Hindawi Publishing Corporation
Advances in Artificial Intelligence
Volume 2009, Article ID 421425, 19 pages
doi:10.1155/2009/421425

Review Article

A Survey of Collaborative Filtering Techniques

Xiaoyuan Su and Taghi M. Khoshgoftaar

Department of Computer Science and Engineering, Florida Atlantic University, 777 Glades Road, Boca Raton, FL 33431, USA

Correspondence should be addressed to Xiaoyuan Su, suxiaoyuan@gmail.com

Received 9 February 2009; Accepted 3 August 2009

Recommended by Jun Hong

As one of the most successful approaches to building recommender systems, collaborative filtering (*CF*) uses the known preferences of a group of users to make recommendations or predictions of the unknown preferences for other users. In this paper, we first introduce CF tasks and their main challenges, such as data sparsity, scalability, synonymy, gray sheep, shilling attacks, privacy protection, etc., and their possible solutions. We then present three main categories of CF techniques: memory-based, model-based, and hybrid CF algorithms (that combine CF with other recommendation techniques), with examples for representative algorithms of each category, and analysis of their predictive performance and their ability to address the challenges. From basic techniques to the state-of-the-art, we attempt to present a comprehensive survey for CF techniques, which can be served as a roadmap for research and practice in this area.

1. Introduction

In everyday life, people rely on recommendations from other people by spoken words, reference letters, news reports from news media, general surveys, travel guides, and so forth. Recommender systems assist and augment this natural social process to help people sift through available books, articles, webpages, movies, music, restaurants, jokes, grocery products, and so forth to find the most interesting and valuable information for them. The developers of one of the first recommender systems, *Tapestry* [1] (other earlier recommendation systems include rule-based recommenders and user-customization), coined the phrase "collaborative filtering (CF)," which has been widely adopted regardless of the facts that recommenders may not explicitly collaborate with recipients and recommendations may suggest particularly interesting items, in addition to indicating those that should be filtered out [2]. The fundamental assumption of *CF* is that if users X and Y rate n items similarly, or have similar behaviors (e.g., buying, watching, listening), and hence will rate or act on other items similarly [3].

CF techniques use a database of preferences for items by users to predict additional topics or products a new user might like. In a typical *CF* scenario, there is a list of m users $\{u_1, u_2, \ldots, u_m\}$ and a list of n items $\{i_1, i_2, \ldots, i_n\}$, and each user, u_i, has a list of items, Iu_i, which the user has rated, or about which their preferences have been inferred through their behaviors. The ratings can either be explicit indications, and so forth, on a 1–5 scale, or implicit indications, such as purchases or click-throughs [4]. For example, we can convert the list of people and the movies they like or dislike (Table 1(a)) to a user-item ratings matrix (Table 1(b)), in which *Tony* is the *active user* that we want to make recommendations for. There are missing values in the matrix where users did not give their preferences for certain items.

There are many challenges for collaborative filtering tasks (Section 2). CF algorithms are required to have the ability to deal with highly sparse data, to scale with the increasing numbers of users and items, to make satisfactory recommendations in a short time period, and to deal with other problems like synonymy (the tendency of the same or similar items to have different names), shilling attacks, data noise, and privacy protection problems.

Early generation collaborative filtering systems, such as *GroupLens* [5], use the user rating data to calculate the similarity or weight between users or items and make predictions or recommendations according to those calculated similarity values. The so-called memory-based CF methods (Section 3) are notably deployed into commercial systems such as http://www.amazon.com/ (see an example in Figure 1) and

Table 1: An example of a user-item matrix.

(a)

Alice: (like) Shrek, Snow White, (dislike) Superman
Bob: (like) Snow White, Superman, (dislike) spiderman
Chris: (like) spiderman, (dislike) Snow white
Tony: (like) Shrek, (dislike) Spiderman

(b)

	Shrek	Snow White	Spider-man	Super-man
Alice	Like	Like		Dislike
Bob		Like	Dislike	Like
Chris		Dislike	Like	
Tony	Like		Dislike	?

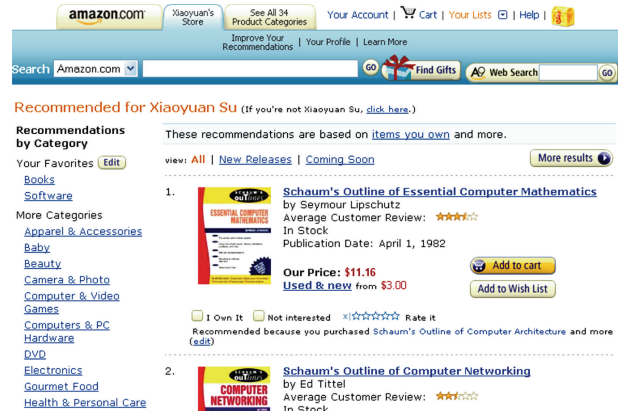

Figure 1: Amazon recommends products to customers by customizing CF systems.

Barnes and Noble, because they are easy-to-implement and highly effective [6, 7]. Customization of *CF* systems for each user decreases the search effort for users. It also promises a greater customer loyalty, higher sales, more advertising revenues, and the benefit of targeted promotions [8].

However, there are several limitations for the memory-based *CF* techniques, such as the fact that the similarity values are based on common items and therefore are unreliable when data are sparse and the common items are therefore few. To achieve better prediction performance and overcome shortcomings of memory-based *CF* algorithms, model-based *CF* approaches have been investigated. Model-based *CF* techniques (Section 4) use the pure rating data to estimate or learn a model to make predictions [9]. The model can be a data mining or machine learning algorithm. Well-known model-based *CF* techniques include Bayesian belief nets (*BNs*) *CF* models [9–11], clustering *CF* models [12, 13], and latent semantic *CF* models [7]. An *MDP* (Markov decision process)-based *CF* system [14] produces a much higher profit than a system that has not deployed the recommender.

Besides collaborative filtering, content-based filtering is another important class of recommender systems. Content-based recommender systems make recommendations by analyzing the content of textual information and finding regularities in the content. The major difference between *CF* and content-based recommender systems is that *CF* only uses the user-item ratings data to make predictions and recommendations, while content-based recommender systems rely on the features of users and items for predictions [15]. Both content-based recommender systems and *CF* systems have limitations. While *CF* systems do not explicitly incorporate feature information, content-based systems do not necessarily incorporate the information in preference similarity across individuals [8].

Hybrid *CF* techniques, such as the *content-boosted CF* algorithm [16] and *Personality Diagnosis (PD)* [17], combine *CF* and content-based techniques, hoping to avoid the limitations of either approach and thereby improve recommendation performance (Section 5).

A brief overview of *CF* techniques is depicted in Table 2.

To evaluate *CF* algorithms (Section 6), we need to use metrics according to the types of *CF* application. Instead of *classification error*, the most widely used evaluation metric for prediction performance of *CF* is Mean Absolute Error (*MAE*). *Precision* and *recall* are widely used metrics for ranked lists of returned items in information retrieval research. *ROC sensitivity* is often used as a decision support accuracy metric.

As drawing convincing conclusions from artificial data is risky, data from live experiments are more desirable for *CF* research. The commonly used *CF* databases are *MovieLens* [18], *Jester* [19], and *Netflix prize* data [20]. In Section 7, we give the conclusion and discussion of this work.

2. Characteristics and Challenges of Collaborative Filtering

E-commerce recommendation algorithms often operate in a challenging environment, especially for large online shopping companies like *eBay* and *Amazon*. Usually, a recommender system providing fast and accurate recommendations will attract the interest of customers and bring benefits to companies. For *CF* systems, producing high-quality predictions or recommendations depends on how well they address the challenges, which are characteristics of CF tasks as well.

2.1. Data Sparsity. In practice, many commercial recommender systems are used to evaluate very large product sets. The user-item matrix used for collaborative filtering will thus be extremely sparse and the performances of the predictions or recommendations of the *CF* systems are challenged.

The data sparsity challenge appears in several situations, specifically, the *cold start* problem occurs when a new user or item has just entered the system, it is difficult to find similar ones because there is not enough information (in some literature, the *cold start* problem is also called the *new user problem* or *new item problem* [21, 22]). New items cannot be recommended until some users rate it, and new

Table 2: Overview of collaborative filtering techniques.

CF categories	Representative techniques	Main advantages	Main shortcomings
Memory-based CF	*Neighbor-based CF (item-based/user-based CF algorithms with Pearson/vector cosine correlation) *Item-based/user-based top-*N* recommendations	*easy implementation *new data can be added easily and incrementally *need not consider the content of the items being recommended *scale well with co-rated items	*are dependent on human ratings *performance decrease when data are sparse *cannot recommend for new users and items *have limited scalability for large datasets
Model-based CF	*Bayesian belief nets CF *clustering CF *MDP-based CF *latent semantic CF *sparse factor analysis *CF using dimensionality reduction techniques, for example, *SVD*, *PCA*	*better address the sparsity, scalability and other problems *improve prediction performance *give an intuitive rationale for recommendations	*expensive model-building *have trade-off between prediction performance and scalability *lose useful information for dimensionality reduction techniques
Hybrid recommenders	*content-based CF recommender, for example, *Fab* *content-boosted CF *hybrid CF combining memory-based and model-based *CF* algorithms, for example, Personality Diagnosis	*overcome limitations of CF and content-based or other recommenders *improve prediction performance *overcome CF problems such as sparsity and gray sheep	*have increased complexity and expense for implementation *need external information that usually not available

users are unlikely given good recommendations because of the lack of their rating or purchase history. *Coverage* can be defined as the percentage of items that the algorithm could provide recommendations for. The *reduced coverage* problem occurs when the number of users' ratings may be very small compared with the large number of items in the system, and the recommender system may be unable to generate recommendations for them. *Neighbor transitivity* refers to a problem with sparse databases, in which users with similar tastes may not be identified as such if they have not both rated any of the same items. This could reduce the effectiveness of a recommendation system which relies on comparing users in pairs and therefore generating predictions.

To alleviate the data sparsity problem, many approaches have been proposed. Dimensionality reduction techniques, such as *Singular Value Decomposition (SVD)* [23], remove unrepresentative or insignificant users or items to reduce the dimensionalities of the user-item matrix directly. The patented *Latent Semantic Indexing* (*LSI*) used in information retrieval is based on *SVD* [24, 25], in which similarity between users is determined by the representation of the users in the reduced space. Goldberg et al. [3] developed *eigentaste*, which applies Principle Component Analysis (*PCA*), a closely-related factor analysis technique first described by Pearson in 1901 [26], to reduce dimensionality. However, when certain users or items are discarded, useful information for recommendations related to them may get lost and recommendation quality may be degraded [6, 27].

Hybrid *CF* algorithms, such as the *content-boosted CF* algorithm [16], are found helpful to address the sparsity problem, in which external content information can be used to produce predictions for *new users* or *new items*. In Ziegler et al. [28], a hybrid collaborative filtering approach was proposed to exploit bulk taxonomic information designed for exact product classification to address the data sparsity problem of *CF* recommendations, based on the generation of profiles via inference of super-topic score and topic diversification [28]. Schein et al. proposed the *aspect model latent variable* method for *cold start* recommendation, which combines both collaborative and content information in model fitting [29]. Kim and Li proposed a probabilistic model to address the *cold start* problem, in which items are classified into groups and predictions are made for users considering the Gaussian distribution of user ratings [30].

