ADDRESSING THE DATA RECENCY PROBLEM IN COLLABORATIVE FILTERING SYSTEMS

by

Yoonsoo Kim

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Master of Science

in

Computer Science

by

_____

September 2004

APPROVED:

_____

Professor David C. Brown, Thesis Advisor

_____

Professor Mark Claypool, Thesis Reader

_____

Professor Michael A. Gennert, Head of Department

ABSTRACT

Recommender systems are being widely applied in many E-commerce sites to suggest products, services, and information items to potential users. Collaborative filtering systems, the most successful recommender system technology to date, help people make choices based on the opinions of other people. While collaborative filtering systems have been a substantial success, there are several problems that researchers and commercial applications have identified: the early rater problem, the sparsity problem, and the large scale problem. Moreover, existing collaborative filtering systems do not consider data recency. For this reason, if a user's preferences have changed over time, the systems might not recognize it quickly. This thesis studies how to apply data recency to collaborative filtering systems to get more predictive accuracy. We define the data recency problem as the negative impact of old data on the predictive accuracy of collaborative filtering systems. In order to mitigate this shortcoming, the combinations of time-based forgetting mechanisms, pruning and non-pruning strategies and linear and kernel functions, are utilized to apply weights. A clustering technique is employed to detect the user's changing preferences. We apply our research approach to the DeliBook dataset. The goal of our experiments is to show that our algorithm that incorporates temporal factors provides better recommendations than existing methods.

Keywords: Collaborative filtering system, Recommender system, Data recency problem, Time-based forgetting strategy, Time-based forgetting function.

ACKNOWLEDGMENTS

I wish to express sincere appreciation to my advisor Professor David C. Brown for introducing me to the field of collaborative filtering systems, and for his uncountable helpful comments and advice about this thesis. I would not have completed my thesis without his contributions.

I would like to thank my reader Professor Mark Claypool for his thorough review of this thesis as well as for many interesting discussions even though he has a lot of work to do.

I would like to thank my independent study advisor Professor Dewon Shon for his explanations that allowed me to clearly understand some techniques used in collaborative filtering systems.

Empirical research into collaborative filtering methods is not possible without suitable datasets. The empirical results presented in this thesis are based on the DeliBook dataset (www.delibook.com) that has been generously provided to me for research purposes. I would like to thank CEO Hyukjoon Yoo of the YoungJin E-Commerce in the Republic of Korea for granting me permission to use the dataset.

Finally, special thanks to my parents, my parents-in-law, and my wife Hyunju who have given me boundless support, encouragement, and motivation. Regrettably, I have to apologize to my lovely son Derrick D. because this thesis has not allowed me much time to play with him.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

Chapter 1
Introduction

Recent rapid expansions in computer use and an enormous increase in the popularity of the internet have motivated many companies to enter the on-line market, resulting in a huge expansion in the number of E-commerce sites. In order to be an outstanding E-commerce site, companies started to consider how to provide users with information about the most desirable items among their numerous products. As a result, recommender systems have been developed, and have been applied to many E-commerce sites.

Generally, there are two types of recommender systems. The early recommender systems used content-based filtering techniques. Content-based filtering systems build a profile of user preference by observing the behavior of an individual user to predict which information would be selected or rejected (Maneeroj et al., 2002). Each user is assumed to operate independently. The preferences are associated with the contents of items selected. As a result, such systems can exploit only information that can be derived from document contents: i.e., the systems are based on information about the content of items rather than on other, similar users' opinions.

On the other hand, collaborative filtering systems suggest items to a particular user based on a database of all user ratings. Collaborative filtering is the most successful recommender system technology to date, and can be used in the recommendation systems of many areas: web pages, documents, news, movies, books, CDs, DVDs, and so forth. Some of the best-known representative

commercial enterprises using these systems are as follows: Amazon.com™ (www.amazon.com), CDNOW.com™ (www.cdnow.com), and eBay.com™ (www.ebay.com).

1.1 Problem Statement

The challenge of collaborative filtering systems is to improve the quality of the recommendations for the users. Users need recommendations they can trust to help them find items they might like (Sarwar et al., 2000a). Many factors should be considered to enhance such predictions. In this thesis, we suggest that temporal factors should be one of the important considerations in collaborative filtering systems.

Existing collaborative filtering systems have focused on dealing with only the users' explicit or implicit data. They do *not* consider the data recency. That is, *when* users purchased the item does not affect the result of a recommendation. For this reason, if a user's interest has changed, the systems might not recognize it quickly. This means that the prediction becomes inaccurate until the system notices the change in the user interest.

We define the data recency problem as the negative impact of old data on the predictive accuracy of collaborative filtering systems.

Suppose that there is a user whose major was Computer Science in her undergraduate years, but she is an MBA student now. She bought a lot of books related to CS during her undergraduate years. However, now she needs to buy Management books because that is what she is studying. Basic collaborative filtering systems would still recommend CS books to her although she has

been buying only Management books recently.

With respect to Table 1.1, consider that we would like to predict user 0's (U_0) preference. There are 4 users and some items purchased by them. Items purchased are in categories A, B, C, and D, respectively. As you can see, user 0 purchased A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8, B_1, B_2, C_1, B_3, C_2, and C_3, over time. The number of the same items purchased by users 0 and 1, users 0 and 2, and users 0 and 3 is 6, 8, and 8, respectively. Whose preference is the most similar to user 0's? Existing collaborative filtering systems might say the answer is users 2 and 3 since user 0's correlation coefficient with user 2 is 0.37 and 0.22 with user 3. Both of which are greater than with user 1's where the value is -0.22. However, considering data recency, our answer to the question would be user 1.

The reason is that the point of the time when user 0 purchased items from category A could be too long ago to be considered with the same weight.

		TIME									
USERS		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
	U_0	A_1	A_2,A_3	A_4,A_5	A_6,A_7	A_8		B_1,B_2	C_1	B_3,C_2	C_3
	U_1	D_1	D_2		B_1		B_2	D_3,C_1	B_3,C_2	C_3	C_4
	U_2	B_1		B_2	A_3		A_4,A_2	A_1	A_7	A_8	A_9
	U_3	C_1		B_1,B_2	A_1	B_3	A_2,A_3		A_4	D_1	D_2

Table 1.1: A sample matrix of users versus items

In order to mitigate this shortcoming, in this thesis we provide techniques that uses data recency and demonstrate that it is an important factor to include in collaborative filtering systems. We propose several possible methods to apply to collaborative filtering systems, and carry out comparison experiments of these methods.

1.2 Thesis Organization

The rest of this thesis is organized as follows. Chapter 2 reviews the underlying algorithms of collaborative filtering systems, the identified problems with collaborative filtering, and previous research that uses temporal data. Chapter 3 presents our approach that incorporates temporal factors. It involves applying time-based forgetting strategies and functions, detecting user changing preferences, applying temporal factors based on the user preferences, and recommendation process. Chapter 4 depicts a particular implementation built using the algorithms described in Chapter 3. Some simplified cases, the experimental design using those cases, and evaluation matrices to show the improvement of predictive accuracy are showed in chapter 5. The results obtained in our experiments are given in Chapter 6. Finally, Chapter 7 consists of the concluding remarks and the future work.

Chapter 2

Background and Related Work

In this chapter, we briefly outline the main ideas of collaborative filtering algorithms that are reported in the literature, investigate the proposed problems in collaborative filtering, and introduce some time-related research.

2.1 Collaborative Filtering Algorithms

Collaborative filtering was developed in order to complement content-based filtering. It has improved over the past decade to the point where a wide variety of algorithms exist for generating recommendations. The systems predict items a particular user might prefer based on a database of ratings or purchases from all users.

In a collaborative filtering system, we first build a matrix of *m* users and *n* items which is a database of ratings or purchases. We then find the most like-minded users by comparing the target user's purchase pattern to others.

The general technique is based on Pearson's correlation coefficient which is used by Resnick et al. (1994) on their GroupLens project. The correlation between users a and b is (Breese et al., 1998):

$$corr(a,b) = \frac{\sum_i (r_{a,i} - \overline{r_a})(r_{b,i} - \overline{r_b})}{\sqrt{\sum_i (r_{a,i} - \overline{r_a})^2 \sum_i (r_{b,i} - \overline{r_b})^2}},$$

where the bars over *r* are the means for the ratings of users *a* and *b*, and the summations over *i* are over the items for which both users *a* and *b* have recorded ratings. The value of *corr*(*a,b*) ranges from -1 to 1. If their purchase patterns are similar, the value is positive; negative, otherwise. When they don't correlate, the value becomes 0.

Finally, using the most like-minded users, the algorithms recommend the N most frequent items that have not been purchased by the target user (Breese et al., 1998; Karypis, 2000; Sarwar et al., 2000a) in the top-*N* recommendation experiment; whereas the prediction experiment employs the following formula which is the prediction on the item j for user a (Resnick et al., 1994):

$$r_{a,j} = \bar{r}_a + \frac{\sum_x corr(a,x)(r_{x,j} - \bar{r}_x)}{\sum_x corr(a,x)},$$

where the summations over *x* range over all users who rated the item *j*.

2.2 Problems with Collaborative Filtering

While collaborative filtering systems have been a substantial success, there are several problems that researchers and commercial applications have identified:

a. *Early rater problem*. Pure collaborative filtering cannot provide predictions for an item when it first appears since there are no users' ratings on which to base the predictions. In order to avoid this problem, Sarwar et al. (1998) present filterbots which are automated rating robots that evaluate new documents as soon as they are published and enter ratings for those documents.

b. *Sparsity problem*. The number of items far exceeds what any individual can hope to absorb, thus matrices containing the ratings for all items for all users are very rare, and actual matrices are sparely filled. Usenet studies have shown a rating density of about 1% in some areas. We can estimate that few people will have read and formed an opinion on even 1/10 of 1% of the over two million books available through the largest bookstores (Sarwar et al., 1998). Thus, Claypool et al. (2001) suggest developing a curious browser which can collect user's implicit ratings such as mouse, scrollbar, and keyboards activities, as well as explicit ones.

c. *Large scale problem*. The more the number of users or items increases, the higher the computational complexity is. Sarwar et al. (1998) try to explore the relationships between items rather than the relationships between users. Hofmann (2003) describes a model-based algorithm designed with probabilistic latent semantic analysis.

