A Comparison of Collaborative-Filtering Recommendation Algorithms for E-commerce

Zan Huang, *Pennsylvania State University*

Daniel Zeng and Hsinchun Chen, *University of Arizona*

This comparison is an initial step toward a metalevel guideline for choosing the best algorithm for a given application with certain data characteristics.

Collaborative filtering is one of the most widely adopted and successful recommendation approaches. Unlike approaches based on intrinsic consumer and product characteristics, CF characterizes consumers and products implicitly by their previous interactions.[1] The simplest example is to recommend the most popular products to all consumers. Researchers are advancing CF technologies in such areas as algorithm design, human-computer interaction design, consumer incentive analysis, and privacy protection.[2,3]

Despite significant progress in CF research, three main problems limit CF's e-commerce applications. First, most research has focused multigraded rating data that explicitly indicate consumers' preferences. However, the available data about consumer-product interactions in e-commerce applications are typically binary transactional data (for example, whether or not someone purchased an item). You can apply CF algorithms for multigraded rating data to binary data, typically with some modest modifications. But these algorithms can't exploit the special characteristics of binary transactional data to achieve more effective recommendation.

The second problem is the lack of understanding of the relative strengths and weaknesses of different types of CF algorithms in e-commerce applications. The need for such comparative studies is evident from the many recent studies that have proposed algorithms but conducted only limited comparisons.

The third problem is data sparsity: a lack of prior transactional and feedback data makes it difficult and unreliable to infer consumer similarity and other patterns for prediction purposes. Research on high-performance algorithms under sparse data is emerging,[4–7] but a deep understanding of them will require substantial additional research.

Our research attempts to address these problems. Our ultimate goal is to develop a metalevel guideline that "recommends" an appropriate recommendation algorithm for a given application demonstrating certain data characteristics. Toward that end, we've evaluated six types of representative CF algorithms with different e-commerce data sets, and we've assessed the algorithms' effectiveness with sparse data.

The six algorithms

We evaluated the popular user-based and item-based correlation/similarity algorithms, three algorithms designed to alleviate the sparsity problem, and a proposed algorithm based on link analysis. Several previous studies have evaluated multiple recommendation algorithms,[2,8] but they focused mainly on variations of the user-based algorithm. Furthermore, these

1541-1672/07/$25.00 © 2007 IEEE
Published by the IEEE Computer Society

studies typically compared the proposed algorithms only with user-based algorithms. So, a comprehensive understanding of existing recommendation algorithms' performance is still far from complete.

We use the following notation to describe CF problems. The problem input is an $M \times N$ *interaction matrix* $A = (a_{ij})$ associated with M consumers $C = \{c_1, c_2, \ldots, c_M\}$ and N products $P = \{p_1, p_2, \ldots, p_N\}$. We focus on recommendations based on transactional data. That is, a_{ij} can take the value of either 0 or 1, with 1 representing an observed transaction between c_i and p_j (for example, c_i has purchased p_j) and 0 representing the absence of a transaction. We consider a CF algorithm's output to be *potential scores* of products for individual consumers that represent possibilities of future transactions. The recommendations consist of a ranked list of K products with the highest potential scores for a target consumer.

The user-based algorithm

This well-studied algorithm predicts a target consumer's future transactions by aggregating the observed transactions of similar consumers. The algorithm first computes a consumer similarity matrix $WC = (wc_{st})$, $s, t = 1, 2, \ldots, M$. It calculates the similarity score wc_{st} on the basis of the row vectors of A using a vector similarity function.[2] A high wc_{st} indicates that consumers s and t might have similar preferences because they previously purchased many of the same products. $WC \cdot A$ gives the products' potential scores for each consumer. In the resulting matrix, the element at the cth row and pth column aggregates the scores of the similarities between consumer c and other consumers who have purchased product p. In other words, the more similar the set of consumers who bought the target product is to the target consumer, the more likely the target consumer will be interested in that product.

The item-based algorithm

This algorithm differs from the user-based algorithm only in that it computes product similarities instead of consumer similarities. It first computes a product similarity matrix $WP = (wp_{st})$, $s, t = 1, 2, \ldots, N$. It calculates wp_{st} on the basis of the column vectors of A. A high wp_{st} indicates that products s and t are similar in that many consumers have purchased both of them. $A \cdot WP$ gives the products' potential scores for each consumer. In the resulting matrix, the element at the cth row and pth column aggregates the scores of the similarities between product p and other products that consumer c has purchased.

The intuition behind this algorithm is similar to that of the user-based algorithm: the more similar the target consumer's purchased products are to the target product, the more likely he or she will be interested in that product. This algorithm can provide higher efficiency and comparable or better recommendation quality than the user-based algorithm for many data sets.[9]

The dimensionality-reduction algorithm

This algorithm condenses the original interaction matrix and generates recommendations based on the condensed, less-sparse matrix to alleviate the sparsity problem.[7] It applies standard *singular-vector decomposition* to decompose the interaction matrix A into $U \cdot Z \cdot V'$, where U and V are two orthogonal matrices of size $M \times R$ and $N \times R$, respectively, and R is the rank of matrix A. Z is a diagonal matrix of size $R \times R$ having all singular values of matrix A as its diagonal entries. The algorithm then reduces Z by retaining only the k largest singular values, to obtain Z_k. It reduces U and V accordingly to obtain U_k and V_k. So, $U_k \cdot Z_k \cdot V_k'$ provides the best lower-rank approximation of the original interaction matrix A that preserves the primary data patterns existing in the data after the "noises" are removed. Consumer similarities can then be derived from the compact representation based on U_k and $Z_k^{1/2}$. The algorithm then generates recommendations in the same fashion as the user-based algorithm.

> The more similar the target consumer's purchased products are to the target product, the more likely he or she will be interested in that product.

The generative-model algorithm

This algorithm uses latent class variables to explain the patterns of interactions between consumers and products.[4,10] Typically, you can use one latent class variable to represent the unknown cause that governs consumer-product interactions. To generate A, the algorithm employs this probabilistic process:

1. Select a consumer with probability $P(c)$.
2. Choose a latent class with probability $P(z|c)$ for the chosen consumer c.
3. Generate an interaction between consumer c and product p (that is, setting a_{cp} to be 1) with probability $P(p|z)$, given that the latent class z was chosen for consumer c.

So, the probability of observing an interaction between c and p is

$$P(c,p) = \sum_z P(c)P(z \mid c)P(p \mid z)$$

With A comprising the observed data, the algorithm estimates the relevant probabilities and conditional probabilities using a maximum-likelihood procedure called *expectation maximization*. On the basis of the estimated probabilities, $P(c, p)$ gives the potential score of product p for consumer c. This algorithm in effect clusters consumers and products around the latent classes and therefore alleviates the data-sparsity problem.

