An Ordinal Regression Model with SVD Hebbian Learning for Collaborative Recommendation

TE-MIN CHANG[1], WEN-FENG HSIAO[2] AND WEI-LUN CHANG[3]
[1]*Department of Information Management*
National Sun-Yat Sen University
Kaohsiung, 804 Taiwan
[2]*Department of Information Management*
National Pingtung Institute of Commerce
Pingtung, 90004 Taiwan
[3]*Department of Business Administration*
Tamkang University
New Taipei City, 25137 Taiwan

The Internet is disseminating more and more information as it continues to grow. This large amount of information, however, can cause an information overload problem for users. Recommender systems to help predict user preferences for new information can ease users' mental loads. The model-based collaborative filtering (CF) approach and its variants for recommender systems have recently received considerable attention. Nonetheless, two issues should be carefully considered in practical applications. First, the data reliability of the rating matrix can affect the prediction performance. Second, most current models view the measurement scale of output classes as nominal instead of ordinal ratings. This study proposes a model-based CF approach that deals with both issues. Specifically, this approach uses the Hebbian learning rule to facilitate singular value decomposition in reducing noisy and redundant data, and employs support vector ordinal regression to build up the models. The results of the experiments conducted in this study show that the proposed approach outperforms other methods, especially under data of mild data sparsity and large-scale conditions. The feasibility of the proposed approach is justified accordingly.

Keywords: recommender systems; collaborative filtering; model-based CF; data reliability; ordinal scale

1. INTRODUCTION

The Internet is disseminating increasingly more information as it continues to grow. However, this large amount of information leads to an information overload problem, which prevents users from acquiring the online information they desire. To compensate for their insufficient information processing ability, users have adopted techniques such as information filtering to exclude a plenty of irrelevant information and reduce the workloads.

Recommender systems that employ information filtering techniques are a useful tool when users cannot describe their requirements precisely as keywords. Recommender systems adopt simple ideas in a manner similar to our daily decision-making process. Specifically, these systems recommend desired information by analysing users' preferences, or the preferences of like-minded people.

Filtering techniques in recommender systems are usually classified as content-based filtering and collaborative filtering. Collaborative filtering has recently received much attention because it only involves user ratings on items instead of well-structured content descriptions, which are harder to obtain. Collaborative filtering algorithms can be further grouped into two general classes: memory-based and model-based [1, 2]. Memory-based algorithms are heuristics that predict recommendation ratings for the active user based on the entire ratings collected in the rating matrix. Traditional user-based (calculating user correlation), item-based (calculating item correlation), and trust-based (adding trust component) CF approaches belong to this category. However, memory-based algorithms are subject to computational inefficiency with a great number of prediction demands, and generally cause scalability problems.

On the other hand, model-based algorithms make rating predictions based on the model learned from the collected rating matrix. They are computationally efficient once the model has been developed [3]. Several studies show that model-based algorithms can generate more accurate prediction results than their memory-based counterparts [4].

Two issues should be addressed in model-based CF algorithms though. First, input ratings may not be reliable, and noisy ratings can be detrimental to the learned model and the quality of the generated recommendations. Second, in current situations, models adopted in a machine learning framework are usually in the form of classifiers. Most classifiers use a nominal measurement scale of rating data and ignore the fact that they stand for the degree of user's preferences, which are of ordinal nature. Therefore, their performance may deteriorate because of classifier construction with inappropriate data manipulation.

This study thus proposes a model-based CF algorithm that considers issues of data reliability and data measurement scale. Specifically, this study adopts the Hebbian learning rule to facilitate singular value decomposition (SVD) of the sparse rating matrix to reduce noisy and redundant data. It then employs support vector ordinal regression (SVOR) to build prediction classifiers with data of ordinal scale. The proposed model-based CF approach is aimed to yield good recommendation results.

This paper is organized as follows. Section 2 reviews the relevant literature regarding this research. Section 3 illustrates the proposed approach. Section 4 presents three experiments conducted to justify the feasibility of the proposed approach. Finally, Section 5 presents concluding remarks and future work.

2. RELATED WORKS

2.1 Collaborative Filtering

Recommender systems emerged in the mid-1990s as an effective means to help users deal with information overload situations through providing personalized recommendations [2]. Researchers have developed various approaches in recommender systems. Balabanović and Shoham [5] classified these approaches into three categories: content-based, collaborative, and hybrid methods. Content-based methods recommend items that are similar to a given user's past experiences and preferences. Collaborative methods

recommend items that are highly rated by users whose characteristics similar to the given user. Finally, hybrid methods combine both approaches to providing better results.

Collaborative methods have recently received much attention. Many studies pointed out their superiority over content-based methods, which may suffer from content limitation and over-specification problems. The filtering algorithms adopted in collaborative methods generally fall into two classes: memory-based and model-based [1, 2]. Most of the earlier collaborative filtering algorithms are memory-based. These algorithms employ user-item ratings in a matrix form to make rating predictions. Typical memory-based algorithms find users that are most similar to the active user by calculating the correlation between both users. The unknown item ratings for an active user can be predicted by considering the item ratings from the active user's neighbours. On the other hand, model-based algorithms build a model from the collected user-item ratings for prediction purposes. Models are an abstraction of detailed user profiles, and can be in the form of clusters, Bayesian networks, classifiers, probabilistic relations. They are computationally efficient to make recommendations once the model is developed [3].

The *k*-means clustering method [6-8] is a simple model-based CF algorithm that is analogous to the memory-based CF with user similarity. However, this approach uses cluster profiles (the centroid of the cluster) instead of individual user profiles. The algorithm begins to assign the first *k* users as the centroids of *k* clusters. Each subsequent user is then assigned to one of the *k* clusters based on the minimum distance (highest similarity) to the *k* centroids. After all users are assigned to *k* clusters, each cluster centroid is adjusted based on the users in the cluster. The algorithm repeats this process until reaching a specified objective function (i.e., minimizing the intra-cluster variance). Recommendations for an unrated item *s* to an active user *c* are based on which cluster *c* belongs to and the rating regarding *s* contained in the centroid of this cluster. This approach alleviates the computational time for recommendation prediction because only one cluster profile is chosen each time in the prediction process.

