Expert Systems with Applications 28 (2005) 189–199

Expert Systems with Applications

www.elsevier.com/locate/eswa

Detection of the customer time-variant pattern for improving recommender systems

Sung-Hwan Min*, Ingoo Han

Graduate School of Management, Korea Advanced Institute of Science and Technology, 207-43 Cheongrangri-dong, Dongdaemun-gu, Seoul 130-722, South Korea

Abstract

Due to the explosion of e-commerce, recommender systems are rapidly becoming a core tool to accelerate cross-selling and strengthen customer loyalty. There are two prevalent approaches for building recommender systems—content-based recommending and collaborative filtering. So far, collaborative filtering recommender systems have been very successful in both information filtering and e-commerce domains. However, the current research on recommendation has paid little attention to the use of time-related data in the recommendation process. Up to now there has not been any study on collaborative filtering to reflect changes in user interest.

This paper suggests a methodology for detecting a user's time-variant pattern in order to improve the performance of collaborative filtering recommendations. The methodology consists of three phases of profiling, detecting changes, and recommendations. The proposed methodology detects changes in customer behavior using the customer data at different periods of time and improves the performance of recommendations using information on changes.

Keywords: Recommender system; Collaborative filtering; Change mining

1. Introduction

In recent years there has been the explosive growth in the amount of information. The rapid spread of the Internet has made it easy for a firm to develop a new style of e-business via one-to-one marketing. One of the issues required to establish the good relations between a firm and its customers is how to provide appropriate information that matches revealed interests of customers. The need for new marketing strategies in e-commerce, such as one-to-one marketing, web personalization, and customer relationship management (CRM) has been stressed both in research as well as in practice (Mobasher, Cooley, & Srivastava, 2000; Sarwar, Karypis, Konstan, & Riedl, 2000).

A recommender system is the information filtering that applies data analysis techniques to the problem of helping customers find the products they would like to purchase by producing a predicted likeness score or a list of recommended products for a given customer (Sarwar et al., 1998). Due to an explosion of e-commerce, recommender systems are rapidly becoming a core tool for accelerating cross-selling and strengthening customer loyalty. Recommender systems have been used in many web sites to recommend various items including books, movies, music, news articles, etc. There are two prevalent approaches for building recommender systems—content-based recommending and collaborative filtering (CF).

CF is a general approach to personalized information filtering. CF systems work by collecting user feedback in the form of ratings for items in a given domain and exploit the similarities and differences among the profiles of several users in determining how to recommend an item. On the other hand, content-based methods provide recommendations by comparing representations of content that interests the users (Pazzani, 1999). Conventional content-based recommendation systems generate recommendations based on a comparison between the representations of content features and user profiles. So far, CF recommender systems have been very successful in both information filtering domains and e-commerce domains, and many researchers have presented variations of CF to increase its performance (Aggarwal et al., 1999; Basu, Hirsh, & Cohen, 1998; Breese et al., 1998; Cheung, Kwok, Law, & Tsui,

* Corresponding author. Tel.: +822 958 3131; fax: +822 958 3604.
E-mail address: shmin@kgsm.kaist.ac.kr (S.-H. Min).

0957-4174/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2004.10.001

2003; Kohrs & Merialdo, 1999; Ungar & Foster, 1998). However, most recommender systems do little to detect or predict the changes in preference, although the customer interest does vary from time to time in the world. Especially for an internet-based company, it is of crucial importance knowing what is changing and how it has been changed because it allows the management to provide the right products and services to suit the changing market needs.

So far, there are only a small number of studies that consider time factors and find changed rules using association rule algorithms (Liu & Hsu, 1996; Song, Kim, & Kim, 2001). Furthermore, there has not been any research on CF to reflect changes in user interest. Finding neighbors in traditional CF is crucial for accurate recommendations because recommendations are based on the ratings of an active user's neighbors. For example, if an active user has preference for a small-sized car, his or her neighbors should be selected among the people who also have preference for a small-sized car. But, if his or her preference has moved from a small-sized car to a large-sized car, then his or her neighbors should be changed immediately. But the current CF algorithms are not adaptive to these situations dynamically, which results in the false recommendations. Our research focuses on these situations.

The purpose of this research is to develop a methodology which detects customers' time-variant patterns and define the characteristics of the patterns. Furthermore, this research seeks to improve recommendations by using information on changing patterns.

The proposed methodology consists of three phases of profiling, detecting changes, and recommendations. The proposed methodology detects the changes in user interest by using customer data at different periods of time and improves the performance of recommendations using information on changes.

The rest of this paper is organized as follows. Section 2 describes the research background. Section 3 presents the proposed methodology. Section 4 explains the results of the evaluation experiment. Section 5 presents the summary and future research issue.

2. Research background

2.1. Recommender systems

A recommender system is a technology that helps merchandisers implement a one-to-one marketing strategy. The recommender system analyzes a database of consumer preferences to overcome the limitations of segment-based mass marketing by presenting each consumer with a personal set of recommendations (Schafer, Konstan, & Riedl, 2001). Recommender systems are used by e-commerce sites to suggest products and provide consumers with information to help them decide which products to purchase. Recommender systems have been implemented by many big Web retailers such as Amazon.com and CDNow.com. Recommender systems help overcome information overload by providing personalized suggestions based on a history of a user's likes and dislikes. There are two prevalent approaches to building recommender systems—Collaborative Filtering (CF) and Content-based recommending.

2.1.1. Content-based approach

In the content-based approach, the system analyzes the content of items and creates a profile that is a representation of a user's interest in terms of items such as keyword, phrases and features. Then the content-based systems analyze the content of items unknown to the user, compare it with the profile and estimate which of those items are interesting to the user. This method is effective for recommending textual documents since it is able to recommend only those items that are 'understandable' to computers (Herlocker, Konstan, & Riedl, 2000). Content-based recommendation systems analyze the textual information about preferred items and recommend new items by finding items with similar information. Since the content-based method is appropriate when there is rich content information, it has been applied to recommend news articles or web pages. Content-based systems have the advantages of directness and simplicity, but they lack the sophistication of CF systems which analyze the behaviors of peer groups for making recommendations (Melville et al., 2000).