The spreading-activation algorithm

This approach addresses the sparsity problem by exploring transitive associations between consumers and products in a bipartite *consumer-product graph* that corresponds with A.[5] We adopt the spreading-activation algorithms developed in associative information retrieval to efficiently transverse the consumer-product graph to explore the transitive associations. We use the *Hopfield net* algorithm,[5] which provides competitive performance in recommendation applications.

This approach represents consumers and products as nodes in a graph, each with an activation level μ_j, $j = 1, \ldots, N$. To generate recommendations for consumer c, we set the corresponding node at activation level 1 ($\mu_c = 1$). We set the activation levels of all other nodes at 0. After initialization, the algorithm repeatedly performs this activation procedure:

$$\mu_j(t+1) = f_s\left[\sum_{i=0}^{n-1} t_{ij}\mu_i(t)\right]$$

where f_s is the continuous sigmoid transformation function or other normalization

functions; t_{ij} equals η if i and j correspond to an observed transaction and 0 otherwise. The algorithm stops when the activation levels of all nodes converge. The final activation levels μ_j of the product nodes give the potential scores of all products for consumer c.

This algorithm efficiently explores the *connectedness* of a consumer-product pair in the consumer-product graph. Connectedness relates to the number of paths between the pair and their lengths and is the predictor of the occurrence of future interaction.

The link-analysis algorithm

Link-analysis algorithms have found significant application in Web page ranking and social-network analysis (notably, HITS[11] and PageRank[12]). Our algorithm adapts the HITS (Hypertext-Induced Topic Selection) algorithm in the recommendation context.[13]

The original HITS algorithm distinguishes between two types of Web pages that pertain to a certain topic:

- *Authoritative* pages contain definitive high-quality information.
- *Hub* pages are comprehensive lists of links to authoritative pages.

A given Web page i in the Web graph has two distinct measures of merit: its authority score a_i and its hub score h_i. The quantitative definitions of the two scores are recursive. A page's authority score is proportional to the sum of the hub scores of pages linking to it. Conversely, its hub score is proportional to the authority scores of the pages to which it links. These definitions translate to a set of linear equations:

$$a_i = \sum_j G_{ji} h_j$$

and

$$h_i = \sum_j G_{ij} a_j$$

where G is the matrix representing the links in the Web graph. Using the vector notation $a = (a_1, a_2, \ldots, a_n)$ and $h = (h_1, h_2, \ldots, h_n)$, we can express the equations in compact matrix form: $a = G' \cdot h = G' \cdot G \cdot a$ and $h = G \cdot a = G \cdot G' \cdot h$. The equations' solutions correspond to eigenvectors of $G' \cdot G$ (for a) and $G \cdot G'$ (for h). Computationally, it's often more efficient to start with arbitrary values of a and h and repeatedly apply $a = G' \cdot h$ and $h = G \cdot a$ with a certain normalization procedure at each iteration. Subject to some mild assumptions, this iterative procedure will converge to the solutions.[11]

In our recommendation application, the consumer-product relationship forms a bipartite graph consisting of consumer and product nodes. A link between a consumer node c and a product node p indicates that p could represent part of c's interest and that c could partially represent product p's consumer base. Compared to Web page ranking, recommendation requires the identification of products of interest to individual consumers rather than generally popular products. So, we adapt the following original authority and hub score definitions. We define a *product representativeness* score $pr(p, c^0)$ of product p with respect to consumer c^0, which we view as a measure of p's "authority" in terms of its level of interest to c^0. Similarly, we define a *consumer representativeness* score $cr(c, c^0)$ of c with respect to consumer c^0, which measures how well c, as a "hub" for c^0, associates with products of interest to c^0.

Compared to Web page ranking, recommendation requires the identification of products of interest to individual consumers rather than generally popular products.

Instead of a vector representation (as with the Web page authority and hub scores), we use $PR = (pr_{ik})$ to denote an $N \times M$ product representativeness matrix, where $pr_{ik} = pr(i, k)$ represents products p_k's representativeness score for consumer C_i, and $CR = (cr_{it})$ to denote an $M \times M$ consumer representativeness matrix, where $cr_{it} = cr(i, t)$ represents consumer c_t's representativeness score for consumer c_i. Using the recursive definition of authority and hub scores, we define the product and consumer representativeness scores as $PR = A' \cdot CR$ and $CR = A \cdot PR$. Intuitively, the sum of the representativeness scores of products linked to a consumer gives the consumer representativeness score, and vice versa.

These obvious extensions of the score definitions have two inherent problems. First, if a consumer has links to all products, that consumer will have the highest representativeness scores for all target consumers. However, such a consumer's behavior actually provides little information for predicting the target consumer's behavior. A more fundamental problem is that with the convergence property that we mentioned earlier, PR and CR will converge to matrices with identical columns. This amounts to scores representing product ranking independent of particular consumers, thus providing only limited value for recommendation.

To address these problems, we redefine the consumer representativeness score: $CR = B \cdot PR + CR^0$, where $B = (b_{ij})$ is an $M \times N$ matrix derived from A,

$$b_{ij} = \frac{a_{ij}}{\left(\sum_j a_{ij}\right)^{\gamma}}$$

and CR^0 is the source consumer representativeness score matrix,

$$cr^0_{ij} = \begin{cases} \eta & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

(that is, $CR^0 = \eta I_M$, where I_M is an $M \times M$ identity matrix). This new definition borrows ideas similar to those successfully applied in network spreading-activation models. The introduction of B normalizes the representativeness score a consumer receives from linked products by dividing it by the total number of products he or she is linked to. In other words, to be considered equally representative of the target consumer's interest, a consumer who has purchased more products than another consumer must have more overlapping purchases with the target consumer.

The parameter γ controls the extent to which a consumer is penalized for making many purchases. In spreading-activation research, such an adjustment is well studied in modeling the decay of strength that gets spread from active nodes to their neighbors within a network. We've experimented with various values of γ in the range of 0 to 1; our final experiments used 0.9. We include the source matrix CR^0 to maintain the high representativeness scores for the target consumers themselves and to customize the score updates for each consumer. To maintain consistent levels of consumer self-representativeness, in the actual computation we normalize the matrix multiplication result $B \cdot PR$ before adding CR^0. The normalization is such that each col-

umn of $B \cdot PR$ (corresponding to the consumer representativeness score for each target consumer) adds up to 1. Such normalization also helps avoid numerical problems during repeated multiplication of large-scale matrices.

In summary, our link-analysis algorithm follows these steps:

1. Construct the interaction matrix A and the associating matrix B on the basis of the sales transaction data: $A = (a_{ij})$ and $B = (b_{ij})$, where

$$b_{ij} = \frac{a_{ij}}{\left(\sum_j a_{ij}\right)^{\gamma}}$$

2. Set the source consumer representativeness matrix CR^0 to be ηI_M, and let it be the initial consumer representativeness matrix: $CR(0) = CR^0$.
3. At each iteration t, perform the following:

   a. $PR(t) = A' \cdot CR(t-1)$.
   b. $CR(t) = B \cdot PR(t)$.
   c. Normalize $CR(t)$ such that

$$\sum_{i=1}^{M} cr_{ij} = 1$$

   d. $CR(t) = CR(t) + CR^0$.