Billsus and Pazzani [4] proposed a model-based approach that treats collaborative filtering as a classification task. The first step in their approach is to deal with missing values in the rating matrix by introducing Boolean features for each user. They employed singular value decomposition to reduce features, discard uninformative features, and account for feature interdependencies. This approach leaves only significant singular vectors as training examples. The final step was to employ a neural network to train those examples and build a prediction model.

2.2 Singular Value Decomposition Analysis

Reducing dimensionality techniques are often employed in the areas of data mining, information retrieval, image recognition, and recommender systems. The dimension reduction process can transform high-dimensional data into a lower dimension, where the features of data are represented.

Singular value decomposition (SVD) was first adopted by Deerwester et al. [9] to discover the underlying semantic relationships in text documents. SVD is a useful factoring matrix method that factors a matrix into a multiplication of a left singular vector matrix (orthonormal), a singular value matrix (diagonal), and a right singular vector ma-

trix (orthonormal). The first k largest singular value and the corresponding left and right singular vectors are retained to obtain a truncated SVD that is the closest rank-k approximation to the original matrix. Berry et al. [10] applied SVD to information retrieval areas and obtained less noisy data through SVD analysis to uncover the latent associations between terms and documents.

In the context of collaborative filtering, however, it is not possible to perform SVD decomposition directly using data in the rating matrix because there could be many missing values in the matrix. Before employing SVD, it is necessary to fill in or estimate these values. The most intuitive way is to fill zeros in those empty entries because the ratings are usually based on a 5-point Likert scale. However, this large amount of zeros can make SVD very biased toward unobserved values [11]. Sarwar et al. [12] attempted to tackle this issue by substituting overall means for missing values. Unfortunately, this could also make SVD biased toward the employed means. Zhang et al. [13] combined SVD with the expectation-maximization (EM) procedure to find a low-dimensional model that optimizes the log-likelihood of observed values. Nonetheless, this approach is computationally costly because it requires one SVD computation for each EM iteration before it converges.

Introducing a new numeric value for the missing value is generally inappropriate because this could conflate the new value and the observed ratings. Therefore, Gorrell [14] proposed an alternative approach that utilizes SVD with missing values. In that approach, SVD is performed based on the generalized Hebbian algorithm with only observed values. The significance of this approach lies in not only the SVD factorization of incomplete matrices (i.e., without filling in missing values in advance), but also the adoption of a simple learning rule (only a few lines of code are needed) with minimal memory requirements. This approach is highly accurate in collaborative filtering and scales easily to a matrix with billions of potential values [11]. Webb [15] (known as Simon Funk, his pseudonym) employed this concept in the Netflix contest (http://www.netflixprize.com). Webb expressed the learning rule in "two lines of code," as follows:

$$\mathit{userValue[user]} \mathrel{+}= \mathit{lrate} * \mathit{err} * \mathit{movieValue[movie]}; \tag{1}$$

$$\mathit{movieValue[movie]} \mathrel{+}= \mathit{lrate} * \mathit{err} * \mathit{userValue[user]}; \tag{2}$$

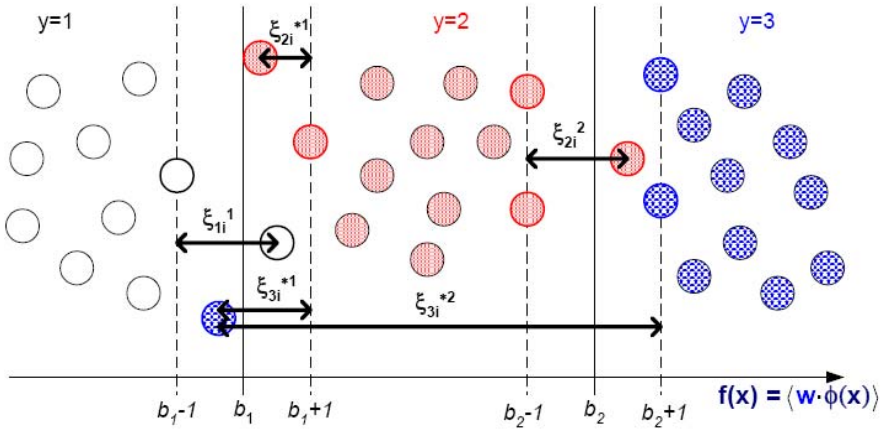
where *lrate* is the learning rate and *err* is the error between the predicted value and the actual value.

2.3 Support Vector Ordinal Regression

Classification is an important task in many research areas. Researchers have developed a variety of techniques to perform classification tasks. Among them, support vector machines (SVM) [16] have shown their power of classification accuracy over many other classifiers. For example, Miha et al. [17] showed that recommendation performance of SVM is better than that of other classification methods, even when data are sparse.

Unfortunately, typical SVMs can only solve classification problems with classes of nominal scales, which is not suitable for collaborative filtering tasks with ordinal output classes. Shashua and Levin [18] proposed a modified support vector method to deal with ordinal output classes referred to as ranking learning or ordinal regression. Their method finds an optimal separation distance together with penalty for errors subject to the deter-

mination of thresholds that represent parallel hyperplanes for classes (ordinal ranks). An example (a data point) contributes errors only at the threshold that separates its adjacent classes. Chu and Keerthi [19] extended this approach by adding implicit constraints on the ordinal inequalities of the thresholds. In their method, an example can contribute errors at all thresholds separating all classes. For example, in Fig. 1, the misclassified example of class 3 in b_1 would only contribute errors at threshold b_2 in Shashua and Levin's settings [18] whereas in Chu and Keerthi's approach [19], this example would also contribute errors at threshold b_1. Thus, all thresholds b_i's are ordered without explicit expression, and the ordinal regression performance is better.



(b_j's are thresholds, and ξ_{ki}^{j}, ξ_{ki}^{*j} are errors)
Fig. 1. Implicit constraints on the thresholds (Chu and Kerthi[19]).

3. PROPOSED APPROACH

This study proposes a model-based CF algorithm that deals with issues of data reliability and data scale. For data reliability issue, we employ the SVD analysis with the generalized Hebbian learning rule in the rating matrix with missing values. This approach avoids the potential biases of the generated recommendation results toward filling specific values. By doing so, not only redundant and noisy data are removed, but also a complete (approximate) rating matrix is generated by multiplying the resulting matrices together. For data scale issue, we employ the support vector ordinal regression (SVOR) model to classify data of ordinal ratings.