There is much research on the content-based approach. Basu et al. (1998) attempted to recommend the items that are similar to items the user has liked in the past, and Pazzani (1999) proposed content-based systems which search for the contents of items, and based on a sample of a user's preferences, learns to predict which items the user will like. Other examples of content-based filtering are Boolean search indexes, where the query is a set of keywords combined by Boolean operators; probabilistic retrieval systems, where the probabilistic reasoning is used to determine the probability that a document meets a user's information need (Fuhr & Buckley, 1991).

2.1.2. Collaborative filtering approach

CF is the most successful recommender system technology to date, and has been used in many successful recommender systems on the Web. CF systems recommend products to a target customer based on the opinions of other customers (Sarwar et al., 2000). Generally, CF uses a database of user preferences to predict additional topics or products that a new (or an active) user might like. So the main task in CF is to locate the peer group and predict the utility of the items to a particular user based on the votes from the peer group (Konstan et al., 1997). Traditionally, the problem space of CF can be formulated as a matrix of users versus items with each cell representing a user's rating on a specific item (Resnick, Iacovou, Suchak, Bergstrom, &

Riedl, 1994). Under this formulation, the problem is to predict the values for the specific empty cells.

Currently, CF algorithms can be classified into memory-based and model-based algorithms (Breese, Heckerman, & Kadie, 1998). Memory-based algorithms repeatedly scan the preference (or people) database to locate the peer groups for the active users. A prediction is then computed by weighting the votes of users in the peer groups. The people in the peer groups are identified based on their similarity or nearness in tastes to the active user. Consequently, these algorithms can be equivalently called correlation-based or nearest-neighbor collaborative filters. Model-based algorithms infer a user model from the database of rating histories. The user model is then consulted for predictions. Model-based algorithms require more time to train but can provide predictions in a shorter time in comparison to nearest-neighbor algorithms (Schafer et al., 2001). The most popular memory-based algorithms used in CF are correlation or vector similarity. Using this kind of algorithm, a subset of appropriate users or the peer group is chosen based on their similarity to the active user, then a weighted aggregate of their rankings is used to generate predictions for the active user.

In recent years, much research on CF has focused on the development of new algorithmic models for producing recommendations. These include rule induction (Basu et al., 1998), clustering (Kohrs & Merialdo, 1999; Ungar & Foster, 1998), graph theory (Aggarwal, Wolf, Wu, & Yu, 1999), latent semantic indexing (Billsus & Pazzani, 1998), and Bayesian networks (Breese et al., 1998).

2.2. Self-organizing map

The Self-Organizing Map (SOM) is a method for unsupervised learning, based on a grid of artificial neurons whose weights are adapted to match input vectors in a training set. It was first described by Teuvo Kohonen, and so is sometimes referred to as a Kohonen map (Kohonen et al., 1996). It converts complex, non-linear statistical relationships between high-dimensional data items into simple geometric relationships on a low-dimensional display. As it thereby compresses information while preserving the most important topological and metric relationships of the primary data items on the display, it may also be thought to produce some kind of abstractions. These two aspects, visualization and abstraction, can be utilized in a number of ways in complex tasks such as process analysis, machine perception, control, and communication.

The algorithm is explained most easily in terms of a set of artificial neurons, each having its own physical location on the output map, which take part in a winner-take-all process (a competitive network) where a node with its weight vector closest to the vector of inputs is declared the winner and its weights are adjusted making them closer to the input vector. Each node has a set of neighbors. When this node wins a competition, the neighbors' weights are also changed. They are not changed much though. The further the neighbor is from the winner, the smaller its weight change. This process is then repeated for each input vector, over and over, for a number of cycles. Different inputs produce different winners (Mangiameli et al., 1996).

3. Methodology

3.1. Overall procedure

This paper suggests a methodology for detecting the users' time-variant pattern in order to improve performance of CF recommendations. The methodology consists of three phases of profiling, detecting changes and recommendation as shown in Fig. 1.

In phase 1, raw data is preprocessed and transformed for detecting changes. Original input data of traditional CF, user-to-item matrix rarely contain each user's comparable data at different timeframes due to the sparse data set. So, it is difficult to compare two periods. To solve this problem, input data reduction (item dimension reduction) methods are needed. In phase 1, we reduce the dimension of input space using item hierarchy in order to get information for comparing a user's pattern at different timeframes. We can find which cluster each user belongs to at different timeframes by using reduced input data. We can also calculate auto-similarity which is used to detect time-variant patterns.

In phase 2, we use two methods to detect a customer's time-variant patterns. The cluster change module compares clusters of an active user at different timeframes, and records the information if there is any change. Auto-similarity is defined as a user's degree of change at different timeframes and it is calculated by an item reduction matrix using item hierarchy. In phase 2, we can find users whose patterns change and know when the users' pattern changes. Phase 3 is the recommendation process which consists of static and dynamic recommendations. Static recommendation means a recommendation using static similarity while dynamic recommendation is a proposed recommendation methodology which can detect a customer's time-variant pattern and recommend items using dynamically updated information. In the following sections, we will explain each module in detail.

3.2. Phase 1: profiling—preprocessing for detecting changes

In phase 1, raw data is preprocessed and transformed for detecting changes. First, we analyze the item, compose item hierarchy and the SOM cluster.

3.2.1. Item hierarchy

In general, input data of a CF problem is a user-to-rating matrix as shown in Table 1a. But in order to detect the time-variant pattern for an active user, we need to add a time dimension to the original input data. Table 1b shows