   Perform steps 3a to 3d until reaching convergence or the specified number of iterations T (setting T to 5 was sufficient in our experiments).

An illustration of the algorithms

A simple hypothetical interaction matrix illustrates the six algorithms. Figure 1's left panel shows the interaction matrix and consumer-product graph of a simple recommendation data set involving three consumers, four products, and seven observed transactions (shown as 1's in the matrix and as links in the graph). The panels on the figure's right show the algorithms' computational steps.

For the user-based and item-based algorithms, figure 1 shows the consumer similarity matrix WC and product similarity matrix WP. The calculation follows normalized procedures[2] that take into account the global frequency of consumers and products participating in transactions. The two resulting potential score matrices show a small difference but have similar patterns in rows and columns.

For the dimensionality-reduction algorithm, the figure shows the singular-vector decomposition results. It then shows the compact consumer representation based on a rank-2 approximation and the consumer similarity matrix computed using the cosine-similarity function based on this compact representation.

For the generative-model algorithm, we use two latent classes. The figure presents the estimates of the latent class probabilities $P(z)$ and the conditional probabilities $P(z|c)$ and $P(z|p)$.

For the spreading-activation algorithm, the figure shows the activation scores of the consumer and product nodes for several iterations, using each consumer as the starting node.

Finally, for the link-analysis algorithm, we show the normalized version of A, B, and CR^0. We then show PR and CR for the first three iterations of computation. In this example, to make the matrix multiplication calculations easy to follow, we don't perform the normalization at each iteration.

> We adopted four metrics regarding the relevance, coverage, and ranking quality of ranked-list recommendation: precision, recall, F-measure, and rank score.

An interesting comparison in this simple example involves the recommendation of product p_1 to consumer c_1. The user-based and item-based algorithms exploit only the direct consumer-product neighborhood information; so, both algorithms result in a potential score of 0 for the pair $<c_1, p_1>$. When exploring the transitive-neighborhood information, we can consider c_3 as a transitive neighbor of c_1 through a common neighbor c_2 and can consider p_2 and p_4 as transitive neighbors of p_1 through a common neighbor p_3. The other four algorithms indeed obtain nonzero potential scores for $<c_1, p_1>$, showing their ability to capture transitive associations to make recommendations.

Evaluation of algorithms

Our testing set consisted of each consumer's most recent interactions, which constituted 20 percent of the entire data set. The training set comprised the remaining 80 percent. To understand the algorithms' performance under sparse data, we also studied recommendation performance on a *reduced training set*. To create this set, we randomly selected 40 percent of the consumer's total interactions from the entire training set (the *unreduced training set*).

As we mentioned earlier, we set all the algorithms to generate a ranked list of K products. For each consumer, we measured the recommendation quality on the basis of the number of *hits* (recommendations that matched the products in the testing set) and their positions in the ranked list.

We adopted four metrics regarding the relevance, coverage, and ranking quality of ranked-list recommendation from the literature.[2] First, *precision* is

$$P_c = \frac{\text{Number of hits}}{K}$$

Recall is

$$R_c = \frac{\text{Number of hits}}{\text{Number of products in the testing set with which consumer } c \text{ interacted}}$$

The *F-measure* is

$$F_c = \frac{2 \times P_c \times R_c}{P_c + R_c}$$

Finally, the *rank score* is

$$RS_c = \sum_j \frac{q_{cj}}{2^{(j-1)/(h-1)}}$$

where j is the index for the ranked list, h is the viewing half-life (the rank of the product on the list such that there's a 50 percent chance the user will purchase that product), and

$$q_{cj} = \begin{cases} 1 \text{ if } j \text{ is in } c\text{'s testing set} \\ 0 \text{ otherwise} \end{cases}$$

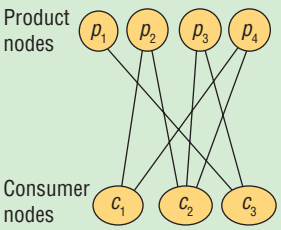
John Breese, David Heckerman, and Carl Kadie proposed this measure,[2] which many follow-up studies adopted to evaluate the recommendation list's ranking quality.[5,8,9]

Precision and recall are essentially competing measures. As the number of recommendations K increases, we expect to see

Input

	p_1	p_2	p_3	p_4
c_1	0	1	0	1
c_2	0	1	1	1
c_3	1	0	1	0

Interaction matrix: A



Consumer-product graph

User-based algorithm

	c_1	c_2	c_3
c_1	1	0.82	0
c_2	0.82	1	0.2
c_3	0	0.2	1

Consumer similarity matrix: WC

	p_1	p_2	p_3	p_4
c_1	0	1.82	0.82	1.82
c_2	0.2	1.82	1.2	1.82
c_3	1	0.2	1.2	0.2

Potential score matrix: $WC \cdot A$

Item-based algorithm

	p_1	p_2	p_3	p_4
p_1	1	0	0.92	0
p_2	0	1	0.15	1
p_3	0.92	0.15	1	0.15
p_4	0	1	0.15	1

Product similarity matrix: WP

	p_1	p_2	p_3	p_4
c_1	0	2	0.29	2
c_2	0.92	2.15	1.29	2.15
c_3	1.92	0.15	1.92	0.15

Potential score matrix: $A \cdot WP$

Dimensionality-reduction algorithm

c_1	0.56	0.45	–0.70
c_2	0.78	0.00	0.63
c_3	0.28	–0.89	–0.35

$\cdot$

2.19	0	0
0	1.41	0
0	0	0.46

$\cdot$

	p_1	p_2	p_3	p_4
	0.13	0.61	0.48	0.61
	–0.6	0.32	–0.6	0.32
	–0.8	–0.2	0.6	–0.2

Singular-vector decomposition: $A = U \cdot Z \cdot V'$

c_1	0.83	0.53	0
c_2	1.15	0.00	0
c_3	0.41	–1.06	0

Compact consumer representation: $U \cdot Z_2^{1/2}$

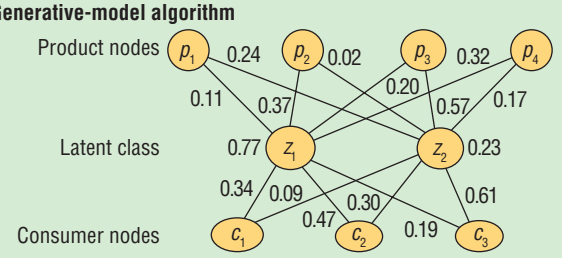
	c_1	c_2	c_3
c_1	1	0.84	–0.2
c_2	0.84	1	0.36
c_3	–0.2	0.36	1

Consumer similarity matrix

	p_1	p_2	p_3	p_4
c_1	–0.2	1.84	0.64	1.84
c_2	0.36	1.84	1.36	1.84
c_3	1	0.16	1.36	0.16