Figure 2 shows the process of the proposed approach, which includes three steps. Step 1 is to process the collected data using SVD with the generalized Hebbian learning rule. Step 2 transforms data from the approximate rating matrix (or combined with the original one) into examples. Finally, Step 3 builds SVOR-based models for each individual user, and adopts those models to makes recommendations for unrated items. The following section describes these steps.

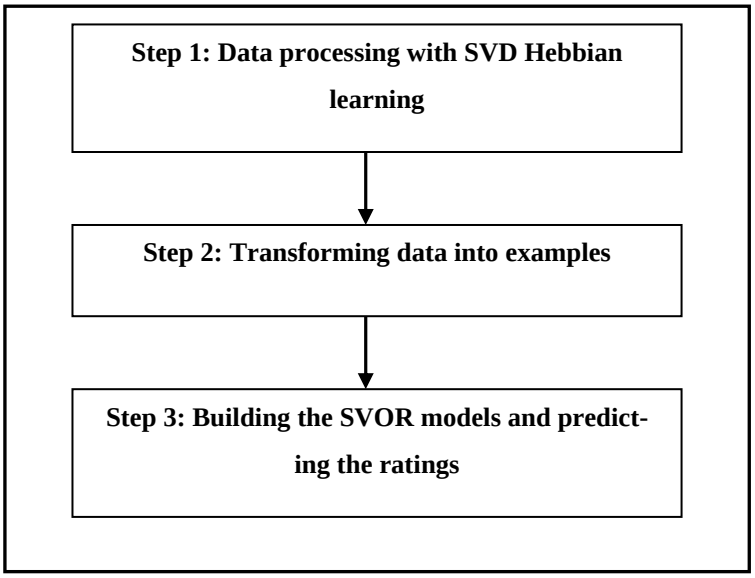


Fig. 2. Steps of the proposed approach.

3.1 Data Processing with SVD Hebbian Learning

The first step in our approach is to reduce possible noises and redundancy in the rating data to enhance their reliability. This study employs SVD with the Hebbian learning rule [14, 15, 20] for this purpose. The matrix factorization problem is formally stated as follows: Given a matrix $\boldsymbol{D}$, find singular vector matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ such that $\boldsymbol{U}\boldsymbol{V}^{\boldsymbol{T}} \approx \boldsymbol{D}$ (where the singular value matrix is absorbed into $\boldsymbol{U}$ and $\boldsymbol{V}$).

To apply the learning rule, the objective function was set to minimize the errors in the L^2 norm while preventing overfitting through the regularization that is commonly adopted in statistics and machine learning. The objective function is defined as

$$\underset{\boldsymbol{U},\,\boldsymbol{V}}{argmin} \; || \; \boldsymbol{D} - \boldsymbol{U}\boldsymbol{V}^{\boldsymbol{T}} \; ||^2 + \gamma(||\boldsymbol{U}||^2 + ||\boldsymbol{V}||^2) \tag{3}$$

where γ is the regularization parameter.

The Hebbian learning rule to estimate the first paired singular vectors of $\boldsymbol{D}$, $\boldsymbol{U_1}$ (first column vector of $\boldsymbol{U}$) and $\boldsymbol{V_1}$ (first column vector of $\boldsymbol{V}$) is

$$\boldsymbol{U_1} = \boldsymbol{U_1} + \eta\,[\boldsymbol{e}\boldsymbol{V_1} - \gamma\boldsymbol{U_1}] \qquad \boldsymbol{V_1} = \boldsymbol{V_1} + \eta\,[\boldsymbol{e}^{\boldsymbol{T}}\boldsymbol{U_1} - \gamma\boldsymbol{V_1}] \tag{4}$$

for every i, j such that $\boldsymbol{D_{ij}}$ exists. Here, η is the learning rate, and $\boldsymbol{e}$ is the error defined by $\boldsymbol{D} - \boldsymbol{U_1}V_1^T$. This is repetitively done until $\boldsymbol{U_1}$ and $\boldsymbol{V_1}$ converge. Figure 3 lists the pseudo code of the Rank 1 algorithm to find the first paired singular vectors ($\boldsymbol{u_1}$, $\boldsymbol{v_1}$) as the estimates of ($\boldsymbol{U_1}$, $\boldsymbol{V_1}$). Notice that the ith element in $\boldsymbol{u_1}$ and j^{th} element in $\boldsymbol{v_1}$ will be adjusted if the corresponding d_{ij} in $\boldsymbol{D}$ exists. Therefore, all elements in $\boldsymbol{u_1}$ and $\boldsymbol{v_1}$ will be adjusted as long as there are no zero rows and zero columns in $\boldsymbol{D}$.

As Gorrell[14] indicated, it is possible to remove the projection of the first paired singular vectors from $\boldsymbol{D}$, i.e., $\boldsymbol{D}_{rem} = \boldsymbol{D} - \boldsymbol{U_1}V_1^T$, and use the same Hebbian learning rule to

find the first paired singular vectors of $\boldsymbol{D}_{rem}$. This turns out to be exactly the second paired singular vectors of $\boldsymbol{D}$, $\boldsymbol{U_2}$, and $\boldsymbol{V_2}$. With this property, this procedure can be repeated k times until the significant k^{th} paired singular vectors of $\boldsymbol{D}$ are found. By putting singular vectors $\boldsymbol{U_i}$'s together into $\boldsymbol{U}$ and singular vectors $\boldsymbol{V_i}$'s together into $\boldsymbol{V}$, the multiplication of $\boldsymbol{U}$ and $\boldsymbol{V^T}$ becomes the low-rank (k) approximation of $\boldsymbol{D}$. Figure 4 lists the pseudo code of the Rank k algorithm for the complete learning algorithm which recursively calls the Rank 1 algorithm to find the successive paired singular vectors.

Rank 1 algorithm

```
procedure RANK-1(D)
   initialize u1, v1
   define η, γ
   repeat
      for each dij exists
         dij' = u1(i) × v1(j)
         u1(i) = u1(i) + η [(dij − dij') v1(j) − γ u1(i)]
         v1(j) = v1(j) + η [(dij − dij') u1(i) − γ v1(j)]
      end for
until stopping criterion met
return (u1, v1)
end procedure
```

Fig. 3. Hebbian learning SVD with the first paired singular vectors.