2.3 Time-related Research

There are some researchers who have considered temporal factors as capable of playing a significant role to improve the accuracy in dealing with users' transaction data.

Chalmers et al. (1998) developed a system that recommends URLs based on the context of an individual's recent activity by using a path model. The path model is a developed approach to information access that stemmed from collaborative filtering. Like collaborative filtering, the path model primarily interprets information by means of the people who use it. The path is a time-ordered history of a person's use of information objects. Unlike collaborative

filtering, context or current activity is an essential part of the approach.

Billsus and Pazzani (1999) developed an intelligent agent, named NewsDude, which is designed to compile a daily news program for individual users. Based on feedback from the user, the system automatically adapts to the user's preferences and interests. The system has two separate user models: one represents the user's short-term interests; the other represents the user's long-term interests. The short-term model is learned from the most recent observations only. If the short-term model cannot classify the story at all, it is passed on to the long-term model. The purpose of the long-term user model is thus to model a user's general preferences for news stories and compute predictions for stories that could not be classified by the short-term model.

Koychev and Schwab (2000) presented a method for gradual forgetting, which is applied for learning drifting concepts, and claimed that the ability to adapt fast to the user current interests is an important feature for recommender systems. Concept changes (aka, drifting), whether gradual or abrupt, occur over time. The evidences for changes in a concept are represented by the training examples, which are distributed over time. Hence, the old observation can become irrelevant to the current period so that the learned knowledge could be out-of date. They experimented with recommendations for the users of ELFI (ELectronic Finding Information) which is a content-based filtering system that provides information about research funding, and with STAGGER which is an incremental learning system that dynamically tracks changes of concepts by applying gradual forgetting function.

Kukar (2003) suggested that many of the databases available for Statistical, Machine Learning and Data Mining algorithms violate the assumption that the data they use is a random sample drawn from a stationary distribution since

they were gathered over months or years, and the underlying processes generating them may have changed during this time, sometimes radically (i.e., concept drift). He reviewed several techniques for dealing with concept drift in Machine Leaning and Data Mining framework and evaluated linear and kernel functions for dealing with concept drift in clinical studies with a case study of coronary artery disease diagnostics.

Chapter 3

Proposed Approach

This thesis proposes a new technology designed to apply data recency to pure collaborative filtering systems to cope with the data recency problem. We claim that user preferences are not stationary and can drift with time. As time goes by, a user's preferences can vary depending on personal or social issues. Thus, her/his future selections of items should be more affected by her/his recent data than by past data.

In this thesis, we do *not* focus on the prediction of the exact rating a user would have given to a target item. We would much rather like to have a system that can recommend items that are more likely to be purchased by a user in the future. Thus, our approach is designed for E-commerce sites by producing a list of Top-*N* recommended items for a target user.

This chapter describes the concept of how to apply temporal factors to collaborative filtering systems. It includes the exploitation of time-based forgetting techniques, a clustering technique for detecting a user's changing preferences, and the combined approaches of these two methods. Also, a recommendation process is provided.

3.1 Applying Temporal Factors

In this section, we introduce some time-based forgetting strategies and functions. The purchasing behavior of the users can be seen as occurring over time.

The main idea is to treat the observations as time series data and to give more impact to more recent data by applying time-based forgetting strategies.

3.1.1 Time-based Forgetting Strategies

In general, there are two different strategies. One is to consider the recent data only, rather than the whole data (i.e., Pruning). The other is to use all the data (i.e., non-Pruning).

Applying only the most recent observations may lead to collaborative filtering systems that can adjust more rapidly to user changing interests. The simplest of methods presented by Billsus and Pazzani (1999) is to restrict a window size to the *n* most recently rated. The window is of fixed size, and the oldest observation is dropped whenever a new one comes in. However, this method seems to exacerbate the sparsity problem more than existing collaborative filtering systems. Thus, in this thesis, a window size varies depending on the user's current preferences. This is the more reasonable method than setting the window to a fixed size.

3.1.2 Time-based Forgetting Functions

Two functions are used, the linear and kernel functions. These functions provide each data item with a different weight based on its time of occurrence. Hence, they make the most recent observations more significant for our algorithms than the old ones. That is how we try to alleviate the fact that the old data may sometimes be detrimental to the predictive accuracy.

Koychev (2000) suggested using a linear gradual forgetting function, defined

as follows:

$$w_t = -\frac{2l}{n-1}(t-1)+1+l\,,$$

where t is a counter of observations starting from the most recent one and going back over time, n is the length of the observed sequence, and $l \in [0,1]$ is a parameter. By varying l, the slope of the linear function can be adjusted.

Another forgetting function, which is non-linear, is a kernel function suggested by Kukar (2003), defined as follows:

$$w_t = \frac{1}{\sqrt{2\pi}k} e^{-\frac{d^2}{2k^2}},$$

where $d=t/n$ is a relative time distance to the training example from the past, and $k \in [0,1]$ is a parameter. By varying k, the slope of the kernel function can be adjusted.

The graphs of the linear and kernel functions are given in Figures 3.1 and 3.2. In the linear function, as the parameter l increases, the slope gets steep. If we want to give nearly even impact to all observations, a lower value of parameter l should be given. Allowing the parameter l to be zero means giving no weight to any observation; whereas allowing the parameter l to be one means giving no weight to only the oldest data item.

On the contrary, for the kernel function, as the parameter k increases, the slope gets more gentle. If we would like to give the most impact to the most recent data, using a kernel function might be a better choice than using a linear function by giving the lowest value to the parameter k.

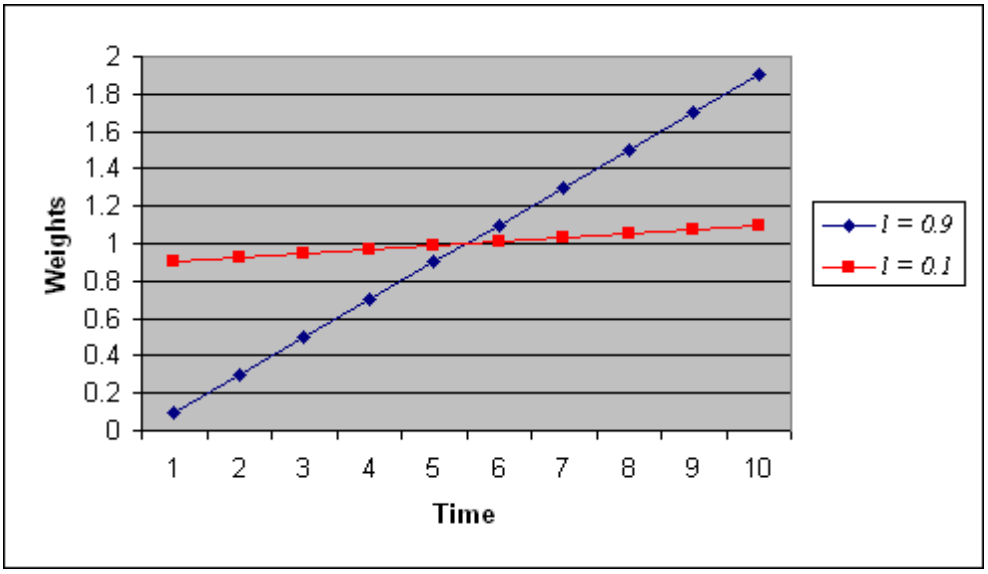

Figure 3.1: Linear function

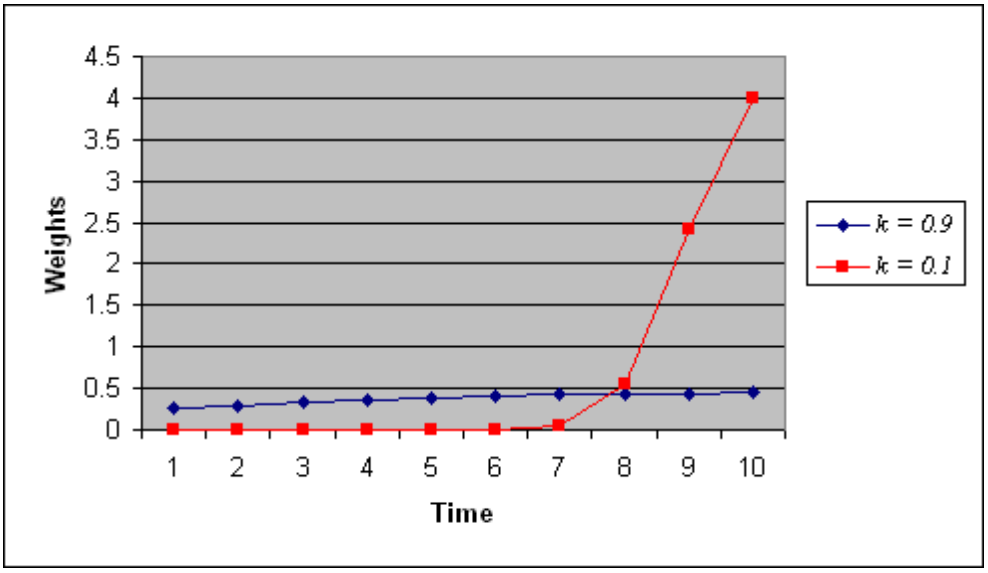

Figure 3.2: Kernel function

3.2 Detecting User Changing Preferences

The main idea in this section is that user interests can be tracked over time by

measuring the dissimilarities between the categories of a user's purchased items in order to identify the point in time at which the users changed their preferences. The time point will be used as a cut-off line in pruning or as a place to apply a different parameter in a non-pruning strategy.

Specifically, we are interested in being able to classify data items from each user into current or past clusters. The "current cluster" has items relevant to the user's current preference and the "past cluster" has items irrelevant to the user's current preference.