Model-based *CF* algorithms, such as *TAN-ELR* (tree augmented naïve Bayes optimized by extended logistic regression) [11, 31], address the sparsity problem by providing more accurate predictions for sparse data. Some new model-based *CF* techniques that tackle the sparsity problem include the *association retrieval technique*, which applies an associative retrieval framework and related spreading activation algorithms to explore transitive associations among users through their rating and purchase history [32]; Maximum margin matrix factorizations (*MMMF*), a convex, infinite dimensional alternative to low-rank approximations and standard factor models [33, 34]; ensembles of *MMMF* [35]; multiple imputation-based *CF* approaches [36]; and imputation-boosted *CF* algorithms [37].

2.2. Scalability. When numbers of existing users and items grow tremendously, traditional *CF* algorithms will suffer serious scalability problems, with computational resources going beyond practical or acceptable levels. For example, with tens of millions of customers (M) and millions of distinct catalog items (N), a *CF* algorithm with the complexity of $O(n)$ is already too large. As well, many systems need to react immediately to online requirements and make recommendations for all users regardless of their purchases and ratings history, which demands a high scalability of a *CF* system [6].

Dimensionality reduction techniques such as *SVD* can deal with the scalability problem and quickly produce good quality recommendations, but they have to undergo expensive matrix factorization steps. An incremental *SVD CF* algorithm [38] precomputes the *SVD* decomposition using existing users. When a new set of ratings are added to the database, the algorithm uses the *folding-in* projection technique [25, 39] to build an incremental system without recomputing the low-dimensional model from scratch. Thus it makes the recommender system highly scalable.

Memory-based *CF* algorithms, such as the item-based *Pearson correlation CF* algorithm can achieve satisfactory scalability. Instead of calculating similarities between all pairs of items, item-based *Pearson CF* calculates the similarity only between the pair of co-rated items by a user [6, 40]. A simple Bayesian *CF* algorithm tackles the scalability problem by making predictions based on observed ratings [41]. Model-based *CF* algorithms, such as clustering *CF* algorithms, address the scalability problem by seeking users for recommendation within smaller and highly similar clusters instead of the entire database [13, 42–44], but there are tradeoffs between scalability and prediction performance.

2.3. Synonymy. Synonymy refers to the tendency of a number of the same or very similar items to have different names or entries. Most recommender systems are unable to discover this latent association and thus treat these products differently. For example, the seemingly different items "*children movie*" and "*children film*" are actual the same item, but memory-based *CF* systems would find no match between them to compute similarity. Indeed, the degree of variability in descriptive term usage is greater than commonly suspected. The prevalence of synonyms decreases the recommendation performance of *CF* systems.

Previous attempts to solve the synonymy problem depended on intellectual or automatic term expansion, or the construction of a thesaurus. The drawback for fully automatic methods is that some added terms may have different meanings from intended, thus leading to rapid degradation of recommendation performance [45].

The *SVD* techniques, particularly the *Latent Semantic Indexing* (*LSI*) method, are capable of dealing with the synonymy problems. *SVD* takes a large matrix of term-document association data and construct a *semantic* space where terms and documents that are closely associated are placed closely to each other. *SVD* allows the arrangement of the space to reflect the major associative patterns in the data, and ignore the smaller, less important ones. The performance of *LSI* in addressing the synonymy problem is impressive at higher *recall* levels where *precision* is ordinarily quite low, thus representing large proportional improvements. However, the performance of the *LSI* method at the lowest levels of *recall* is poor [25].

The *LSI* method gives only a partial solution to the *polysemy* problem, which refers to the fact that most words have more than one distinct meaning [25].

2.4. Gray Sheep. *Gray sheep* refers to the users whose opinions do not consistently agree or disagree with any group of people and thus do not benefit from collaborative filtering [46]. *Black sheep* are the opposite group whose idiosyncratic tastes make recommendations nearly impossible. Although this is a failure of the recommender system, non-electronic recommenders also have great problems in these cases, so *black sheep* is an acceptable failure [47].

Claypool et al. provided a hybrid approach combining content-based and *CF* recommendations by basing a prediction on a weighted average of the content-based prediction and the *CF* prediction. In that approach, the weights of the content-based and *CF* predictions are determined on a per-user basis, allowing the system to determine the optimal mix of content-based and *CF* recommendation for each user, helping to solve the *gray sheep* problem [46].

2.5. Shilling Attacks. In cases where anyone can provide recommendations, people may give tons of positive recommendations for their own materials and negative recommendations for their competitors. It is desirable for *CF* systems to introduce precautions that discourage this kind of phenomenon [2].

Recently, the *shilling attacks models* for collaborative filtering system have been identified and their effectiveness has been studied. Lam and Riedl found that item-based *CF* algorithm was much less affected by the attacks than the user-based *CF* algorithm, and they suggest that new ways must be used to evaluate and detect *shilling attacks* on recommender systems [48]. Attack models for shilling the item-based *CF* systems have been examined by Mobasher et al., and alternative *CF* systems such as hybrid *CF* systems and model-based *CF* systems were believed to have the ability to provide partial solutions to the bias injection problem [49]. O'Mahony et al. contributed to solving the *shilling attacks* problem by analyzing robustness, a recommender system's resilience to potentially malicious perturbations in the customer/product rating matrix [50].

Bell and Koren [51] used a comprehensive approach to the *shilling attacks* problem by removing global effects in the data normalization stage of the neighbor-based *CF*, and working with residual of global effects to select neighbors. They achieved improved *CF* performance on the *Netflix* [20] data.

2.6. Other Challenges. As people may not want their habits or views widely known, *CF* systems also raise concerns about

Table 3: The Nexflix Prize Leaderboard as of July 2009.

Rank	Team	Best RMSE score	Improvement (%)
1	BellKor's Pragmatic Chaos	0.8556	10.07
2	Grand Prize Team	0.8571	9.91
3	Opera Solutions and Vandelay United	0.8573	9.89
4	Vandelay Industries!	0.8579	9.83
5	Pragmatic Theory	0.8582	9.80
6	BellKor in BigChaos	0.8590	9.71
7	Dace	0.8605	9.55
8	Opera Solutions	0.8611	9.49
9	BellKor	0.8612	9.48
10	BigChaos	0.8613	9.47

Table 4: A simple example of ratings matrix.

	I_1	I_2	I_3	I_4
U_1	4	?	5	5
U_2	4	2	1	
U_3	3		2	4
U_4	4	4		
U_5	2	1	3	5

personal privacy. Miller et al. [4] and Canny [52] find ways to protect users' privacy for *CF* recommendation tasks.

Increased noise (or sabotage) is another challenge, as the user population becomes more diverse. Ensembles of maximum margin matrix factorizations [35] and instance selection techniques [53] are found useful to address the noise problems of *CF* tasks. As Dempster-Shafer (*DS*) theory [54, 55] and imputation techniques [56] have been successfully applied to accommodate imperfect and noisy data for knowledge representation and classification tasks, they are also potentially useful to deal with the noise problem of *CF* tasks.

Explainability is another important aspect of recommender systems. An intuitive reasoning such as "you will like this book because you liked those books" will be appealing and beneficial to readers, regardless of the accuracy of the explanations [57].

2.7. The Netflix Prize Challenge. Launched in October 2006, the Netflix prize challenge [20] attracted thousands of researchers to compete in the million-dollar-prize race for a most improved performance for movie recommendations. The challenge is featured with a large-scale industrial dataset (with 480,000 users and 17,770 movies), and a rigid performance metric of RMSE (see detailed description in Section 6).

Up to July, 2009, the Leaderboard on the Netflix prize competition is as Table 3, in which the leading team "BellKor in Pragmatic Chaos" (with 10.05% improved RMSE over the Netflix movie recommendation system: Cinematch) based their solution on a merged model of latent factor and neighborhood models [58]. Some interesting research papers on the Netflix prize challenge can be found in the 2008 KDD Netflix Workshop (http://netflixkddworkshop2008.info/).

3. Memory-Based Collaborative Filtering Techniques

Memory-based *CF* algorithms use the entire or a sample of the user-item database to generate a prediction. Every user is part of a group of people with similar interests. By identifying the so-called neighbors of a new user (or active user), a prediction of preferences on new items for him or her can be produced.

The neighborhood-based *CF* algorithm, a prevalent memory-based *CF* algorithm, uses the following steps: calculate the similarity or weight, $w_{i,j}$, which reflects distance, correlation, or weight, between two users or two items, i and j; produce a prediction for the active user by taking the weighted average of all the ratings of the user or item on a certain item or user, or using a simple weighted average [40]. When the task is to generate a top-N recommendation, we need to find k most similar users or items (nearest neighbors) after computing the similarities, then aggregate the neighbors to get the top-N most frequent items as the recommendation.

3.1. Similarity Computation. Similarity computation between items or users is a critical step in memory-based collaborative filtering algorithms. For item-based *CF* algorithms, the basic idea of the similarity computation between item i and item j is first to work on the users who have rated both of these items and then to apply a similarity computation to determine the similarity, $w_{i,j}$, between the two co-rated items of the users [40]. For a user-based *CF* algorithm, we first calculate the similarity, $w_{u,v}$, between the users u and v who have both rated the same items.

There are many different methods to compute similarity or weight between users or items.

3.1.1. Correlation-Based Similarity. In this case, similarity $w_{u,v}$ between two users u and v, or $w_{i,j}$ between two items i and j, is measured by computing the *Pearson correlation* or other correlation-based similarities.

Pearson correlation measures the extent to which two variables linearly relate with each other [5]. For the user-based algorithm, the *Pearson correlation* between users u and v is

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

where the $i \in I$ summations are over the items that both the users u and v have rated and $\overline{r}_u$ is the average rating of the co-rated items of the uth user. In an example in Table 4, we have $w_{1,5} = 0.756$.

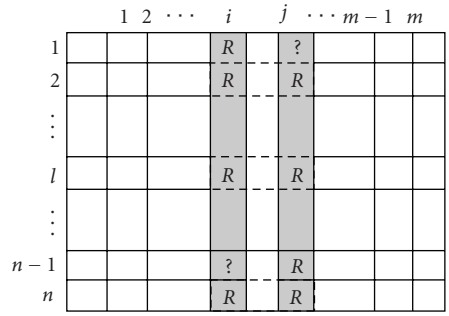

Figure 2: item-based similarity ($w_{i,j}$) calculation based on the co-rated items i and j from users 2, l and n.

For the item-based algorithm, denote the set of users $u \in U$ who rated both items i and j, then the *Pearson Correlation* will be

$$w_{i,j} = \frac{\sum_{u\in U}\left(r_{u,i} - \overline{r}_i\right)\left(r_{u,j} - \overline{r}_j\right)}{\sqrt{\sum_{u\in U}\left(r_{u,i} - \overline{r}_i\right)^2}\sqrt{\sum_{u\in U}\left(r_{u,j} - \overline{r}_j\right)^2}}, \quad (2)$$

where $r_{u,i}$ is the rating of user u on item i, $\overline{r}_i$ is the average rating of the ith item by those users, see Figure 2 [40].

Some variations of item-based and user-based *Pearson correlations* can be found in [59]. The *Pearson correlation*-based *CF* algorithm is a representative *CF* algorithm, and is widely used in the *CF* research community.

Other correlation-based similarities include: *constrained Pearson correlation*, a variation of *Pearson correlation* that uses midpoint instead of mean rate; *Spearman rank correlation*, similar to *Pearson correlation*, except that the ratings are ranks; and *Kendall's* τ *correlation*, similar to the *Spearman rank correlation*, but instead of using ranks themselves, only the relative ranks are used to calculate the correlation [3, 60].

Usually the number of users in the computation of similarity is regarded as the neighborhood size of the active user, and similarity based *CF* is deemed as neighborhood-based *CF*.