Potential score matrix

Generative-model algorithm



		p_1	p_2	p_3	p_4
$P(z\|p)$	c_1	0.027	0.142	0.057	0.06
$P(z)$	c_2	0.057	0.141	0.115	0.116
$P(z\|c)$	c_3	0.058	0.003	0.115	0.11

Potential score matrix

$\sum_z P(c)P(z|c)P(p|z)$

Spreading-activation algorithm

Target	c_1					c_2				c_3				
Iteration	1	2	3	4	5	1	2	3	4	1	2	3	4	5
c_1	0.0474	0.0207	0.0191	0.019	0.01831		0.0197	0.0191	0.01902				0.0186	0.019
c_2		0.0197	0.0191	0.0194	0.01903	0.0474	0.0217	0.0195	0.01939		0.0188	0.0187	0.0193	0.01937
c_3				0.0183	0.01901		0.0188	0.0187	0.019	0.0474	0.0207	0.0191	0.019	0.01902
p_1					**0.01903**			0.0183	**0.01865**	0.0474	0.0197	0.0187	0.0187	**0.01866**
p_2	0.0474	0.0197	0.0191	0.019	**0.01902**	0.0474	0.0197	0.0191	**0.01903**			0.0183	0.0187	**0.01901**
p_3			0.0183	0.0187	**0.01938**	0.0474	0.0197	0.0191	**0.01903**	0.0474	0.0197	0.0191	0.019	**0.01903**
p_4	0.0474	0.0197	0.0191	0.019	**0.01865**	0.0474	0.0197	0.0191	**0.01903**			0.0183	0.0183	**0.01901**

Link-analysis algorithm

	p_1	p_2	p_3	p_4
c_1	0	1	0	1
c_2	0	1	1	1
c_3	1	0	1	0

Interaction matrix: A

	p_1	p_2	p_3	p_4
c_1	0	0.62	0	0.62
c_2	0	0.46	0.46	0.46
c_3	0.62	0	0.62	0

Matrix: B

	c_1	c_2	c_3
c_1	1	0	0
c_2	0	1	0
c_3	0	0	1

Consumer representativeness source matrix: CR^0

Product representativeness matrix: PR

	p_1	p_2	p_3	p_4
c_1	0	1	0	1
c_2	0	1	1	1
c_3	1	0	1	0

	p_1	p_2	p_3	p_4
c_1	0	3.16	0.93	3.16
c_2	0.62	3.62	3.01	3.62
c_3	2.23	0.46	2.69	0.46

	p_1	p_2	p_3	p_4
c_1	0.57	8.24	3.93	8.24
c_2	2.23	10.2	7.98	10.2
c_3	4.03	2.25	5.71	2.25

Consumer representativeness matrix: CR

	c_1	c_2	c_3
c_1	2.23	0.93	0
c_2	1.23	2.39	0.62
c_3	0	0.46	2.23

Iteration 1

	c_1	c_2	c_3
c_1	4.89	3.36	0.57
c_2	4.46	5.75	2.23
c_3	0.57	1.68	4.03

Iteration 2

	c_1	c_2	c_3
c_1	11.2	9.46	2.77
c_2	12.6	14.2	6.28
c_3	2.77	4.73	7

Iteration 3

Figure 1. Six recommendation algorithms illustrated with a simple hypothetical interaction matrix.

Table 1. Data set characteristics.*

Data set	Number of consumers	Number of products	Number of transactions	Density level (%)	Average number of purchases per consumer	Average sales per product
Apparel	1,000 (~4 million)	7,328 (~128,000)	9,332 (~16 million)	0.1273 (~0.0031)	9.33 (~4)	1.27 (~125)
Book	851 (~2,000)	8,566 (~9,700)	13,902 (~18,000)	0.1907 (0.0928)	16.34 (~9)	1.62 (~1.86)
Movie	1,000 (~3,500)	2,900 (~6,000)	50,748 (~1 million)	1.7499 (4.7619)	50.75 (~166)	17.50 (~285.71)

* The statistics of the complete data sets appear in parentheses.

lower precision and higher recall. So, precision, recall, and the F-measure might be sensitive to the number of recommendations. The number of recommendations has a similar but minor effect on the rank score, because the rank score of individual recommendations decreases exponentially with the list index j.

To complement precision and recall, we also adopted a *receiver operating characteristics curve*-based measure, which several recent recommendation evaluation studies have also used.[6,8] The ROC curve attempts to measure the extent to which a learning system can successfully distinguish between signal (relevance) and noise (nonrelevance). For our recommendation task, we defined relevance on the basis of consumer-product pairs: a recommendation that corresponds to a transaction in the testing set is deemed relevant; otherwise, it's deemed nonrelevant. The ROC curve's x-axis and y-axis are the percent of nonrelevant recommendations and the percent of relevant recommendations, respectively. The entire set of nonrelevant recommendations consists of all possible consumer-product pairs that appear neither in the training set (not recommended) nor in the testing set (not relevant). The entire set of relevant recommendations corresponds to the transactions in the testing set. As we increase the number of recommendations from one to the total number of products, we can plot an ROC curve based on the corresponding relevance and nonrelevance percentages at each step.

The *area under the ROC curve* (AUC) completely characterizes the algorithm's performance across all possible numbers of recommendations. An ideal recommendation model that ranks all of a consumer's future purchases at the top of that consumer's recommendation list should steeply increase in the beginning and flatten out in the end, with the AUC close to 1. A random recommendation model would be a 45-degree line with a 0.5 AUC.

For precision, recall, and the F-measure, the overall metric for each algorithm was an average value over all consumers tested. We derived an aggregated rank score RS for all consumers tested:

$$RS = 100 \frac{\sum_c RS_c}{\sum_c RS_c^{\max}}$$

where $RS_c^{\max}$ is the maximum achievable rank score for consumer c if all future purchases are at the top of a ranked list. We used the AUC directly as an overall measure for all consumers.

An experimental study and observations

Our experiment involved samples from three e-commerce data sets. The first was an apparel data set from a leading US online clothing merchant. This set contained three months of data covering approximately 16 million transactions (household-product pairs) involving approximately 4 million households and 128,000 products.

The second was a book data set from a Taiwan online bookstore. It contained three years of data for a sample of 2,000 customers and involved approximately 18,000 transactions and 9,700 books.

The third was a movie-rating data set from the MovieLens project (http://movielens.umn.edu). It contained approximately 1 million ratings on approximately 6,000 movies given by 3,500 users over three years. For this data set, we treated a rating on product p by consumer c as a transaction ($a_{cp} = 1$) and ignored the actual rating. Other recent studies, such as that by Mukund Deshpande and George Karypis,[9] have also used this type of adaptation. Assuming that a user rates a movie on the basis of his or her experience with it, we recommend only whether a consumer will watch a movie in the future; we don't deal with whether he or she will like it.