Schwab et al. (2000) employed a univariate significance analysis in their content-based recommendation component to determine a user's interest in specific values of the detailed view (DV) features. It is based on the idea that attribute values in random samples are normally distributed. If the value appears in the selected DVs significantly more frequently than in a random sample, the user is assumed to be interested in it. On the other hand, if the selection frequency is lower, the user is not interested in that value. However, they do not consider temporal factors in deciding interest.

By contrast, Crabtree and Soltysiak (1998) tried to derive user interest profiles automatically by monitoring user web and e-mail habits. They claimed that since user interests are not likely to remain constant, any interest learning algorithm must track changes in the user interests. To incrementally update the profile of a user, they employed a clustering algorithm which identifies interests, and then clusters them together to form interest themes. The main effort of their research is to see if the clustering approach to generating the user interests can be used to track changes in interests over time. They applied the clustering technique to a series of documents generated during a time period t, and then to a series of items generated in time period t+1. They then compared

clusters generated at time window t+1 with those generated in time window t by employing a dot-product for a similarity measure.

Our approach starts with applying a general time window method. To classify the items, we need to suppose that each item already has its own predefined category (e.g., Family, Health, Management, and etc.). All items purchased by user 0 shown in Table 1.1 are used as an example. A time window three-months wide is used for clustering. For our first run, items from months 1 through 3 inclusive are classified into each category. The time window is then advanced one month, and the next three months worth of items are classified (months 2 through 4 inclusive). This is repeated throughout the time period.

Table 3.1 shows the periods in which the purchased items appeared for the user 0. As we can see, each category appears or disappears over the examined period of time. In our approach, we disregard the item frequencies on each period since the item frequencies might affect the user's current interest more than the temporal factors. Thus, we only consider their two different features of appear and disappear which correspond to 1 and 0, respectively.

Month	**Category A**	**Category B**	**Category C**
1 ~ 3	✓		
2 ~ 4	✓		
3 ~ 5	✓		
4 ~ 6	✓		
5 ~ 7	✓	✓	
6 ~ 8		✓	✓
7 ~ 9		✓	✓
8 ~ 10		✓	✓

Table 3.1: Interests changing over time for user 0

We first identify categories in the most recent period, which are categories B and C. And then we go back to compare the lengths of periods of these two.

We now assume that user 0's preference is category B as it shows the largest continuous most-recent interest.

Next, we measure the dissimilarities between category B and the others. Calculating the dissimilarity between matching categories from one period to the next provides an indication of the time difference between two categories. This will help determine points at which interests changed.

The dissimilarity measure is given by using the Jaccard coefficient. The Jaccard coefficient is used in measuring the dissimilarity between two objects only for binary variables (Han and Kamber, 2001). It is defined as follows:

$$d(i, j) = \frac{r + s}{q + r + s},$$

where q is the number of variables that equal 1 for both objects i and j, r is the number of variables that equal 1 for object i but that are 0 for object j, and s is the number of variables that equal 0 for object i but that are 1 for object j. The dissimilarity value $d(i, j)$ ranges from 0 to 1. And, a higher value indicates more dissimilarity between the two.

We adopt this equation to our approach, where the categories take the role of the objects. The dissimilarity calculations are:

$$d(B, A) = \frac{3 + 4}{1 + 3 + 4} = 0.875,$$

$$d(B, C) = \frac{1 + 0}{3 + 1 + 0} = 0.250.$$

These measurements suggest that categories B and A are unlikely to have a

similar temporal pattern since they have a higher dissimilar value. Of course categories B and C are more likely to have a similar temporal pattern. As we can see in these results, we expect that s/he is currently interested in category C more than category A relative to category B. Therefore, we could estimate the first purchase point of time of categories B and C as the point of time of her/his changed interest, as shown in Figure 3.3.

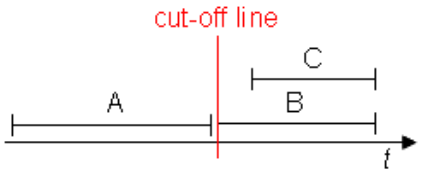

Figure 3.3: Determining a cut-off line

However, some worst cases may exist in this approach. Suppose that a user's interest is category A since it holds a lot of recently purchased items, and there is another category B that also holds recently purchased items but occupies much smaller period than category A, as shown in Figure 3.4. Intuitively, we can see two categories are not similar over the examined period of time (i.e., the dissimilarity value is not low). If we regard category B as a "past cluster", the co-existence of current and past occurs at some period and if this is the situation, the time period before this point should normally be a "past cluster". This is a clear violation of our assumption that category A is the user's interest. Therefore, it is more reasonable that category B as well as category A should be in the "current cluster" although the dissimilar value is not low.

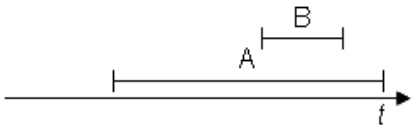

Figure 3.4: The worst case of interests changing over time

3.3 Applying Temporal Factors based on the User Preferences

Once the user's current preference has been detected it can be given more weight by the recommender system. We do this by using the time sensitive approaches described above in combination with the clusters of preferences. A variety of combinations of techniques are possible.

This section proposes some combined approaches using the techniques described in sections 3.1 and 3.2. The pruning or non-pruning strategies, clustering or non-clustering analysis, weights, linear or kernel function, or non-weights can all be varied.

3.3.1 Pruning Strategy

The first combined approach is to combine the pruning strategy and the time-based forgetting functions based on the user preferences. The cut-off line is not the same for every user, but different according to their preferences. It is fixed depending on the clustering technique by comparing the dissimilarity between categories on each user described in section 3.2.

Figures 3.5 and 3.6 show a user purchase pattern changing from category A to category B over time. After detecting the user's interests, we identify that category B is in her/his "current cluster"; whereas category A is in her/his "past cluster". We first assign the time of the first data item of category B to the cut-off line, and then apply the time-forgetting functions to observations from the cut-off line.

One or two parameters can be used to give weights to the observations in applying the time-based forgetting functions.

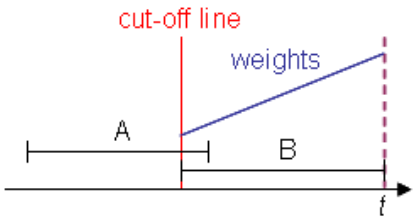

Figure 3.5: Applying a single parameter for pruning strategy

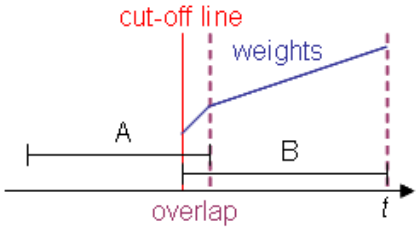

Figure 3.6: Applying two parameters for pruning strategy

The first method is to use a single parameter. In Figure 3.5, the cut-off line includes some data in category A as well as all the data in category B. One possible approach is to consider only the items in category B; whereas, the other is to consider some items in category A which overlap with category B, as well as all items in category B.

In the second case, we employ two parameters. In Figure 3.6, the observations from the cut-off line can be divided into two periods, overlapped and non-overlapped. We apply different parameters to each period, respectively, since the overlapped period may be regarded as the period of interest changing from category A to category B.

However, applying two parameters rather than one may cause an excessive

overlap with the period of a “current cluster” given the worst case described in section 3.2. Its result is that too many items in a “current cluster” could be given lower weights.

3.3.2 Non-Pruning Strategy

The second method is not to use pruning, but to use the time-based forgetting functions based on the user preferences. Figures 3.7, 3.8, and 3.9 show the parameters varied into two or three values depending on a user’s interests. That is why we would like to give different impact to each period.

Once all observations are classified into cluster groups, we can divide the periods into two or three parts. If these two groups are separate as shown in Figure 3.7, we apply two different parameters to the recent and the past group, respectively. Of course, the first weight of the first group should be scaled to be less than the last weight of the second group.

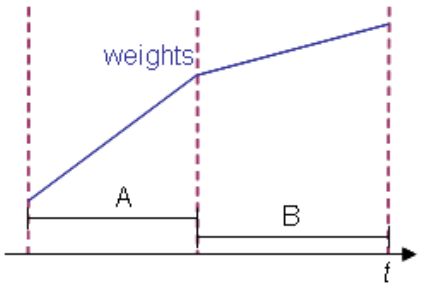

Figure 3.7: Applying two parameters to separate case for non-pruning strategy

On the other hand, in Figures 3.8 and 3.9, two groups are somewhat overlapped. The observations can be divided into three periods, the recent, the overlapped and the past periods. We can use two or three parameters to these periods, as shown in Figures 3.8 and 3.9, respectively. Of course, we need to

adjust the first weights of each period to be less than the last weight of the previous period. However, applying three parameters may also suffer from the excessive overlapping problem in the worst case, as mentioned before.

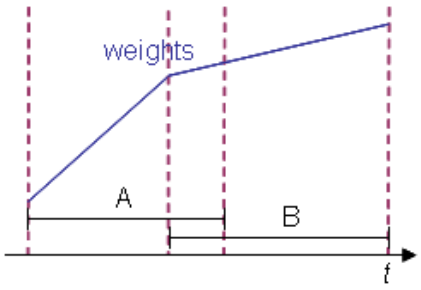

Figure 3.8: Applying two parameters to overlapped case for non-pruning strategy

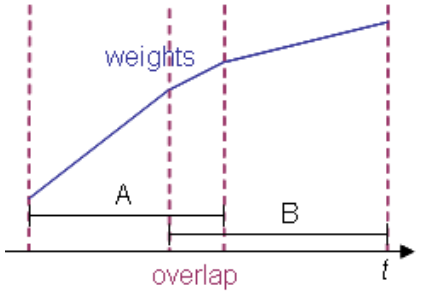

Figure 3.9: Applying three parameters to overlapped case for non-pruning strategy

3.4 The Recommendation Process

We have described how to apply temporal factors, detect changing user interests, and combine these techniques. Now, we look into the procedure of the recommendation with the techniques in detail. There are three steps: Building matrices, Neighborhood formation, and Generation of recommendation. Each procedure of our algorithm follows the general steps of typical collaborative filtering systems.