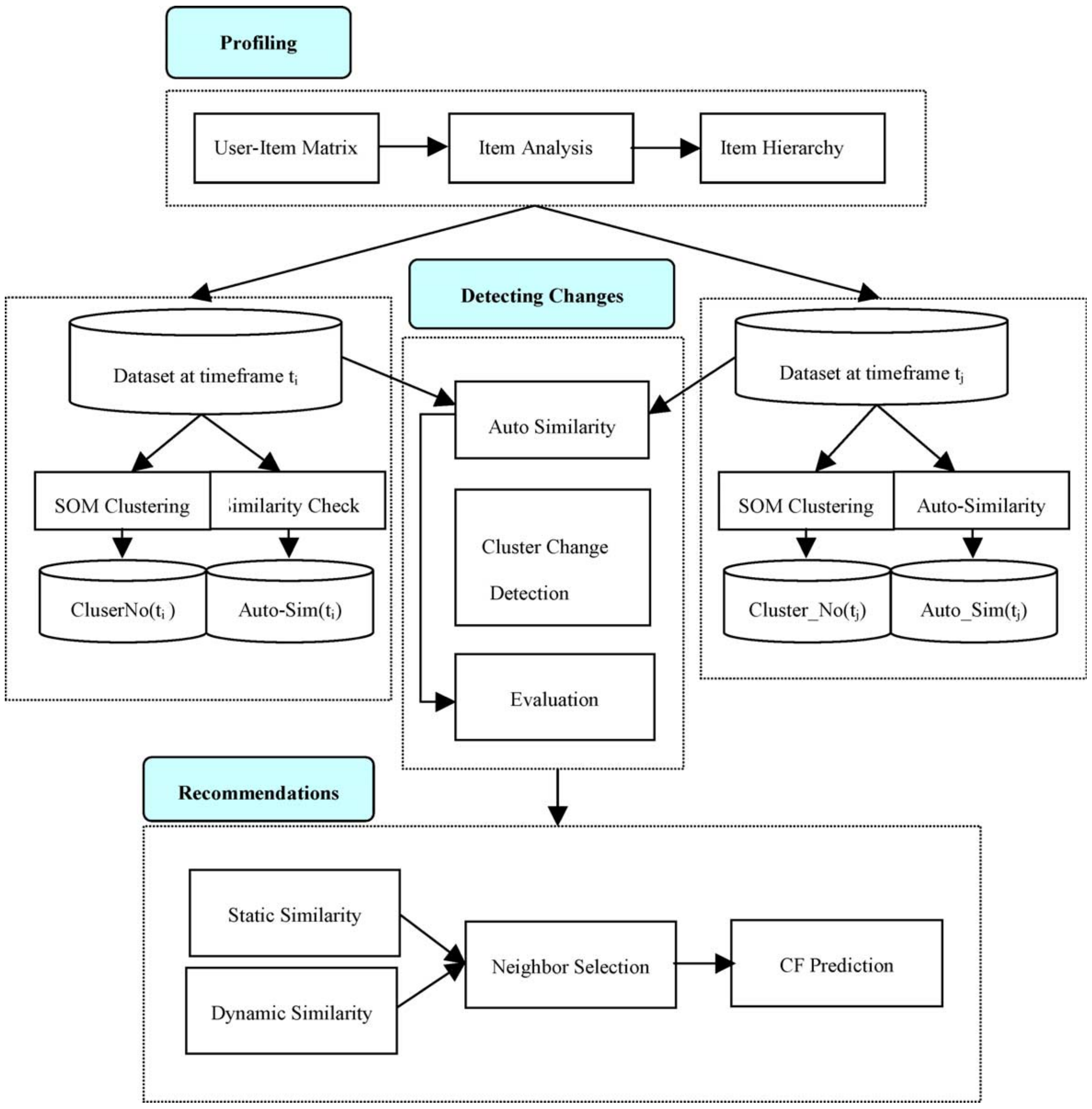

Fig. 1. Research model.

item ratings for an active user at different timeframes. As shown in Table 1b, it is difficult to compare the ratings at different timeframes to detect a time-variant pattern. For example, if we calculate the correlation between the ratings of an active user at $T1$ and the ratings at $T2$, we can easily see that the result is zero because there is no item which is rated at both $T1$ and $T2$. Correlations between other timeframes mostly come to zero in a traditional CF problem because of the sparse data set. For this reason, we cannot compare interests of an active user at different timeframes. To solve this problem, input data reduction (item dimension reduction) methods are needed. In this paper, we use a hierarchy of items shown in Fig. 2 whose leaf nodes represent items and non-leaf nodes represent a higher-level category to reduce the dimension of input data space. Yu et al. (2000) proposed a multi-level item-based CF

Table 1a
Original CF input data (User-to-Item Rating Matrix)

	Item 1	Item 2	Item 3	Item 4	⋯	Item *n*
User 1		2				
User 2	3		1		…	
⋮						1
Active User	1	5	2	5		3
User *n*						

Table 1b
Time-to-Item Rating Matrix of an active user

Time	Item 1	Item 2	Item 3	Item 4	⋯	Item *n*
*T*1	1					
*T*2		5	2			
*T*3				5	…	3
⋮						
Tn						

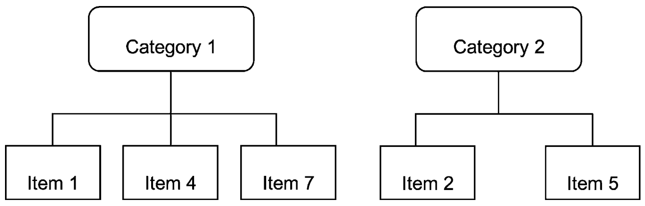

Fig. 2. Example of item hierarchy.

algorithm. We apply this concept to our proposed dynamic CF. In the proposed method, the relationship between items 1 and 7 can be found via a hierarchical structure. As a result, it becomes possible to compute the similarity between the ratings of a user having bought item 1 at timeframes *T*1 and the ratings of the same user having bought item 7 at timeframes *T*2 as shown in Table 1c. The ratings for higher-level categories are derived from the actual ratings for the items. The rating for a category is defined as the average ratings for the items in that category as follows

$$\mathrm{cr}_{a,k} = \sum_{i \in \mathrm{category},k} \frac{1}{RN_k} r_{a,i}$$

In above equation, $\mathrm{cr}_{a,k}$ is the derived rating of category *k* of user A, RN_k is the number of rated items that belong to that category, and $r_{a,i}$ is the rating of item *i* of user A. These derived ratings of non-leaf level nodes are incorporated in computing both auto-similarity and cluster change at different timeframes. In Section 4, we describe the experiments in which we use genre data as category information.

3.2.2. SOM clustering

In phase 1, customers with similar interests are clustered by SOM and this output is used as a base for detecting a cluster change of an active user. In the SOM clustering process, category ratings, calculated using item hierarchy information, are used as input data in order to find the cluster of an active user at different timeframes. One user may belong to the same cluster at different timeframes, but another user may belong to a different cluster at different timeframes. Different clusters at different timeframes means that the user may have a time-variant pattern. The change detection module compares clusters of an active user at different timeframes. If there is a big change in a cluster at any timeframe, the user's interest is interpreted as having changed at that timeframe.