To allow meaningful testing of the recommendation algorithms, our samples included consumers who had interacted with five to 100 products. This range constraint resulted in 1,000 consumers each for the apparel and movie-rating data sets and 851 consumers for the book data set. Table 1 shows details of the final samples we used. In addition to the numbers of consumers, products, and transactions, we also report each data set's *density level*—that is, the percentage of elements valued at 1 in the interaction matrix. The movie-rating data set has the highest density level (1.75 percent), followed by the book data set (0.19 percent) and then the apparel data set (0.13 percent). We also report the average number of purchases per consumer and the average number of interacting consumers. The comparison of the three data sets is consistent with the three density measures. The statistics of the complete data sets appear in parentheses in table 1.

Following the evaluation procedure we described in the section "Algorithm evaluation metrics," we prepared an unreduced training set, a reduced training set, and a testing set for evaluation. So, we had six experimental configurations (three data sets by two training sets) for each algorithm. We set the number of recommendations at $K = 10$ and the rank score's half-life at $h = 2$. In addition to the six CF algorithms, we included as a benchmark a simple approach (called the "top-N most popular" or "top-N" algorithm) that recommends the 10 most popular unseen products.

On the basis of the existing literature and our understanding of the algorithms, we expected these findings:

- Most algorithms should generally perform better with the unreduced (dense) training set.
- Algorithms designed to alleviate the sparsity problem should generally outperform user-based and item-based algorithms and

Table 2. Experimental results for six collaborative-filtering algorithms.*

Measure	Algorithm	Data set: Apparel Reduced	Apparel Unreduced	Book Reduced	Book Unreduced	Movie Reduced	Movie Unreduced	Average algorithm rank
Precision	User-based	0.0041	0.0088	0.0103	0.0202	**0.0460**	0.0648	4.17
	Item-based	0.0044	0.0096	0.0045	0.0091	**0.0460**	**0.0759**	4.50
	Dimensionality-reduction	0.0050	0.0064	0.0097	0.0191	0.0384	0.0530	5.50
	Generative-model	**0.0069**	0.0066	**0.0283**	**0.0260**	0.0377	0.0471	3.83
	Spreading-activation	**0.0063**	**0.0130**	0.0201	**0.0231**	**0.0437**	0.0607	3.67
	Link-analysis	**0.0081**	**0.0133**	**0.0218**	**0.0267**	**0.0471**	0.0624	**1.67**
	Top-*N* most popular	**0.0069**	0.0062	**0.0268**	**0.0258**	0.0373	0.0452	4.67
Recall	User-based	0.0359	**0.0663**	0.0534	0.1041	**0.0503**	0.0686	4.33
	Item-based	0.0359	**0.0711**	0.0251	0.0454	**0.0538**	**0.0864**	4.33
	Dimensionality-reduction	0.0411	0.0408	0.0564	0.1026	0.0428	0.0580	5.17
	Generative-model	0.0429	0.0356	**0.1616**	**0.1320**	0.0382	0.0468	3.83
	Spreading-activation	**0.0531**	**0.0863**	0.1070	0.1155	0.0457	0.0618	3.33
	Link-analysis	**0.0703**	**0.0891**	0.1212	**0.1282**	**0.0510**	0.0649	**2.17**
	Top-*N* most popular	0.0429	0.0326	**0.1553**	**0.1316**	0.0377	0.0440	4.83
F-measure	User-based	0.0072	0.0153	0.0165	0.0320	**0.0458**	0.0634	4.33
	Item-based	0.0077	0.0166	0.0072	0.0142	**0.0473**	**0.0769**	4.17
	Dimensionality-reduction	0.0088	0.0109	0.0157	0.0305	0.0386	0.0528	5.33
	Generative-model	**0.0113**	0.0107	**0.0454**	**0.0406**	0.0362	0.0447	4.00
	Spreading-activation	**0.0111**	**0.0219**	0.0320	0.0362	0.0426	0.0583	3.67
	Link-analysis	**0.0144**	**0.0224**	0.0349	**0.0415**	**0.0466**	0.0605	**1.83**
	Top-*N* most popular	**0.0113**	0.0100	**0.0431**	**0.0405**	0.0357	0.0425	4.67
Rank score	User-based	1.8750	4.6519	3.8270	7.8635	4.5250	7.4500	4.17
	Item-based	2.0313	4.8527	1.0261	3.1281	4.2167	**8.7500**	4.33
	Dimensionality-reduction	3.4896	3.0120	3.0227	6.9486	4.1000	6.8000	5.17
	Generative-model	1.8490	1.4056	**12.8120**	**10.7443**	4.1250	5.1000	4.67
	Spreading-activation	3.7500	5.2209	8.6800	9.4955	4.7500	7.4000	3.00
	Link-analysis	**4.4035**	**6.4074**	9.9902	**10.3835**	**5.3387**	7.4319	**2.00**
	Top-*N* most popular	2.0052	1.3889	**12.8397**	**10.7814**	3.7000	5.0750	4.67
AUC (area under the receiver-operating-characteristics curve)	User-based	0.4308	0.4628	0.3798	0.5287	0.7984	0.8544	5.67
	Item-based	0.4463	0.4810	0.3873	0.5558	**0.8289**	**0.8930**	4.33
	Dimensionality-reduction	0.4908	0.5663	0.5535	0.6894	0.8051	0.8602	3.00
	Generative-model	**0.6166**	**0.6164**	**0.7246**	**0.7630**	0.7849	0.8179	**2.33**
	Spreading-activation	0.4510	0.5194	0.4908	0.6569	0.7962	0.8325	4.67
	Link-analysis	0.4603	0.5852	0.6333	0.7462	0.7789	0.8058	3.83
	Top-*N* most popular	0.5296	0.5575	0.6487	0.7013	0.7772	0.8045	4.17
Number of target consumers		320	498	601	674	1,000	1,000	

* The boldfaced precision, recall, and F-measures correspond to the algorithms that didn't significantly differ from the highest measure in each configuration at the 10 percent significance level. The boldfaced rank score and AUC measures correspond to the best-performing algorithm in each configuration. Boldfaced average ranks are the top average ranks.

the top-*N* algorithm, especially for the reduced (sparse) training sets.

- The link-analysis algorithm, which takes the global link structure into consideration and features flexible control for penalizing frequent consumers and products, should generally outperform the other algorithms.