3.1.2. Vector Cosine-Based Similarity. The similarity between two documents can be measured by treating each document as a vector of word frequencies and computing the cosine of the angle formed by the frequency vectors [61]. This formalism can be adopted in collaborative filtering, which uses users or items instead of documents and ratings instead of word frequencies.

Formally, if R is the $m \times n$ user-item matrix, then the similarity between two items, i and j, is defined as the cosine of the n dimensional vectors corresponding to the ith and jth column of matrix R.

Vector cosine similarity between items i and j is given by

$$w_{i,j} = \cos\left(\vec{i}, \vec{j}\right) = \frac{\vec{i} \bullet \vec{j}}{\|\vec{i}\| * \|\vec{j}\|}, \quad (3)$$

where "$\bullet$" denotes the dot-product of the two vectors. To get the desired similarity computation, for n items, an $n \times n$ similarity matrix is computed [27]. For example, if the vector $\vec{A} = \{x_1, y_1\}$, vector $\vec{B} = \{x_2, y_2\}$, the vector cosine similarity between $\vec{A}$ and $\vec{B}$ is

$$w_{A,B} = \cos\left(\vec{A}, \vec{B}\right) = \frac{\vec{A} \bullet \vec{B}}{\|\vec{A}\| * \|\vec{B}\|} = \frac{x_1 x_2 + y_1 y_2}{\sqrt{x_1^2 + y_1^2}\sqrt{x_2^2 + y_2^2}}. \quad (4)$$

In an actual situation, different users may use different rating scales, which the *vector cosine similarity* cannot take into account. To address this drawback, *adjusted cosine similarity* is used by subtracting the corresponding user average from each co-rated pair. The *Adjusted cosine similarity* has the same formula as *Pearson correlation* (2). In fact, *Pearson correlation* performs cosine similarity with some sort of normalization of the user's ratings according to his own rating behavior. Hence, we may get negative values with *Pearson correlation*, but not with *cosine similarity*, supposing we have an n-point rating scale.

3.1.3. Other Similarities. Another similarity measure is conditional probability-based similarity [62, 63]. As it is not commonly-used, we will not discuss it in detail in this paper.

3.2. Prediction and Recommendation Computation. To obtain predictions or recommendations is the most important step in a collaborative filtering system. In the neighborhood-based *CF* algorithm, a subset of nearest neighbors of the active user are chosen based on their similarity with him or her, and a weighted aggregate of their ratings is used to generate predictions for the active user [64].

3.2.1. Weighted Sum of Others' Ratings. To make a prediction for the active user, a, on a certain item, i, we can take a weighted average of all the ratings on that item according to the following formula [5]:

$$P_{a,i} = \overline{r}_a + \frac{\sum_{u\in U}\left(r_{u,i} - \overline{r}_u\right) \cdot w_{a,u}}{\sum_{u\in U}\left|w_{a,u}\right|}, \quad (5)$$

where $\overline{r}_a$ and $\overline{r}_u$ are the average ratings for the user a and user u on all other rated items, and $w_{a,u}$ is the weight between the user a and user u. The summations are over all the users $u \in U$ who have rated the item i. For the simple example in Table 4, using the user-based *CF* algorithm, to predict the rating for U_1 on I_2, we have

$$\begin{aligned} P_{1,2} &= \overline{r}_1 + \frac{\sum_u\left(r_{u,2} - \overline{r}_u\right) \cdot w_{1,u}}{\sum_u\left|w_{1,u}\right|} \\ &= \overline{r}_1 + \frac{\left(r_{2,2} - \overline{r}_2\right)w_{1,2} + \left(r_{4,2} - \overline{r}_4\right)w_{1,4} + \left(r_{5,2} - \overline{r}_5\right)w_{1,5}}{\left|w_{1,2}\right| + \left|w_{1,4}\right| + \left|w_{1,5}\right|} \\ &= 4.67 + \frac{(2 - 2.5)(-1) + (4 - 4)0 + (1 - 3.33)0.756}{1 + 0 + 0.756} \\ &= 3.95. \end{aligned} \quad (6)$$

Note the above prediction is based on the neighborhood of the active users.

3.2.2. Simple Weighted Average. For item-based prediction, we can use the *simple weighted average* to predict the rating, $P_{u,i}$, for user u on item i [40]

$$P_{u,i} = \frac{\sum_{n \in N} r_{u,n} w_{i,n}}{\sum_{n \in N} |w_{i,n}|}, \quad (7)$$

where the summations are over all other rated items $n \in N$ for user u, $w_{i,n}$ is the weight between items i and n, $r_{u,n}$ is the rating for user u on item n.

3.3. Top-N Recommendations. Top-N recommendation is to recommend a set of N top-ranked items that will be of interest to a certain user. For example, if you are a returning customer, when you log into your http://amazon.com/ account, you may be recommended a list of books (or other products) that may be of your interest (see Figure 1). Top-N recommendation techniques analyze the user-item matrix to discover relations between different users or items and use them to compute the recommendations. Some models, such as association rule mining based models, can be used to make top-N recommendations, which we will introduce in Section 4.

3.3.1. User-Based Top-N Recommendation Algorithms. User-based top-N recommendation algorithms firstly identify the k most similar users (nearest neighbors) to the active user using the *Pearson correlation* or vector-space model [9, 27], in which each user is treated as a vector in the m-dimensional item space and the similarities between the active user and other users are computed between the vectors. After the k most similar users have been discovered, their corresponding rows in the user-item matrix R are aggregated to identify a set of items, C, purchased by the group together with their frequency. With the set C, user-based *CF* techniques then recommend the top-N most frequent items in C that the active user has not purchased. User-based top-N recommendation algorithms have limitations related to scalability and real-time performance [62].

3.3.2. Item-Based Top-N Recommendation Algorithms. Item-based top-N recommendation algorithms have been developed to address the scalability problem of user-based top-N recommendation algorithms. The algorithms firstly compute the k most similar items for each item according to the similarities; then identify the set, C, as candidates of recommended items by taking the union of the k most similar items and removing each of the items in the set, U, that the user has already purchased; then calculate the similarities between each item of the set C and the set U. The resulting set of the items in C, sorted in decreasing order of the similarity, will be the recommended item-based Top-N list [62]. One problem of this method is, when the joint distribution of a set of items is different from the distributions of the individual items in the set, the above schemes can potentially produce suboptimal recommendations. To solve this problem, Deshpande and Karypis [63] developed higher-order item-based top-N recommendation algorithms that use all combinations of items up to a particular size when determining the itemsets to be recommended to a user.

3.4. Extensions to Memory-Based Algorithms

3.4.1. Default Voting. In many collaborative filters, pairwise similarity is computed only from the ratings in the intersection of the items both users have rated [5, 27]. It will not be reliable when there are too few votes to generate similarity values. Also, focusing on intersection set similarity neglects the global rating behavior reflected in a user's entire rating history.

Empirically, assuming some *default voting* values for the missing ratings can improve the *CF* prediction performance. Herlocker et al. [64] accounts for small intersection sets by reducing the weight of users that have fewer than 50 items in common. Chee et al. [13] uses the average of the clique (or small group) as default voting to extend each user's rating history. Breese et al. [9] uses a neutral or somewhat negative preference for the unobserved ratings and then computes the similarity between users on the resulting ratings data.

3.4.2. Inverse User Frequency. The idea of *inverse user frequency* [61] applied in collaborative filtering is that universally liked items are not as useful in capturing similarity as less common items. The inverse frequency can be defined as $f_j = \log(n/n_j)$, where n_j is the number of users who have rated item j and n is the total number of users. If everyone has rated item j, then f_j is zero. To apply *inverse user frequency* while using the vector similarity-based *CF* algorithm, we need to use a transformed rating, which is simply the original rating multiplied by the f_j factor [9].

3.4.3. Case Amplification. Case amplification refers to a transform applied to the weights used in the basic collaborative filtering prediction. The transform emphasizes high weights and punishes low weights [9]:

$$w'_{i,j} = w_{i,j} \cdot \left| w_{i,j} \right|^{\rho - 1}, \quad (8)$$

where ρ is the *case amplification* power, $\rho \geq 1$, and a typical choice of ρ is 2.5 [65]. *Case amplification* reduces noise in the data. It tends to favor high weights as small values raised to a power become negligible. If the weight is high, for example, $w_{i,j} = 0.9$, then it remains high ($0.9^{2.5} \approx 0.8$); if it is low, for example, $w_{i,j} = 0.1$, then it will be negligible ($0.1^{2.5} \approx 0.003$).

3.4.4. Imputation-Boosted CF Algorithms. When the rating data for *CF* tasks are extremely sparse, it will be problematic to produce accurate predictions using the *Pearson correlation-based CF*. Su et al. [37, 66] proposed a framework of *imputation-boosted collaborative filtering (IBCF)*, which first uses an imputation technique to fill in the missing data, before using a traditional *Pearson correlation*-based *CF* algorithm on this completed data to predict a specific

user rating for a specified item. After comprehensively investigating the use of various standard imputation techniques (including mean imputation, linear regression imputation, and predictive mean matching [67] imputation, and Bayesian multiple imputation [68]), and machine learning classifiers [66] (including naïve Bayes, SVM, neural network, decision tree, lazy Bayesian rules) as imputers for *IBCF*, they found that the proposed *IBCF* algorithms can perform very effectively in general, and that *IBCF* using Bayesian multiple imputation, *IBCF-NBM* (a mixture *IBCF* which uses *IBCF* using naïve Bayes for denser datasets and *IBCF* using mean imputation for sparser ones) [37], and *IBCF* using naïve Bayes perform especially well, outperforming the content-boosted *CF* algorithm (a representative hybrid *CF*), and do so without using external content information.

3.4.5. Weighted Majority Prediction. The weighted majority prediction algorithm proposed by Goldman and Warmuth [69] makes its prediction using the rows with observed data in the same column, weighted by the believed similarity between the rows, with binary rating values. The weights (or similarities, with initialized values of 1) are increased by multiplying it by $(2-\gamma)$ when the compared values are same, and decreased by multiplying by γ when different, with $\gamma \in (0,1)$. This update is equivalent to $w_{ii'} = (2-\gamma)^{C_{ii'}}\gamma^{W_{ii'}}$, where $C_{ii'}$ is the number of rows that have the same value as in row i and $W_{ii'}$ is the number of rows having different values. The prediction for a rating on a certain item by the active user is determined by the rating on the item by a certain user, who has the highest accumulated weight value with the active user. This algorithm can be generalized to multiclass data, and be extended from user-to-user similarity to item-to-item similarity and to user-item-combined similarity [70]. One shortcoming of this algorithm is the scalability, when the user number or item number grows over a certain large number n, it will be impractical for the user-to-user or item-to-item similarity computations to update the $O(n^2)$ similarity matrices.

4. Model-Based Collaborative Filtering Techniques

The design and development of models (such as machine learning, data mining algorithms) can allow the system to learn to recognize complex patterns based on the training data, and then make intelligent predictions for the collaborative filtering tasks for test data or real-world data, based on the learned models. Model-based *CF* algorithms, such as Bayesian models, clustering models, and dependency networks, have been investigated to solve the shortcomings of memory-based *CF* algorithms [9, 71]. Usually, classification algorithms can be used as *CF* models if the user ratings are categorical, and regression models and *SVD* methods and be used for numerical ratings.