Rank *k* algorithm

```
procedure RANK-k (D)
      initialize   l = 1, U' = 0, V^T' = 0
define  D^l_rem = D
while l ≤ k
      (u, v) = RANK-1( D^l_rem )
      U'(:, l) = u, V^T'(l, :) = v^T
      D^l_rem = D^l_rem − uv^T
      l = l + 1
end while
```

```
D' = U' × V^T'
return D'
end procedure
```

Fig. 4. Hebbian learning SVD.

3.2 Transforming Data into Examples

The step above not only enhances data reliability, but also obtains a full approximate rating matrix by multiplying the singular vector matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ ($\boldsymbol{U}\boldsymbol{V}^{\boldsymbol{T}} = \boldsymbol{D}' \approx \boldsymbol{D}$). Intuitively, the rating results for the unrated ones without further process can be estimates of predicted recommendations already. Webb [15] adopted this approach in the Netflix contest. However, this study points out that model-based collaborative filtering is more robust because of its generalization ability.

This step transforms data from the approximate rating matrix (or combined with the original one) into examples to build the model. The following discussion illustrates how examples are transformed for each user. Suppose that there are four users and five items, with the original rating matrix and the approximate rating matrix shown in Tables 1 and 2, respectively.

Table 1. Original rating matrix.

	I1	**I2**	**I3**	**I4**	**I5**
U1	—	—	4	2	4
U2	2	2	3	—	1
U3	1	—	1	4	—
U4	3	2	—	—	4

Table 2. Rating matrix by SVD Hebbian learning.

	I1	**I2**	**I3**	**I4**	**I5**
U1	4	3	4	3	3
U2	2	2	3	2	2
U3	2	3	3	3	2

U4	3	3	2	2	3

To transform the examples for user 4, for instance, the system only considers data on items 1, 2, and 5 that are rated by user 4. An example to build the model contains input features that are ratings from users other than user 4, and an associated output that is the actual rating from user 4. This study employs two different approaches for the input features. The first one is to use results from Hebbian learning SVD solely, even when actual ratings exist. The second one is to use results from Hebbian learning SVD only for missing ratings. Consequently, Tables 3 and 4 list the corresponding examples transformed for user 4 under the two different considerations respectively (with values in bold font indicating the estimate ones).

Table 3. Examples for user 4.

	Examples		
	E1	**E2**	**E3**
Input features	**4**	**3**	**3**
	2	**2**	**2**
	2	**3**	**2**
Output	3	2	4

Table 4. Alternative examples for user 4.

	Examples		
	E1	**E2**	**E3**
Input features	**4**	**3**	4
	2	2	1
	1	**3**	**2**
Output	3	2	4

3.3 Building the SVOR Models and Predicting the Ratings

The transformed examples above are used to build support vector ordinal regression (SVOR) models [19]. This is a straightforward process using the open source code provided by the authors.

Finally, after the SVOR models are built for each individual user, they are used to predict the ratings of originally unrated items for the active user. In the previous instance, items 3 and 4 were not originally rated by user 4. Therefore, the trained SVOR model employs input features of I_3 (**4**, **3**, **3**) and I_4 (**3**, **2**, **3**) (or alternative input features of I_3 (4,

3, 1) and I_4 (2, **2**, 4)) to generate the predicted ratings. The SVOR output results could be different from current outputs (**2** and **2**, respectively), which are approximations using only the Hebbian learning SVD.

4. PERFORMANCE EVALUTION

This section presents three experiments and their results to examine the performance of the proposed model-based approach in collaborative filtering (CF). This section also compares the performance of three other approaches: traditional user-based CF [21], pure Hebbian learning SVD (use the resulting approximate rating matrix without building classifiers), and pure implicit SVOR (using means to substitute missing values without running Hebbian learning SVD). For ease of distinction, these are referred to as TCF, HSVD, and SVORIM, respectively, whereas our proposed approach is referred to as HSVD_SVORIM (using HSVD results for all input features) and alt_HSVD_SVORIM (using HSVD results only for missing input features).

4.1 Experimental Design

The experiments in this study adopt the MovieLens dataset (http://movielens.umn.edu) that is commonly applied in collaborative filtering for performance evaluation [22, 23]. MovieLens is a web-based movie recommender system provided by GroupLens Research group at the University of Minnesota. There are currently three datasets in MovieLens. The first dataset includes 100,000 ratings for 1682 movies by 943 users. The second dataset includes 1 million ratings for 3900 movies by 6040 users. The third dataset includes 10 million ratings and 100,000 tags for 10681 movies by 71567 users. Tags refer to user-generated metadata, the rating scale is from 1 to 5 stars, and each user rates at least 20 movies.

The experiments in this study use the first dataset because its size is tractable and operable. Furthermore, in SVORIM, the output classes of all training examples should cover all ordinal ranks (1, 2, 3, 4, and 5, in this case) for them to be learned and distinctly predicted. Therefore, for fair comparison, this study only considers users whose ratings cover the full rating range, with each rank occurring at least twice. This yields a large-scale dataset, called Data LS, with 76,337 ratings for 1674 movies by 522 users. The sparsity of Data LS is 0.912 (= 1 − 76337 / (522 × 1674)). To evaluate the validity of the proposed approach, three small datasets were also selected from Data LS to conduct experiments by manipulating the degrees of sparsity. Table 5 refers to these datasets as Data 1, Data 2, and Data 3, respectively.

This study adopts the mean absolute error (MAE) to measure recommender performance. The MAE calculates the average magnitude of error and compares the predicted rating results with real ratings without considering the sign direction. The MAE is defined as

$$MAE = \frac{\sum_{i=1}^{N} |p_i - q_i|}{N} \tag{5}$$

Table 5. Statistics of datasets employed.

	Data LS	Data 1	Data 2	Data 3
Number of users	522	174	174	174
Number of movies	1674	1416	1509	1409
Number of ratings	76337	19111	24507	27560
Degree of sparsity	0.912	0.922	0.907	0.888
Number of data	76897	19218	24419	27459

where p_i and q_i represent the predicted rating and real rating of the movie i, respectively, and N is the total number of movies predicted (i.e., the number of test data). Lower MAE values refer to better performance.

