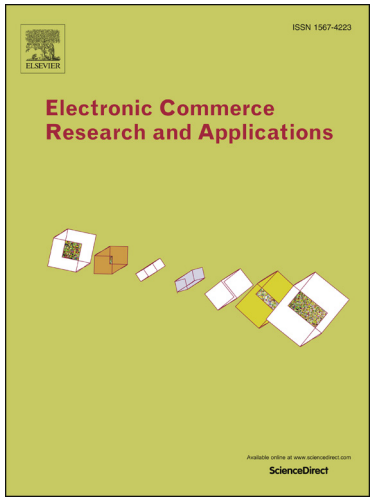
Accepted Manuscript



A Clustering Based Approach to Improving the Efficiency of Collaborative Filtering Recommendation

Chih-Lun Liao, Shie-Jue Lee

PII: S1567-4223(16)30027-8
DOI: http://dx.doi.org/10.1016/j.elerap.2016.05.001
Reference: ELERAP 666

To appear in: *Electronic Commerce Research and Applications*

Received Date: 5 August 2015
Revised Date: 27 March 2016
Accepted Date: 5 May 2016

Please cite this article as: C-L. Liao, S-J. Lee, A Clustering Based Approach to Improving the Efficiency of Collaborative Filtering Recommendation, *Electronic Commerce Research and Applications* (2016), doi: http://dx.doi.org/10.1016/j.elerap.2016.05.001

A Clustering Based Approach to Improving the Efficiency of Collaborative Filtering Recommendation

Chih-Lun Liao[a,1], Shie-Jue Lee[a,b,2,*]

[a] *Department of Electrical Engineering, National Sun Yat-Sen University, Kaohsiung 80424, Taiwan.*
[b] *Electronic Commerce Research Center, and Information Technologies Research Center, National Sun Yat-Sen University, Kaohsiung 80424, Taiwan.*

Abstract

In collaborative filtering recommender systems, products are regarded as features and users are requested to provide ratings to the products they have purchased. By learning from the ratings, such a recommender system can recommend interesting products to users. However, there are usually quite a lot of products involved in E-commerce and it would be very inefficient if every product needs to be considered before making recommendations. We propose a novel approach which applies a self-constructing clustering algorithm to reduce the dimensionality related to the number of products. Similar products are grouped in the same cluster and dissimilar products are dispatched in different clusters. Recommendation work is then done with the resulting clusters. Finally, re-transformation is performed and a ranked list of recommended products is offered to each user. With the proposed approach, the processing time for making recommendations is much reduced. Experimental results show that the efficiency of the recommender system can be greatly improved without compromising the recommendation quality.

Keywords: Collaborative filtering recommender system; correlation graph; self-constructing clustering; dimensionality reduction; ranking algorithm.

*Corresponding author
Email address: leesj@mail.ee.nsysu.edu.tw. (Shie-Jue Lee)
URL: http://itlab.ee.nsysu.edu.tw/ (Shie-Jue Lee)
[1]E-mail: clliao@water.ee.nsysu.edu.tw.
[2]Phone: +886-7-5252000 ext 4141.

1. Introduction

Due to the fast development of E-commerce, nowadays there are a large number of on-line shoppers and a huge amount of products from which people can choose on-line. However, the task of examining and choosing appropriate products from such a large number of products can be not only confusing but also time-consuming. Recommender systems [1, 2] have thus been developed to help people find the products they are interested in and save their time in the search process. For a user, such systems can learn from the recorded experience of all the customers and recommend a preference list of products to the user. During the past several years, recommender systems have been evolving rapidly. Many recommender systems have been developed. In essence, they can be classified into two categories, content-based and collaborative filtering, although a tendency toward hybrid systems [3] has been growing in recent years.

A content-based recommender system [4, 5, 6, 7] makes recommendations to a user based on the content which may include the categories or other attributes of the products. It may also refer to the habits, interests, or preferences of the users. By analyzing these data with some technologies such as Bayesian modeling [8, 9], a content-based recommender system recommends to the user those products that are most appealing. In general, content-based systems require detailed information about products and users. New products can be recommended to the users. However, the information needed is either enormous or hard to get. The products or users have their own attributes. It's difficult to collect the attributes of all the products and users. Furthermore, making sure a product or a user is uniquely represented by the collected attributes is also hard.

A collaborative filtering recommender system [10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25], on the other hand, does not require detailed information about the attributes of the products or the users. Instead, it makes recommendations by applying the interaction information between users and products. Usually, the interaction information is expressed as the user ratings for the purchased products. By learning from the ratings, such a recommender system can recommend a product to a user based on the opinions of other like-minded users on that product. In general,

collaborative filtering recommender systems are simpler and more implementable, and tend to be more appealing and practical in the E-commerce community.

There are usually quite a lot of products to be considered in a recommender system. It would be very inefficient if every product needs to be considered before making recommendations. Dimensionality reduction techniques have been incorporated to produce quickly quality recommendations for large-scale problems. In [26], a variant of K-means called the bisecting K-means clustering algorithm is adopted to group the involved users into different partitions. The neighborhood of a given user is selected by looking into the partition to which the user belongs, and is used for making recommendations to the user. Xue et al. [27] present a smoothing-based approach which employs clusters generated from the training data to provide the basis for data smoothing and neighborhood selection. Honda et al. [28] apply a clustering method for selecting neighbors based on a structural balance theory. Users and products are partitioned into clusters by balancing a general signed graph composed of alternative evaluations on products and users. Ba et al. [29] group the users into clusters according to the attributes, e.g., gender, age, and occupation. Then the user-product rating matrix is decomposed and recombined into a new rating matrix to calculate the similarity between any two users. The iExpand system [30] enhances collaborative filtering by user interest expansion via personalized ranking. It introduces a three-layer representation scheme to help the understanding of the interactions among users, products, and user interests. Latent Dirichlet allocation (LDA) is applied to partition the involved products into clusters. In [31], fuzzy clustering is conducted on users to form user groups. A user group typicality vector is thus constructed for each user. Representing a user by a user group typicality vector instead of product vector can be regarded as a dimension reduction on user representation. Sarwat et al. [32] produce recommendations using a taxonomy of three types of location-based ratings within a single framework. User partitioning and travel penalty are exploited to favor recommendation candidates closer to querying users. In PRM2 [33], personal interest, interpersonal interest similarity, and interpersonal influence are fused into a unified personalized recommendation model, and singular value decomposition (SVD) is used to produce a low-dimensional representation of the original user-product space. BiFu [34] introduces the concepts of

popular products and frequent raters to identify the rating sources for recommendation. To reduce the dimensionality of the rating matrix, K-means [35] is applied to group the users and products into clusters. It also employs the smoothing and fusion technique to overcome the data sparsity and rating diversity. ICRRS [36] is an iterative rating algorithm which is not based on comparing submitted evaluations to an approximation of the final rating scores, and it entirely decouples credibility assessment of the cast evaluations from the ranking itself.

The dimensionality reduction based recommender systems mentioned above have some disadvantages. Some systems [29, 32] require extra attributes about users or products to group the users into clusters. These attributes are usually hard to get in a practical application. Other systems, e.g., [26], [27], and [31], require the number of clusters be given in advance, which is a big burden on the user. Also, the similarity measure most systems adopted for dimensionality reduction only takes the centers of clusters into account. Ignoring the variances of clusters may lead to imprecise results. In this paper, we propose a clustering based approach which applies a self-constructing clustering algorithm to reduce the dimensionality related to the number of products. Similar products are grouped in the same cluster and dissimilar products are dispatched in different clusters. Recommendation work is then done with the resulting clusters called product groups. A correlation graph which shows the inter-relationship among the product groups is created. A series of random walks are then executed and a preference list of product groups is derived for each user. Subsequently, re-transformation, which transforms preference lists of product groups to preference lists of individual products, is performed, and a ranked list of recommended products is finally offered to each user. With the proposed approach, clustering is done totally based on the user-product rating matrix without the necessity of collecting extra attributes about customers and products. Clusters are formed automatically and a pre-determined number of clusters provided by the user is not required. Besides, when measuring the similarity for clustering, we consider both the centers and variances of clusters, resulting in a similarity measure better than that proposed in other methods. Due to dimensionality reduction on the number of products, the processing time for making recommendations by our approach is much reduced. Experimental results show that the efficiency of the

recommender system can be greatly improved without compromising the recommendation quality.