3.4.1 Building Matrices

The first step is to build matrices which are used to find like-minded users. In a typical collaborative filtering system, each user's transaction data can be represented by the items a user has purchased. Just as for a collaborative filtering system, our process starts with construction of an original matrix whose columns are associated with item $i \in I=\{i_1, i_2,\ldots, i_N\}$, and whose rows are associated with user $u \in U=\{u_1, u_2,\ldots, u_M\}$. If a user a purchases an item i, we use one as the element $r_{a,i}$ of the original matrix, and zero otherwise, which is called a Binary feature vector (Huang et al., 2004; Karypis, 2000).

$$r_{a,i} = \begin{cases} 1, & \text{if a user } a \text{ purchased an item } i, \\ 0, & \text{otherwise.} \end{cases}$$

We now build a weighted matrix that holds an element $w_{a,i}$ that is a weight corresponding to the element $r_{a,i}$ in the original matrix. When using the pruning strategy, even though a user purchased an item, if the item is outside the calculated boundary of her/his interests, the element $w_{a,i}$ should be zero; whereas in the other cases, the element $w_{a,i}$ should have a non-zero value, which is calculated by using the time-based forgetting functions.

We then produce a combined matrix by multiplying these two matrices. Its elements contain:

$$p_{a,i} = r_{a,i} \times w_{a,i},$$

where $r_{a,i}$ is in the original matrix and $w_{a,i}$ is in the weighted matrix.

3.4.2 Neighborhood Formation

The most important phase is to measure the similarities between users, as it is used to form a proximity-based neighborhood between a target user and a number of like-minded users. The neighborhood formation process is the learning process for a collaborative filtering system algorithm. The main goal of neighborhood formation is to find for each user u an ordered list of l users $N=\{N_1, N_2, \ldots, N_l\}$ such that $u \notin N$ and $corr(u, N_1)$ is maximum, $corr(u, N_2)$ is the next maximum, and so on.

In our algorithm, we utilize the adjusted Pearson's correlation coefficient by using $p_{u,i}$, which is produced at the building matrices stage, instead of using $r_{u,i}$ in the original matrix, which is given by:

$$corr(a,b) = \frac{\sum_i (p_{a,i} - \overline{p_a})(p_{b,i} - \overline{p_b})}{\sqrt{\sum_i (p_{a,i} - \overline{p_a})^2 \sum_i (p_{b,i} - \overline{p_b})^2}}.$$

3.4.3 Generation of Recommendation

The final step is to derive the top-N recommendations from the neighborhood of users. There are two different techniques: Most-frequent item recommendation and Association rule-based recommendation. In this thesis, we use the most frequent item recommendation method since it can quickly and easily generate the list of the top-N recommendations.

The method looks into the neighborhood N and for each neighbor scans through her/his purchase data and performs a frequency count of the items. When using the pruning strategy, the items eliminated by the cut-off line

should not be counted. After all neighbors are accounted for, the system sorts the items according to their frequency count and simply returns as the recommendation the *N* most frequent items that have not yet been purchased by the target user. The value of *N* could be determined and varied depending on the results of each experiment.

Chapter 4

Implementation

This chapter introduces how to construct our algorithms that incorporated temporal factors into collaborative filtering systems. Some pseudocode is also presented. We have not implemented a complete E-commerce site, but designed and developed the heart of a collaborative filtering system, calling it the Temporal Factors Collaborative Filtering (TFCF) system.

The TFCF system was implemented using C# which is run on the Microsoft .NET Framework. It was tested on a Windows XP Professional based PC with Intel® Pentium 4 processor having a speed of 2.00 GHz and 512 MB of RAM. SQL Server 2000 is employed for the database to build the user/item matrices and to store recommended items, and it is connected via ADO.NET.

The TFCF system is broken into three parts: Graphical User Interface, Database, and Algorithms. Figure 4.1 depicts the architecture of the system.

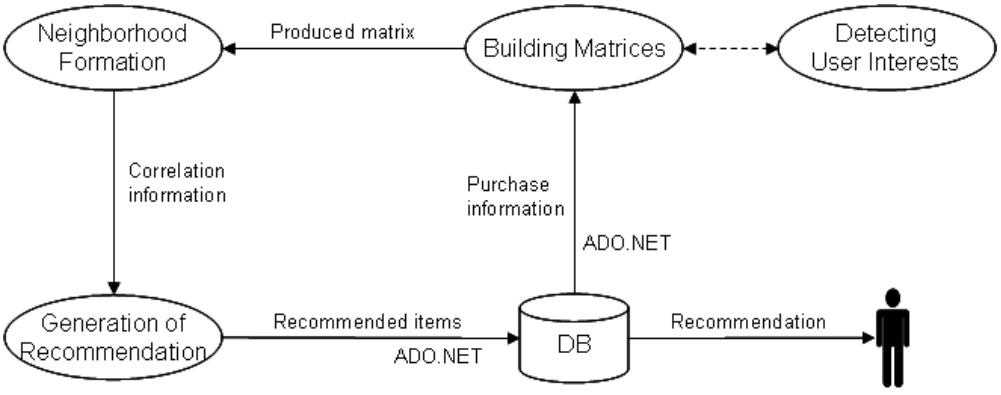

Figure 4.1: The TFCF system architecture

The ovals represent the recommendation steps, and the dotted arrow means the conditional process. The process “Detecting User Interests” is executed only when either the pruning strategy or the non-pruning strategy using two parameters are selected. Consequently it is connected with a dashed line in the Figure.

4.1 Graphical User Interface

The Graphical User Interface (GUI) of the TFCF system is Window-based, as shown in Figure 4.2, in order to provide convenient usage. The interface was designed to allow control of experimentation with the techniques developed. It allows the user to select which techniques and data are to be used in an experiment.

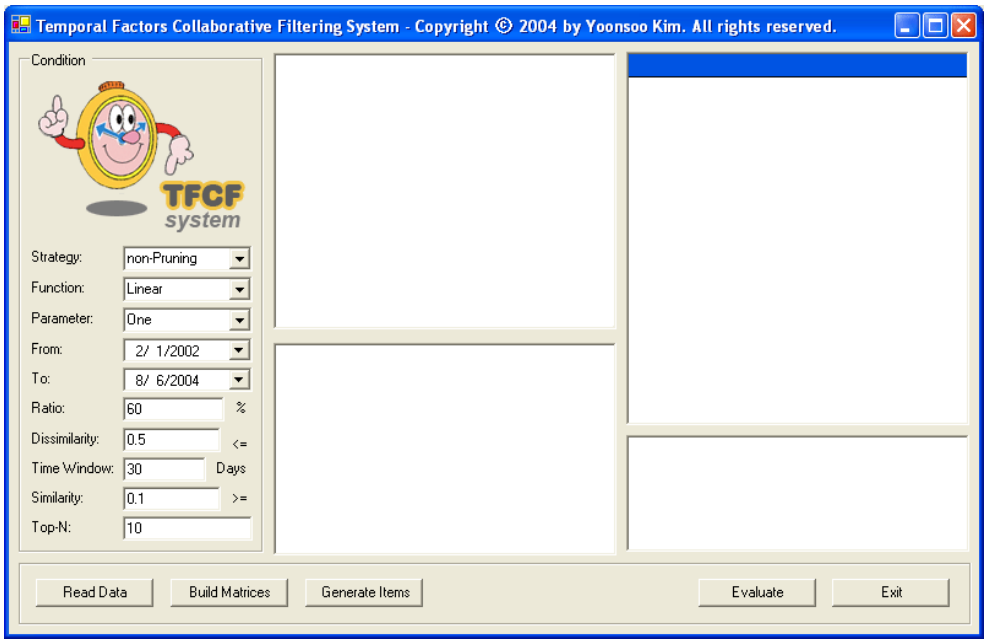

Figure 4.2: The GUI display of the TFCF system

Ten menus are needed since we evaluate our algorithms with various combinations of techniques. The responsibilities of each menu are as follows:

- The “Strategy” menu is for Pruning and non-Pruning.
- The “Function” menu is for Existing, Linear, and Kernel.
- The “Parameter” menu is for One and Two which are the number of parameters used in time-based forgetting functions.
- In the “From” and “To” menus, we can easily decide the periods of the data by choosing specific dates from the real calendars.
- The “Ratio” menu enables us to divide the dataset into the training and test data depending on the value.
- The “Dissimilarity” and “Time Window” menus are used in the cluster analysis to detect the user’s current preference.
- The “Similarity” menu is used in determining the like-minded users for a specific target user.
- The “Top-N” menu is for setting the value *N* of the number of items to be recommended in the generation of recommendation phase.

After determining all of the techniques from those menus, the user pushes the buttons in the bottom of the display. First, clicking the “Read Data” button connects the database and retrieves the order transaction data from the database. Next, the “Build Matrices” button is for building the original, weighted, and produced matrices. The “Generate Items” button is for generating and recommending items to target users. Clicking the “Evaluate” button enables us to

check the predictions using the evaluation metrics of our algorithm discussed in chapter 5. Finally, we can terminate the system by pushing the “Exit” button.

In addition, there are four display areas on the right side of the menus. These boxes show the results of each step of the recommendation process. Figure 4.3 is a screen shot of the TFCF system with results.