The evaluation scheme used in this study is the hold-out validation scheme, which randomly splits the data into a training set and a test set according to a certain percentage. Training data are used in the recommendation process to predict the ratings of unrated items for users, whereas test data are used to assess performance by comparing the predicted and actual results. For small datasets (Data 1 to Data 3), the ratio of training data to test data is 9:1, whereas for Data LS, the ratio is 8:2. Each case is run with two replications. Finally, all experiments were performed on a Dell server with a multiprocessor Intel Xeon CPU E5405 2GHZ, 2 gigabytes of memory, and the Linux (Fedora 11) platform.

4.2 Experiment I

The objective of experiment I is to set up the parameters employed in HSVD, which include learning rate η, regularization parameter γ, maximum number of learning iterations (ep), and rank k (dimensionality). Previous literature provides some guidance for parameters η and γ (e.g., Webb[15]). After some pilot runs, these parameters were set to be $\eta = 0.003$, $\gamma = 0.02$ for the small datasets (Data 1, Data 2, Data 3), and $\eta = 0.002$, $\gamma = 0.015$ for Data LS. The latter settings indicate a slower convergence process because of the large-scale data involved.

Next, we would like to determine the maximum number of learning iterations, and the rank k. For small datasets, ep was set to be 30, 40, or 50, and k ranged from 10 to 120 in increments of 10. Figures 5, 6, and 7 show the performance results for Data 1, Data 2, and Data 3, respectively (with two replications).

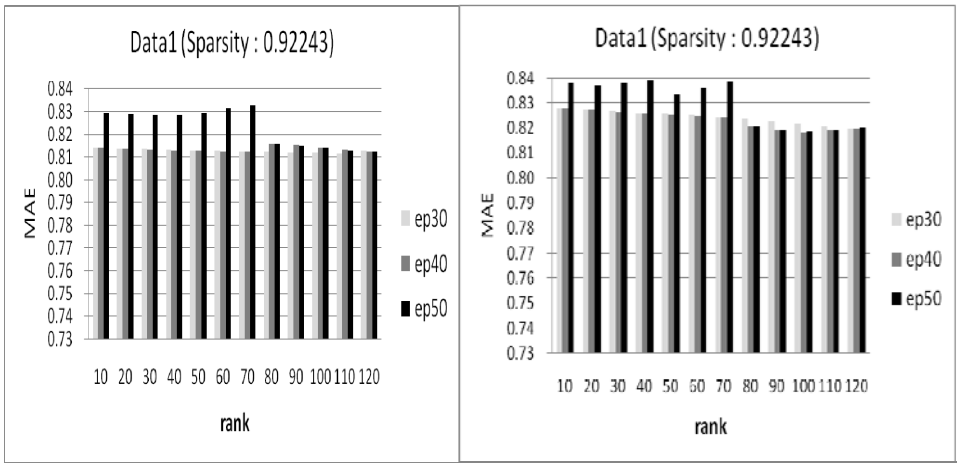


Fig. 5. Parameter settings for Data 1.

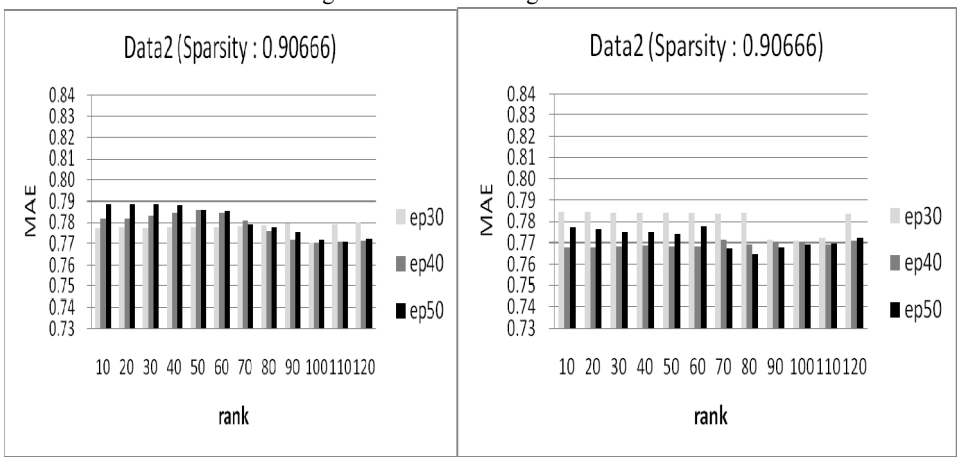


Fig. 6. Parameter settings for Data 2.

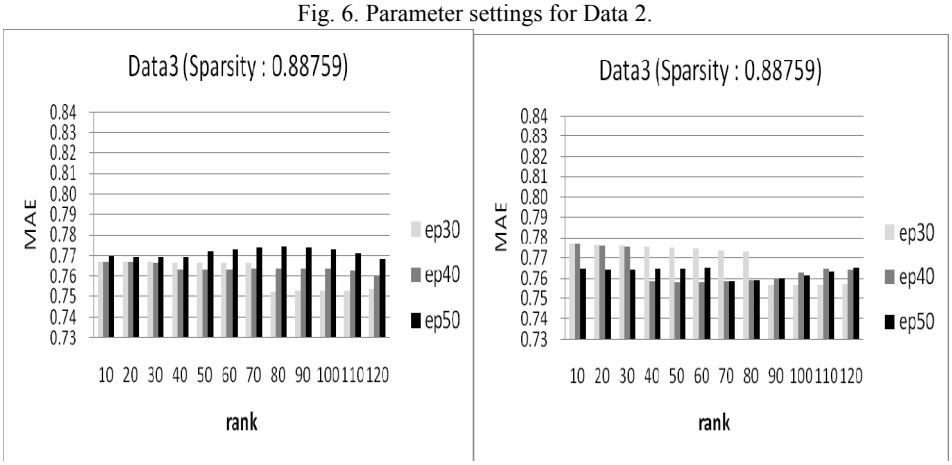


Fig. 7. Parameter settings for Data 3.

The figures above indicate that the performance of HSVD improves from Data 1 to Data 3. This is reasonable because the collaborative method yields better recommendation results with denser data. The performance of each dataset, however, differs under different parameter settings. The maximum number of learning iterations for Data 1 to Data 3 was set to be 50, 40, and 30, respectively, and the optimal rank *k* to be 100 for all cases. With the decrease of sparsity degrees from Data 1 to Data 3, the maximum number of iterations should also decrease in the same direction with possibly faster convergence. In contrast, rank *k* remained stable in the range of 90 to 110 for all three cases. Therefore, we simply set *k* to be 100.

