambridge University Press New York.

Mahalanobis, PC. 1936. On the generalised distance in statistics. Proceedings National Institute of Science. India.

Mitchell, Tom. 1997. *Machine Learning*. McGraw-Hill Book Company.

Moon, Sangkil, Gary J. Russell. 2008. Predicting Product Purchase from Inferred Customer Similarity: An Autologistic Model Approach. *MANAGEMENT SCIENCE* **54**(1) 71–82. doi:10.1287/mnsc.1070.0760. URL `http://mansci.journal.informs.org/cgi/content/abstract/54/1/71`.

Morrison, Donald F. 1967. *Multivariate Statistical Methods*. McGraw-Hill Book Company.

Myers, James H. 1965. Removing halo from job evaluation factor structure. *Journal of Applied Psychology* **49** 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, D.C. 20549.

Nir Friedman, Moises Goldszmidt. 1998. *Learning in Graphical Models*, chap. 15. Kluwer Academic Publishers, 431–432.

Pearl, J. 2000. *Causality: Models, Reasoning, and Inference*. Cambridge University Press.

Resnick, P., N. Iacovou, M. Suchak, P. Bergstrom, J. Riedl. 1994. GroupLens: an open architecture for collaborative filtering of netnews. *Proceedings of the Conference on Computer-Supported Cooperative Work, CSCW'94*.

Rizzo, W. A., F. D. Frank. 1977. Influence of irrelevant cues and alternate forms of graphic rating scales on the halo effect. *Personnel Psychology* **30** 405–417.

Shardanand, Upendra, Pattie Maes. 1995. Social information filtering: Algorithms for automating ẅord of mouth.̈. *CHI*. 210–217.

Shweder, R. A. 1975. How relevant is an individual difference in personality. *Journal of Personality* **43** 455–484.

Shweder, R. A., R. G. D'Andrade. 1980. The systematic distortion hypothesis. *New directions for methodology of behavioral science: Fallible judgment in behavioral research* 37–58.

Si, Luo, Rong Jin. 2003. Flexible mixture model for collaborative filtering. *ICML*. AAAI Press, 704–711.

Steele, F.J. Landy; R.J. Vance; J.L. Barnes-Farrell; J.W. 1980. Statistical control of halo error in performance ratings. *Journal of applied psychology* **65** 501–506.

Thorndike, EL. 1920. A constant error in psychological ratings. *Journal of Applied Psychology* **4** 25–29.

Ungar, L.H., D.P. Foster. 1998. Clustering methods for collaborative filtering. *AAAI Workshop on Recommendation Systems* 112–125.

Voorhees, E.M. 2000. The TREC-8 question answering track report. *NIST SPECIAL PUBLICATION SP* 77–82.

Wells, F. L. 1907. A statistical study of literary merit. *Archives of psychology* **1**.