4.1. Bayesian Belief Net CF Algorithms. A Bayesian belief net (*BN*) is a directed, acyclic graph (*DAG*) with a triplet $\langle N, A, \Theta\rangle$, where each node $n \in N$ represents a random variable, each directed arc $a \in A$ between nodes is a probabilistic association between variables, and Θ is a conditional probability table quantifying how much a node depends on its parents [72]. Bayesian belief nets (*BNs*) are often used for classification tasks.

4.1.1. Simple Bayesian CF Algorithm. The simple Bayesian *CF* algorithm uses a naïve Bayes (*NB*) strategy to make predictions for *CF* tasks. Assuming the features are independent given the class, the probability of a certain class given all of the features can be computed, and then the class with the highest probability will be classified as the predicted class [41]. For incomplete data, the probability calculation and classification production are computed over observed data (the subscript o in the following equation indicates observed values):

$$\text{class} = \arg\max_{j \in \text{classSet}} p\left(\text{class}_j\right)\prod_o P\left(X_o = x_o \mid \text{class}_j\right). \tag{9}$$

The *Laplace Estimator* is used to smooth the probability calculation and avoid a conditional probability of 0:

$$P(X_i = x_i \mid Y = y) = \frac{\#(X_i = x_i, Y = y) + 1}{\#(Y = y) + |X_i|}, \tag{10}$$

where $|X_i|$ is the size of the class set $\{X_i\}$. For an example of binary class, $P(X_i = 0 \mid Y = 1) = 0/2$ will be $(0+1)/(2+2) = 1/4$, $P(X_i = 1 \mid Y = 1) = 2/2$ will be $(2+1)/(2+2) = 3/4$ using the *Laplace Estimator*.

Using the same example in Table 4, the class set is $\{1, 2, \ldots, 5\}$, to produce the rating for U_1 on I_2 using the *simple Bayesian CF* algorithm and the *Laplace Estimator*, we have

$$\begin{aligned}\text{class} &= \arg\max_{c_j \in \{1,2,3,4,5\}} p\left(c_j \mid U_2 = 2, U_4 = 4, U_5 = 1\right)\\ &= \arg\max_{c_j \in \{1,2,3,4,5\}} p\left(c_j\right)P\left(U_2 = 2 \mid c_j\right)P\left(U_4 = 4 \mid c_j\right)\\ &\quad \times P\left(U_5 = 1 \mid c_j\right)\\ &= \arg\max_{c_j \in \{1,2,3,4,5\}} \{0, 0, 0, 0.0031, 0.0019\} = 4\end{aligned} \tag{11}$$

in which $p(5)P(U_2 = 2 \mid 5)P(U_4 = 4 \mid 5)P(U_5 = 1 \mid 5) = (2/3) * (1/7) * (1/7) * (1/7) = 0.0019$.

In Miyahara and Pazzani [10], multiclass data are firstly converted to binary-class data, and then converted to a *Boolean feature vector* rating matrix. These conversions make the use of the *NB* algorithm for *CF* tasks easier, but bring the problems of scalability and the loss of multiclass information for multiclass data. In Miyahara and Pazzani [41], they applied the *simple Bayesian CF* model only on binary data.

Because most real-world *CF* data are multiclass ones, Su and Khoshgoftaar [11] apply the *simple Bayesian CF* algorithm to multiclass data for *CF* tasks, and found *simple Bayesian CF* has worse predictive accuracy but better scalability than the *Pearson correlation-based CF* as it makes

predictions based on observed ratings, and the prediction-making process is less time-consuming.

The simple Bayesian *CF* algorithm can be regarded as memory-based *CF* technique because of its in-memory calculation for *CF* predictions. We put it in this section for the reason that most other Bayesian *CF* algorithms are model-based *CF*s.

4.1.2. NB-ELR and TAN-ELR CF Algorithms. Because of the limitations of the simple Bayesian algorithm for *CF* tasks, advanced *BNs CF* algorithms, with their ability to deal with incomplete data, can be used instead [11]. Extended logistic regression (*ELR*) is a gradient-ascent algorithm [31, 73], which is a discriminative parameter-learning algorithm that maximizes *log conditional likelihood.*

TAN-ELR and *NB-ELR* (tree augmented naïve Bayes [74] and naïve Bayes optimized by *ELR*, resp.) have been proven to have high classification accuracy for both complete and incomplete data [31, 73].

Applied to *CF* tasks, working on real-world multiclass *CF* datasets and using *MAE* as evaluation criterion, the empirical results show that the *TAN-ELR CF* and *NB-ELR CF* algorithms perform significantly better than the *simple Bayesian CF* algorithm, and consistently better than the *Pearson correlation* memory-based *CF* algorithm [11]. However, *TAN-ELR* and *NB-ELR* need a longer time to train the models. A solution is to run the time-consuming training stage offline, and the online prediction-producing stage will take a much shorter time.

4.1.3. Other Bayesian CF Algorithms. Bayesian belief nets with decision trees at each node. This model has a decision tree at each node of the *BNs*, where a node corresponds to each item in the domain and the states of each node correspond to the possible ratings for each item [9]. Their results show that this model has similar prediction performance to *Pearson correlation*-based *CF* methods, and has better performance than Bayesian-clustering and vector cosine memory-based *CF* algorithms.

Baseline Bayesian model uses a Bayesian belief net with no arcs (baseline model) for collaborative filtering and recommends items on their overall popularity [75]. However, the performance is suboptimal.

4.2. Clustering CF Algorithms. A cluster is a collection of data objects that are similar to one another within the same cluster and are dissimilar to the objects in other clusters [76]. The measurement of the similarity between objects is determined using metrics such as *Minkowski* distance and *Pearson correlation*.

For two data objects, $X = (x_1, x_2, \ldots, x_n)$ and $Y = (y_1, y_2, \ldots, y_n)$, the popular *Minkowski* distance is defined as

$$d(X, Y) = \sqrt[q]{\sum_{i=1}^{n} |x_i - y_i|^q}, \tag{12}$$

where n is the dimension number of the object and x_i, y_i are the values of the ith dimension of object X and Y respectively, and q is a positive integer. When $q = 1$, d is *Manhattan* distance; when $q = 2$, d is *Euclidian* distance [76].

Clustering methods can be classified into three categories: partitioning methods, density-based methods, and hierarchical methods [76, 77]. A commonly-used partitioning method is k-means, proposed by MacQueen [78], which has two main advantages: relative efficiency and easy implementation. Density-based clustering methods typically search for dense clusters of objects separated by sparse regions that represent noise. *DBSCAN* [79] and *OPTICS* [80] are well-known density-based clustering methods. Hierarchical clustering methods, such as *BIRCH* [81], create a hierarchical decomposition of the set of data objects using some criterion.

In most situations, clustering is an intermediate step and the resulting clusters are used for further analysis or processing to conduct classification or other tasks. Clustering *CF* models can be applied in different ways. Sarwar et al. [43] and O'Connor and Herlocker [42] use clustering techniques to partition the data into clusters and use a memory-based *CF* algorithm such as a *Pearson correlation*-based algorithm to make predictions for *CF* tasks within each cluster.

Using the k-means method with $k = 2$, the *RecTree* method, proposed by Chee et al. [13], recursively splits the originally large rating data into two sub-clusters as it constructs the *RecTree* from the root to its leaves. The resulting *RecTree* resembles an unbalanced binary tree, of which leaf nodes have a similarity matrix and internal nodes maintain rating centroids of their subtrees. The prediction is made within the leaf node that the active user belongs to. *RecTree* scales by $O(n \log_2(n))$ for off-line recommendation and $O(b)$ for on-line recommendation, where n is the dataset size and b is the partition size, a constant, and it has an improved accuracy over the *Pearson correlation-based CF* when selecting an appropriate size of advisors (cluster of users).

Ungar and Foster [12] clusters users and items separately using variations of k-means and *Gibbs sampling* [82], by clustering users based on the items they rated and clustering items based on the users that rated them. Users can be reclustered based on the number of items they rated, and items can be similarly re-clustered. Each user is assigned to a class with a degree of membership proportional to the similarity between the user and the mean of the class. Their *CF* performance on synthetic data is good, but not good on real data.

A flexible mixture model (*FMM*) extends existing clustering algorithms for *CF* by clustering both users and items at the same time, allowing each user and item to be in multiple clusters and modeling the clusters of users and items separately [15]. Experimental results show that the *FMM* algorithm has better accuracy than the *Pearson correlation-based CF* algorithm and *aspect model* [83].

Clustering models have better scalability than typical collaborative filtering methods because they make predictions within much smaller clusters rather than the entire customer base [13, 27, 44, 84]. The complex and expensive clustering computation is run offline. However, its recommendation quality is generally low. It is possible to

improve quality by using numerous fine-grained segments, but then online user-segment classification becomes almost as expensive as finding similar customers using memory-based collaborative filtering [6]. As optimal clustering over large data sets is impractical, most applications use various forms of greedy cluster generation techniques. For very large datasets, especially those with high dimensionality, sampling or dimensionality reduction is also necessary.

4.3. Regression-Based CF Algorithms. For memory-based *CF* algorithms, in some cases, two rating vectors may be distant in terms of *Euclidean distances* but they have very high similarity using *vector cosine* or *Pearson correlation* measures, where memory-based *CF* algorithms do not fit well and need better solutions. Also, numerical ratings are common in real-life recommender systems. Regression methods that are good at making predictions for numerical values are helpful to address these problems.

A regression method uses an approximation of the ratings to make predictions based on a regression model. Let $X = (X_1, X_2, \ldots, X_n)$ be a random variable representing a user's preferences on different items. The linear regression model can be expressed as

$$Y = \Lambda X + N, \tag{13}$$

where Λ is a $n \times k$ matrix. $N = (N_1, \ldots, N_n)$ is a random variable representing *noise* in user choices, Y is an $n \times m$ matrix with Y_{ij} is the rating of user i on item j, and X is a $k \times m$ matrix with each column as an estimate of the value of the random variable X (user's ratings in the k-dimensional rating space) for one user. Typically, the matrix Y is very sparse.

To remedy this, Canny [52] proposed a *sparse factor analysis*, which replaces missing elements with *default voting* values (the average of some nonmissing elements, either the average by columns, or by rows, or by all), and uses the regression model as the initialization for Expectation Maximization (*EM*) [85] iterations. According to Canny [52], the *sparse factor analysis* has better scalability than *Pearson correlation-based CF* and *Personality Diagnosis (PD)*, a representative hybrid *CF* algorithm [86], and better accuracy than *singular value decomposition (SVD)* [23]. *Sparse factor analysis* also protects user privacy, as it supports computation on encrypted user data [52].

Vucetic and Obradovic [87] proposed a regression-based approach to *CF* tasks on numerical ratings data that searches for similarities between items, builds a collection of simple linear models, and combines them efficiently to provide rating predictions for an active user. They used *ordinary least squares* to estimate the parameters of the linear regression function. Their experimental results show the approach has good performance in addressing the *sparsity*, prediction latency and numerical prediction problems of *CF* tasks. Lemire and Maclachlan [88] proposed *slope one* algorithms to make faster *CF* prediction than memory-based *CF* algorithms.

4.4. MDP-Based CF Algorithms. Instead of viewing the recommendation process as a prediction problem, Shani et al. [14] views it as a sequential optimization problem and uses a *Markov decision processes* (*MDPs*) model [89] for recommender systems.

An *MDP* is a model for sequential stochastic decision problems, which is often used in applications where an agent is influencing its surrounding environment through actions. An *MDP* can be defined as a four-tuple: $\langle S, A, R, Pr \rangle$, where S is a set of states, A is a set of actions, R is a real-valued reward function for each state/action pair, and Pr is the transition probability between every pair of states given each action.