The rest of this paper is organized as follows. The problem to be solved is stated in Section 2. A collaborative filtering recommender system, ItemRank [17], is introduced in Section 3. Our proposed approach of efficiency improvement to ItemRank is described in detail in Section 4. An example for illustration is given in Section 5. Experimental results are presented in Section 6. Finally, a conclusion is given in Section 7.

2. Problem Statement

Suppose there are a set of N users u_i, $1 \leq i \leq N$, and a set of M products p_j, $1 \leq j \leq M$. A user u_i may express his/her evaluation to a product p_j by providing a rating r_{ij}, a positive integer, for p_j. Usually, a higher rating is assumed to indicate a more favorable feedback from the user. If user u_i has not provided a rating for product p_j, $r_{ij} = 0$. Such information can be represented by the following user-product rating matrix $\mathbf{R}$:

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}_1 \\ \mathbf{R}_2 \\ \vdots \\ \mathbf{R}_N \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1M} \\ r_{21} & r_{22} & \dots & r_{2M} \\ \vdots & \vdots & \vdots & \vdots \\ r_{N1} & r_{N2} & \dots & r_{NM} \end{bmatrix} \tag{1}$$

which is an N by M matrix. Note that $\mathbf{R}_i = \begin{bmatrix} r_{i1} & r_{i2} & \dots & r_{iM} \end{bmatrix}$, $1 \leq i \leq N$. For convenience, each row is called a user record and each column is called a product column. The goal of a collaborative filtering recommender system is, given the rating matrix, to recommend a predicted preference list of the products to each user.

3. ItemRank

ItemRank [17] is one of the baseline methods for collaborative filtering recommendation. It applies a random-walk based scoring algorithm to recommend products according to user preferences. ItemRank was chosen in our study since it is less complex, yet performs better, in terms of memory usage and computational cost, than other

baseline systems [37]. Given the rating matrix shown in Eq.(1), ItemRank proceeds with two steps, correlation graph creation and random walks [38]. In the correlation graph creation step, a correlation graph is built from the given ratings. Each product is regarded as a node in the graph. The edge between any two nodes, node p_i and node p_j, $1 \leq i, j \leq M$, has a weight w_{ij} which is the number of users who have provided ratings to both product p_i and product p_j. Note that a user u_k has provided ratings to both products p_i and p_j if $r_{ki} > 0$ and $r_{kj} > 0$. When the correlation graph is completed, one gets the following correlation matrix:

$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1M} \\ w_{21} & w_{22} & \dots & w_{2M} \\ \vdots & \vdots & \vdots & \vdots \\ w_{M1} & w_{M2} & \dots & w_{MM} \end{bmatrix} \tag{2}$$

which is an M by M matrix. Each column of $\mathbf{W}$ is then normalized. In the random walks step, a series of random walks are performed. Consider any user u_i, $1 \leq i \leq N$. Let $\mathbf{S}_i(0)$ be

$$\mathbf{S}_i(0) = \begin{bmatrix} 1/M & 1/M & \dots & 1/M \end{bmatrix}^T \tag{3}$$

which is a vector of size M. The following operation

$$\mathbf{S}_i(t+1) = \alpha \mathbf{W} \mathbf{S}_i(t) + (1-\alpha)\mathbf{R}_i^T \tag{4}$$

is performed repeatedly for $t = 0, 1, 2, \dots$ until convergence is reached. Note that $\alpha \in [0, 1]$ is a user-defined constant. A common choice for α is 0.85. Usually, a reasonable convergence is reached after 20 iterations. Therefore, it is sufficient to apply Eq.(4) 20 times for each user. Let $\mathbf{S}_i$ be the vector after convergence, which is regarded as the predicted preference list of the products for user u_i. Then the products can be recommended to user u_i in the order according to the magnitudes of the elements in $\mathbf{S}_i$. The product corresponding to the largest element in $\mathbf{S}_i$ is the first recommendation, the product corresponding to the second largest element in $\mathbf{S}_i$ is the second recommendation, etc.

4. Proposed Approach

ItemRank encounters the issue of inefficiency. Since there may be a huge number of products involved in E-commerce, the $\mathbf{W}$ matrix, which is of size $M \times M$, can be enormously large. Multiplying $\mathbf{W}$ with $\mathbf{S}_i(t)$ each time in Eq.(4) takes a large amount of time, making ItemRank inefficient for large scale problems. We apply a self-constructing clustering (SCC) algorithm [39, 40] to do dimensionality reduction by grouping products into clusters. Recommendation work is then done with the resulting clusters. As a result, the efficiency of ItemRank can be much improved. Compared with other dimensionality reduction techniques [41, 42, 43, 44], SCC has some advantages. Clusters are formed automatically and a pre-determined number of clusters provided by the user is not required. Besides, when measuring the similarity for clustering, both the centers and variances of clusters are taken into account, resulting in a similarity measure better than that considering only the centers in other methods.

Our approach consists of five steps, user labeling, dimensionality reduction, correlation graph creation, random walks, and re-transformation, as shown in Figure 1. In the user labeling step, SCC is applied to assign the users class labels which help the second step do the job efficiently. In the dimensionality reduction step, SCC is applied again to cluster the products into a number of product groups. Similar products belong to the same product group and dissimilar products belong to different product groups. Since the number of product groups is much smaller than the number of products, the dimensionality involved is much reduced. Then a correlation graph which shows the inter-relationship among the resulting product groups is created in the third step. Based on the correlation graph, a series of random walks are executed and a preference list of product groups is derived for each user in the fourth step. Finally, in the re-transformation step, preference lists of product groups are transformed to preference lists of individual products, and a ranked list of recommended products is offered to each user.

In this study, we chose ItemRank [17] as the target of improvement. However, our approach can also work with other baseline methods to reduce the involved dimensionality and improve their efficiency.

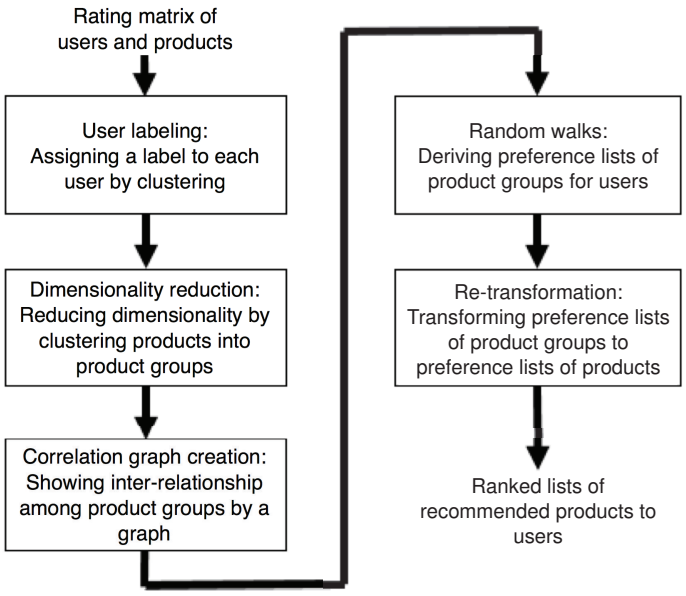


Figure 1: Overview of our approach.