Table 1c
Time-to-Category Rating Matrix

	Category 1			Category *n*	
	Item 1	Item 4	Item 7	Item 2	Item *i*
*T*1	1				
		1			
*T*2			1	5	
	1				
*T*3		5			1
		5			1
…					

3.3. Phase 2: detecting time-variant patterns

Two methods are proposed to detect a customer's time-variant pattern. The cluster change module compares an active user's cluster at different timeframes and records the information if there is any change. Auto-similarity is defined as a user's degree of change at different timeframes and it is calculated by the item reduction matrix using item hierarchy. In this phase, we can find the users whose patterns change and when the users' pattern changes. The proposed methodology for detecting the time-variant pattern is as follows:

Step 1. For each active user, calculate the following change measures for each timeframe.

- Cluster Change
- Auto-similarity

Step 2. Detect if there are larger changes than the given threshold value of each measure.

In the following sections, we will explain each method in detail.

3.3.1. Cluster change

In order to detect users' time-variant patterns, cluster change is measured. First, SOM cluster map is used as a base for detecting the cluster change. Number of cluster is determined by using cluster validity indexes such as Davies–Bouldin (DB) Index (Davies & Bouldin, 1979). It is checked if there is any change in clusters of an active user at different timeframes. The proposed change detection procedure using the cluster change measure is as follows:

Step 1. Determine number of clusters.
Step 2. Find weight vectors of the SOM clustering.
Step 3. Find SOM cluster of an active user at a different timeframe.
Step 4. Check if there is any change in the cluster of active user at a different timeframe.
Step 5. If there is any change, calculate the distances between two clusters using centroid values.
Step 6. If $\mathrm{Dist}(C_{i,T1},C_{i,T2}) > \mathrm{TR}$, conclude that the active user's behavior changed after timeframe *T*1.

■ Definition

$C_{i,T}$:	cluster number of user *i* at timeframe *T*
$\mathrm{Dist}(C_{i,T1},C_{i,T2})$:	distance between $C_{i,T1}$ and $C_{i,T2}$
TR:	threshold value where customer change is decided in cluster change detection module.

3.3.2. Auto-similarity

Similarity in the traditional CF approach means the similarity weight between an active user and neighbors. It is usually calculated by the Pearson correlation coefficient in Eq. (1)

$$S_{a,u} = \frac{\sum_i (r_{a,i} - \bar{r_a})(r_{u,i} - \bar{r_u})}{\sqrt{\sum_i (r_{a,i} - \bar{r_a})^2}\sqrt{\sum_i (r_{u,i} - \bar{r_u})^2}}, \quad \text{where } -1 \le S_{a,u} \le 1 \tag{1}$$

In the above equation $S_{a,u}$ is the similarity between the active user and each of the other users who have the co-rated items with the active user a, i is the index of each item that both user a and user u have rated, $r_{a,i}$ is the rating of user A for item i and $r_{u,i}$ is the rating of user U for item k. And $\bar{r_a}$ denotes the average rating of active user a, $\bar{r_u}$ is the average of the other user's ratings. This similarity is generally used to select the nearest neighbors who have the most similar tastes in order of the values of $S_{a,u}$.

In this paper, we propose using auto-similarity measure to detect a change by the active user. Auto-similarity means the similarity weight between an active user's ratings at different timeframes and it is also calculated by Eq. (2). This measure is used for detecting the degree and the characteristics of changes.

Auto-similarity $\mathrm{AS}(a)_{T1,T2}$ is defined as the similarity between the active user's category ratings at timeframe $T1$ and the same active user's category ratings at timeframe $T2$

$$\mathrm{AS}(a)_{T1,T2} = \frac{\sum_k (\mathrm{cr}(T1)_{a,k} - r(\bar{T1})_a)(\mathrm{cr}(T2)_{a,k} - r(\bar{T2})_a)}{\sqrt{\sum((\mathrm{cr}(T1)_{a,k} - r(\bar{T2})_a)^2}\sqrt{\sum((\mathrm{cr}(T2)_{a,k} - r(\bar{T2})_a)^2}} \tag{2}$$

where k is the index of each category that user A has rated at both timeframe $T1$ and timeframe $T2$, $\mathrm{cr}(T1)_{a,k}$ is user A's category rating for category k at timeframe $T1$, $\mathrm{cr}(T2)_{a,k}$ is the same user A's rating for category k at timeframe $T2$, $r(\bar{T1})_a$ denotes the average category rating of active user A at timeframe $T1$ and $r(\bar{T2})_a$ is the average of the same user's category ratings at timeframe $T2$.

$\mathrm{AS}(a)_{T1,T2}$ is 1 if the active user has exactly the same preference for the category rating at timeframe $T1$ and timeframe $T2$, and is -1 if he or she has a completely different preference between $T1$ and $T2$. When $\mathrm{AS}(a)_{T1,T2}$ is -1, we can conclude that the active user's interest changed after timeframe $T1$. If there is no correlation between the preferences of the category rating at timeframe $T1$ and the category rating at timeframe $T2$ of the active user, $\mathrm{AS}(a)_{T1,T2} = 0$. If user A tends to have a similar category rating for each timeframe, $\mathrm{AS}(a)_{T1,T2}$ becomes a positive number. The absolute value of $\mathrm{AS}(a)_{T1,T2}$ indicates how much user A at timeframe $T1$ tends to agree with the same user at timeframe $T2$ on the category rating that he or she rated at each timeframe. If they tend to have opposite ratings for each category, $\mathrm{AS}(a)_{T1,T2}$ becomes a negative number. The absolute value of $\mathrm{AS}(a)_{T1,T2}$ indicates how much user A at timeframe $T1$ tends to disagree with the same user at timeframe $T2$ on the category rating that he or she rated at each timeframe. $\mathrm{AS}(a)_{T1,T2}$ can be between -1 and 1. The change detection procedure using auto-similarity is as follows:

Step 1. Transform item ratings into category ratings at each timeframe using item hierarchy.

Step 2. Find auto-similarity of the active user at a different timeframe.