An optimal solution to the *MDP* is to maximize the function of its reward stream. By starting with an initial policy $\pi_0(s) = \arg\max_{a \in A} R(s, a)$, computing the reward value function $V_i(s)$ based on the previous policy, and updating the policy with the new value function at each step, the iterations will converge to an optimal policy [90, 91].

In Shani et al. [14], the states of the *MDP* for the *CF* system are k tuples of items, with some null values corresponding to missing items; the actions of the *MDP* correspond to a recommendation of an item; and the rewards in the *MDP* correspond to the utility of selling an item, for example, the net profit. The state following each recommendation is the user's response to that recommendation, such as taking the recommended item, taking the non-recommended item, or selecting nothing. To handle the large action space, it is assumed that the probability that a user buys an item depends on his current state, item, and whether or not the item is recommended, but does not depend on the identity of the other recommended items.

Working on an Israeli online bookstore, *Mitos*, the deployed *MDP*-recommender system produced a much higher profit than the system without using the recommender. Also, the *MDP CF* model performs much better than the simpler Markov chain (*MC*) model, which is simply an *MDP* without actions [14].

The *MDP*-Based *CF* model in Shani et al. [14] can be viewed as approximating a partial observable *MDP* (*POMDP*) by using a finite rather than unbounded window of past history to define the current state. As the computational and representational complexity of *POMDPs* is high, appropriate approaches to tackling these problems must be developed, which are generally classified into three broad strategies: value function approximation [92], policy based optimization [84, 93], and stochastic sampling [94]. The application of these strategies to *CF* tasks may be an interesting direction of future research.

4.5. Latent Semantic CF Models. A *Latent semantic CF* technique relies on a statistical modeling technique that introduces latent class variables in a mixture model setting to discover user communities and prototypical interest profiles. Conceptionally, it decomposes user preferences using overlapping user communities. The main advantages of this technique over standard memory-based methods are its higher accuracy and scalability [7, 95].

The *aspect model*, proposed by Hofmann and Puzicha [83], is a probabilistic latent-space model, which models individual ratings as a convex combination of rating factors. The latent class variable is associated with each observed pair of {user, item}, with the assumption that users and items are independent from each other given the latent class variable. The performance of the *aspect model* is much better than the clustering model working on the *EachMovie* dataset [96].

A *multinomial model* is a simple probabilistic model for categorical data [9, 97] that assumes there is only one type of user. A *multinomial mixture model* assumes that there are multiple types of users underlying all profiles, and that the rating variables are independent with each other and with the user's identity given the user's type [98]. A *user rating profile* (*URP*) model [97] combines the intuitive appeal of the *multinomial mixture model* and *aspect model* [83], with the high-level generative semantics of *Latent Dirichlet Allocation* (*LDA*, a generative probabilistic model, in which each item is modeled as a finite mixture over an underlying set of users) [99]. *URP* performs better than the *aspect model* and *multinomial mixtures models* for *CF* tasks.

4.6. Other Model-Based CF Techniques. For applications in which ordering is more desirable than classifying, Cohen et al. [100] investigated a two-stage *order learning CF* approach to learning to order. In that approach, one first learns a preference function by conventional means, and then orders a new set of instances by finding the total ordering that best approximates the preference function, which returns a confidence value reflecting how likely that one is preferred to another. As the problem of finding the total ordering is *NP*-complete, a greedy-order algorithm is used to obtain an approximately optimal ordering function. Working on *EachMovie* [96], this order learning *CF* approach performs better than a nearest neighbor *CF* algorithm and a linear regression algorithm.

Association rule based CF algorithms are more often used for top-*N* recommendation tasks than prediction ones. Sarwar et al. [27] describes their approach to using a traditional *association rule mining* algorithm to find rules for developing top-*N* recommender systems. They find the top-*N* items by simply choosing all the rules that meet the thresholds for support and confidence values, sorting items according to the confidence of the rules so that items predicted by the rules that have a higher confidence value are ranked higher, and finally selecting the first *N* highest ranked items as the recommended set [27]. Fu et al. [101] develop a system to recommend web pages by using an a priori algorithm to mine association rules over users' navigation histories. Leung et al. proposed a collaborative filtering framework using fuzzy association rules and multi-level similarity [102].

Other model-based CF techniques include a *maximum entropy approach*, which clusters the data first, and then in a given cluster uses maximum entropy as an objective function to form a conditional maximal entropy model to make predictions [17]. A *dependency network* is a graphical model for probabilistic relationships, whose graph is potentially cyclic. The probability component of a *dependency network* is a set of conditional distributions, one for each node given its parents. Although less accurate than Bayesian belief nets, *dependency networks* are faster in generating predictions and require less time and memory to learn [75]. *Decision tree CF models* treat collaborative filtering as a classification task and use decision tree as the classifier [103]. *Horting* is a graph-based technique in which nodes are users and edges between nodes are degrees of similarity between users [104]. *Multiple multiplicative factor models* (*MMFs*) are a class of causal, discrete latent variable models combining factor distributions multiplicatively and are able to readily accommodate missing data [105]. *Probabilistic principal components analysis* (*pPCA*) [52, 106] determines the principal axes of a set of observed data vectors through maximum-likelihood estimation of parameters in a latent variable model closely related to factor analysis. *Matrix factorization* based *CF* algorithms have been proven to be effective to address the scalability and sparsity challenges of *CF* tasks [33, 34, 107]. *Wang* et al. showed how the development of collaborative filtering can gain benefits from information retrieval theories and models, and proposed probabilistic relevance *CF* models [108, 109].

5. Hybrid Collaborative Filtering Techniques

Hybrid *CF* systems combine *CF* with other recommendation techniques (typically with content-based systems) to make predictions or recommendations.

Content-based recommender systems make recommendations by analyzing the content of textual information, such as documents, URLs, news messages, web logs, item descriptions, and profiles about users' tastes, preferences, and needs, and finding regularities in the content [110]. Many elements contribute to the importance of the textual content, such as observed browsing features of the words or pages (e.g., term frequency and inverse document frequency), and similarity between items a user liked in the past [111]. A content-based recommender then uses heuristic methods or classification algorithms to make recommendations [112]. Content-based techniques have the *start-up* problem, in which they must have enough information to build a reliable classifier. Also, they are limited by the features explicitly associated with the objects they recommend (sometimes these features are hard to extract), while collaborative filtering can make recommendations without any descriptive data. Also, content-based techniques have the overspecialization problem, that is, they can only recommend items that score highly against a user's profile or his/her rating history [21, 113].

Other recommender systems include *demographic-based recommender systems*, which use user profile information such as gender, postcode, occupation, and so forth [114]; *utility-based recommender systems* and *knowledge-based recommender systems*, both of which require knowledge about how a particular object satisfies the user needs [115, 116]. We will not discuss these systems in detail in this work.

Hoping to avoid limitations of either recommender system and improve recommendation performance, hybrid *CF* recommenders are combined by adding content-based characteristics to *CF* models, adding *CF* characteristics to content-based models, combining *CF* with content-based or other systems, or combining different *CF* algorithms [21, 117].

5.1. Hybrid Recommenders Incorporating CF and Content-Based Features. The *content-boosted CF* algorithm uses *naïve Bayes* as the content classifier, it then fills in the missing values of the rating matrix with the predictions of the content predictor to form a *pseudo rating matrix*, in which observed ratings are kept untouched and missing ratings are replaced by the predictions of a content predictor. It then makes predictions over the resulting *pseudo ratings matrix* using a *weighted Pearson correlation*-based *CF* algorithm, which gives a higher weight for the item that more users rated, and gives a higher weight for the active user [16] (see an illustration in Table 5). The *content-boosted CF* recommender has improved prediction performance over some pure content-based recommenders and some pure memory-based *CF* algorithms. It also overcomes the *cold start* problem and tackles the *sparsity* problem of *CF* tasks. Working on reasonably-sized subsets instead of the original rating data, Greinemr et al. used *TAN-ELR* [31] as the content-predictor and directly applied the *Pearson correlation-based CF* instead of a weighted one on the *pseudo rating matrix* to make predictions, and they achieved improved *CF* performance in terms of *MAE* [118].

Ansari et al. [8] propose a Bayesian preference model that statistically integrates several types of information useful for making recommendations, such as user preferences, user and item features, and expert evaluations. They use Markov chain Monte Carlo (*MCMC*) methods [119] for sampling-based inference, which involve sampling parameter estimation from the full conditional distribution of parameters. They achieved better performance than pure collaborative filtering.

The recommender *Fab*, proposed by Balabanović and Shoham [117], maintains user profiles of interest in web pages using content-based techniques, and uses *CF* techniques to identify profiles with similar tastes. It can then recommend documents across user profiles. Sarwar et al. [120] implemented a set of knowledge-based "*filterbots*" as artificial users using certain criteria. A straightforward example of a *filterbot* is a *genrebot*, which bases its opinion solely on the genre of the item, for example, a "*jazzbot*" would give a full mark to a CD simply because it is in the jazz category, while it would give a low score to any other CD in the database. Mooney and Roy [121] use the prediction from the *CF* system as the input to a content-based recommender. Condiff et al. [113] propose a Bayesian mixed-effects model that integrates user ratings, user, and item features in a single unified framework. The *CF* system *Ripper*, proposed by Basu et al. [71], uses both user ratings and contents features to produce recommendations.

Table 5: Content-boosted CF and its variations (a) content data and originally sparse rating data (b) pseudorating data filled by content predictor (c) predictions from (weighted) Pearson CF on the pseudo rating data.

(a)

	Content information				Rating matrix				
	Age	Sex	Career	zip	I_1	I_2	I_3	I_4	I_5
U_1	32	F	writer	22904			4		
U_2	27	M	student	10022	2		4	3	
U_3	24	M	engineer	60402		1			
U_4	50	F	other	60804		3	3	3	3
U_5	28	M	educator	85251	1				

(b)

Pseudo rating data				
I_1	I_2	I_3	I_4	I_5
2	3	**4**	3	2
2	2	**4**	**3**	2
3	**1**	3	4	3
3	**3**	**3**	**3**	**3**
1	2	4	1	2

(c)

Pearson-CF prediction				
I_1	I_2	I_3	I_4	I_5
2	3	**4**	2	3
3	4	**2**	**2**	3
3	**3**	2	3	3
3	**3**	**3**	**3**	**3**
1	3	1	2	2

5.2. Hybrid Recommenders Combining CF and Other Recommender Systems. A *weighted hybrid recommender* combines different recommendation techniques by their weights, which are computed from the results of all of the available recommendation techniques present in the system [115]. The combination can be linear, the weights can be adjustable [46], and *weighted majority voting* [110, 122] or *weighted average voting* [118] can be used. For example, the *P-Tango* system [46] initially gives *CF* and content-based recommenders equal weight, but gradually adjusts the weighting as predictions about user ratings are confirmed or disconfirmed. The strategy of the *P-Tango* system is similar to boosting [123].

A *switching hybrid recommender* switches between recommendation techniques using some criteria, such as confidence levels for the recommendation techniques. When the *CF* system cannot make a recommendation with sufficient confidence, then another recommender system such as a content-based system is attempted. Switching hybrid recommenders also introduce the complexity of parameterization for the switching criteria [115].

Other hybrid recommenders in this category include *mixed hybrid recommenders* [124], *cascade hybrid recommenders* [115], *meta-level recommenders* [110, 115, 117, 125], and so forth.