Finally, we consider the parameter settings for Data LS. In this case, ep was set to be 90, 100, or 110, and *k* varies from 10 to 100 in increments of 10. The reason for setting ep higher is that a slower convergence is necessary to avoid results of local optima for large-scale data. On the other hand, it is not necessary to increase rank *k* proportionally because it only matters with singular values to represent significant components in the original matrix. Figure 8 shows the performance results. The maximum number of learning iterations required for Data LS was set to be 110, and the optimal rank *k* was 80 accordingly.

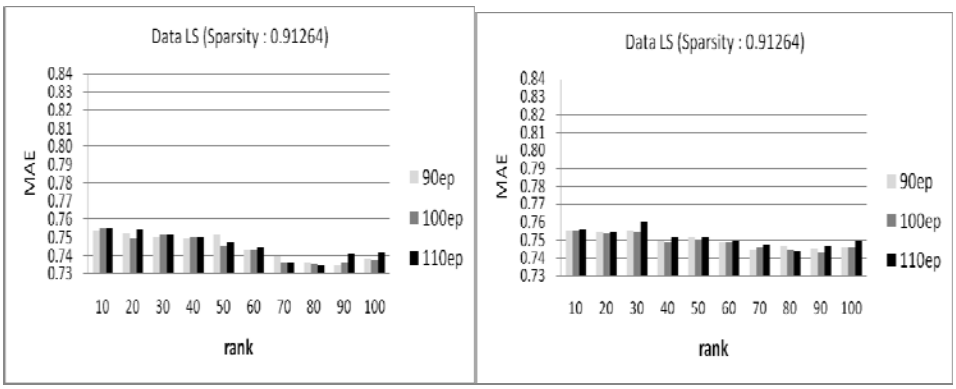


Fig. 8. Parameter settings for Data LS.

4.3 Experiment II

We compared the performances of HSVD_SVORIM and alt_HSVD_SVORIM with those of TCF, HSVD, and SVORIM by using the small datasets. Figures 9, 10, and 11 show the results for Data 1, Data 2, and Data 3, respectively.

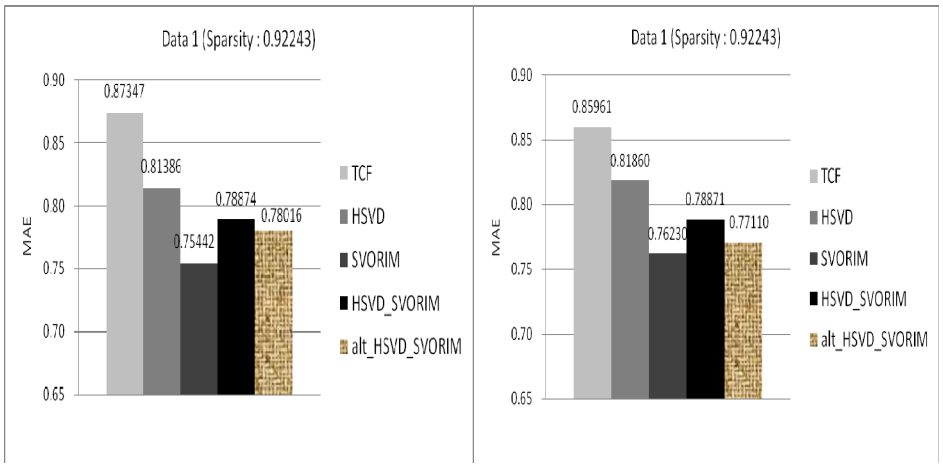


Fig. 9. Performance comparison for Data 1.

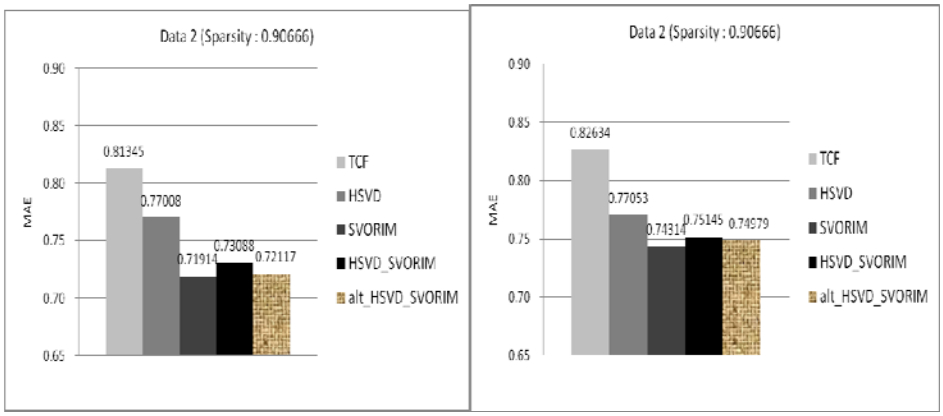


Fig. 10. Performance comparison for Data 2.

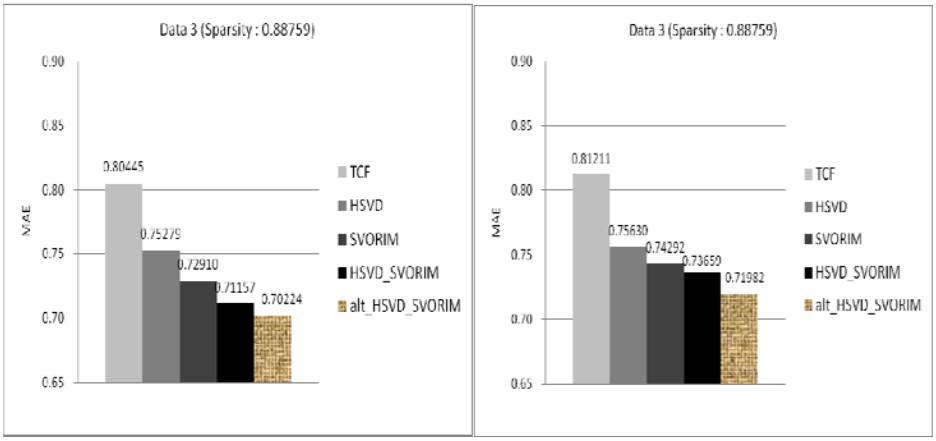


Fig. 11. Performance comparison for Data 3

The results above reveal a similar pattern, as pointed out in Experiment I. That is, all methods considered perform better and better from Data 1 to Data 3, as the data become denser. In addition, HSVD, SVORIM, HSVD_SVORIM and alt_HSVD_SVORIM perform much better than TCF. This means that the data cleaning process with Hebbian SVD and model-based CF enhances performance.