- $\mathrm{AS}(a)_{T1,T2}$: similarity weight between the active user's category rating at $T1$ and $T2$

Step 3. Detect characteristics of changes.

(1) Changed Interest:
If $\mathrm{AS}(a)_{T1,T2} < \mathrm{TS}$, conclude that the active user's behavior changed after timeframe $T1$.(TS; given a threshold value smaller than zero where customer change is decided)
(2) Added Interest:
If $|AS(a)_{T1,T2}| < \mathrm{TA}$, conclude that the new active user's interest is added after timeframe $T1$.(TA; given threshold value where we conclude that new interest is added)
(3) No change:
$\mathrm{AS}(a)_{T1,T2} > \mathrm{TN}$, conclude that there is no change.(TN; given threshold value larger than zero)

3.4. Phase 3: recommendations

The traditional CF algorithms are not adaptive to the situations where customers' patterns change. This may result in the false recommendations. In phase 3, we can find the active user's neighbors dynamically according to his or her changing pattern by using the dynamic similarity. Phase 3 is the recommendation process, which consists of static and dynamic recommendations. A static recommendation means recommendation using static similarity while dynamic recommendation is a proposed recommendation methodology which can detect a customer's time-variant pattern and recommend items using dynamically updated information. The procedure for the proposed recommendation is as follows:

Step 1. Detecting change
– Detect customer behavior change.
Step 2. Dynamic Similarity Calculation
– Identify the similarity weight between active user and neighbors.

$$S_{a,u} = \frac{\sum_i (r_{a,i} - \bar{r_a})(r_{a,i} - \bar{r_a})}{\sqrt{\sum_i (r_{a,i} - \bar{r_{aa}})^2}\sqrt{\sum_i (r_{u,i} - \bar{r_u})^2}} \text{(if no change)} \tag{3}$$

$$S_{a,u} = \frac{\sum_{i\in I_1} RW_{a,i}(r_{a,i}-\bar{r_a})(r_{u,i}-\bar{r_u}) + \sum_{i\in I_2} RW_{a,i}(r_{a,i}-\bar{r_a})(r_{u,i}-\bar{r_u})}{\sqrt{\sum_i (r_{a,i}-\bar{r_a})^2}\sqrt{\sum_i (r_{u,i}-\bar{r_u})^2}} \quad (4)$$

■ Definition

-$T(r_{a,i})$: the time when active user A rated item i
-RW$_{a,i}$: dynamic weight which is 1 if $T(r_{a,i})<t$ and $1+w|AS(a)_{(t-1),(t)}|$ (w>0) if $T(r_{a,i})>t$

where I_1 is the item set which consists of items that user A rated before time t, I_2 is the item set which consists of items that user A rated after time t

Step 3. Neighbor Selection

– Find a neighborhood of users with similar preferences.

Step 4. Prediction

– Predict the active users' rating unanswered based on neighborhood ratings.

After $S_{a,u}$ is obtained, the predicted numerical rating $r_{a,x}$ of the active user for a target item x is calculated as follows

$$P_{a,x} = \bar{r}_a + \frac{\sum_{v\in \text{Raters}}(r_{v,x}-\bar{r_v})S_{a,v}}{\sum_{v\in \text{Raters}}|S_{a,v}|}$$

where raters are the users who rate the target item, $\bar{r_a}$ is the average rating of user A, and $S_{a,v}$ it the similarity weight between the active user and neighbor v.

The basic example of computation for dynamic recommendation is as follows:

Example of computation

(a) Initial user's ratings

	Item 1	Item 2	Item 3	⋯	Item *i*
*U*1	1	4	5		2
*U*2		2	1		
*U*3	5		2		4
⋮					

(b) Calculate user's category rating by item analysis (content-based approach)

	Category 1			Category *n*	
	Item 3	Item 4	Item 7	Item 2	Item *i*
*U*1	5	3	4	1	2
				1.5	
*U*2	1	1	2	4	5
				4.5	
⋮					

(c) If an active user is given

	Item 1	Item 2	Item 3	⋯	Item i
Ua	3	2	5		1

(d) Detect change

	*T*1	*T*2	*T*3	*Tn*
Cluster change	C06	C06	C12	C12
Auto-similarity	–	0.9	−0.7	0.2

(e) Dynamic similarity

$$S_{a,u} = \frac{\sum_{i\in I_1}(r_{a,i}-\bar{r_a})(r_{u,i}-\bar{r_u}) + \sum_{i\in I_2} 1.4(r_{a,i}-\bar{r_a})(r_{u,i}-\bar{r_u})}{\sqrt{\sum_i (r_{a,i}-\bar{r_a})^2}\sqrt{\sum_i (r_{u,i}-\bar{r_u})^2_u}}, \text{ if } w = 0.5$$

where I_1 is the item set that consists of items that user A rated before timeframe *T*2, I_2 is the item set that consists of items that user A rated after timeframe *T*2.

(f) Neighbor selection

– Find the neighborhood of users with similar preferences.

(g) Prediction

– Predict the active users' rating unanswered based on neighborhood ratings.

4. Experimental evaluation

4.1. Data set

For experiments we use the EachMovie database, provided by Comaq Systems Research Center (http://www.research.compaq.com/SRC/eachmovie). The dataset contains explicit rating data provided by each user for various movies. The EachMovie dataset has rating information on 1628 movies by 72,916 users during an 18-month-period from 1996. Users rated various numbers of movies by the number of stars 0–5. In these experiments, we normalize the rating values to values between 0 and 1. Other than the rating information, the database contains the movie title, genre, each user's age, and gender.

We assume that the items are classified into a multi-level (hierarchical) category, and we use the category information to compute the similarity between an active user's ratings at different timeframes. Even when an active user does not have rating information on the same item at different timeframes, the same user's change measures can be computed if some of the rated items belong to the same category in higher level. In this experiment we use genre data as category information.

4.2. Evaluation metrics

Recommender systems research has used several types of measures for evaluating the quality of a recommender system. They can be mainly categorized into two classes: statistical accuracy metrics and decision support accuracy metrics.

Statistical accuracy metrics evaluate the accuracy of a system by comparing the numerical recommendation scores against the actual user ratings for the user-item pairs in the test dataset. Mean Absolute Error (MAE) between ratings and predictions is a widely used metric. Decision support accuracy metrics evaluate how effective a prediction engine is at helping a user to select high-quality items from the set of all items. The most commonly used decision support accuracy metrics are reversal rate, weighted errors and ROC sensitivity (Sarwar et al., 1998).