Many papers empirically compared the performance of hybrid recommenders with the pure *CF* and content-based methods and found that hybrid recommenders may make more accurate recommendations, especially for the *new user* and *new item* situations where a regular *CF* algorithm cannot make satisfactory recommendations. However, hybrid recommenders rely on external information that is usually not available, and they generally have increased complexity of implementation [110, 115, 126].

5.3. Hybrid Recommenders Combining CF Algorithms. The two major classes of *CF* approaches, memory-based and model-based *CF* approaches, can be combined to form hybrid *CF* approaches. The recommendation performances of these algorithms are generally better than some pure memory-based *CF* algorithms and model-based *CF* algorithms [22, 86].

Probabilistic memory-based collaborative filtering (PMCF) combines memory-based and model-based techniques [22]. They use a mixture model built on the basis of a set of stored user profiles and use the posterior distribution of user ratings to make predictions. To address the *new user problem*, an active learning extension to the *PMCF* system can be used to actively query a user for additional information when insufficient information is available. To reduce the computation time, *PMCF* selects a small subset called *profile space* from the entire database of user ratings and gives predictions from the small *profile space* instead of the whole database. *PMCF* has better accuracy than the *Pearson correlation-based CF* and the *model-based CF using naïve Bayes*.

Personality diagnosis (PD) is a representative hybrid *CF* approach that combines memory-based and model-based *CF* algorithms and retains some advantages of both algorithms [86]. In *PD*, the active user is assumingly generated by choosing one of the other users uniformly at random and adding Gaussian noise to his or her ratings. Given the active user's known ratings, we can calculate the probability that he or she is the same "personality type" as other users, and the probability he or she will like the new items. *PD* can also be regarded as a clustering method with exactly one user per cluster. Working on *EachMovie* [96] and *CiteSeer* [127], *PD* makes better predictions than *Pearson correlation*-based and *vector similarity*-based *CF* algorithms and the two model-based algorithms, Bayesian clustering and Bayesian network, investigated by Breese et al. [9].

As an ensemble classifier is able to give more accurate prediction than a member classifier, a hybrid *CF* system that combines different *CF* algorithms using an ensemble scheme will also be helpful to improve predictive performance of *CF* tasks [118].

6. Evaluation Metrics

The quality of a recommender system can be decided on the result of evaluation. The type of metrics used depends on the type of *CF* applications. According to Herlocker et al. [60], metrics evaluating recommendation systems can be broadly classified into the following broad categories: *predictive accuracy metrics*, such as Mean Absolute Error (*MAE*) and its variations; *classification accuracy metrics*, such as *precision*, *recall*, *F1-measure*, and *ROC sensitivity*; *rank accuracy metrics*, such as *Pearson's product-moment correlation*, *Kendall's Tau*, *Mean Average Precision (MAP)*, *half-life utility* [9], and *normalized distance-based performance metric* (*NDPM*) [128].

We only introduce the commonly-used *CF* metrics *MAE*, *NMAE*, *RMSE*, and *ROC sensitivity* here. For other *CF* performance metrics of recommendation quality, see [60]. There are other evaluations of recommender systems including usability evaluation [129] and so forth.

6.1. Mean Absolute Error (MAE) and Normalized Mean Absolute Error (NMAE). Instead of *classification accuracy* or *classification error*, the most widely used metric in *CF* research literature is *Mean Absolute Error (MAE)* [3, 60], which computes the average of the absolute difference between the predictions and true ratings

$$MAE = \frac{\sum_{\{i,j\}} \left| p_{i,j} - r_{i,j} \right|}{n}, \tag{14}$$

where n is the total number of ratings over all users, $p_{i,j}$ is the predicted rating for user i on item j, and $r_{i,j}$ is the actual rating. The lower the *MAE*, the better the prediction.

Different recommender systems may use different numerical rating scales. *Normalized Mean Absolute Error* (*NMAE*) normalizes *MAE* to express errors as percentages of full scale [3]:

$$NMAE = \frac{MAE}{r_{\max} - r_{\min}}, \tag{15}$$

where $r_{\max}$ and $r_{\min}$ are the upper and lower bounds of the ratings.

6.2. Root Mean Squared Error (RMSE). *Root Mean Squared Error (RMSE)* is becoming popular partly because it is the *Netflix prize* [20] metric for movie recommendation performance:

$$RMSE = \sqrt{\frac{1}{n} \sum_{\{i,j\}} \left(p_{i,j} - r_{i,j} \right)^2}, \tag{16}$$

where n is the total number of ratings over all users, $p_{i,j}$ is the predicted rating for user i on item j, and $r_{i,j}$ is the actual rating again. *RMSE* amplifies the contributions of the absolute errors between the predictions and the true values.

Although accuracy metrics have greatly helped the field of recommender systems, the recommendations that are most accurate are sometimes not the ones that are most useful to users, for example, users might prefer to be recommended with items that are unfamiliar with them, rather than the old favorites they do not likely want again [130]. We therefore need to explore other evaluation metrics.

Table 6: Confusion matrix.

Actual	Predicted	
	Positive	Negative
Positive	TruePositive	FalseNegative
Negative	FalsePositive	TureNegative

6.3. ROC Sensitivity. An *ROC (Receiver Operating Characteristic)* curve is a two-dimensional depiction of classifier performance, on which *TPR* (true positive rate) is plotted on the *Y*-axis and *FPR* (false positive rate) is plotted on the *X*-axis. For the confusion matrix in Table 6, we have *TPR = TruePositive/(TotalPositive)*, and *FPR = FalsePositive/(TotalNegative)*. By tuning a threshold value, all the items ranked above it are deemed observed by the user, and below unobserved, thus the system will get different prediction values for different threshold values to draw the *ROC* curve of {*FPR*, *TPR*} points [60].

Generally, if one *ROC* curve is consistently dominant over another, the system represented by the former curve has better prediction performance. But in actual situations, *ROC* curves may intersect with each other.

Variations of the *ROC* metric include *GROC* (global *ROC*) and *CROC* (customer *ROC*) [29].

ROC sensitivity is a measure of the diagnostic power of a *CF* system. Operationally, it is given by the *Area Under the ROC Curve* (*AUC*). The calculation of *AUC* can be the actual area under the *ROC* curve for binary class problems. We can also use the strategy from [131] to estimate AUC:

$$AUC = \frac{S_0 - n_0(n_0 + 1)/2}{n_0 n_1} \tag{17}$$

where n_0 and n_1 are the numbers of negative and positive examples respectively, and $S_0 = \Sigma_{r_i}$, where r_i is the rank of ith positive example in the ranked list. From the above equation, the *AUC* is essentially a measure of the quality of a ranking.

For multiclass problems, we can use estimated *AUC*, which can be the weighted average of the *AUC*s obtained by taking each class as the reference class in turn (i.e., making it class 0 and all other classes class 1). The weight of a class's *AUC* is the class's frequency in the data [129, 132].

The performance of the recommender system with a bigger *AUC* value is better.

7. Conclusions

Collaborative filtering (*CF*) is one of the most successful recommender techniques. Broadly, there are memory-based *CF* techniques such as the neighborhood-based *CF* algorithm; model-based *CF* techniques such as Bayesian belief nets *CF* algorithms, clustering *CF* algorithms, and *MDP*-based *CF* algorithms; and hybrid *CF* techniques such as the content-boosted *CF* algorithm and *Personality diagnosis*.

As a representative memory-based *CF* technique, neighborhood-based *CF* computes similarity between users or items, and then use the weighted sum of ratings or simple weighted average to make predictions based on the similarity values. *Pearson correlation* and *vector cosine similarity* are commonly used similarity calculations, which are usually conducted between co-rated items by a certain user or both users that have co-rated a certain item. To make top-*N* recommendations, *neighborhood-based* methods can be used according to the similarity values. Memory-based *CF* algorithms are easy to implement and have good performances for dense datasets. Shortcomings of memory-based *CF* algorithms include their dependence on user ratings, decreased performance when data are sparse, new users and items problems, and limited scalability for large datasets, and so forth [11, 42, 133]. Memory-based *CF* on imputed rating data and on dimensionality-reduced rating data will produce more accurate predictions than on the original sparse rating data [24, 25, 37].

Model-based *CF* techniques need to train algorithmic models, such as Bayesian belief nets, clustering techniques, or *MDP*-based ones to make predictions for *CF* tasks. Advanced Bayesian belief nets *CF* algorithms with the ability to deal with missing data are found to have better performance than simple *Bayesian CF* models and *Pearson correlation*-based algorithms [11]. Clustering *CF* algorithms make recommendations within small clusters rather than the whole dataset, and achieve better scalability. An *MDP*-based *CF* algorithm incorporates the users' action of taking the recommendation or not into the model, and the optimal solution to the *MDP* is to maximize the function of its reward stream. The *MDP*-based *CF* algorithm brings profits to the customized system deploying it. There are downsides of model-based *CF* techniques, for example, they may not be practical when the data are extremely sparse, the solutions using dimensionality reduction or transformation of multiclass data into binary ones may decrease their recommendation performance, the model-building expense may be high, and there is a tradeoff between prediction performance and scalability for many algorithms.

Most hybrid *CF* techniques combine *CF* methods with content-based techniques or other recommender systems to alleviate shortcomings of either system and to improve prediction and recommendation performance. Besides improved performance, hybrid *CF* techniques rely on external content information that is usually not available, and they generally have increased complexity.

It is always desirable to design a *CF* approach that is easy to implement, takes few resources, produces accurate predictions and recommendations, and overcomes all kinds of challenges presented by real-world *CF* applications, such as data sparsity, scalability, synonymy, privacy protection, and so forth. Although there is no cure-all solution available yet, people are working out solutions for each of the problems. To alleviate the sparsity problem of *CF* tasks, missing-data algorithms such as *TAN-ELR* [31], imputation techniques such as Bayesian multiple imputation [68], and dimensionality reduction techniques such as *SVD* [23] and matrix factorization [107] can be used. Clustering *CF* algorithms and other approaches such as an incremental-*SVD CF* algorithm [38] are found promising in dealing with the scalability problem. Latent semantic indexing (*LSI*) is helpful to handle the synonymy problem. And *sparse factor analysis* is found helpful to protect user privacy [52].

Besides addressing the above challenges, future *CF* techniques should also be able to make accurate predictions in the presence of shilling attacks and noisy data, and be effectively applied in fast-growing mobile applications as well.

There are many evaluation metrics for *CF* techniques. The most commonly used metric for prediction accuracy include mean absolute error (*MAE*), *recall* and *precision*, and *ROC sensitivity*. Because artificial data are usually not reliable due to the characteristics of *CF* tasks, real-world datasets from live experiments are more desirable for *CF* research.

Acknowledgment

The authors are grateful to Drs. Miroslav Kubat and Moiez A. Tapia for their help during the early stage of this paper and also to Drs. Xingquan Zhu, Russ Greiner, Andres Folleco, and Amri Napolitano for their comments. Their thanks also go to Dr. Jun Hong and other editors/reviewers of the AAI Journal for their work as well as to many anonymous reviewers for their comments. This work was supported by the Data Mining and Machine Learning and Empirical Software Engineering and Laboratories at Florida Atlantic University.

References

[1] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," *Communications of ACM*, vol. 35, no. 12, pp. 61–70, 1992.

[2] P. Resnick and H. R. Varian, "Recommender systems," *Communications of the ACM*, vol. 40, no. 3, pp. 56–58, 1997.

[3] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, "Eigentaste: a constant time collaborative filtering algorithm," *Information Retrieval*, vol. 4, no. 2, pp. 133–151, 2001.

[4] B. N. Miller, J. A. Konstan, and J. Riedl, "PocketLens: toward a personal recommender system," *ACM Transactions on Information Systems*, vol. 22, no. 3, pp. 437–476, 2004.