Results

Table 2 presents the five recommendation quality measures for the six experimental configurations. The boldfaced precision, re-

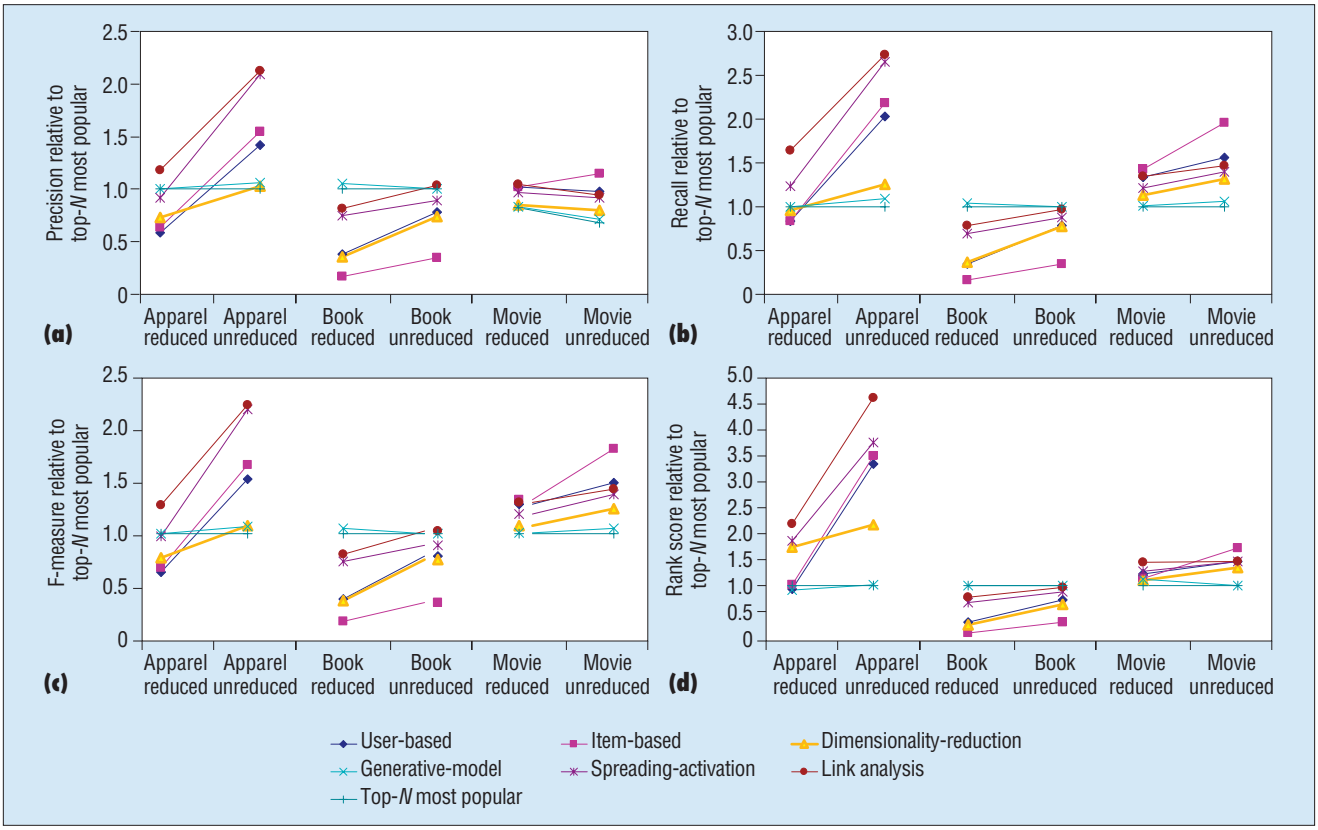


Figure 2. Performance measures relative to the top-*N* algorithm: (a) precision, (b) recall, (c) F-measure, and (d) rank score.

call, and F-measures correspond to the algorithms that didn't significantly differ from the highest measure in each configuration at the 10 percent significance level. The boldfaced rank score and AUC measures correspond to the best-performing algorithm in each configuration.

To summarize each algorithm's overall performance across different data sets, we also report each algorithm's average rank across the six configurations. For example, for precision, the link-analysis algorithm's average rank is 1.67, which corresponds to the average of its ranks for individual configurations (1, 1, 3, 1, 1, and 3). Boldfaced average ranks are the top average ranks.

Because CF algorithms can recommend only those products appearing in the training transactions, no successful recommendation is possible for consumers who don't have recommendable products in the testing set. To make the performance measures meaningful, we evaluated only the recommendations for target consumers for whom successful recommendations were possible. So, each reduced training set had a different number of target consumers from its related unreduced training set (see the bottom of table 2).

For easy visualization of the results, figure 2 presents the algorithms' relative precision, recall, F-measure, and rank score computed as the actual measures divided by the corresponding measures for the top-*N* algorithm. For example, the link-analysis algorithm's precision for the unreduced apparel training set was 2.13, meaning its precision was 113 percent higher than that achieved by the top-*N* algorithm.

Figure 3a presents each algorithm's AUC across the six configurations. Figures 3b and 3c show the actual ROC curves for the reduced and unreduced apparel training sets as examples.

Observations

On the basis of these results, we report five observations.

First, all algorithms performed better with the unreduced data. The performance measures in table 2 were generally better with the unreduced training sets than with the reduced training sets. The difference is even more significant when we take into account the numbers of target consumers. For example, for the apparel data, the average precision of 0.81 percent for 320 target consumers under the reduced training set should be adjusted to 0.52 percent when compared with the average precision of 1.33 percent for 498 target consumers under the unreduced training set ($0.81\% \times 320/498$). The general upward pattern for lines in figures 2 and 3 visually demonstrates this finding. There were only four exceptions in the 105 data algorithm-measure configurations (3 data sets × 7 algorithms × 5 metrics) after we adjusted for the number of target consumers. For the top-*N* and generative-model algorithms, recall and the rank score were slightly higher under the reduced book training set than under the unreduced book training set.

Second, the link-analysis algorithm generally performed the best across all configurations except for the movie-rating training sets. Table 2 shows that this algorithm achieved the highest average ranks for precision, recall, the F-measure, and the rank score (1.67, 2.17, 1.83, and 2). The spreading-activation algorithm achieved the second-highest average ranks (3.67, 3.33, 3.67, and 3). The average ranks for all other algorithms were between 4