To make further comparison, the results can be rearranged for easy visualization (Fig. 12). First consider the data-cleaning effect on the recommendation performance. The performance improvement from HSVD is not as significant as that from SVORIM, HSVD_SVORIM, and alt_HSVD_SVORIM. As noticed, HSVD removes noise and redundancy in the original rating matrix. The cleaned rating matrix, however, only helps predictions in a primary way because it has a tendency to reduce the variations of a user's ratings. Without a sufficient rating range for the user, the predicted ratings are prone to errors.

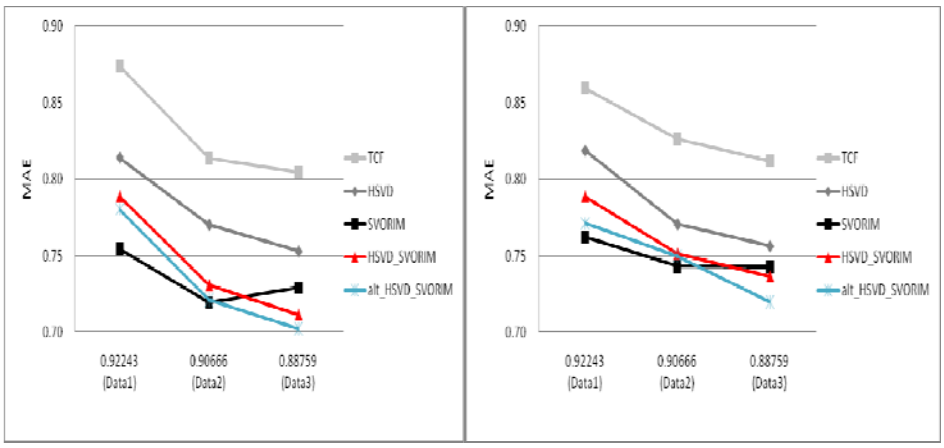


Fig. 12. Overall performance comparison.

Next, this study examines the effects of using SVOR models in CF on the recommendation performance. The model-based CF method, SVORIM in this case, significantly improves recommendation performance. These results agree with previous studies that besides scalability compensation, model-based methods can generate better recommendations than their memory-based counterparts.

In addition, a comparison of SVORIM, HSVD_SOVRIM, and alt_HSVD_SOVRIM yields an interesting result. Both HSVD_SOVRIM and alt_HSVD_SOVRIM outperform SVORIM for Data 3, perform similarly to SVORIM for Data 2, and yet fail to compete with SVORIM for Data 1. This implies that the data-cleaning process does not help the model-based method much when applied to sparse data (Data 1 in this case). However, it helps significantly when applied to dense data, which may include a lot of noise (Data 3 in this case). Therefore, incorporating a data-cleaning process into the model-based method can be feasible under circumstances of mildly sparse data.

Finally, we compare HSVD_SOVRIM and alt_HSVD_SOVRIM and find out that in general, alt_HSVD_SVORIM performs better than HSVD_SOVRIM. This is reasonable because, as explained before, the cleaned rating matrix from HSVD has a tendency to

reduce the variations of a user's ratings. Therefore, if we use HSVD results for all input features, these features may not be distinct enough for the classifier to distinguish.

4.4 Experiment III

This experiment compares TCF, HSVD, SVORIM, HSVD_SVORIM, and alt_HSVD_SVORIM using the large dataset (Data LS). As indicated in the data descriptions, Data LS contains 522 users and 1,674 movies with 76,337 ratings. Its number of ratings is 3 times as large as that for the small datasets. Figure 13 shows the performance result.

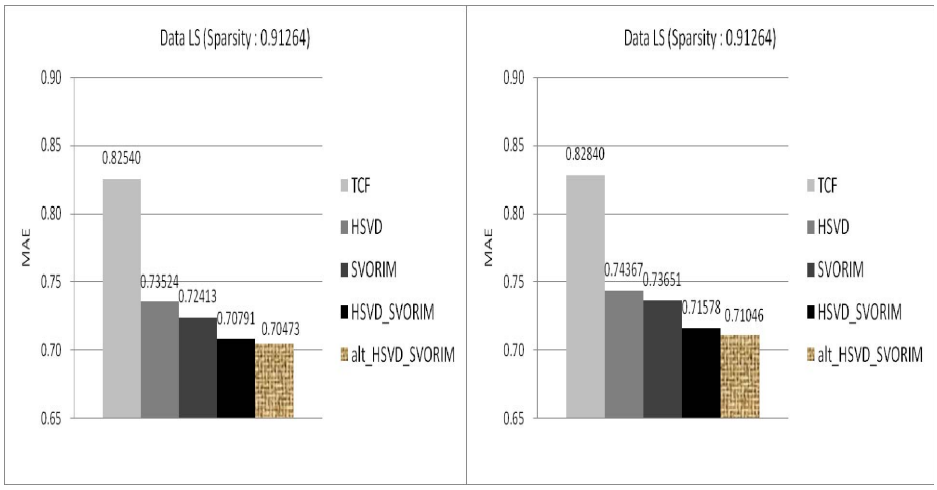


Fig. 13. Performance comparison for Data LS.

Not surprisingly, this experiment reveals patterns similar to the small datasets. TCF performs worst, followed by HSVD. In addition, both HSVD_SVORIM and alt_HSVD_SVORIM perform better than SVORIM. Based on the high sparsity degree in Data LS (0.91264), it is unclear why data-cleaning process also helps the model-based method significantly. However, the number of ratings in Data LS is almost 3 times that of the small datasets. Therefore, chances for noisy and redundant data become higher and a data-cleaning process such as HSVD can improve performance. Finally, alt_HSVD_SVORIM performs better than HSVD_SVORIM. These experimental results confirm the feasibility of applying the proposed approach and data-cleaning process followed by the model-based collaborative method.