[5] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "Grouplens: an open architecture for collaborative filtering of netnews," in *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, pp. 175–186, New York, NY, USA, 1994.

[6] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: item-to-item collaborative filtering," *IEEE Internet Computing*, vol. 7, no. 1, pp. 76–80, 2003.

[7] T. Hofmann, "Latent semantic models for collaborative filtering," *ACM Transactions on Information Systems*, vol. 22, no. 1, pp. 89–115, 2004.

[8] A. Ansari, S. Essegaier, and R. Kohli, "Internet recommendation systems," *Journal of Marketing Research*, vol. 37, no. 3, pp. 363–375, 2000.

[9] J. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI '98)*, 1998.

[10] K. Miyahara and M. J. Pazzani, "Collaborative filtering with the simple Bayesian classifier," in *Proceedings of the 6th Pacific Rim International Conference on Artificial Intelligence*, pp. 679–689, 2000.

[11] X. Su and T. M. Khoshgoftaar, "Collaborative filtering for multi-class data using belief nets algorithms," in *Proceedings of the International Conference on Tools with Artificial Intelligence (ICTAI '06)*, pp. 497–504, 2006.

[12] L. H. Ungar and D. P. Foster, "Clustering methods for collaborative filtering," in *Proceedings of the Workshopon Recommendation Systems*, AAAI Press, 1998.

[13] S. H. S. Chee, J. Han, and K. Wang, "RecTree: an efficient collaborative filtering method," in *Proceedings of the 3rd International Conference on Data Warehousing and Knowledge Discovery*, pp. 141–151, 2001.

[14] G. Shani, D. Heckerman, and R. I. Brafman, "An MDP-based recommender system," *Journal of Machine Learning Research*, vol. 6, pp. 1265–1295, 2005.

[15] L. Si and R. Jin, "Flexible mixture model for collaborative filtering," in *Proceedings of the 20th International Conference on Machine Learning (ICML '03)*, vol. 2, pp. 704–711, Washington, DC, USA, August 2003.

[16] P. Melville, R. J. Mooney, and R. Nagarajan, "Content-boosted collaborative filtering for improved recommendations," in *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI '02)*, pp. 187–192, Edmonton, Canada, 2002.

[17] D. Y. Pavlov and D. M. Pennock, "A maximum entropy approach to collaborative filtering in dynamic, sparse, high-dimensional domains," in *Advances in Neural Information Processing Systems*, pp. 1441–1448, MIT Press, Cambridge, Mass, USA, 2002.

[18] MovieLens data, http://www.grouplens.org/.

[19] Jester data, http://shadow.ieor.berkeley.edu/humor/.

[20] Netflix prize, http://www.netflixprize.com/.

[21] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734–749, 2005.

[22] K. Yu, A. Schwaighofer, V. Tresp, X. Xu, and H.-P. Kriegel, "Probabilistic memory-based collaborative filtering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 1, pp. 56–69, 2004.

[23] D. Billsus and M. Pazzani, "Learning collaborative information filters," in *Proceedings of the 15th International Conference on Machine Learning (ICML '98)*, 1998.

[24] T. Landauer, M. Littman, and Bell Communications Research (Bellcore), "Computerized cross-language document retrieval using latent semantic indexing," US patent no. 5301109, April 1994.

[25] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society for Information Science*, vol. 41, no. 6, pp. 391–407, 1990.

[26] K. Pearson, "On lines and planes of closest fit to systems of points in space," *Philosophical Magazine*, vol. 2, pp. 559–572, 1901.

[27] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl, "Analysis of recommendation algorithms for E-commerce," in *Proceedings of the ACM E-Commerce*, pp. 158–167, Minneapolis, Minn, USA, 2000.

[28] C.-N. Ziegler, G. Lausen, and L. Schmidt-Thieme, "Taxonomy-driven computation of product recommendations," in *Proceedings of the 13th International Conference on Information and Knowledge Management (CIKM '04)*, pp. 406–415, Washington, DC, USA, November 2004.

[29] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock, "Methods and metrics for cold-start recommendations," in *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '02)*, 2002.

[30] B. M. Kim and Q. Li, "Probabilistic model estimation for collaborative filtering based on items attributes," in *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI '04)*, pp. 185–191, Beijing, China, September 2004.

[31] R. Greinemr, X. Su, B. Shen, and W. Zhou, "Structural extension to logistic regression: discriminative parameter learning of belief net classifiers," *Machine Learning*, vol. 59, no. 3, pp. 297–322, 2005.

[32] Z. Huang, H. Chen, and D. Zeng, "Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering," *ACM Transactions on Information Systems*, vol. 22, no. 1, pp. 116–142, 2004.

[33] N. Srebro, J. D. M. Rennie, and T. Jaakkola, "Maximum-margin matrix factorization," in *Advances in Neural Information Processing Systems*, vol. 17, pp. 1329–1336, 2005.

[34] J. D. M. Rennie and N. Srebro, "Fast maximum margin matrix factorization for collaborative prediction," in *Proceedings of the 22nd International Conference on Machine Learning (ICML '05)*, Bonn, Germany, August 2005.

[35] D. DeCoste, "Collaborative prediction using ensembles of maximum margin matrix factorizations," in *Proceedings of the 23rd International Conference on Machine Learning (ICML '06)*, pp. 249–256, Pittsburgh, Pa, USA, June 2006.

[36] H. Noh, M. Kwak, and I. Han, "mproving the prediction performance of customer behavior through multiple imputation," *Intelligent Data Analysis*, vol. 8, no. 6, pp. 563–577, 2004.

[37] X. Su, T. M. Khoshgoftaar, and R. Greiner, "A mixture imputation-boosted collaborative filter," in *Proceedings of the 21th International Florida Artificial Intelligence Research Society Conference (FLAIRS '08)*, pp. 312–317, Coconut Grove, Fla, USA, May 2008.

[38] B. M. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Incremental SVD-based algorithms for highly scaleable recommender systems," in *Proceedings of the 5th International Conference on Computer and Information Technology (ICCIT '02)*, 2002.

[39] M. W. Berry, S. T. Dumais, and G. W. O'Brien, "Using linear algebra for intelligent information retrieval," *SIAM Review*, vol. 37, no. 4, pp. 573–595, 1995.

[40] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th International Conference on World Wide Web (WWW '01)*, pp. 285–295, May 2001.

[41] K. Miyahara and M. J. Pazzani, "Improvement of collaborative filtering with the simple Bayesian classifier," *Information Processing Society of Japan*, vol. 43, no. 11, 2002.

[42] M. O'Connor and J. Herlocker, "Clustering items for collaborative filtering," in *Proceedings of the ACM SIGIR Workshop on Recommender Systems (SIGIR '99)*, 1999.

[43] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl, "Recommender systems for large-scale E-commerce: scalable neighborhood formation using clustering," in *Proceedings of the 5th International Conference on Computer and Information Technology (ICCIT '02)*, December 2002.

[44] G.-R. Xue, C. Lin, Q. Yang, et al., "Scalable collaborative filtering using cluster-based smoothing," in *Proceedings of the ACM SIGIR Conference*, pp. 114–121, Salvador, Brazil, 2005.

[45] S. K. Jones, "A statistical interpretation of term specificity and its applications in retrieval," *Journal of Documentation*, vol. 28, no. 1, pp. 11–21, 1972.

[46] M. Claypool, A. Gokhale, T. Miranda, et al., "Combining content-based and collaborative filters in an online newspaper," in *Proceedings of the SIGIR Workshop on Recommender Systems: Algorithms and Evaluation*, Berkeley, Calif, USA, 1999.

[47] J. McCrae, A. Piatek, and A. Langley, "Collaborative Filtering," http://www.imperialviolet.org/.

[48] S. K. Lam and J. Riedl, "Shilling recommender systems for fun and profit," in *Proceedings of the 13th International World Wide Web Conference (WWW '04)*, pp. 393–402, New York, NY, USA, May 2004.

[49] B. Mobasher, R. Burke, R. Bhaumik, and C. Williams, "Effective attack models for shilling item-based collaborative filtering systems," in *Proceedings of the WebKDD Workshop*, August 2005.

[50] M. O'Mahony, N. Hurley, N. Kushmerick, and G. Silvestre, "Collaborative recommendation: a robustness analysis," *ACM Transactions on Internet Technology*, vol. 4, no. 4, pp. 344–377, 2004.

[51] R. Bell and Y. Koren, "Improved neighborhood-based collaborative filtering," in *Proceedings of KDD Cup and Workshop*, 2007.

[52] J. Canny, "Collaborative filtering with privacy via factor analysis," in *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 238–245, Tampere, Finland, August 2002.

[53] K. Yu, X. Xu, J. Tao, M. Ester, and H.-P. Kriegel, "Instance selection techniques for memory-based collaborative filtering," in *Proceedings of the SIAM International Conference on Data Mining (SDM '02)*, April 2002.

[54] G. Shafer, *A Mathematical Theory of Evidence*, Princeton University Press, Princeton, NJ, USA, 1976.

[55] D. Cai, M. F. McTear, and S. I. McClean, "Knowledge discovery in distributed databases using evidence theory," *International Journal of Intelligent Systems*, vol. 15, no. 8, pp. 745–761, 2000.

[56] T. M. Khoshgoftaar and J. V. Hulse, "Multiple imputation of software measurement data: a case study," in *Proceedings of the International Conference on Software Engineering and Knowledge Engineering (SEKE '06)*, 2006.

[57] Y. Koren, "Tutorial on recent progress in collaborative filtering," in *Proceedings of the the 2nd ACM Conference on Recommender Systems*, 2008.

[58] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '08)*, pp. 426–434, Las Vegas, Nev, USA, August 2008.

[59] M. R. McLaughlin and J. L. Herlocker, "A collaborative filtering algorithm and evaluation metric that accurately model the user experience," in *Proceedings of 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '04)*, pp. 329–336, Sheffield, UK, 2004.

[60] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Transactions on Information Systems*, vol. 22, no. 1, pp. 5–53, 2004.

[61] G. Salton and M. McGill, *Introduction to Modern Information Retrieval*, McGraw-Hill, New York, NY, USA, 1983.

[62] G. Karypis, "Evaluation of item-based top-N recommendation algorithms," in *Proceedings of the International Conference on Information and Knowledge Management (CIKM '01)*, pp. 247–254, Atlanta, Ga, USA, November 2001.

[63] M. Deshpande and G. Karypis, "Item-based top-N recommendation algorithms," *ACM Transactions on Information Systems*, vol. 22, no. 1, pp. 143–177, 2004.

[64] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl, "An algorithmic framework for performing collaborative filtering," in *Proceedings of the Conference on Research and Development in Information Retrieval (SIGIR '99)*, pp. 230–237, 1999.

[65] D. Lemire, "Scale and translation invariant collaborative filtering systems," *Information Retrieval*, vol. 8, no. 1, pp. 129–150, 2005.

[66] X. Su, T. M. Khoshgoftaar, X. Zhu, and R. Greiner, "Imputation-boosted collaborative filtering using machine learning classifiers," in *Proceedings of the 23rd Annual ACM Symposium on Applied Computing (SAC '08)*, pp. 949–950, Ceará Fortaleza, Brazil, March 2008.

[67] R. J. A. Little, "Missing-data adjustments in large surveys," *Journal of Business & Economic Statistics*, vol. 6, no. 3, pp. 287–296, 1988.

[68] D. B. Rubin, *Multiple Imputation for Nonresponse in Surveys*, John Wiley & Sons, New York, NY, USA, 1987.