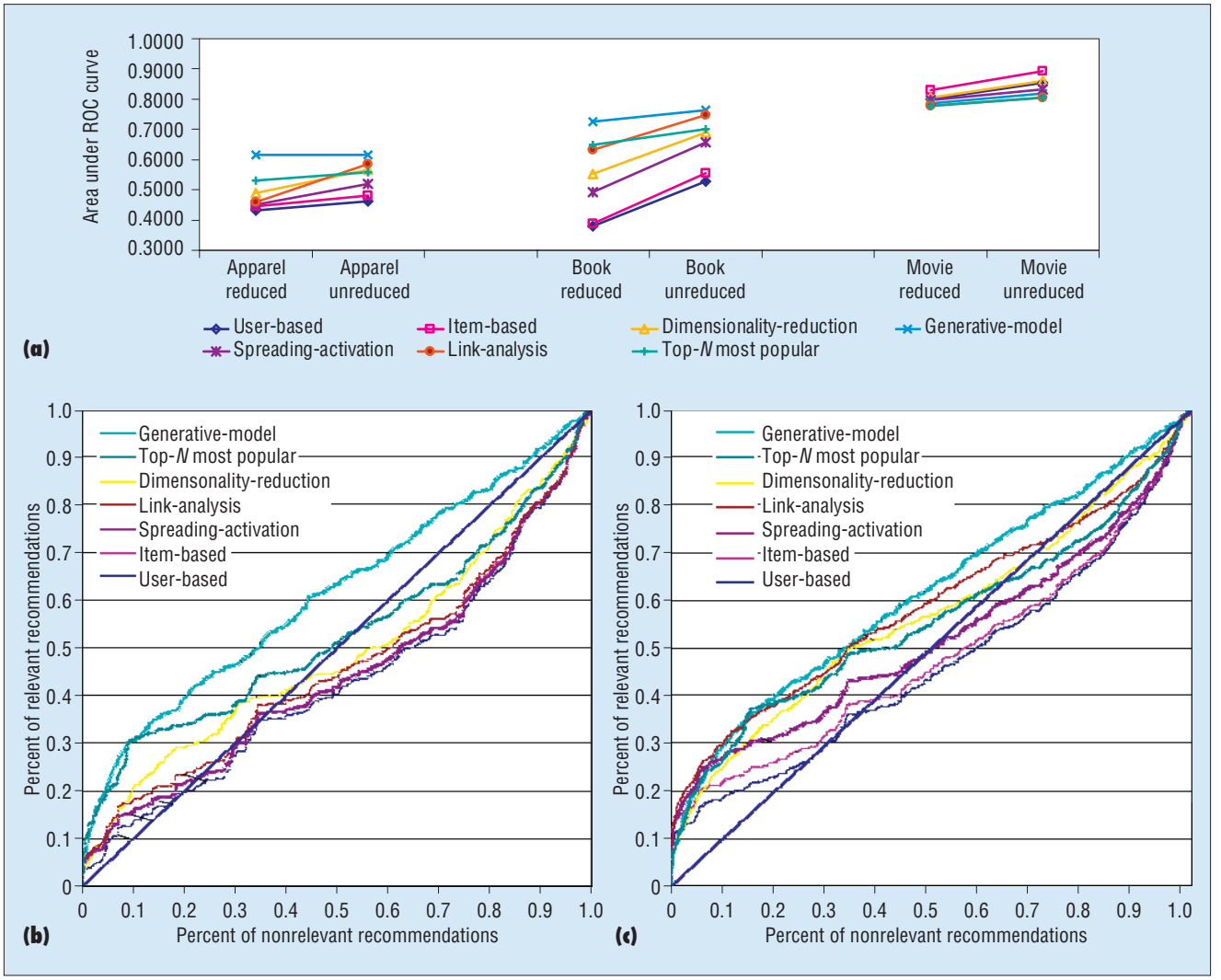


Figure 3. The (a) area under the receiver-operating-characteristics curve (AUC) for each algorithm and the actual ROC curves for the (b) reduced and (c) unreduced apparel training sets.

and 6. The good performance of both the link-analysis and spreading-activation algorithms seems to indicate that graph-based algorithms can effectively exploit additional valuable information (such as transitive associations) in sales transactions. The link-analysis algorithm's dominance was most evident with the unreduced apparel training sets. Compared to the top-*N* algorithm, its precision, recall, and F-measure were about 150 percent higher and its rank score was more than 350 percent higher.

Third, the AUC results generally differ from those for the other four measures. The generative-model algorithm achieved the best AUC measures for the reduced apparel training set and the book training sets. The link-analysis algorithm achieved the best AUC measures for the unreduced apparel training set. The item-based algorithm achieved the best AUC measures for the movie-rating training sets. The generative-model algorithm achieved the best average rank of 2.33.

As we mentioned earlier, the ROC curves completely characterize an algorithm's performance for all possible numbers of recommendations. However, the AUC measure provides only an overall summary of an algorithm's performance, which doesn't necessarily correspond to its performance in practical recommendation settings. For example, consumers are typically much more interested in the quality of the top 10 recommendations; the algorithm's performance at recommendations 100 to 1,000 is practically irrelevant. Consider the ROC curves for the unreduced apparel training set in figure 3c. The ROC curves for the link-analysis algorithm and the generative-model algorithm showed comparable performance, which corresponds to the similar AUC measure in table 2 (0.5852 and 0.5663, respectively). However, a more meaningful comparison is that the link-analysis algorithm's ROC curve is much steeper than that of the generative-model algorithm at the beginning. This difference corresponds to precision and recall that are significantly higher than those for the generative-model algorithm (0.0133 and 0.0891 compared to 0.0066 and 0.0356, respectively).

Fourth, most other algorithms showed mixed performance under different training sets. The item-based algorithm performed exceptionally well for the movie-rating train-

ing sets but had relatively lower performance with the apparel training sets and the worst performance with the book training sets. This algorithm's good performance with the movie-rating training sets might be associated with those sets' much higher transaction density (1.75 percent) and average sales per product (17.50). The dimensionality-reduction algorithm's performance was consistently mediocre across all configurations, and the user-based algorithm was almost always dominated by other algorithms. The generative-model algorithm's performance was always comparable to or slightly better than the top-*N* algorithm. It performed relatively well with the book training sets and the reduced apparel training set but performed worst with the movie-rating training sets and the unreduced apparel training set.

Finally, despite the top-*N* algorithm's simplicity, it wasn't necessarily the worst performer for many configurations. This is especially the case for the book training sets. On the other hand, a closer examination of the actual recommendations from various algorithms showed that products recommended by many of the CF algorithms differed from the top-*N* algorithm recommendations. CF-based recommendations therefore still provide value to consumers, complementing simple popularity-based recommendations such as those from the top-*N* algorithm.

Computational efficiency

We measured the algorithms' computational efficiency using the unreduced apparel training set as the testbed. Figure 4 summarizes the algorithms' total computing times. (We implemented the algorithms in Python, a common scripting language, for fast prototyping. We expect that using a more efficient implementation environment, such as the C programming language, can achieve a speedup of 10 to 50.)

The dimensionality-reduction algorithm required the longest runtime. This was because after reduction, the interaction matrix becomes dense, and computing pairwise consumer similarity requires significant CPU cycles. The item-based algorithm was slow owing to the large number of products (relative to the number of consumers). Both the spreading-activation and link-analysis algorithms require only a small number of iterations to achieve satisfactory recommendation quality, thus requiring modest computing times (less than that for the item-based algorithm). The spreading-activation algorithm was especially fast because it computes recommendations only for target consumers, as opposed to computing recommendations for all consumers in batch mode. The generative-model algorithm was generally very efficient mainly because it requires a small number of hidden classes for quality recommendations and because the expectation maximization converges quickly.