4.1. Self-Constructing Clustering (SCC)

Given a set $\mathbf{X}$ of n patterns $\mathbf{x}_1$, $\mathbf{x}_2$, ..., $\mathbf{x}_n$, with $\mathbf{x}_i = \langle x_{i1}, x_{i2}, \ldots, x_{ip} \rangle$ for $1 \leq i \leq n$, the purpose of the SCC algorithm is to group these patterns into a collection of clusters, with similar patterns being grouped in the same cluster and dissimilar patterns being dispatched in different clusters. Let K be the number of currently existing clusters, named as G_1, G_2, ..., and G_K, respectively. Each cluster G_j has mean $\mathbf{m}_j = \langle m_{j1}, m_{j2}, \ldots, m_{jp} \rangle$ and deviation $\boldsymbol{\sigma}_j = \langle \sigma_{j1}, \sigma_{j2}, \ldots, \sigma_{jp} \rangle$ which stand for the average and standard deviation, respectively, of all the patterns contained in G_j . Let s_j be the size of cluster G_j, i.e., the number of patterns contained in G_j. Initially, we have $K = 0$, indicating that no clusters exist at the beginning. For each pattern $\mathbf{x}_i$, $1 \leq i \leq n$, we calculate the membership degree of $\mathbf{x}_i$ in each existing cluster, $\mu_{G_j}(\mathbf{x}_i)$, by

$$\mu_{G_j}(\mathbf{x}_i) = \prod_{q=1}^{p} \exp\left[-\left(\frac{x_{iq} - m_{jq}}{\sigma_{jq}} \right)^2 \right] \tag{5}$$

for $1 \leq j \leq K$. We say that $\mathbf{x}_i$ passes the similarity test on cluster G_j if

$$\mu_{G_j}(\mathbf{x}_i) \geq \rho \tag{6}$$

where ρ, $0 \leq \rho \leq 1$, is a predefined threshold. Note that ρ plays an important role in this clustering algorithm. A bigger ρ results in smaller clusters, while a smaller ρ results in larger clusters. As ρ increases, the number of clusters also increases. Two cases may occur. Firstly, there are no existing clusters on which $\mathbf{x}_i$ has passed the similarity test. In this case, we assume that $\mathbf{x}_i$ is not similar enough to any existing cluster and a new cluster G_h, $h = K + 1$, is created with

$$\mathbf{m}_h = \mathbf{x}_i, \;\; \boldsymbol{\sigma}_h = \boldsymbol{\sigma}_0 \tag{7}$$

where $\boldsymbol{\sigma}_0 = \langle \sigma_0, \sigma_0, \ldots, \sigma_0 \rangle$ is a user-defined constant vector. Of course, the number of clusters is increased by 1 and the size of cluster G_h, s_h, should be initialized to 1, i.e.,

$$K = h, \;\; s_h = 1. \tag{8}$$

Secondly, there are existing clusters on which $\mathbf{x}_i$ has passed the similarity test. In this case, let cluster G_t be the cluster with the largest membership degree, i.e.,

$$t = arg \max_{1 \leq j \leq K} (\mu_{G_j}(\mathbf{x}_i)). \tag{9}$$

We regard $\mathbf{x}_i$ to be most similar to cluster G_t, and $\mathbf{m}_t$ and $\boldsymbol{\sigma}_t$ of cluster G_t should be modified to include $\mathbf{x}_i$ as its member. The modification to cluster G_t is described as follows:

$$\sigma_{tj} = \sqrt{A - B} + \sigma_0, \tag{10}$$

$$A = \frac{(s_t - 1)(\sigma_{tj} - \sigma_0)^2 + s_t \times {m_{tj}}^2 + {x_{ij}}^2}{s_t}, \tag{11}$$

$$B = \frac{s_t + 1}{s_t} (\frac{s_t \times m_{tj} + x_{ij}}{s_t + 1})^2, \tag{12}$$

$$m_{tj} = \frac{s_t \times m_{tj} + x_{ij}}{s_t + 1} \tag{13}$$

for $1 \leq j \leq p$, and

$$s_t = s_t + 1. \tag{14}$$

Note that K does not change in this case. This process is iterated until all the patterns have been processed. Consequently, we obtain K clusters for $\mathbf{X}$.

4.2. Step 1: User Labeling

To do dimensionality reduction effectively, we need to assign class labels to the users. The idea is to group the users into clusters [32]. Similar users are grouped into the same cluster, and dissimilar users are grouped into different clusters. Then all the users in one group are assigned a unique class label. We use the SCC algorithm for this purpose. Many other clustering algorithms [45, 46, 35] also can do the job, but they require the number of classes to be decided in advance, which is usually difficult in practice. With the SCC algorithm, we only need to provide some meaningful constants during the clustering process.

To apply SCC, we determine the similarity among the users based on the ratings they have provided for the products. However, people are different in personality. For giving ratings, some users are generous and the given scores tend to be high, while others are less generous and the given scores tend to be low. Let's regard the ratings of a user as a waveform. It is reasonable to assume that two users are similar to each other if their waveforms are closely matched except for a certain offset between these two waveforms. Therefore, we normalize the user record of user u_i as follows:

$$\begin{aligned} Q_i &= \sum_{k=1}^{M} r_{ik}, \\ x_{ij} &= \frac{r_{ij}}{Q_i},\ 1 \le j \le M \end{aligned} \tag{15}$$

for $1 \le i \le N$. Let $\mathbf{x}_i = \langle x_{i1}, x_{i2}, \ldots, x_{iM} \rangle$, $1 \le i \le N$, and $\mathbf{X} = \{\mathbf{x}_i | 1 \le i \le N\}$.

We apply the SCC algorithm on $\mathbf{X}$. Suppose z clusters, G_1, G_2, …, G_z, are obtained. Each cluster is regarded as a class, and we have z classes, labeled as c_1, c_2, …, c_z, respectively. For all the users contained in cluster G_j, $1 \le j \le z$, we assign class label c_j to these users. As a result, we expand the original set $\mathbf{R}$ into another set $\mathbf{R}'$ which contains N entries $(\mathbf{R}_1, y_1)$, $(\mathbf{R}_2, y_2)$, …, $(\mathbf{R}_N, y_N)$ where $y_i \in \{c_1, c_2, \ldots, c_z\}$, $1 \le i \le N$.

4.3. Step 2: Dimensionality Reduction

In this step, we reduce the dimensionality M associated with the products using a similar idea proposed in [40]. For each product p_j, $1 \leq j \leq M$, we construct a feature pattern $\mathbf{x}_j = \langle x_{j1}, x_{j2}, \ldots, x_{jz} \rangle$ by

$$x_{jk} = P(c_k|p_j) = \frac{\sum_{d=1}^{N} r_{dj} \times \delta_{dk}}{\sum_{d=1}^{N} r_{dj}},\ 1 \leq k \leq z \tag{16}$$

for $1 \leq j \leq M$, where δ_{dk} is defined as

$$\delta_{dk} = \begin{cases} 1, & \text{if } y_d = c_k; \\ 0, & \text{otherwise.} \end{cases} \tag{17}$$

Therefore, we have M feature patterns $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M$, each having z components. Let $\mathbf{Y} = \{\mathbf{x}_i | 1 \leq i \leq M\}$.

Next, we apply the SCC algorithm on $\mathbf{Y}$. Suppose we obtain q clusters, G_1, G_2, $\ldots$, and G_q. Note that the products contained in a cluster are similar to each other. It is reasonable to employ the cluster to represent all the products contained in this cluster. Since there are q clusters, a user record with M components can be replaced by a new record with q components. In this way, we can reduce the high dimensionality M to a low dimensionality q. Let $\mathbf{T}$ be the reducing matrix:

$$\mathbf{T} = \begin{bmatrix} t_{11} & t_{12} & \ldots & t_{1q} \\ t_{21} & t_{22} & \ldots & t_{2q} \\ \vdots & \vdots & \vdots & \vdots \\ t_{M1} & t_{M2} & \ldots & t_{Mq} \end{bmatrix} \tag{18}$$

where

$$t_{ij} = \mu_{G_j}(\mathbf{x}_i) \tag{19}$$

is the membership degree of $\mathbf{x}_i$ in cluster G_j as defined in Eq.(5), for $1 \leq i \leq M$ and $1 \leq j \leq q$. Then we transform the high-dimensional $\mathbf{R}$, which is an $N \times M$ matrix, to a low-dimensional $\mathbf{B}$ by

$$\mathbf{B} = \begin{bmatrix} \mathbf{B}_1 \\ \mathbf{B}_2 \\ \vdots \\ \mathbf{B}_N \end{bmatrix} = \mathbf{RT} = \begin{bmatrix} \mathbf{R}_1 \\ \mathbf{R}_2 \\ \vdots \\ \mathbf{R}_N \end{bmatrix} \mathbf{T} \tag{20}$$

which is an $N \times q$ matrix. Note that

$$\mathbf{B}_i = \begin{bmatrix} b_{i1} & b_{i2} & \dots & b_{iq} \end{bmatrix} \tag{21}$$

for $1 \leq i \leq N$. For convenience, we call each column in **B** a product group. Thus we have q product groups, named as g_1, g_2, ..., and g_q, respectively. By this, user records with M components, each component corresponding to one product, become new user records with q components, each component corresponding to one product group. Recall that ItemRank works with **R**. But our approach will work with **B**. Since q is usually much smaller than M, the dimensionality can be reduced and the efficiency can be improved significantly.