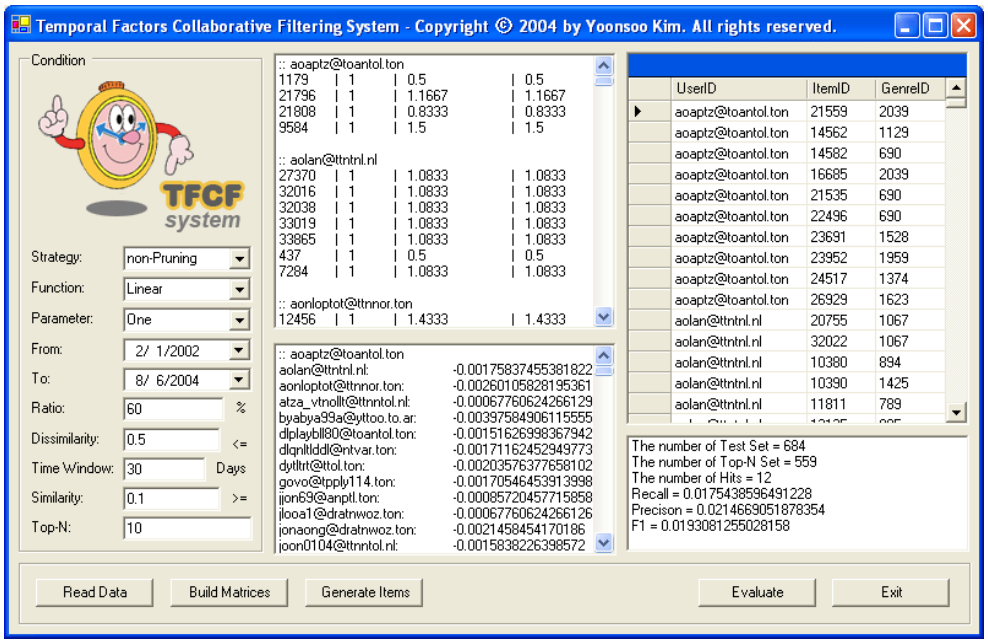

Figure 4.3: A screen shot of the TFCF system with results

4.2 Database

Order information and the list of recommendations are stored in a SQL Server 2000 database. The GUI and algorithms connect to the database via ADO.NET. ADO.NET represents a new approach for building applications that interact with databases. It also provides a powerful, flexible, and appropri-

ate method for accessing data in modern applications that are more widely distributed than with previous technologies, such as ADO, OLE DB, RDO, DAO, and ODBC.

4.3 Algorithms

In this section, we present some pseudocode that describes the core of the TFCF system. The algorithms are written in C# which is an object-oriented language. C# is the next phase in the evolution of C and C++, and was developed expressly for Microsoft's .NET platform.

Clicking the "Read Data" button is used to read the dataset from the database. Before reading, we need to calculate the ratio depending on the dates of the "From" and "To" menus in order to divide the data into the training and test sets. And, then we make the indexes of users, items, and genres data by retrieving the data from the database. The indexes are used as keys in the user/item matrices.

We now build three user/item matrices by pushing the "Build Matrices" button. The original matrix, whose columns are associated with the index of item and whose rows are associated with the index of user, can be built easily. And then we need to calculate the weights by using the linear or kernel functions. Calculating the weights is varied depending on the "Strategy" and "Parameter" menus. If we choose Pruning from the "Strategy" menu, we need to identify a cut-off line. If Two from the "Parameter" menu and non-Pruning from the "Strategy" menu are selected, we need to identify the point at which a different parameter is applied. It is achieved by measuring the dissimilarity value using the Jaccard coefficient (see Algorithm 4.1).

```
private void JaccardCoefficient()
{
   Set q, r, and t to 0

   While not at end of category list
      While not at end of time window list
         If clusterArray (current category, time window) is 1 Then
            If clusterArray (category, time window) is 1 Then
               Increment q
            Else
               Increment r
            End If
         Else
            If clusterArray (category, time window) is 1 Then
               Increment s
            End If
         End If
      End While
   End While

   Compute dissimilarity as (r+s) / (q+r+s)
}
```

Algorithm 4.1: Jaccard coefficient

We now know in which categories the user is interested currently. We can build the weighted matrix by applying the weights to each observation in the current cluster with one parameter when we choose the pruning strategy, or in the current and past clusters with two parameters when we choose the non-pruning strategy.

After producing these two matrices, we find the neighborhood users by measuring similarities between a particular user and others using the Pearson's correlation coefficient (see Algorithm 4.2).

Finally, we store recommended items into a Recommendation table of the database. The recommended items are given by the Top-*N* algorithm which counts frequencies on items purchased by the neighborhood users and adopts

the top-*N* frequent ones (see Algorithm 4.3).

```
private void ComputeCorrelation()
{
   Compute the means of users

   While not at end of target users list
      r_a ← mean of target user
      While not at end of training users list
         r_b ← mean of training user
         While not at end of items list
            a_i ← Subtract r_a from the value of matrix
            b_i ← Subtract r_b from the value of matrix
            Add saa ← a_i^2
            Add sbb ← b_i^2
            sab ← Multiple a_i by b_i
         End While
      End While
   End While

   Compute corr as sab / sqrt(saa * sbb)
}
```

Algorithm 4.2: Pearson's correlation coefficient

```
private void GenerateTopN()
{
   While not at end of target users list
      While not at end of training users list
         If training user is closest neighbor to target user Then
            While not at end of items list
               Increment frequentArray (item)
            End While
         End If
      End While
      Sort frequentArray
      While not at end of items list
         If item is not purchased by the target user Then
            Store the item into recommend table
         End If
      End While
   End While
}
```

Algorithm 4.3: Top-*N* algorithm

4.4 Miscellaneousness

To build the original matrix, an $m \times n$ size of array is needed which is the case for typical collaborative filtering algorithms. However, the TFCF system needs three matrices of the same size for the original, weighted, produced matrices. The larger the number of users and items is, the higher the possibility of getting an out-of-memory error. Unfortunately, the TFCF system could suffer from the complexity problem more than existing collaborative filtering systems.

It is very important that the system is bug-free in order to get a good result. Hence, we have debugged the system by comparing the calculations of each step in the system with those done by hand using an Excel tool.

Chapter 5
Experimental Evaluation

This chapter provides the five simple cases of user purchase patterns and demonstrates the design of experiments with these cases. We also review some methodologies to evaluate the recommendation performance of our algorithms.

5.1 Datasets

Empirical research into collaborative filtering methods is not possible without suitable datasets. A lot of empirical research has used the MovieLens or EachMovie dataset.

The MovieLens dataset were collected by the GroupLens Research Project. This dataset consists of 100,000 ratings (1~5) from 943 users on 1,682 movies. Each user has rated at least 20 movies. The data was collected through the MovieLens web site (movielens.umn.edu) during 17 months period from September 19, 1997 through April 22, 1998. The features for the movies are: Unknown, Action, Adventure, Animation, Children's, Comedy, Crime, Documentary, Drama, Fantasy, Film-Noir, Horror, Musical, Mystery, Romance, Sci-Fi, Thriller, War, or Western. The dataset has three tables, such as User, Item, and Data.

The EachMovie dataset were colleted by the Compaq Systems Research Center for 18 months from 1995 through 1997. There are 72,916 users and

2,811,983 ratings (0.0~1.0) for 1,628 different movies. The features for the movies are: Action, Animation, Art/Foreign, Classic, Comedy, Drama, Family, Horror, Romance, or Thriller. The dataset has three tables, such as Person, Movie, and Vote.

Although these two datasets have the time information of rating on items, the datasets may not be pertinent enough for our thesis to expect an improvement in accuracy of recommendations. This is due to the fact that the time information is not the date on which people rate the movies immediately after choosing or watching them. That is, a lot of people rate almost all items on the same day or within a short period. The consequence is that it was impossible to determine the change in users' preferences, as they are not spread out over time.

Therefore, the dataset we utilized in our experiments is from DeliBook which is a real dataset currently used in the on-line bookstore (www.delibook.com) debuted in 2002 and managed by YoungJin E-Commerce. The dataset is being used with their permission under certain privacy restrictions. There are 5,392 users, 9,204 books, and 17,479 order transactions for about 30 months from February 1, 2002 through August 6, 2004 in their dataset.

The categories for the books are: Family, Health, Management/Economy, Students, Cartoon, Literature, Dictionary, Social Science, Test, Travel, History/Culture, Foreign Language, Kids, Humanities, Biography, Self-development, Nature/Science, Magazine, Professionals, Religion, Textbooks, Youth, Recommendations, Hobby/Sports, Computer/Internet, or Original Texts. The dataset has six primary attributes: MemberID, BookID, BookName, GenreID, GenreName, and OrderDate.

Figures 5.1, 5.2, and 5.3 are the histograms of the distributions of data in the

DeliBook dataset. Figure 5.1 depicts the number of purchased items for each month. As we can see, the dataset seems to have sufficient items ordered per month for use in our experiments, except for the first 6 months and the last month. The average number of purchased items is about 583 in each month.