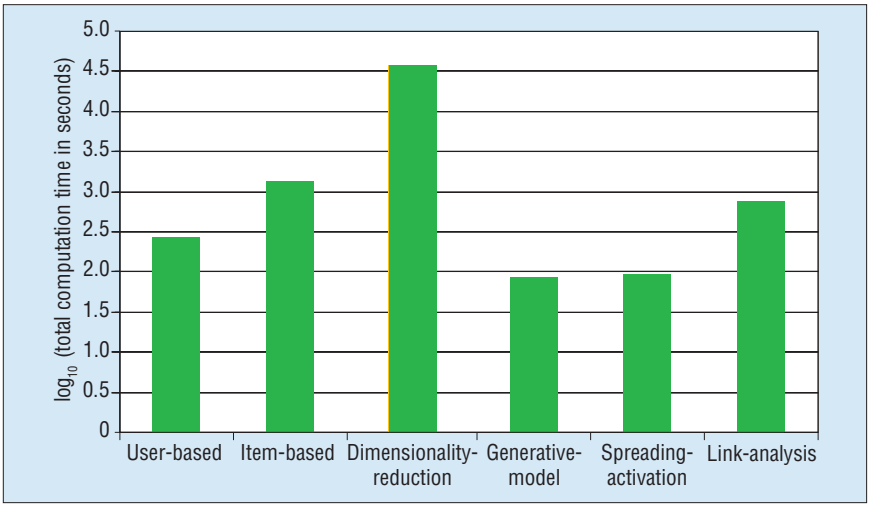


Figure 4. The six algorithms' computational efficiency.

As our results show, the consumer-product interaction matrix's overall sparsity level and row-column density clearly influenced the algorithms' relative performance. However, such data characteristics are far from complete and lack predictive power as a foundation for the metalevel guideline we suggested earlier. Additional prescriptive descriptors of the interaction data must be developed. These descriptors should probably include measures from the graph/network topological-modeling literature, such as node degree distribution, average path length, and cluster coefficient. We're exploring customized versions of these descriptors for recommendation analysis and are using them to guide further comparisons of recommendation algorithms. ■

Acknowledgments

This research was partly supported by US National Science Foundation Digital Library Initiative-II grant IIS-9817473 and NSF Information Technology Research grant IIS-0114011. Daniel Zeng was supported partly by Chinese Academy of Sciences grant 2F05N01, a Ministry of Science and Technology, China, 973 grant 2006CB705500, and National Science Foundation of China grant 60621001.

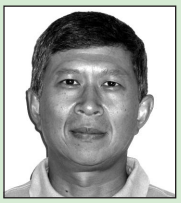
References

1. P. Resnick et al., "An Open Architecture for Collaborative Filtering of Netnews," *ACM Conf. Computer-Supported Cooperative Work*, ACM Press, 1994, pp. 175–186.
2. J.S. Breese, D. Heckerman, and C. Kadie, "Empirical Analysis of Predictive Algorithms for Collaborative Filtering," *Proc. 14th Conf. Uncertainty in Artificial Intelligence* (UAI 98), Morgan Kaufmann, 1998, pp. 43–52.
3. P. Resnick and H. Varian, "Recommender Systems," *Comm. ACM*, vol. 40, no. 3, 1997, pp. 56–58.
4. T. Hofmann, "Latent Semantic Models for Collaborative Filtering," *ACM Trans. Information Systems*, vol. 22, no. 1, 2004, pp. 89–115.
5. Z. Huang, H. Chen, and D. Zeng, "Applying Associative Retrieval Techniques to Alleviate the Sparsity Problem in Collaborative Filtering," *ACM Trans. Information Systems*, vol. 22, no. 1, 2004, pp. 116–142.
6. M. Papagelis, D. Plexousakis, and T. Kutsuras, "Alleviating the Sparsity Problem of Collaborative Filtering Using Trust Inferences," *Proc. 3rd Int'l. Conf. Trust Management* (iTrust 05), Springer, 2005, pp. 224–239.
7. B. Sarwar et al., "Application of Dimensionality Reduction in Recommender Systems: A Case Study," *Proc. WebKDD Workshop at the ACM SIGKDD*, 2000; http://glaros.dtc.umn.edu/gkhome/node/122.
8. J.L. Herlocker et al., "Evaluating Collabora-

The Authors

Zan Huang is an assistant professor in the Department of Supply Chain and Information Systems at Pennsylvania State University's Smeal College of Business. His primary research interests include data analysis and modeling research in recommender systems, financial applications, scientific and technology innovations, and bioinformatics. He received his PhD in management information systems from the University of Arizona. His dissertation received the 2005 ACM SIGMIS Best Dissertation Award. Contact him at 419 Business Bldg., University Park, PA 16802; zanhuang@psu.edu.

Daniel Zeng is an associate professor and the director of the Intelligent Systems and Decisions Lab in the University of Arizona's Department of Management Information Systems. He's also an affiliated professor at the Institute of Automation, Chinese Academy of Sciences. His research interests include software agents and their applications, social computing, security informatics, infectious disease informatics, computational support for auctions and negotiations, and recommender systems. He received his PhD in industrial administration from Carnegie Mellon University. He's the IEEE Intelligent Transportation Systems Society's vice president for technical activities and chair of the INFORMS College on Artificial Intelligence. Contact him at the MIS Dept., Univ. of Arizona, 1130 E. Helen St., Tucson, AZ 85721; zeng@eller.arizona.edu.



Hsinchun Chen is the McClelland Professor of Management Information Systems at the University of Arizona, where he was the Andersen Consulting Professor of the Year in 1999. His research interests include digital libraries; intelligence analysis; biomedical informatics; data, text, and Web mining; knowledge management; and Web computing. He received his PhD in information systems from New York University. He serves on the editorial board of the *ACM Transactions on Information Systems*, the *IEEE Transactions on Intelligent Transportation Systems*, the *IEEE Transactions on Systems, Man, and Cybernetics*, the *Journal of the American Society for Information Science and Technology*, and *Decision Support Systems*. He's a scientific counselor and advisor for the US National Library of Medicine, the Academia Sinica (Taiwan), and the National Library of China (China). Contact him at the MIS Dept., Univ. of Arizona, 1130 E. Helen St., Tucson, AZ 85721; hchen@eller.arizona.edu.

tive Filtering Recommender Systems," *ACM Trans. Information Systems*, vol. 22, no. 1, 2004, pp. 5–53.

9. M. Deshpande and G. Karypis, "Item-Based Top-*N* Recommendation Algorithms," *ACM Trans. Information Systems*, vol. 22, no. 1, 2004, pp. 143–177.
10. L.H. Ungar and D.P. Foster, "A Formal Statistical Approach to Collaborative Filtering," *Proc. 1998 Conf. Automated Learning and Discovery* (CONALD 98), 1998; www.cis.upenn.edu/~ungar/papers/CONALD.ps.
11. J. Kleinberg, "Authoritative Sources in a Hyperlinked Environment," *J. ACM*, vol. 46, no. 5, 1999, pp. 604–632.
12. S. Brin and L. Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine," *Proc. 7th Int'l World Wide Web Conf.*, Elsevier, 1998, pp. 107–117.
13. Z. Huang, D. Zeng, and H. Chen, "A Link Analysis Approach to Recommendation under Sparse Data," *Proc. 2004 Americas Conf. Information Systems*, 2004; http://www.personal.psu.edu/faculty/h/u/huz2/Zan/papers/link.recommend.pdf.