4.4. Step 3: Correlation Graph Creation

In this step, we create a correlation graph which shows the inter-relationship among the q product groups [47]. Since we work with **B**, instead of **R**, we derive the graph in a somewhat different way. Each product group is regarded as a node, and thus we have q nodes in the graph. The weight w_{ij} between node g_i and node g_j, $1 \leq i, j \leq q$, is calculated by

$$w_{ij} = \begin{cases} 0, \text{ if } i = j \\ \sum_{k=1}^{N} limit(\frac{b_{ki}}{b_{kj}}), \text{ otherwise} \end{cases} \tag{22}$$

where

$$limit(\frac{a_1}{a_2}) = \begin{cases} 0, \text{ if } a_1 = 0 \text{ or } a_2 = 0 \\ \frac{a_1}{a_2}, \text{ if } a_1 < a_2 \\ 1, \text{ otherwise} \end{cases} \tag{23}$$

For the case of $a_1 < a_2$, we have $w_{ij} = \frac{a_1}{a_2} < 1$ since it is less preferable for a user to go from a more favorable node, having the value of a_2, to a less favorable node, having the value of a_1. For the case of $a_1 \geq a_2$, we might as well set $w_{ij} = \frac{a_1}{a_2} \geq 1$ to encourage a user to go from a less favorable node to a more favorable. However, if w_{ij} is too large, some certain product groups may dominate and prevent the other product groups from being discriminated and properly recommended. Therefore, we set a cap

limit, 1, on the value of w_{ij} in this case. When the correlation graph is completed, we have the following correlation matrix:

$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1q} \\ w_{21} & w_{22} & \dots & w_{2q} \\ \vdots & \vdots & \vdots & \vdots \\ w_{q1} & w_{q2} & \dots & w_{qq} \end{bmatrix} \tag{24}$$

which is a q by q matrix. Each column of $\mathbf{W}$ is then normalized, i.e.,

$$\begin{aligned} Q_j &= \sum_{k=1}^{q} w_{kj}, \\ w_{ij} &= \frac{w_{ij}}{Q_j},\ 1 \le i \le q \end{aligned}$$

for $1 \le j \le q$.

4.5. Step 4: Random Walks

In this random walks step, a series of random walks [48, 38, 17, 49] are performed. Consider any user u_i, $1 \le i \le N$. Let $\mathbf{V}_i(0)$ be

$$\mathbf{V}_i(0) = \begin{bmatrix} 1/q & 1/q & \dots & 1/q \end{bmatrix}^T \tag{25}$$

which is a vector of size q. The following operation

$$\mathbf{V}_i(t+1) = \alpha \mathbf{W}\mathbf{V}_i(t) + (1-\alpha)\mathbf{B}_i^T \tag{26}$$

is performed repeatedly for $t = 0, 1, 2, \dots$ until convergence is reached. Note that $\mathbf{W}$ is the matrix of Eq.(24), $\mathbf{B}_i$ is the vector of Eq.(21), and $\alpha \in [0, 1]$ is a user-defined constant. As in [17], α is chosen to be 0.85 and Eq.(4) is applied 20 times to reach a reasonable convergence for each user. Let $\mathbf{V}_i$ be the vector after convergence, which has q components. Then $\mathbf{V}_i$ is the derived preference list of product groups for user u_i, $1 \le i \le N$.

4.6. Step 5: Re-Transformation

Each vector $\mathbf{V}_i$, $1 \le i \le N$, obtained for user u_i in the previous step contains q values since q product groups are involved. However, we are interested in recommending individual products, instead of product groups, to each user. Therefore, we have

to transform $\mathbf{V}_i$ to $\mathbf{S}_i$ which contains preferences of individual products. Recall that, from Eq.(19), the membership degrees of $\mathbf{x}_j$, $1 \leq j \leq M$, in G_1, G_2, ..., and G_q are $t_{j1} = \mu_{G_1}(\mathbf{x_j})$, $t_{j2} = \mu_{G_2}(\mathbf{x_j})$, ..., and $t_{jq} = \mu_{G_q}(\mathbf{x_j})$, respectively. First, we normalize each column of $\mathbf{T}$ in Eq.(18), such that

$$\begin{aligned} Q_k &= \sum_{j=1}^{M} t_{jk}, \\ t_{jk} &= \frac{t_{jk}}{Q_k},\ 1 \leq j \leq M \end{aligned}$$

for $1 \leq k \leq q$. For each row, this calculates the proportion product p_j contributes to each product group. Then we have

$$S_i[j] = t_{j1}V_i[1] + t_{j2}V_i[2] + t_{j3}V_i[3] + \ldots + t_{jq}V_i[q] \tag{27}$$

where $S_i[j]$ is the jth component of $\mathbf{S}_i$ and $V_i[k]$, $1 \leq k \leq q$, is the kth component of $\mathbf{V}_i$. Note that t_{jk} is the proportion product p_j contributes to product group g_k and $V_i[k]$ indicates the preference of product group g_k for user u_i. Therefore, $t_{jk}V_i[k]$ is the preference of product p_j for user u_i in terms of product group g_k. Summing up together over all the product groups, as shown in Eq.(27), results in the preference of product p_j for user u_i. Eventually, we end up with the predicted preference list $\mathbf{S}_i$ of products for user u_i. Like ItemRank, the products can be recommended to user u_i in the order according to the magnitudes of the elements in $\mathbf{S}_i$.

4.7. Complexity Analysis

We give a rough analysis on the computational cost of our approach here. In the user labeling step, we have to calculate the similarity between each user and every existing cluster. Recall that N is the number of users, M is the number of products, each user has M components, and z is the number of class labels. The time complexity of this step is $O(NzM)$. In the dimensionality reduction step, we have to compute the similarity between each feature pattern and every existing cluster. Since the number of feature patterns is M, the number of product groups is q, and each feature pattern contains z components, the time complexity of the dimensionality reduction step is $O(Mqz)$. In the correlation graph creation step, each weight w_{ij}, $1 \leq i, j \leq q$, of

Table 1: A user-product rating matrix **R**

	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	p_{11}	p_{12}
u_1	5	0	2	0	5	2	3	0	4	0	0	5
u_2	0	4	0	3	0	3	3	0	2	0	4	0
u_3	4	0	2	0	4	0	2	2	3	0	0	4
u_4	3	4	0	4	0	0	3	0	2	0	4	0
u_5	0	2	5	3	0	4	0	5	0	4	2	0
u_6	5	0	0	0	4	3	0	2	4	2	0	5
u_7	0	4	0	3	2	2	3	0	0	0	3	0
u_8	2	3	4	0	0	4	0	5	0	4	0	0
u_9	4	0	2	2	4	0	0	0	3	0	0	5
u_{10}	0	2	4	0	0	3	0	4	0	4	2	2

Eq.(22) has to be computed. Therefore, the complexity of this step is $O(q^2)$. In the random walks step, Eq.(26) has to be iteratively computed. For each iteration, $\mathbf{WV}_i(t)$ is computed, requiring q^2 multiplications. Since 20 iterations are performed before convergence, the complexity of this step is $O(q^2)$ for a user and $O(Nq^2)$ for all the users. Finally, for a given user, Eq.(27) has to be computed M times, each time involving q multiplications and $q-1$ additions. The complexity of this step is $O(NqM)$ for all the users. Therefore, the time complexity of our approach to recommend products to all the users is $O(NzM)+O(Mqz)+O(q^2)+O(Nq^2)+O(NqM)$. Since $z < N$, the complexity can be simplified approximately to $O(NzM + Nq^2 + NqM)$.