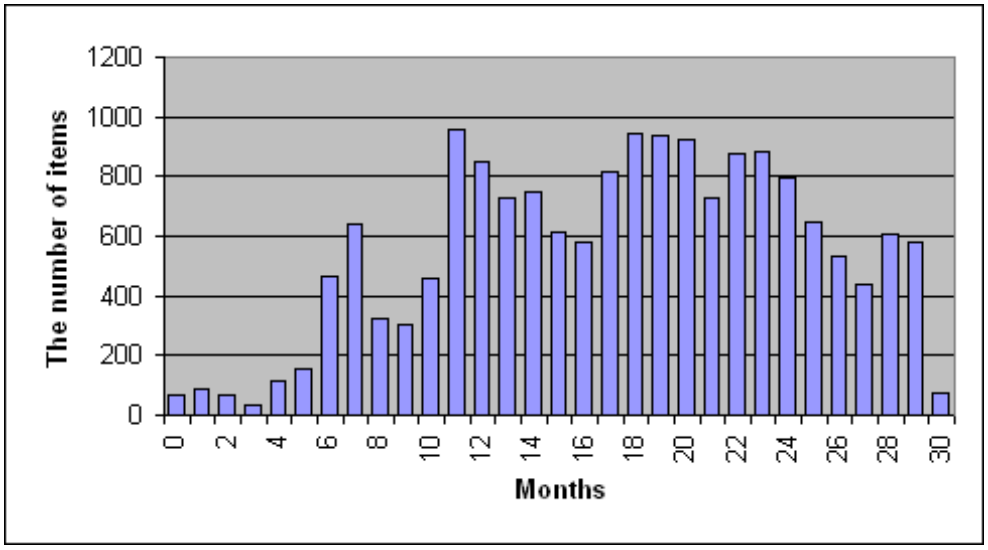

Figure 5.1: The number of purchased items in months

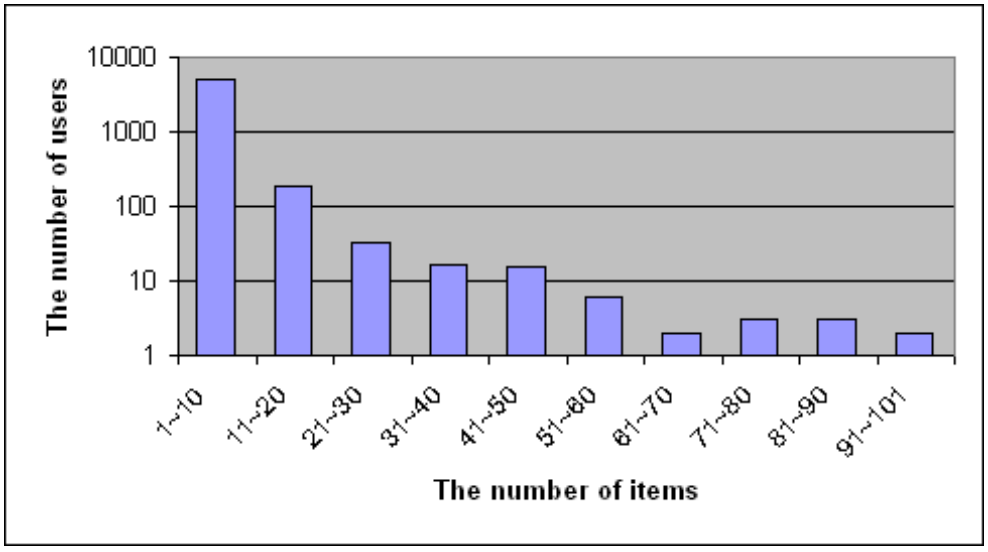

Figure 5.2: The number of users versus number of items purchased by them

Figure 5.2 shows that the number of users and the number of items purchased by them. Note that the *y* axis is shown in log scale. Each user purchased about 3 items on average, and the number of users who purchased at most 10 items is 5,131 which are 95% of all users.

Figure 5.3 shows the number of users ordered by the difference between the earliest and latest dates of order for each user. The value 0 means that the users purchased items within one month. The number of users who have the value 0 is 4,569. Among them, 4,287 users, 80% of the dataset, purchased items in one day.

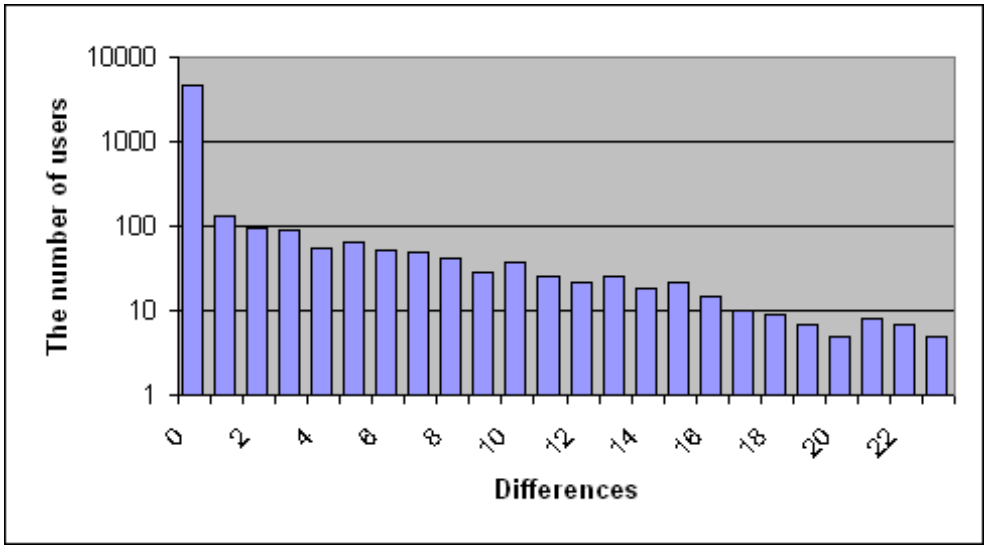

Figure 5.3: The number of users in month differences

Just as with many other datasets, this dataset seems to have a sparsity problem. As mentioned before, the dataset has 5,392 users, 9,204 books, and 17,479 order transactions. Thus, the size of the user/item matrix is 5,392 x 9,204 elements. The MovieLens and EachMovie datasets are 93.70% and 97.63% sparse, respectively; whereas the DeliBook dataset is 99.97% sparse since the ratio of the number of non-zero elements versus the size of the matrix is

17,479 / 49,627,968. This value means that the dataset might be insufficient to identify similarities in users' interests and moreover insufficient to recommend frequent items. However, although the DeliBook dataset is probably too sparse to be perfect for our thesis, we could not help but use this dataset, since in spite of our every endeavor, it was the only appropriate dataset that we obtained permission to use in our research.

5.2 Evaluation Cases

In this section, we describe some simplified cases of the distribution of data over time to be used in detecting user changing interests. These cases are intended to express the simple patterns of user transactions data. We deal with only the cases of two and three categories of user purchases in order to give a precise explanation.

Suppose that a user purchased items in categories A and B over time. Figure 5.4 (Case 1) shows two distributions that are separate; whereas Figure 5.5 (Case 2) shows two distributions that are close. We might expect a similar result from these two cases since we assume that s/he now likes the category of recently purchased items. We could thus declare that s/he no longer likes category A, but is now interested in category B.

Figure 5.4: Case 1: A and B separate

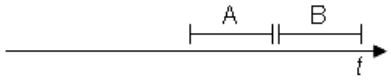

Figure 5.5: Case 2: A and B close

With respect to Figure 5.6 (Case 3), two distributions are somewhat overlapped. We could say that s/he is interested in category B; whereas her/his degree of interest in category A is based on the result of measuring their similarities over time. On the other hand, two distributions are completely overlapped in whole periods in Figure 5.7 (Case 4). Obviously, s/he is interested in both categories.

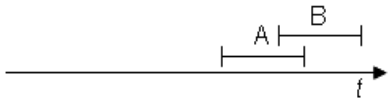

Figure 5.6: Case 3: A and B somewhat overlapped

Figure 5.7: Case 4: A and B completely overlapped

There exist three distributions in Figure 5.8 (Case 5). The distribution of category A is separated from those of category B and C; whereas the distributions of category B and C are overlapped. We might say that s/he no longer likes category A, but is interested in category B depending on the result of measuring their similarities over time, and is definitely interested in category C.

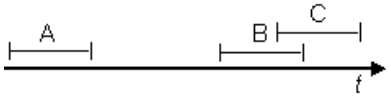

Figure 5.8: Case 5: A in the left side; B and C in the right side

Besides these above cases, we might also consider the cases of four or more categories, but we would like to leave the more intricate cases for future work. In addition, we expect the more complex cases to be composed of the simple

cases discussed above.

5.3 Experimental Design

This section describes how to design the experiments used to evaluate the accuracy of our research approach. It also explains how to select training and test data (including target data) from the DeliBook dataset, as well as how to determine the dissimilarity values and the size of the neighborhood in the top-*N* recommendation.

We evaluate the pruning and non-pruning strategies, respectively, as shown in Table 5.1. All the testing is compared with the plain test of the existing collaborative filtering algorithm under the same experimental conditions in order to determine how much impact temporal factors have.

Method		**Cases 1,2**	**Case 3**	**Case 4**	**Real**
Existing algorithm					
Pruning Strategy	No Weights				
	Linear Function - with one parameter				
Non-Pruning Strategy	Linear Function - with one parameter - with two parameters				
	Kernel Function - with one parameter				

Table 5.1: The experimental design chart

When using the pruning strategy, the experiments of applying no weights and applying the linear function with a single parameter are provided. We compare these two cases to see how similar or different their results are.

When using the non-pruning strategy, the linear and kernel functions with a

single parameter are tested. Also, after clustering items on each user depending her/his current preference we apply two parameters to the linear function in the experiment.

In order to evaluate the quality of our algorithm, we selected 823 users who purchased items over more than one months' period since there is no point in applying the time sensitive approach to the data of users who purchased items within one month. We split the data into training and test sets. Unlike typical collaborative filtering research, we divided the data on the basis of the order time instead of the amount. We used 60% as the training set to build a predictive model, and the remaining most recent 40% was used to evaluate the predictive accuracy of the model. Among the training and test data, we retrieved 69 users who purchased at least 10 items in 12 or more months as target users.

We also employed some "fake data" as target users. The artificial data represents Cases 1/2, 3 and 4 described in section 5.2 in order to show obvious examples and verify our approach. Each case has the same users and items, but different order dates corresponding to the four cases. We randomly selected items whose frequencies are three or four in order to reduce the frequent bias. And, every item was used only once without duplication to prevent the users of the "fake data" from being each other's neighborhood. We allowed the last two observations of each user to be a test set. Ten users' data per case are given and are shown in Appendices.

The dissimilarity value in a clustering technique plays a significant role: in the pruning strategy it is used to determine the cut-off line and in the non-pruning strategy it is used to apply the appropriate value of the parameters to data. As we adopt a lower dissimilarity value, more items are included in a current cluster; on the other hand, a higher dissimilarity value limits her interest to only

the latest purchased categories. Thus, we first tried dissimilarity values from 0.8 in decrements of 0.1. We realized that when applying 0.6 or more the results were similar to that when using the non-pruning strategy. In addition, all the cases of applying 0.5 or less got the same results. Hence, in our experiment, we set 0.5 as the dissimilarity value to see the apparent differences between the results of the pruning and non-pruning strategies.

A time window three-months wide is used to detect user preferences for the clustering. Since we selected users who purchased within more than one month as the training set, the shortest purchasing gap can be one to two months. Hence, even if the gap is the shortest, the window width allows the user's data to have at least two time periods corresponding to rows in Table 2. And, it also enables the Jaccard coefficient to be applied.