[69] S. A. Goldman and M. K. Warmuth, "Learning binary relations using weighted majority voting," *Machine Learning*, vol. 20, no. 3, pp. 245–271, 1995.

[70] A. Nakamura and N. Abe, "Collaborative filtering using weighted majority prediction algorithms," in *Proceedings of the 15th International Conference on Machine Learning (ICML '98)*, 1998.

[71] C. Basu, H. Hirsh, and W. Cohen, "Recommendation as classification: using social and content-based information in recommendation," in *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI '98)*, pp. 714–720, Madison, Wis, USA, July 1998.

[72] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, San Francisco, Calif, USA, 1988.

[73] B. Shen, X. Su, R. Greiner, P. Musilek, and C. Cheng, "Discriminative parameter learning of general Bayesian network classifiers," in *Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence*, pp. 296–305, Sacramento, Calif, USA, November 2003.

[74] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Machine Learning*, vol. 29, no. 2-3, pp. 131–163, 1997.

[75] D. Heckerman, D. M. Chickering, C. Meek, R. Rounthwaite, and C. Kadie, "Dependency networks for inference, collaborative filtering, and data visualization," *Journal of Machine Learning Research*, vol. 1, no. 1, pp. 49–75, 2001.

[76] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Morgan Kaufmann, San Francisco, Calif, USA, 2001.

[77] X. Su, M. Kubat, M. A. Tapia, and C. Hu, "Query size estimation using clustering techniques," in *Proceedings of the 17th International Conference on Tools with Artificial Intelligence (ICTAI '05)*, pp. 185–189, Hong Kong, November 2005.

[78] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the 5th Symposium on Math, Statistics, and Probability*, pp. 281–297, Berkeley, Calif, USA, 1967.

[79] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD '96)*, 1996.

[80] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "OPTICS: ordering points to identify the clustering structure," in *Proceedings of ACM SIGMOD Conference*, pp. 49–60, June 1999.

[81] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: an efficient data clustering method for very large databases," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, vol. 25, pp. 103–114, Montreal, Canada, June 1996.

[82] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, no. 6, pp. 721–741, 1984.

[83] T. Hofmann and J. Puzicha, "Latent class models for collaborative filtering," in *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI '99)*, pp. 688–693, 1999.

[84] A. Ng and M. Jordan, "PEGASUS: a policy search method for large MDPs and POMDPs," in *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI '00)*, Stanford, Calif, USA, 2000.

[85] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society. Series B*, vol. 39, no. 1, pp. 1–38, 1977.

[86] D. M. Pennock, E. Horvitz, S. Lawrence, and C. L. Giles, "Collaborative filtering by personality diagnosis: a hybrid memory- and model-based approach," in *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI '00)*, pp. 473–480, 2000.

[87] S. Vucetic and Z. Obradovic, "Collaborative filtering using a regression-based approach," *Knowledge and Information Systems*, vol. 7, no. 1, pp. 1–22, 2005.

[88] D. Lemire and A. Maclachlan, "Slope one predictors for online rating-based collaborative filtering," in *Proceedings of the SIAM Data Mining Conference (SDM '05)*, 2005.

[89] R. E. Bellman, *Dynamic Programming*, Princeton University Press, Princeton, NJ, USA, 1962.

[90] R. A. Howard, *Dynamic Programming and Markov Processes*, MIT Press, Cambridge, Mass, USA, 1960.

[91] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, Mass, USA, 1998.

[92] M. Hauskrecht, "Incremental methods for computing bounds in partially observable Markov decision processes," in *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI '97)*, pp. 734–739, Providence, RI, USA, July 1997.

[93] P. Poupart and C. Boutilier, "VDCBPI: an approximate scalable algorithm for large POMDPs," in *Proceedings of the 18th Annual Conference on Neural Information Processing Systems (NIPS '04)*, 2004.

[94] M. Kearns, Y. Mansour, and A. Y. Ng, "A sparse sampling algorithm for near-optimal planning in large Markov decision processes," *Machine Learning*, vol. 49, no. 2-3, pp. 193–208, 2002.

[95] T. Hofmann, "Unsupervised learning by probabilistic latent semantic analysis," *Machine Learning*, vol. 42, no. 1-2, pp. 177–196, 2001.

[96] EachMovie dataset, http://www.grouplens.org/node/76.

[97] B. Marlin, "Modeling user rating profiles for collaborative filtering," in *Neural Information Processing Systems*, 2003.

[98] B. Marlin, *Collaborative filtering, a machine learning perspective*, M.S. thesis, Department of Computer Science, University of Toronto, 2004.

[99] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, no. 4-5, pp. 993–1022, 2003.

[100] W. W. Cohen, R. E. Schapire, and Y. Singer, "Learning to order things," *Journal of Artificial Intelligence Research*, vol. 10, pp. 243–270, 1999.

[101] X. Fu, J. Budzik, and K. J. Hammond, "Mining navigation history for recommendation," in *Proceedings of the International Conference on Intelligent User Interfaces (IUI '00)*, pp. 106–112, 2000.

[102] C. W. K. Leung, S. C. F. Chan, and F. L. Chung, "A collaborative filtering framework based on fuzzy association rules and multi-level similarity," *Knowledge and Information Systems*, vol. 10, no. 3, pp. 357–381, 2006.

[103] D. Nikovski and V. Kulev, "Induction of compact decision trees for personalized recommendation," in *Proceedings of the ACM Symposium on Applied Computing*, vol. 1, pp. 575–581, Dijon, France, April 2006.

[104] C. C. Aggarwal, J. L. Wolf, K. Wu, and P. S. Yu, "Horting hatches an egg: a new graph-theoretic approach to collaborative filtering," in *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '99)*, pp. 201–212, 1999.

[105] B. Marlin and R. S. Zemel, "The multiple multiplicative factor model for collaborative filtering," in *Proceedings of the 21st International Conference on Machine Learning (ICML '04)*, pp. 576–583, Banff, Canada, July 2004.

[106] M. E. Tipping and C. M. Bishop, "Probabilistic principal component analysis," *Journal of the Royal Statistical Society. Series B*, vol. 21, no. 3, pp. 611–622, 1999.

[107] G. Takács, I. Pilászy, B. Németh, and D. Tikk, "Investigation of various matrix factorization methods for large recommender systems," in *Proceedings of the IEEE International Conference on Data Mining Workshops (ICDM '08)*, pp. 553–562, Pisa, Italy, December 2008.

[108] J. Wang, S. Robertson, A. P. de Vries, and M. J. T. Reinders, "Probabilistic relevance ranking for collaborative filtering," *Information Retrieval*, vol. 11, no. 6, pp. 477–497, 2008.

[109] J. Wang, A. P. de Vries, and M. J. T. Reinders, "Unified relevance models for rating prediction in collaborative filtering," *ACM Transactions on Information Systems*, vol. 26, no. 3, pp. 1–42, 2008.

[110] M. J. Pazzani, "A framework for collaborative, content-based and demographic filtering," *Artificial Intelligence Review*, vol. 13, no. 5-6, pp. 393–408, 1999.

[111] T. Zhu, R. Greiner, and G. Haubl, "Learning a model of a web user's interests," in *Proceedings of the 9th International Conference on User Modeling (UM '03)*, vol. 2702, pp. 65–75, Johnstown, Pa, USA, June 2003.

[112] M. Pazzani and D. Billsus, "Learning and revising user profiles: the identification of interesting web sites," *Machine Learning*, vol. 27, no. 3, pp. 313–331, 1997.

[113] M. K. Condiff, D. D. Lewis, D. Madigan, and C. Posse, "Bayesian mixed-effects models for recommender systems," in *Proceedings of ACM SIGIR Workshop of Recommender Systems: Algorithm and Evaluation*, 1999.

[114] B. Krulwich, "Lifestyle finder: intelligent user profiling using large-scale demographic data," *Artificial Intelligence Magazine*, vol. 18, no. 2, pp. 37–45, 1997.

[115] R. Burke, "Hybrid recommender systems: survey and experiments," *User Modelling and User-Adapted Interaction*, vol. 12, no. 4, pp. 331–370, 2002.

[116] R. H. Guttman, *Merchant differentiation through integrative negotiation in agent-mediated electronic commerce*, M.S. thesis, School of Architecture and Planning, MIT, 1998.

[117] M. Balabanović and Y. Shoham, "Content-based, collaborative recommendation," *Communications of the ACM*, vol. 40, no. 3, pp. 66–72, 1997.

[118] X. Su, R. Greiner, T. M. Khoshgoftaar, and X. Zhu, "Hybrid collaborative filtering algorithms using a mixture of experts," in *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI '07)*, pp. 645–649, Silicon Valley, Calif, USA, November 2007.

[119] A. E. Gelfand and A. F. M. Smith, "Sampling-based approaches to calculating marginal densities," *Journal of the American Statistical Association*, vol. 85, pp. 398–409, 1990.

[120] B. M. Sarwar, J. A. Konstan, A. Borchers, J. Herlocker, B. Miller, and J. Riedl, "Using filtering agents to improve prediction quality in the grouplens research collaborative filtering system," in *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW '98)*, pp. 345–354, Seattle, Wash, USA, 1998.

[121] R. J. Mooney and L. Roy, "Content-based book recommendation using learning for text categorization," in *Proceedings of the Workshop on Recommender Systems: Algorithms and Evaluation (SIGIR '99)*, Berkeley, Calif, USA, 1999.

[122] J. A. Delgado, *Agent-based information filtering and recommender systems on the internet*, Ph.D. thesis, Nagoya Institute of Technology, February 2000.

[123] R. E. Schapire, "A brief introduction to boosting," in *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI '99)*, pp. 1401–1405, 1999.

[124] B. Smyth and P. Cotter, "A personalized TV listings service for the digital TV age," in *Proceedings of the 19th International Conference on Knowledge-Based Systems and Applied Artificial Intelligence (ES '00)*, vol. 13, pp. 53–59, Cambridge, UK, December 2000.

[125] M. Balabanović, "Exploring versus exploiting when learning user models for text recommendation," *User Modelling and User-Adapted Interaction*, vol. 8, no. 1-2, pp. 71–102, 1998.

[126] A. Popescul, L. H. Ungar, D. M. Pennock, and S. Lawrence, "Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments," in *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence (UAI '01)*, pp. 437–444, 2001.

[127] CiteSeer ResearchIndex, digital library of computer science research papers, http://citeseer.ist.psu.edu.

[128] Y. Y. Yao, "Measuring retrieval effectiveness based on user preference of documents," *Journal of the American Society for Information Science*, vol. 46, no. 2, pp. 133–145, 1995.

[129] R. Sinha and K. Swearingen, "Comparing recommendations made by online systems and friends," in *Proceedings of the DELOS-NSF Workshop on Personalization and Recommender Systems in Digital Libraries*, 2001.

[130] S. M. McNee, J. Riedl, and J. A. Konstan, "Accurate is not always good: how accuracy metrics have hurt recommender systems," in *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI '06)*, 2006.

[131] D. J. Hand and R. J. Till, "A simple generalisation of the area under the ROC curve for multiple class classification problems," *Machine Learning*, vol. 45, no. 2, pp. 171–186, 2001.

[132] F. Provost and P. Domingos, "Well-trained pets: improving probability estimation trees," CDER Working Paper IS-00-04, Stern School of Business, New York University, 2000.

[133] Z. Huang, W. Chung, and H. Chen, "A graph model for E-commerce recommender systems," *Journal of the American Society for Information Science and Technology*, vol. 55, no. 3, pp. 259–274, 2004.