5. Example

Here we give an example to illustrate our approach. Consider the user-product rating matrix **R** shown in Table 1. Note that there are 10 users, $N = 10$, 12 products, $M = 12$, and 65 ratings provided by the users. For example, user u_1 has provided 7 ratings to products p_1, p_3, p_5, p_6, p_7, p_9, and p_{12}, respectively. In the user-labeling step, class label c_1 is assigned to users u_1, u_3, u_6, and u_9, class label c_2 is assigned to users u_2, u_4, and u_7, while class label c_3 is assigned to users u_5, u_8, and u_{10}. Therefore, we

Table 2: Feature patterns in the second step

$\mathbf{x}_1 = \langle 0.783, 0.130, 0.087 \rangle$	$\mathbf{x}_2 = \langle 0.000, 0.632, 0.368 \rangle$
$\mathbf{x}_3 = \langle 0.316, 0.000, 0.684 \rangle$	$\mathbf{x}_4 = \langle 0.133, 0.667, 0.200 \rangle$
$\mathbf{x}_5 = \langle 0.895, 0.105, 0.000 \rangle$	$\mathbf{x}_6 = \langle 0.238, 0.238, 0.524 \rangle$
$\mathbf{x}_7 = \langle 0.357, 0.643, 0.000 \rangle$	$\mathbf{x}_8 = \langle 0.222, 0.000, 0.778 \rangle$
$\mathbf{x}_9 = \langle 0.778, 0.222, 0.000 \rangle$	$\mathbf{x}_{10} = \langle 0.143, 0.000, 0.857 \rangle$
$\mathbf{x}_{11} = \langle 0.000, 0.733, 0.267 \rangle$	$\mathbf{x}_{12} = \langle 0.905, 0.000, 0.095 \rangle$

have $z = 3$. In the dimensionality reduction step, we calculate the feature patterns as shown in Table 2. After applying the SCC algorithm, three clusters G_1, G_2, and G_3 are obtained. The reducing matrix **T** is

$$\mathbf{T} = \begin{bmatrix} 0.982 & 0.109 & 0.120 \\ 0.028 & 0.891 & 0.219 \\ 0.091 & 0.100 & 0.963 \\ 0.066 & 1.000 & 0.149 \\ 0.983 & 0.060 & 0.056 \\ 0.126 & 0.382 & 0.821 \\ 0.189 & 0.764 & 0.080 \\ 0.040 & 0.082 & 0.977 \\ 0.945 & 0.140 & 0.083 \\ 0.018 & 0.065 & 0.908 \\ 0.024 & 0.938 & 0.118 \\ 0.938 & 0.039 & 0.076 \end{bmatrix}. \tag{28}$$

By applying Eq.(20), we reduce **R** to **B** which is shown in Table 3. Note that there are three product groups, g_1, g_2, and g_3. Next, we proceed with the correlation graph creation step. By applying Eq.(22), we have a correlation graph, as shown in Figure 2, and the following normalized correlation matrix:

$$\mathbf{W} = \begin{bmatrix} 0.000 & 0.440 & 0.497 \\ 0.504 & 0.000 & 0.503 \\ 0.496 & 0.560 & 0.000 \end{bmatrix}. \tag{29}$$

Table 3: The reduced rating matrix **B**

	g_1	g_2	g_3
u_1	19.294	4.855	5.395
u_2	3.236	14.031	4.666
u_3	15.086	3.144	5.292
u_4	5.872	14.212	2.711
u_5	1.529	9.353	17.738
u_6	17.806	2.979	7.763
u_7	3.162	12.549	3.673
u_8	3.186	5.487	16.551
u_9	15.697	3.491	3.552
u_{10}	2.952	5.870	14.682

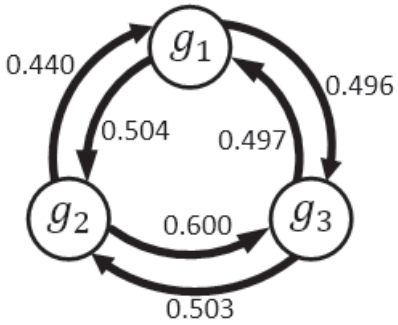


Figure 2: Correlation graph.

In the random walks step, we apply Eq.(26) iteratively for each user. For instance, after convergence, we have $\mathbf{V}_9 = \begin{bmatrix} 0.395 & 0.432 & 0.440 \end{bmatrix}^T$. Finally, in the re-transformation step, we transform $\mathbf{V}_i$ to $\mathbf{S}_i$, $1 \leq i \leq 10$, by Eq.(27). For example, $\mathbf{V}_9$ is transformed to $\mathbf{S}_9$ for user u_9. The components of $\mathbf{S}_9$ are shown in Table 4. Since p_6 possesses the biggest preference value, 0.126, in $\mathbf{S}_9$, p_6 is the first recommendation to user u_9, followed by p_4, p_3, etc.

Table 4: The derived $\mathbf{S}_9$ for user u_9

p_1	p_2	p_3	p_4	p_5	p_6
0.109	0.108	0.110	0.115	0.099	0.126
p_7	p_8	p_9	p_{10}	p_{11}	p_{12}
0.097	0.105	0.106	0.095	0.102	0.095

6. Experimental Results

To evaluate the performance of our proposed self-constructing clustering (SCC) based approach, we conduct a set of experiments on several benchmark data sets. For convenience, we call our approach SCC in the remaining of this section. We also compare our SCC approach with some other collaborative filtering recommender systems. Three metrics are adopted for comparison on recommendation accuracy, degree of agreement (DOA) [17], mean absolute error [50], and root mean squared error (RMSE) [33, 50]. A 5-fold cross validation is adopted for our experiments. In each experiment, the entries in a data set are split randomly into 5 different subsets. Then 5 runs are performed. Each time, four of the 5 subsets are used for training and the remaining one is used for testing. The results of the 5 runs are then averaged. Let $\mathcal{P}$ be the set of all products, $\mathcal{L}_i$ be the set containing the products user u_i has rated in the training set, and $\mathcal{T}_i$ be the set containing the products user u_i has rated in the testing set. It is required that none of $\mathcal{L}_i$ is empty, i.e., $\mathcal{L}_i \neq \emptyset, 1 \leq i \leq N$. DOA is defined by

$$\text{DOA} = \frac{\sum_{u_i \in \mathcal{U}} \text{DOA}_i}{|\mathcal{U}|} \tag{30}$$

where $\mathcal{U}$ is the set of all the users, $|\mathcal{U}|$ is the size of $\mathcal{U}$, and DOA_i measures for user u_i the percentage of product pairs ranked in the correct order with respect to the total number of product pairs. DOA_i is computed as follows. Let $check\_order_i$ be defined by

$$check\_order_i(p_j, p_k) = \begin{cases} 1, \text{ if } PP_i^j \geq PP_i^k \\ 0, \text{ otherwise} \end{cases} \tag{31}$$

where PP_i^j and PP_i^k indicate the predicted preferences for user u_i of products p_j and p_k, respectively. Then DOA_i is formulated as

$$\text{DOA}_i = \frac{\sum_{p_j \in \mathcal{T}_i \wedge p_k \in \mathcal{NW}_i} check\_order_i(p_j, p_k)}{|\mathcal{T}_i| \times |\mathcal{NW}_i|} \tag{32}$$

where

$$\mathcal{NW}_i = \mathcal{P} - (\mathcal{L}_i \cup \mathcal{T}_i) \tag{33}$$

which is the set of products user u_i has neither rated in the training set nor in the testing set. Note that a high DOA value indicates the superiority of a recommender system. An ideal recommender system results in a 100% DOA, i.e., $\text{DOA} = 1.0$. To compute MAE or RMSE, we have to convert predicted preferences to corresponding predicted scores. Let $\hat{r}_{ij}$ be the predicted score corresponding to the predicted preference of product p_j for user u_i. Then $\hat{r}_{ij}$ is computed by

$$\hat{r}_{ij} = r_{a,i} + \frac{\sum_{k=1 \wedge k \neq i}^{N} [Sim(u_i, u_k) \times (r_{kj} - r_{a,k})]}{\sum_{k=1 \wedge k \neq i}^{N} Sim(u_i, u_k)} \tag{34}$$

where $r_{a,i}$ is the average of the ratings in $\mathcal{L}_i$, $r_{a,k}$ is the average of the ratings in $\mathcal{L}_k$, r_{kj} is the rating for product j in $\mathcal{L}_k$, and $Sim(u_i, u_k)$ indicates the similarity between user u_i and user u_k defined by the Cosine of the predicted preference lists $\mathbf{S}_i$ and $\mathbf{S}_k$:

$$Sim(u_i, u_k) = \frac{\mathbf{S}_i \cdot \mathbf{S}_k}{\sqrt{\mathbf{S}_i \cdot \mathbf{S}_i} \sqrt{\mathbf{S}_k \cdot \mathbf{S}_k}} \tag{35}$$

where $\cdot$ is the inner product operator for vectors. Then MAE and RMSE are defined as follows:

$$\text{MAE} = \frac{\sum_{i=1}^{N} \sum_{j=1 \wedge r_{ij} \in \mathcal{T}_i}^{M} |r_{ij} - \hat{r}_{ij}|}{\sum_{i=1}^{N} |\mathcal{T}_i|}, \tag{36}$$

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N} \sum_{j=1 \wedge r_{ij} \in \mathcal{T}_i}^{M} (r_{ij} - \hat{r}_{ij})^2}{\sum_{i=1}^{N} |\mathcal{T}_i|}}. \tag{37}$$

Apparently, a low MAE or RMSE value indicates the superiority of a recommender system. An ideal recommender system results in no errors, i.e., MAE $= 0$ and or RMSE $= 0$. Note that in each of the following experiments, the predicted preference lists are derived from the training set and all DOA, MAE, and RMSE are measured on the testing set. Also, no overlapping exists between the training set and the testing set in any experiment.