The size of the neighborhood has significant impact on the recommendation quality (Herlocker et al., 1999). It also should be determined by varying its size in experiments. Thus, we first tried from 0.6 in decrements of 0.1 as the size of the neighborhood. However, we identified that when the value is greater than or equal to 0.4, no items were recommended due to the sparsity problem. Therefore, we realized that setting 0.1 as the value gets better recommendations than setting 0.2 or 0.3, and it is appropriate to recommend 10 items to the target users in the top-*N* recommendation.

5.4 Evaluation Metrics

In this section, we discuss some techniques used for testing the accuracy of our algorithms. Generally, there are two different accuracy metrics. One is predictive accuracy metrics that empirically measure how well a system can predict

an exact rating value for a specific item. The other is classification accuracy metrics that measure how close a system's predicted ranking of items for a user differs from the user's true ranking of preference (Herlocker et al., 2004).

In this thesis, we are not interested in the most accurate prediction of an exact rating which a user would have given to the target item. Rather we would like to have a system that can accurately recommend items that are liked by the user. Therefore, we use three metrics often used to evaluate classification accuracy, namely, Recall, Precision, and F1, to determine the quality of collaborative filtering systems using data recency techniques.

Classification accuracy metrics measure the frequency with which the recommender system makes correct or incorrect decisions about whether an item is good (Herlocker et al., 2004). The metrics are thus relevant for tasks such as finding good items when users have true binary preferences. Recall represents the probability that a relevant item will be selected, and precision represents the probability that a selected item is relevant. The higher the recall and the precision, the more accurately the recommendations classify.

	Relevant	**Irrelevant**	**Total**
Recommended	*a*	*b*	*a+b*
Not Recommended	*c*	*d*	*c+d*
Total	*a+c*	*b+d*	*a+b+c+d*

Table 5.2: The classification of actual and recommended items

The recall is defined as:

$$\text{Recall} = \frac{\text{Number of relevant items recommended}}{\text{Total number of relevant items in the database}} = \frac{a}{a+c}.$$

The precision is defined as:

$$\text{Precision} = \frac{\text{Number of relevant items recommended}}{\text{Total number of items recommended}} = \frac{a}{a+b}.$$

The main goal of our evaluations is to look into the test set and match items with our top-*N* set. Hence, Sarwar et al. (2000a) rewrote recall and precision, as follows:

$$\text{Recall} = \frac{\text{test} \cap \text{top-}N}{\text{test}},$$

$$\text{Precision} = \frac{\text{test} \cap \text{top-}N}{N},$$

where test is the data excluding training data, and top-*N* is the data in a Recommendation table from the database.

Precision and recall depend on the separation of relevant and non-relevant items. They are less appropriate for domains with non-binary granularity of true preference.

These two measures are, however, often conflicting in nature (Sarwar et al., 2000a). For instance, increasing the number *N* in the top-*N* set tends to increase recall but decreases precision. The fact that both are critical for the quality judgment leads us to use a combination of the two. Several approaches have been taken to combine recall and precision into a single metric. One approach is the F1 metric which combines precision and recall into a single number (Herlocker et al., 2004; Sarwar et al., 2000a). The value of F1 ranges from 0 to 1, with a higher value indicating the best performance. Here we assign equal importance to recall and precision, resulting in the following definition for F1:

$$F1 = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}.$$

The recall, precision, F1 metrics are important measures in evaluating collaborative filtering algorithms. Among these metrics, we believe that the most important measure is the recall metric when considering from the sales' perspective since no matter how many matching items the system actually recommended, the actual number of purchased matching items is what counts. However, the user cannot look at too many recommendations, hence precision matters.

Chapter 6
Experimental Results

In this chapter, we present our experimental results regarding our techniques for generating recommendations. Our goal is to evaluate the qualities and performances of the methods provided by the various combinations described in chapter 3 by comparing existing algorithm. As mentioned before, we experimented with data from two different target users: i.e., the fake and real data.

6.1 Using Fake Data

We first tried the experiments with fake data that was made to fit selected temporal cases (see Appendix). Figures 6.1, 6.2, and 6.3 show the results of the recall, precision, and F1 when using Cases 1/2, 3, and 4 in section 5.2, respectively. Cases 1 and 2 have two categories that are not overlapped. Case 3 has two categories are somewhat overlapped. Case 4 has two categories completely overlapped.

As we can see in Figure 6.1 which shows the results when using Cases 1 and 2, the recall, precision, and F1 of our methods have improved 0.0% to 20.0%, 0.0% to 4.0%, and 0.0% to 6.7% with average improvements of 13.0%, 2.7%, and 4.4%, respectively. All of our methods except the pruning strategy with the linear function got better accuracy performances than existing algorithm. We can identify that when users' preferences have changed apparently without overlapped categories, considering only the recent data and applying weights

can give better performances. However, we know that applying weights to the recent data could give an adverse effect on predictions since it could give less impact than applying weights to all observations.

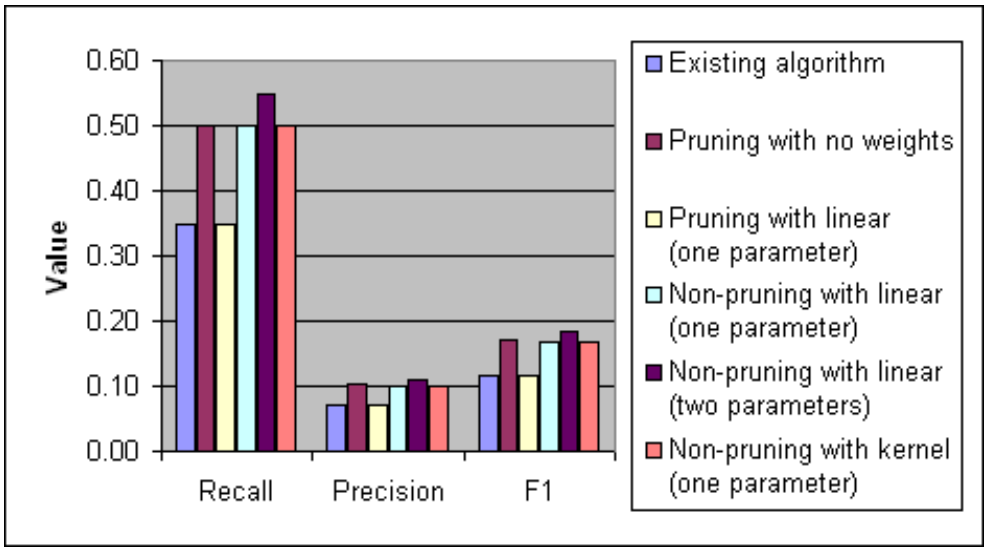

Figure 6.1: The experimental results of cases 1 and 2

With respect to Figure 6.2 which is the results when using Case 3, the recall, precision, and F1 of our methods have improved 0.0% to 20.0%, 0.0% to 4.0%, and 0.0% to 6.7% with average improvements of 10.0%, 2.0%, and 3.3%, respectively. Similar to cases 1 and 2, all of our methods except the pruning strategy with the linear function got better performance accuracy than the existing algorithm. The non-pruning strategy with the linear function and with the kernel function using one parameter have improved less than those of cases 1 and 2 compared to the existing algorithm; whereas the pruning strategy with no weights and the non-pruning strategy with linear using two parameters still give better performance. That is because in the overlap period, the items of the users' current preferences can be given less weight than the items of the user' past preferences.

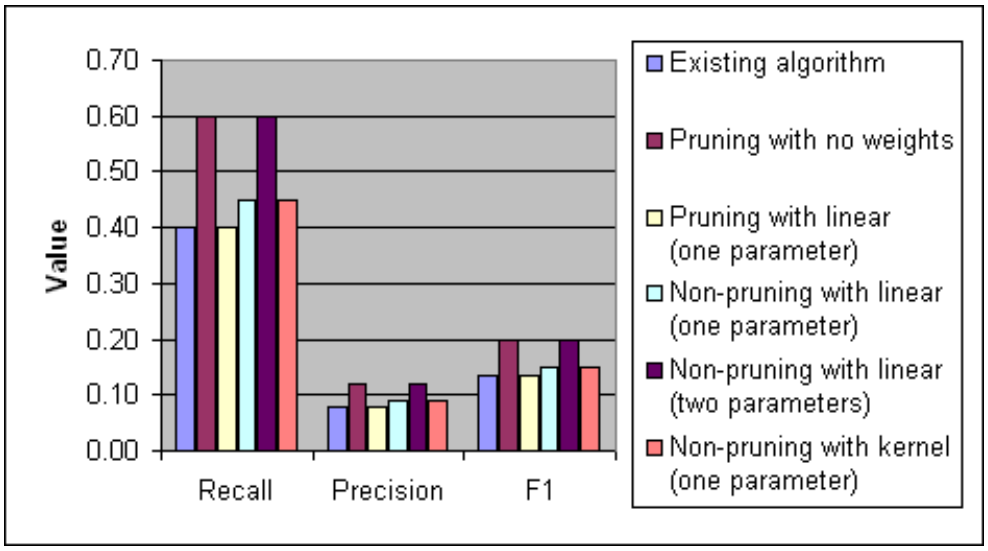

Figure 6.2: The experimental results of case 3

Looking at Figure 6.3 which is the results when using Case 4, the recall, precision, and F1 of our methods have improved 0.0% to 10.0%, 0.0% to 2.0%, and 0.0% to 3.3% with average improvements of 6.0%, 1.2%, and 2.0%, respectively. Only the pruning strategy with the linear function using one parameter and the non-pruning strategy with the linear and kernel functions using one parameter got better performance accuracy than the existing algorithm. We cannot see any differences in the pruning strategy with no weights and the non-pruning strategy with linear using two parameters compared to the existing algorithm. The reason is that the users' preferences have not changed. It means that there are no items outside of the cut-off line since the users are interested in both categories.

To make a long story short, through our experiments with the fake data we identified that the results of using the linear and kernel functions are almost the same and they can also give better performance regardless of the change in users' preferences. Particularly, in the case where users' preferences have ap-

parently changed, considering the most recent items or applying different parameters to their current and past clusters can be expected to perform the best predictive accuracy.