We used MAE as our choice of evaluation metric to report prediction experiments because it is commonly used and easy to interpret. MAE is a measure of the deviation of recommendations from their true user-specified values. For each ratings-prediction pair $\langle p_i, q_i \rangle$ this metric treats the absolute error between them, i.e. $|p_i - q_i|$ equally. The MAE is computed by first summing these absolute errors of the N corresponding ratings-prediction pairs and then computing the average. Formally

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

where N is the number of ratings in the test data set.

The lower the MAE, the more accurately the recommendation engine predicts user ratings. In previous research Sarwar et al. (1998) has shown that MAE and ROC provide the same ordering of different experimental schemes in terms of prediction quality.

4.3. Experimental method

First we selected 1200 users with more than 100 rated items. We divided the data set into a training set and a test portion. Before starting full experimental evaluation of different algorithms, we determined the sensitivity of different parameters. We fixed the optimum values of these parameters from the sensitivity plots and used them for the rest of the experiments.

To compare the performance of the proposed dynamic CF algorithm we use the traditional CF algorithm as the benchmark model. The traditional CF recommendation employs the Pearson nearest neighbor algorithm.

4.4. Experimental results

Experimental results are mainly divided into sensitivity analysis and performance results. In assessing the quality of recommendations, we first determined the sensitivity of some parameters before running the main experiments. These parameters include the number of clusters, the neighborhood size, timeframe, and the threshold for auto-similarity (TS).

4.4.1. Number of clusters

The prediction of the correct number of clusters is a fundamental problem in unsupervised classification problems. Many clustering algorithms such as SOM and k-means clustering require the definition of the number of clusters beforehand. In this experiment, we use the Davies–Bouldin (DB) Index, which is one of the most common cluster validity indexes for finding the number of clusters (Davies & Bouldin, 1979). The Davies–Bouldin index aims at identifying sets of clusters that are compact and well separated. For any partition $U \leftrightarrow X : X_1 \cup \cdots X_i \cup \cdots X_c$, where X_i represents the ith cluster of that partition, the Davies–Bouldin validation index, DB, is defined as

$$\mathrm{DB}(U) = \frac{1}{c} \sum_{i=1}^{c} \max_{i \neq j} \left\{ \frac{\Delta(X_i) + \Delta(X_j)}{\delta(X_i + X_j)} \right\}$$

where $\delta(X_i + X_j)$ defines the distance between clusters X_i and X_j (intercluster distance), $\Delta(X_k)$ represents the intracluster distance of cluster X_k and c is the number of clusters of partition U. Small values of DB correspond to clusters that are compact, and whose centers are far away from each other. Therefore, the cluster configuration that minimizes DB is taken as the optimal number of clusters, c. Fig. 3 shows DB indexes for each clusters by SOM. Though cluster 2 is the smallest in the DB index, this segmentation is not suitable for the base of cluster change because the segmentation only separates customers into people who like movies and people who dislike movies. So we select eight as our final number of clusters, which has the second smallest DB index value. Table 2 presents the results of SOM clustering when the number of clusters is 8.

4.4.2. Sensitivity of the neighborhood size

The size of the neighborhood has a significant impact on the prediction quality (Herlocker et al., 2000). To determine the sensitivity of this parameter, we performed an experiment where we varied the number of neighbors to

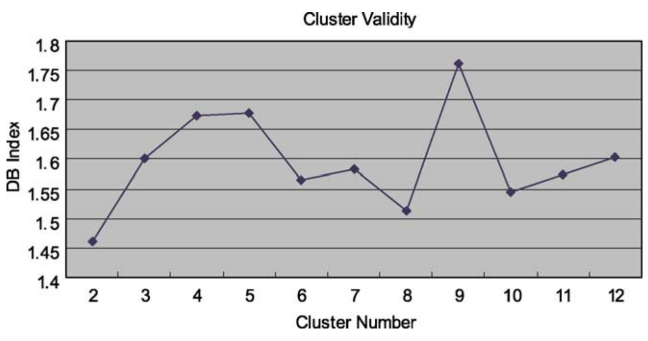

Fig. 3. Davies–Bouldin validity indexes for each cluster.

Table 2
Cluster centroid

Genre	Cluster							
	1	2	3	4	5	6	7	8
Action	0.58	**0.6**	***0.36***	0.49	**0.73**	**0.63**	0.4	**0.66**
Animation	**0.69**	***0.24***	0.53	0.57	**0.77**	**0.66**	***0.35***	**0.61**
Art_Foreign	**0.68**	**0.66**	***0.25***	**0.65**	**0.76**	**0.72**	0.52	***0.28***
Classic	**0.77**	**0.77**	0.57	**0.7**	**0.86**	**0.78**	**0.63**	**0.75**
Comedy	0.55	0.51	***0.31***	0.45	**0.69**	0.58	***0.34*** >	0.52
Drama	**0.67**	**0.65**	***0.37***	0.59	**0.77**	**0.68**	0.48	0.59
Family	**0.67**	***0.39***	0.46	0.48	**0.74**	**0.6**	***0.31***	0.59
Horror	***0.31***	**0.61**	***0.14***	0.48	**0.73**	**0.65**	0.41	0.56
Romance	**0.7**	0.59	0.43	0.54	**0.73**	**0.63**	***0.39***	0.53
Thriller	0.55	**0.63**	***0.31***	0.52	**0.74**	**0.65**	0.43	**0.65**

Bold numbers mean likes and italicized and bold numbers are dislikes.

be used and computed MAE. Our results are shown in Fig. 4. We can observe that the size of neighborhood does affect the quality of prediction. The CF algorithm improves as we increase the neighborhood size from 10 to 40. After that, the rate of increase diminishes and the curve tends to be flat. We used 70 as our choice of neighborhood size.