6.1. Data Sets

Five data sets, MovieLens, Yahoo Movie, Amazon Video, BookCrossing, and Epinions, are used for the experiments. The MovieLens data set [51] was made publicly available by GroupLens Research which operates a movie recommender. It contains 943 users, 1,682 products (movies), and 100,000 user ratings. In other words, $N = 943$ and $M = 1,682$. Each entry in the data set is represented as a triple (u_i, p_j, r_{ij}) where $r_{ij} \in \{1, 2, 3, 4, 5\}$. Therefore, there are 100,000 such entries in the MovieLens data set. The Yahoo Movie data set contains the Yahoo! Movies community's preferences for various movies [52]. It contains 7,642 users, 11,916 products, and 221,367 user ratings. Each entry in the data set is represented as a triple (u_i, p_j, r_{ij}) where $r_{ij} \in \{1, 2, 3, 4, 5\}$. The Amazon Video data set contains a small fraction of the Amazon Instant video products review which spans s period of 18 years including 143.7 million reviews up to July 2014 [53]. It contains 2,978 users, 31,102 products, and 99,816 user ratings. Each entry in the data set is represented as a triple (u_i, p_j, r_{ij}) where $r_{ij} \in \{1, 2, 3, 4, 5\}$. The BookCrossing data set is a fraction of the original which was collected by Cai-Nicolas Ziegler in a 4-week crawl from the BookCrossing company [54]. It contains 4,981 users, 9,846 products (books), and 109,018 users ratings. Each entry in the data set is represented as a triple (u_i, p_j, r_{ij}) where $r_{ij} \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$. The Epinions data set was collected by Paolo Massa in a 5-week crawl from the Epinion.com web site [55]. It contains 2,322 users, 4,571 products, and 136,984 user ratings. Each entry in the data set is represented as a triple (u_i, p_j, r_{ij}) where $r_{ij} \in \{1, 2, 3, 4, 5\}$. The characteristics of these five data sets are summarized in Table 5

Table 5: Characteristics of the data sets used for experiments

	# of users	# of products	# of ratings
MovieLens	943	1682	100000
Yahoo Movie	7642	11916	221367
Amazon Video	2978	31102	99816
BookCrossing	4981	9846	109018
Epinions	2322	4571	136984

6.2. Results and Discussion

We show the effectiveness, both accuracy and efficiency, of our approach, SCC, by comparing it with some other collaborative filtering methods, including ItemRank [17], iExpand [30], PRM2 [33], BiFu [34], and ICRRS [36]. As described earlier, ItemRank does not employ any dimensionality reduction technique at all, iExpand uses LDA to group the products into clusters, PRM2 applies SVD to produce a low-dimensional representation of the original user-product space, BiFu applies K-means to cluster the users and products, and ICRRS uses a reduction technique which decouples credibility assessment of the cast evaluations from the ranking itself. For a fair comparison, we wrote programs for ItemRank, iExpand, PRM2, BiFu, ICRRS, and SCC. All the programs were written in C++ in Visual Studio 2013. We used a computer with Intel(R) Core(TW) i7-4790K CPU, 4.00GHz, 32GB of RAM, and 64 bits windows 7 to run the programs. For the sake of fairness, we used the same number of clusters for grouping the users or products in SCC, iExpand, and BiFu. Since LDA and K-means require the number of clusters to be decided in advance, for each case, we ran SCC first to obtain a desired number of clusters and this number was then used for clustering in iExpand and BiFu. Table 6 shows the number of clusters the involved users and products are partitioned into for SCC, iExpand, and BiFu. For example, the products of the MovieLens data set are partitioned into 25 clusters for iExpand, BiFu, and SCC, and the users are partitioned into 5 clusters for BiFu and SCC.

Table 7, Table 8, and Table 9 show comparisons on MAE, RMSE, and DOA, respectively, among ItemRank, iExpand, PRM2, BiFu, ICRRS, and SCC. Note that in

Table 6: The number of clusters for the involved users and products

	Products (iExpand, BiFu, SCC)	Users (BiFu, SCC)
MovieLens	25	5
Yahoo Movie	24	7
Amazon Video	26	14
BookCrossing	56	24
Epinions	73	33

Table 7: Comparisons among different methods on MAE for five data sets

	ItemRank	iExpand	PRM2	BiFu	ICRRS	SCC
MovieLens	**0.757**	0.836	**0.757**	0.782	1.188	**0.757**
Yahoo Movie	**0.753**	0.826	0.754	0.773	1.263	0.754
Amazon Video	**1.061**	1.080	**1.061**	1.071	1.900	**1.061**
BookCrossing	1.210	**1.192**	1.205	1.250	1.680	1.206
Epinions	**0.804**	0.952	0.805	0.866	1.244	0.805

Table 8: Comparisons among different methods on RMSE for five data sets

	ItemRank	iExpand	PRM2	BiFu	ICRRS	SCC
MovieLens	0.960	1.042	**0.959**	1.013	1.408	0.960
Yahoo Movie	**1.038**	1.114	1.039	1.215	1.522	1.039
Amazon Video	1.352	**1.334**	1.345	1.433	2.354	1.345
BookCrossing	1.595	**1.566**	1.587	1.722	2.096	1.588
Epinions	**1.044**	1.196	**1.044**	1.191	1.464	**1.044**

Table 9: Comparisons among different methods on DOA for five data sets

	ItemRank	iExpand	PRM2	BiFu	ICRRS	SCC
MovieLens	0.853	0.730	0.615	0.583	0.704	**0.976**
Yahoo Movie	0.939	0.697	0.567	0.542	0.512	**0.979**
Amazon Video	0.652	0.721	0.506	0.516	0.495	**0.841**
BookCrossing	0.715	0.655	0.509	0.515	0.516	**0.989**
Epinions	0.746	0.576	0.494	0.494	0.485	**0.907**

Table 10: Comparisons among different methods on efficiency (sec) for five data sets

	ItemRank	iExpand	PRM2	BiFu	ICRRS	SCC
MovieLens	43.83	95.76	13.65	30.67	1.26	**0.26**
Yahoo Movie	17359.42	4924.84	33.59	5399.62	156.86	**7.47**
Amazon Video	43678.72	1331.25	16.34	13777.72	56.87	**8.99**
BookCrossing	9131.29	800.01	16.30	2946.20	29.69	**9.48**
Epinions	837.08	513.24	20.72	487.90	5.82	**3.20**

these tables, the value obtained by the best method for each case is shown in boldface. For MAE and RMSE, the smaller the value a method obtains, the better the method performs. On the contrary, for DOA, the larger the value a method obtains, the better the method performs. As can be seen, ItemRank and SCC perform pretty well in terms of MAE and RMSE for all the five data sets. For example, both ItemRank and SCC have the lowest value, 1.061, in MAE for the Amazon Video data set, and both have the lowest value, 1.044, in RMSE for the Epinions data set. PRM2 performs pretty well in MAE and RMSE for most of the data sets, and so does iExpand for some of the data sets. As for DOA, SCC performs best for all the five data sets. For example, SCC has 0.979, which is the highest among the methods, in DOA for the Yahoo Movie data set. Table 10 shows comparisons on execution time, in seconds, among different methods. We can see that SCC runs much faster than the other methods. Note that in ItemRank, the dimensionality involved is the number of products, M, so the size of the correlation matrix used in Eq.(4) is $M \times M$. For example, the size is 31102×31102 for the Amazon Video data set. Handling this large matrix consumes a lot of time. As a result,

Table 11: Breakdown of the execution time for ItemRank

	size	CGC	RW	total (sec)
MovieLens	1682×1682	1.25	42.58	43.83
Yahoo Movie	11916×11916	147.29	17212.13	17359.42
Amazon Video	31102×31102	699.89	42978.83	43678.72
BookCrossing	9846×9846	47.38	9083.91	9131.29
Epinions	4571×4571	5.14	831.95	837.08

ItemRank takes 43678.72 seconds for Amazon Video, as shown in the table. Instead, SCC applies the self-constructing clustering to reduce the dimensionality, grouping the 31,102 products into only 26 product groups. So the size of the correlation matrix used in Eq.(26) is reduced to 26×26. As a result, SCC could run very fast, in 8.99 seconds, for the Amazon Video data set. BiFu and iExpand perform dimensionality reduction by K-means and LDA, respectively, which are heavily time-consuming. Therefore, iExapnd and BiFu run much slower than SCC. For example, iExpand takes 1331.25 seconds and BiFu takes 13777.72 seconds, while SCC only takes 8.99 seconds, for the Amazon Video data set. PRM2 and ICRRS employ different reduction techniques. However, they still run slower than SCC, as shown in Table 10.