5. CONCLUSION

The recent explosion in Internet growth has resulted in a huge amount of information disseminated on the Web. This in turn causes an information overload problem for users. With limited information processing ability to handle enormous information simultaneously, users may have difficulty finding the information they desire. Thus, recommender

systems have been developed to help users perform recommendation tasks.

This study proposes a model-based CF approach that considers the issues of data reliability and data measurement scale. Three experiments were conducted to examine the performance of the proposed approach. Results show that all CF variants (HSVD, SVORIM, HSVD_SVORIM and alt_HSVD_SVORIM) outperform traditional CF. The SVD method with Hebbian learning performs worst because of its tendency to reduce variations in user ratings. The proposed approach is comparable to, or outperforms, pure SVOR under mildly sparse data and large-scale data conditions. These results justify the practicability of the proposed approach in CF recommendation applications.

Although the results of this study seem promising, some issues require further study. First, the parameter k, the number of significant singular vectors in HSVD, is determined empirically in Experiment I. While the main purpose of this work is to show how data cleaning process (with k reduced) and model-based ordinal regression can help improve CF recommendation performance, the work on determining the optimal value of k remains an interesting and essential research issue to pursue in the future.

Second, the classifier adopted here, SVOR, has some limitations: it cannot learn and predict if the output classes of learning examples do not cover the full rating range (e.g., ranks of 1, 2, 3, 4, and 5 in this case). Therefore, if an active user only rates items with ratings of 2, 3, and 4 (as it is common for people to avoid extreme ratings), then SVOR cannot be used to generate predicted recommendations for this user. Further modification of SVOR or other classifiers that can deal with ordinal data should be considered.

REFERENCES

1. Breese, J. S., Heckerman, D., and Kadie, C. Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, 1998.
2. Adomavicius, G., and Tuzhilin, A. 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 2005; 17(6): 734-749.
3. Sandvig, J. J., Mobasher, B., and Burke, R. A survey of collaborative recommendation and the robustness of model-based algorithms. IEEE Data Engineering Bulletin 2008; 31(2): 3-13.
4. Billsus, D., and Pazzani, M. J. Learning collaborative information filters. In: Proceedings of the Fifteenth International Conference on Machine Learning, 1998.
5. Balabanović, M., & Shoham, Y. Fab: content-based, collaborative recommendation. Communications of the ACM 1997; 40(3): 66-72.
6. Kim, T. H., Ryu, Y. S., Park, S. I., and Yang, S. B. An improved recommendation algorithm in collaborative filtering. E-Commerce and Web Technologies, Lecture Notes in Computer Science 2002; 2455: 254-261.
7. Mobasher, B., Burke, R., and Sandvig, J. J. Model-based collaborative filtering as a defense against profile injection attacks. In: Proceedings of the National Conference on Artificial Intelligence, 2006.

8. Paterek, A. Improving regularized singular value decomposition for collaborative filtering. In: Proceedings of KDD Cup and Workshop, 2007.
9. Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. Indexing by latent semantic analysis. Journal of the American Society for Information Science 1990; 41(6): 391-407.
10. Berry, M. W., Dumais, S. T., and O'Brien, G. W. Using linear algebra for intelligent information retrieval. SIAM Review 1995; 37(4): 573-595.
11. Mehta, B., Hofmann, T., and Nejdl, W. Robust collaborative filtering. In: Proceedings of the 2007 ACM Conference on Recommender Systems, 2007.
12. Sarwar, B. M., Karypis, G., Konstan, J. A., and Riedl, J. Application of dimensionality reduction in recommender system–a case study. In: ACM WebKDD 2000 Web Mining for E-Commerce Workshop, 2000.
13. Zhang, S., Wang, W., Ford, J., Makedon, F., and Pearlman, J. Using singular value decomposition approximation for collaborative filtering. In: Proceedings of the 7th IEEE Conference on E-Commerce, 2005.
14. Gorrell, G. Generalized hebbian algorithm for incremental singular value decomposition in natural language processing. In: Proceedings of 11th Conference of the European Chapter of the Association for Computational Linguistics, 2006.
15. Webb, B. Netflix update: Try this at home. Simon Funk's personal blog. http://sifter.org/~simon/journal/20061211.html, 2006.
16. Vapnik, V. N. The nature of statistical learning theory. Springer, New York; 1995.
17. Miha, G., Dunja, M., Blaž, F., and Marko, G. Data sparsity issues in the collaborative filtering framework. Advances in Web Mining and Web Usage Analysis 2006; 4198: 58-76.
18. Shashua, A., and Levin, A. Ranking with large margin principle: Two approaches. Advances in Neural Information Processing Systems 2003; 15: 961-968.
19. Chu, W., and Keerthi, S. S. Support vector ordinal regression. Neural Computation 2007; 19(3): 792-815.
20. Koren, Y., Bell, R., and Volinsky C. Matrix factorization techniques for recommender systems. Computer 2009; 42 (8): 30-37.
21. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. GroupLens: An open architecture for collaborative filtering of netnews. In: Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work, 1994.
22. Anand, D. and Bharadwaj, K. Enhancing accuracy of recommender system through adaptive similarity measures based on hybrid features. Intelligent Information and Database Systems, Lecture Notes in Computer Science 2010; 5991: 1-10.
23. Sun, X., Wang, H., and Li, J. Satisfying privacy requirements: One step before anonymization. Advances in Knowledge Discovery and Data Mining, Lecture Notes in Computer Science 2010; 6118: 181-188.

Te-Min Chang (張德民) received the Ph.D. degree from School of Industrial Engineering, Purdue University in 1996. He is now an Associate Professor in Department of Information Management at the National Sun Yat-sen University, Kaohsiung, Taiwan. His research interests include information retrieval, data mining, text mining, recommender systems, and decision theory.

Wen-Feng Hsiao (蕭文峰) received the Ph.D. degree from Department of Information Management, National Sun Yat-sen University in 2001. He is now an Associate Professor in Department of Information Management at the National Pingtung Institute of Commerce, Pingtung, Taiwan. His research interest includes data mining, text mining, programming language teaching theory, cognitive decision theory and verbal information evaluation.

Wei-Lung Chang (張瑋倫) received the Ph.D. degree from Department of Management Information Systems, National Chengchi University in 2007. He is now an Associate Professor in Department of Business Administration at the Tamkang University, New Taipei City, Taiwan. His research interests include information management, information economy, electronic commerce and service science.