4.4.3. *Sensitivity of the timeframe size*

To experimentally determine the impact of the timeframe size on the quality of the prediction, we selectively varied the value of the timeframe size. We classified timeframes into CT (cumulative time) and IT (independent time) as shown in Fig. 5. If time is cumulative, the timeframe is CT, otherwise, the time is IT. For example, in case of 'CT=2' we compare the timeframe between January 1 and February 28 and the timeframe between March 1 and April 30. After that, we also compare the timeframe between January 1 and April 30 and the timeframe between May 1 and June 30. But, in case of 'IT=2' we compare the timeframe between January 1 and February 28 and the timeframe between March 1 and April 30. After that, we also compare the timeframe between March 1 and April 30 and the timeframe between May 1 and June 30.

Using the train data set, we precomputed the user similarity using different timeframe sizes. We then used the test data set to compute MAE and plotted the values. Fig. 6 shows the plots at different timeframe values. It can be observed from the plots that the MAE values get better as we use CT and are smallest at 2 months. We thus set CT at 2 months as our choice of timeframe size.

4.4.4. *Sensitivity of the change threshold value (TS)*

To experimentally determine the impact of the change threshold value on the quality of the prediction, we selectively varied the value of threshold to be used for similarity computation from −0.01 to −1 in increments of −0.1. A threshold value of −0.1 means that we only considered −0.1 as the best threshold values for detecting change. If the auto-similarity value of the active user is smaller than −0.1, we conclude that the active user's pattern changed.

Fig. 7 shows the plots at different TS values. TCF-All means recommendations of all users' ratings in the test data set by traditional CF while DCF-All describes recommendations of all users' ratings in the test data set by the proposed dynamic CF. TCF-Changed User means recommendations of ratings of users who are detected as changed users, in the test data set by traditional CF while DCF-Changed User describes recommendations of changed users' ratings in the test data set by the proposed dynamic CF.

As we can see from the results, MAE values of TCF-All have no relation to the change threshold value and MAE values of both TCF-Changed User and DCF-Changed User increase as the change threshold value increases. MAE values of TCF-Changed User are larger than those of DCF-Changed User at all change threshold sizes. This indicates that the proposed model can improve the performance of the recommendation. MAE values of DCF-All have minimum value at −0.4. So we used −0.4 as our choice of auto-similarity threshold size.

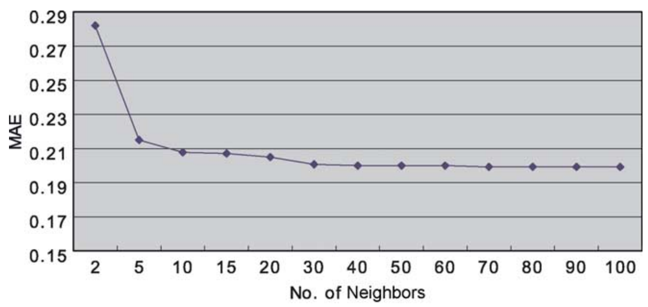

Fig. 4. Sensitivity of the neighborhood size.

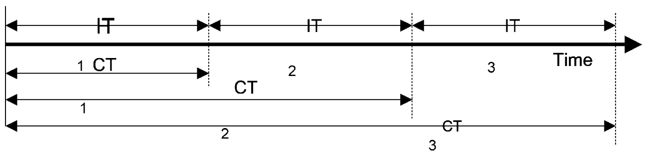

Fig. 5. Classification of the timeframe.

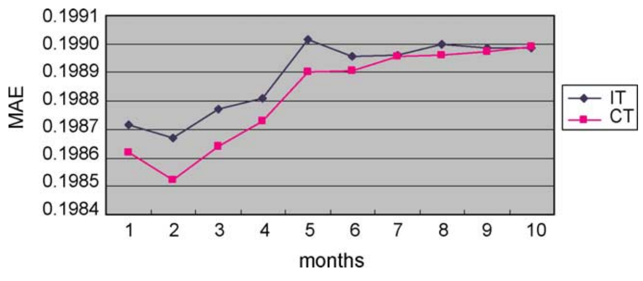

Fig. 6. Sensitivity of the timeframe size.

4.4.5. Performance results

Once we obtain the optimal values of the parameters, we compared both of our dynamic CF algorithms with the benchmark traditional CF algorithm. We present the results in Table 3. In Table 3, RWCF1 describes the simple recent weighted CF model and RWCF2 means the exponentially recent weighted CF model. In the TCF model, $RW_{a,i}$ in Eq. (4) is 1, in the RWCF1 model $RW_{a,i}$ is $k_1T(r_{a,i})$ and $k_2[T(r_{a,i})]^2$ in RWCF2 where k_1 and k_2 are constant larger than zero. Table 3 presents the performance of the competing models according to the metric of mean square error of recommendation. It can be observed that the proposed dynamic CF algorithm outperforms the traditional CF algorithm. In addition, a set of pairwise t-tests in Table 4 indicates that the differences were statistically significant.

Dynamic CF reflects better user preference than traditional CF at the 5% significance level. These results show that the proposed dynamic CF algorithm is more accurate than the traditional CF algorithm.

5. Conclusion

In this paper, we propose a novel methodology for CF which can detect a user's time-variant patterns and

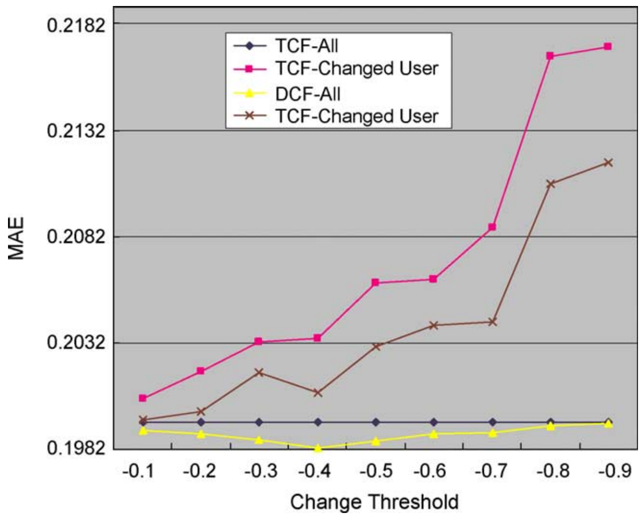

Fig. 7. Sensitivity of change threshold value.