The correlation matrix $\mathbf{W}$ in Eq.(4) has the size of $M\times M$ and that in Eq.(26) has the size of $q\times q$. Therefore, intuitively, SCC might run $(\frac{M}{q})^2$ times faster than ItemRank. Consider MovieLens as an example. SCC might run $(\frac{1682}{25})^2 \approx 4500$ faster than ItemRank for MovieLens. However, from Table 10, SCC only runs $\frac{43.83}{0.18} \approx 244$ times faster than ItemRank. One reason is that the other steps involved also take time. Table 11 and Table 12 show the breakdown of the execution time for ItemRank and SCC, respectively. Note that the size column in these tables indicates the size of the correlation matrix involved. Recall that ItemRank has two steps: correlation graph creation (CGC) and random walks (RW), and SCC has five steps: user labeling (UL), dimensionality reduction (DR), correlation graph creation (CGC), random walks (RW), and re-transformation (RT). Consider the case of MovieLens. In the total 43.83 seconds, ItemRank spent 1.25 seconds in the CGC step and 42.58 seconds in the RW

Table 12: Breakdown of the execution time for SCC

	size	UL	DR	CGC	RW	RT	total (sec)
MovieLens	25×25	0.03	0.07	0.00	0.05	0.11	0.26
Yahoo Movie	24×24	1.94	3.59	0.01	0.22	1.70	7.47
Amazon Video	26×26	2.45	4.48	0.00	0.11	1.93	8.99
BookCrossing	56×56	1.88	4.21	0.03	0.80	2.56	9.48
Epinions	73×73	0.50	1.24	0.03	0.65	0.79	3.20

step. Among the total 0.26 seconds of SCC, 0.03 seconds was spent in the UL step, 0.07 seconds in the DR step, nearly 0.0 seconds in the CGC step, 0.05 seconds in the RW step, and 0.11 seconds in the RT step. If we look into the RW column in both tables, the ratio of these columns is $\frac{42.58}{0.05} \approx 850$ which is much bigger than 244. Note that in addition to the correlation matrix, there are other factors involved in Eq.(4) and Eq.(26). These factors account for the gap between 4,500 and 850 for the case of MovieLens.

Choosing an appropriate value for ρ in SCC is problem dependent and often should go through a trial and error process. As mentioned in Section 4, the choice of ρ may affect the result of our clustering algorithm. A bigger ρ results in smaller clusters and increases the number of clusters. Therefore, as ρ increases, the number of product groups, q, obtained in the dimensionality reduction step also increases. As analyzed in Section 4, the complexity of SCC is proportional to Nq^2 and NqM. Therefore, as ρ increases, the execution time also increases. For example, if we increase ρ a little bit for Yahoo Movie, q increases from 24 to 30. The execution time therefore increases from 7.47 seconds to 8.43 seconds. However, we have found that MAE, RMSE, and DOA are not sensitive to q or ρ. For Yahoo Movie, when q increases from 24 to 30, MAE, RMSE, and DOA almost keep unchanged.

7. Conclusion

In a collaborative filtering recommender system, such as ItemRank, products are regarded as features. However, there are usually quite a lot of products involved in

E-commerce, and it would be very inefficient if every product needs to be considered before making recommendations. We have presented an approach which applies a self-constructing clustering algorithm to reduce the dimensionality related to the number of products. As a result, the processing time by our approach for making recommendations is much reduced. Experimental results have shown that the efficiency of the recommender system is greatly improved without compromising the recommendation quality.

Acknowledgments

This work was supported by the Ministry of Science and Technology under the grants MOST-103-2221-E-110-047-MY2 and MOST-104-2221-E-110-052-MY2, by "Aim for the Top University Plan" of the National Sun Yat-Sen University and Ministry of Education, and by a grant jointly sponsored by National Sun Yat-Sen University and National University of Kaohsiung. The authors would like to express their gratitude to the anonymous referees and Associate Editor for their constructive comments which greatly helped improve the quality of the manuscript.

References

[1] C. Porcel, J. Moreno, E. Herrera-Viedma, A multi-disciplinar recommender system to advice research resources in university digital libraries, Expert Systems with Applications 36 (2009) 12520–12528.

[2] F. Ricci, L. Rokach, B. Shapira, Introduction to recommender systems handbook, Recommender Systems Handbook, Springer, 2011.

[3] A. Gatzioura, M. Sànchez-Marrè, A case-based recommendation approach for market basket data, IEEE Intelligent Systems 30 (1) (2014) 20–27.

[4] M. Balabanović, Y. Shoham, Fab: Content-based, collaborative recommendation, Communications of ACM 40 (3) (1997) 66–72.

[5] P. Melville, R. J. Mooney, R. Nagarajan, Content-boosted collaborative filtering for improved recommendations, in: 18th National Conference on Artificial Intelligence, Edmonton, Canada, 2002, pp. 187–192.

[6] S.-L. Huang, Designing utility-based recommender systems for e-commerce: Evaluation of preference-elicitation methods, Electronic Commerce Research and Applications 10 (2011) 398–407.

[7] L. Liu, N. Mehandjiev, D.-L. Xu, Context similarity metric for multidimensional service recommendation, International Journal of Electronic Commerce 18 (1) (2013) 73–104.

[8] M. K. Condliff, D. D. Lewis, D. Madigan, Bayesian mixed-effects models for recommender systems, in: ACM SIGIR Workshop on Recommender Systems – Algorithms and Evaluation, 1999.

[9] I. Rish, An empirical study of the naive bayes classifier, in: International Joint Conferences on Artificial Intelligence(IJCAI) Workshop on Empirical Methods in AI, 2002, pp. 41–46.

[10] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, J. Riedl, Grouplens: An open architecture for collaborative filtering of netnews, in: ACM 1994 Conference on Computer Supported Cooperative Work, 1994.

[11] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, J. Riedl, Grouplens: Applying collaborative filtering to usenet news, Communications of the ACM 40 (3) (1997) 77–87.

[12] B. M. Sarwar, G. Karypis, J. A. Konstan, J. T. Riedl, Application of dimensionality reduction in recommender systems – a case study, in: ACM WEBKDD workshop, 2000.

[13] D. M. Blei, A. Y. Ng, M. I. Jordan, Latent dirichlet allocation, Journal of Machine Learning Research 3 (2003) 993–1022.

[14] J. L. Herlocker, J. A. Konstan, L. G. Terveen, J. T. Riedl, Evaluating collaborative filtering recommender systems, ACM Transactions on Information Systems 22 (1) (2004) 5–53.

[15] S. Funk, Netflix, A modified SVD algorithm tends to make a mess of sparsely observed movies or users, http://sifter.org/~simon/journal/20061211.html.

[16] F. Fouss, A. Pirotte, J.-M. Renders, M. Saerens, Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation, IEEE Transactions on Knowledge and Data Engineering 19 (3) (2007) 355–369.

[17] A. Pucci, M. Gori, M. Maggini, A random-walk based scoring algorithm applied to recommender engines, Lecture Notes in Computer Science – Advances in Web Mining and Web Usage Analysis 4811 (2007) 127–146.

[18] R. Salakhutdinov, A. Mnih, Probabilistic matrix factorization, in: Neural Information Processing Systems 20 (NIPS'07), 2008, pp. 1257–1264.

[19] N. N. Liu, M. Zhao, Q. Yang, Probabilistic latent preference analysis for collaborative filtering, in: ACM Conference on Information and Knowledge Management, 2009, pp. 759–766.

[20] K.-J. Kim, H. Ahn, Collaborative filtering with a user-item matrix reduction technique, International Journal of Electronic Commerce 16 (1) (2011) 107–128.

[21] M. Jiang, P. Cui, R. Liu, Q. Yang, Fei, W. Zhu, S. Yang, Social contextual recommendation, in: 21st ACM international conference on Information and knowledge management, 2012, pp. 45–54.

[22] C. Porcel, A. Tejeda-Lorente, M. Martinez, E. Herrera-Viedma, A hybrid recommender system for the selective dissemination of research resources in a technology transfer office, Information Sciences 184 (2012) 1–19.

[23] A. N. Nikolakopoulos, M. Kouneli, J. Garofalakis, A novel hierarchical approach to ranking-based collaborative filtering, Communications in Computer and Information Science – Engineering Applications of Neural Networks 384 (2013) 50–59.

[24] G. Guo, J. Zhang, D. Thalmann, N. Yorke-Smith, Leveraging prior ratings for recommender systems in e-commerce, Electronic Commerce Research and Applications 13 (2014) 440–455.

[25] H. Cui, M. Zhu, Collaboration filtering recommendation optimization with user implicit feedback, Journal of Computational Information Systems 10 (14) (2014) 5855–5862.

[26] B. M. Sarwar, G. Karypis, J. Konstan, J. Riedl, Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering, in: 5th International Conference on Computer and Information Technology, 2002.

[27] G.-R. Xue, C. Lin, Q. Yang, W. Xi, H.-J. Zeng, Y. Yu, Z. Chen, Scalable collaborative filtering using cluster-based smoothing, in: ACM SIGIR Conference, 2005.

[28] K. Honda, A. Notsu, H. Ichihashi, Collaborative filtering by user-item clustering based on structural balancing approach, International Journal of Computer Science and Network Security 8 (12) (2008) 190–195.

[29] Q. Ba, X. Li, Z. Bai, Clustering collaborative filtering recommendation system based on SVD algorithm, IEEE International Conference on Software Engineering and Service Science (2013) 963–967.

[30] Q. Liu, E. Chen, H. Xiong, C. H. Q. Ding, J. Chen, Enhancing collaborative filtering by user interest expansion via personalized ranking, IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics 42 (1) (2012) 218–233.

[31] Y. Cai, H. fung Leung, Q. Li, H. Min, J. Tang, J. Li, Typicality-based collaborative filtering recommendation, IEEE Transactions on Knowledge and Data Engineering 26 (3) (2014) 766–779.

[32] M. Sarwat, J. J. Levandoski, A. Eldawy, M. F. Mokbel, LARS*: An efficient and scalable location-aware recommender system, IEEE Transactions On Knowledge And Data Engineering 26 (6) (2014) 1384–1399.

[33] X. Qian, H. Feng, G. Zhao, T. Mei, Personalized recommendation combining user interest and social circle, IEEE Transactions on Knowledge and Data Engineering 26 (7) (2014) 1763–1777.

[34] D. Zhang, C.-H. Hsu, M. Chen, Q. Chen, N. Xiong, J. Lloret, Cold-start recommendation using bi-clustering and fusion for large-scale social recommender systems, IEEE Transactions on Emerging Topics in Computing 2 (2) (2014) 239–250.

[35] J. MacQueen, Some methods for classification and analysis of multivariate observations, in: 5-th Berkeley Symposium on Mathematical Statistics and Probability, Vol. 1, 1967, pp. 281–297.

[36] M. Allahbakhsh, A. Ignjatovic, An iterative method for calculating robust rating scores, IEEE Transactions on Parallel and Distributed Systems 26 (2) (2015) 340–350.

[37] M. Gori, A. Pucci, ItemRank: A random-walk based scoring algorithm for recommender engines, in: IJCAI, 2007, pp. 2766–2771.

[38] Z. Gyöngyi, H. Garcia-Molina, J. Pedersen, Combating web spam with trustrank, in: 30th International Conference on Very Large Data Bases (VLDB), Morgan Kaufmann, 2004, pp. 576–587.

[39] S.-J. Lee, C.-S. Ouyang, A neuro-fuzzy system modeling with self-constructing rule generation and hybrid SVD-based learning, IEEE Transactions on Fuzzy Systems 11 (3) (2003) 341–353.

[40] J.-Y. Jiang, R.-J. Liou, S.-J. Lee, A fuzzy self-constructing feature clustering algorithm for text classification, IEEE Transactions on Knowledge and Data Engineering 23 (3) (2011) 335–349.

[41] N. Slonim, N. Tishby, The power of word clusters for text classification, in: 23rd European Colloquium on Information Retrieval Research (ECIR-01), 2001.

[42] R. Bekkerman, R. El-Yaniv, N. Tishby, Y. Winter, Distributional word clusters versus words for text categorization, The Journal of Machine Learning Research 3 (3) (2003) 1183–1208.

[43] H. Li, T. Jiang, K. Zhang, Efficient and robust feature extraction by maximum margin criterion, IEEE Transactions on Neural Networks 17 (1) (2006) 157–165.

[44] J. Yan, B. Zhang, N. Liu, S. Yan, Q. Cheng, W. Fan, Q. Yang, W. Xi, Z. Chen, Effective and efficient dimensionality reduction for large-scale and streaming data preprocessing, IEEE Transactions on Knowledge and Data Engineering 18 (3) (2006) 320–333.

[45] V. S. Tseng, C.-P. Kao, A novel similarity-based fuzzy clustering algorithm by integrating PCM and mountain method, IEEE Transactions on Fuzzy Systems 15 (6) (2007) 1188–1196.

[46] Z. Zhang, H. Cheng, S. Zhang, W. Chen, Q. Fang, Clustering aggregation based on genetic algorithm for documents clustering, in: IEEE Congress on Evolutionary Computation, 2008, pp. 3156–3161.

[47] Z. Huang, W. Chung, H. Chen, A graph model for e-commerce recommender systems, Journal of the American Society for Information Science and Technology 55 (3) (2004) 259–274.

[48] D. Harel, Y. Koren, On clustering using random walks, Lecture Notes in Computer Science – Foundations of Software Technology and Theoretical Computer Science 2245 (2001) 18–41.

[49] H. Yildirim, M. S. Krishnamoorthy, A random walk method for alleviating the sparsity problem in collaborative filtering, in: 2008 ACM conference on Recommender systems, 2008, pp. 131–138.

[50] D. Wu, G. Zhang, J. Lu, A fuzzy preference tree-based recommender system for personalized business-to-business e-services, IEEE Transactions on Fuzzy Systems 23 (1) (2015) 29–43.

[51] Movielens dataset, http://www.grouplens.org/node/73#attachments.

[52] Yahoo movie dataset, https://www.yahoo.com/movies/.

[53] Amazon video dataset, http://www.amazon.com/.

[54] Bookcrossing dataset, http://www.bookcrossing.com/.

[55] Epinions dataset, http://www.epinions.com/.

Graphical Abstract

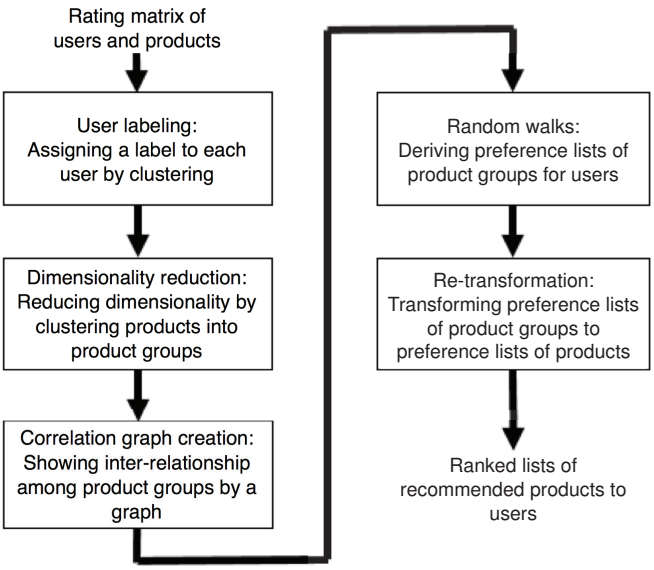


Figure 1: Overview of our approach.

Highlights of the Paper

- Developing an efficient collaborative filtering recommender system for E-commerce.
- Applying a self-constructing clustering algorithm to reduce the dimensionality related to the products.
- Reducing the huge correlation graph to a much smaller graph for faster computation.
- Demonstrating that efficiency is greatly Improved without degradation of the recommendation quality.