Table 3
Performance results

Model		MAE
Proposed model	DCF	0.19823
	RWCF1	0.19908
	RWCF2	0.19925
Traditional model	TCF	0.19917

DCF, Dynamic Collaborative Filtering; TCF, Traditional Collaborative Filtering; RWCF1, Simple Recent weighting CF; RWCF2, Exponentially Recent weighting CF.

dynamically reflect this information for more accurate recommendations.

The proposed methodology consists of three phases of profiling, detecting changes and recommendation. In phase 1, raw data is preprocessed and transformed for detecting changes. In phase 2, we use two methods to detect customer's time-variant patterns, cluster change module and auto-similarity module. Phase 3 is the recommendation process, which consists of static and dynamic recommendation. Static recommendation means a recommendation using static similarity while dynamic recommendation is a proposed recommendation methodology which can detect customer's time-variant pattern and recommend items using information on the change point and characteristics of change in order to improve recommendations.

We conducted an experiment to evaluate the proposed methodology on the EachMovie data set and compared them with the traditional CF algorithm. The results show the proposed methodology's effectiveness of detecting changed behavior, and its improvement in making recommendations.

In this paper, we apply the concept of time to CF algorithm and propose a new methodology which can detect customers' time-variant patterns. We also propose a measure for classifying the characteristics of changes in customer interest. The proposed methodology improves the performance of CF recommendations. This research is expected to be easily extended into the variations of CF algorithm and to help marketer to catch customer's changing needs.

In this paper, there are some limitations. First, we used only EachMovie data set for experiments. It is needed to evaluate our methodology using other data set. Second,

Table 4
Paired t-test

		p-value			
		DCF	RWCF1	RWCF2	TCF
Proposed model	DCF	.	0.033**	0.029**	0.031**
	RWCF1		.	0.335	0.463
	RWCF2			.	0.428
Traditional model	TCF				.

** Significant at the 0.05 level.

we did not consider change in the customer cluster structure. We assumed that cluster structure does not change in a short term.

In our future work, we intend to evaluate our methodology using other data set. We would also like to expand this model to apply to variations of CF. Furthermore, we intend to develop a new performance measure which is appropriate for dynamic recommendations.

References

Aggarwal, C. C., Wolf, J. L., Wu, K. -L., & Yu, P. S. (1999). Horting hatches an egg: a new graph-theoretic approach to collaborative filtering. *Proceedings of ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 201–212.

Basu, C., Hirsh, H., & Cohen, W. (1998). *Recommendation as classification: using social and content-based information in recommendation* pp. 11–15 *Proceedings of the 1998 workshop on recommender systems*. Menlo Park, CA: AAAI Press.

Billsus, D., & Pazzani, M. J. (1998). Learning collaborative information filters. *Machine learning proceedings of the fifteenth international conference (ICML'98).*

Breese, J. S., Heckerman, D., & Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. *Proceedings of the 14th conference on uncertainty in artificial intelligence (UAI-98)*, pp. 43–52.

Cheung, K., Kwok, J. T., Law, M. H., & Tsui, K. (2003). Mining customer product ratings for personalized marketing. *Decision Support Systems*, *35*, 231–243.

Davies, D. L., & Bouldin, D. W. (1979). A cluster separation measure. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, *1*(2), 224–227.

Fuhr, N., & Buckley, C. (1991). A probabilistic learning approach for document indexing. *Transactions on Information Systems*, *9*(3), 223–248.

Herlocker, J. L., Konstan, J. A., & Riedl, J. (2000). Explaining collaborative filtering recommendations. *Proceedings on the ACM 2000 conference on computer supported cooperative work, Philadelphia*, pp. 241–250.

Kohonen, T., Hynninen, J., Kangas, J., & Zaaksonen, J. (1996). *SOM_PAK: The self-organizing map program package*. Technical Report A31. Helsinki University of Technology, Laboratory of Computer and Information Science, FIN-02150.

Kohrs, A., & Merialdo, B. (1999). *Clustering for collaborative filtering applications In Computational intelligence for modeling, control and automation (CIMCA'99), Vienna*. IOS Press.

Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Gordon, L. R., & Riedl, J. (1997). GroupLens: applying collaborative filtering to Usenet news. *Communications of the ACM*, *40*(3), 77–87.

Liu, B., & Hsu, W. (1996). Post-analysis of learned rules. *Proceedings of the 13th national conference on artificial intelligence (AAAI-96)*, pp. 828–834.

Mangiameli, P., Chen, S. K., & West, O. (1996). A comparision of SOM neural network and hierachical clustering methods. *European Journal of Operation Research*, *93*, 402–417.

Melville, P., Mooney, R.J., Nagarajan, R. (2000). Content-boosted collaborative filtering for improved recommendations. *Proceedings of the 18th national conference on artificial intelligence (AAAI-2002), Edmonton, Canada, July 2002*, pp. 187–192.

Mobasher, B., Cooley, R., & Srivastava, J. (2000). Automatic personalization based on Web usage mining. *Communications of the ACM*, *43*(8), 142–151.

Pazzani, M. J. (1999). A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, *13*(5/6), 393–408.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. (1994). GroupLens: an open architecture for collaborative filtering of netnews. *CSCW*, *94*, 175–186.

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2000). Analysis of recommendation algorithms for e-commerce. *Proceedings of ACM E-Commerce 2000 conference*, pp. 158–167.

Sarwar, B. M., Konstan, J. A., Borchers, A., Herlocker, J. L., Miller, B. N., & Riedl, J. (1998). Using filtering agents to improve prediction quality in the grouplens research collaborative filtering system. *Proceedings of CSCW'98, Seattle, WA*.

Schafer, J. B., Konstan, J. A., & Riedl, J. (2001). Electronic commerce recommender applications. *Data Mining and Knowledge Discovery*, *5*(1/2), 115–153.

Song, H. S., Kim, J. K., & Kim, S. H. (2001). Mining the change of customer behavior in an internet shopping mall. *Expert Systems with Applications*, *21*, 157–168.

Ungar, L.H., & Foster, D.P. (1998). Clustering methods for collaborative filtering. *Proceedings of the 1998 workshop on recommender systems*, pp. 114–129.

Yu, K.A., Choi, S., Kim, J. (2000). Improving the performance of collaborative recommendation by using muli-level similarity computation. *IASTED international conference on artificial intelligence and soft computing*, http://www.research.compaq.com/SRC/eachmovie