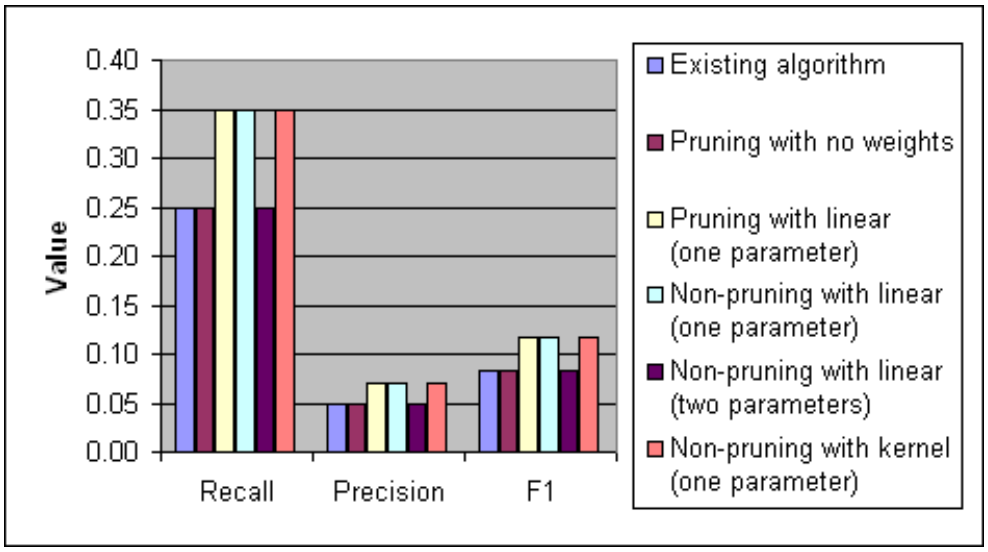

Figure 6.3: The experimental results of case 4

6.2 Using Real Data

Next we experimented with real data. In order to ensure that our results are statistically accurate, for each of the experiments we perform three different trials by varying the amount of the dataset used: the first 60%, the first 80%, and 100% of the data. The results are the averages over these three runs. The other conditions are given with the same as previous experiments.

Looking at the results in Figures 6.4, 6.5, and 6.6, the recall, precision, and F1 performances of our methods have improved 0.16% to 0.28%, 0.34% to 0.41%, and 0.24% to 0.32% with average improvements of 0.23%, 0.38%, and 0.29%. When using the first 60%, there are no differences (in some cases

rather worse) between our methods and the existing algorithm. The wider order date period of data we use the higher differences we get. This suggests that better predictions could be achieved through using a wider range of data.

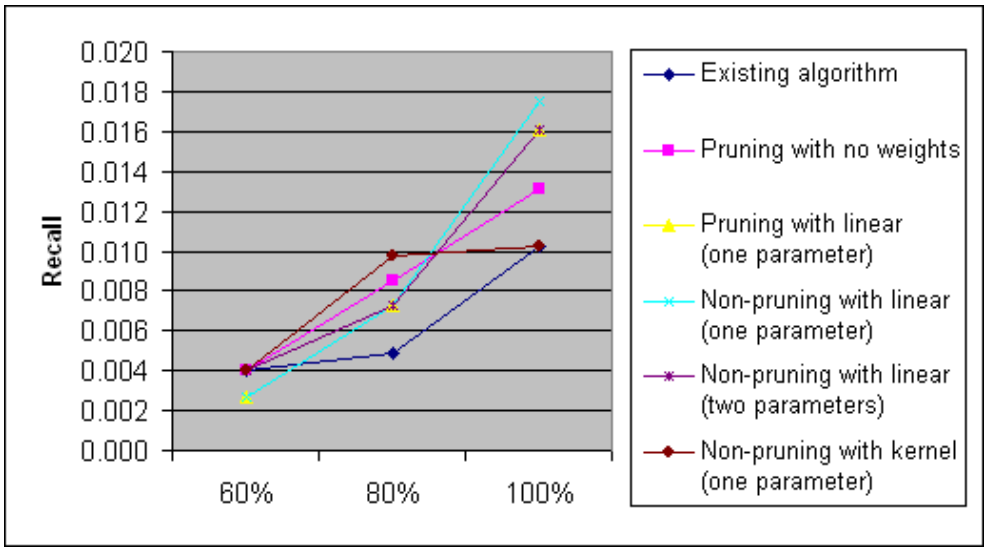

Figure 6.4: The recall accuracy when using real data

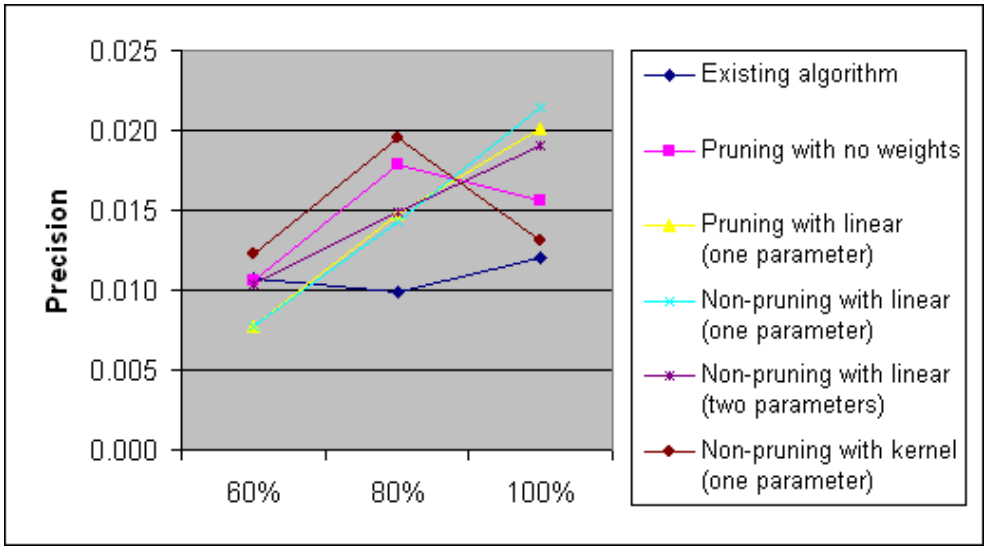

Figure 6.5: The precision accuracy when using real data

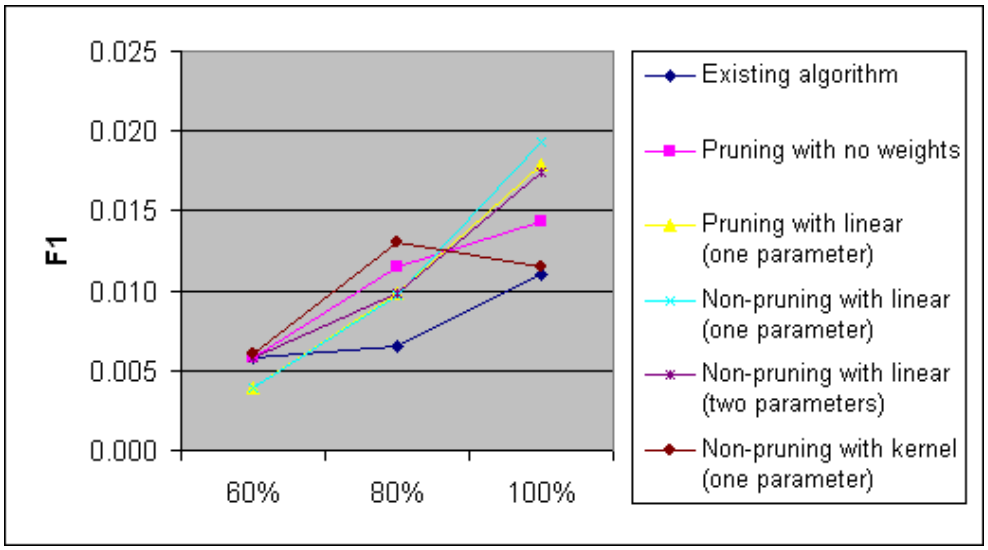

Figure 6.6: The F1 accuracy when using real data

All of the methods we proposed generally showed improvements in prediction compared to the existing algorithm. Especially, applying the linear function with either the pruning or non-pruning strategy showed the biggest improvement on each metric.

When using fake data the change in user preference is clear and distinct. However, with real data the change is more gradual. This means that clustering may be approximate. This makes the results for the pruning strategy different in these two cases, with real data giving poorer results.

We also discovered that the pruning strategy could subject some target users to be recommended nothing. The reason is that due to discarding (i.e., pruning) items in their past clusters there are no similar users on which to base the recommendations.

With fake data, using the kernel function revealed a similar result as using the linear function, but with real data, we get less performance. The reason is that

with fake data, the number of data in all users' training set is the same, whereas it is not the same with real data. Thus, when using the first 80% of the data we saw the best performance, but when using 100% it was not the best performance. We might say that using the kernel function tends to show irregular performance.

In addition, the recommended lists of some target users who did not change their preferences remain the same as when using the existing algorithm. Similarly, whether selecting the pruning or non-pruning strategy when employing the linear function the recommendation lists of the target users is the same.

Although our methods have improved the predictive accuracy, the differences are lower than our expectation and are disappointing. As mentioned in section 5.1, we realized that the dataset is too sparse to see how many recommended items really matched the purchased ones among the test set, since no matter which methods we select, only at most 12 items were matched.

For that reason, in order to make the impact of the temporal factors more apparent, we need an assumption that the hits of the matching genres can be used on behalf of those of the exact matching items. Since even though there are no matching items between the test and top-*N* sets, users' probabilities of repurchasing items among genres in their test sets are higher.

For instance, consider the recommendation $\{A_1, A_5, B_3, B_7\}$ and the actual purchase $\{A_2, A_6, B_4, B_8\}$. With exact matching we get a 0% prediction, but by matching genre's (i.e., "A" instead of "A_1") we get a 100% prediction of two A's and two B's.

Figure 6.7 shows the experimental result when considering matching genres

between the test and top-*N* sets. The Recall, Precision, and F1 performances of our methods have improved by 0% to 2.8%, -0.5% to 3.3%, and -0.4% to 3.0% with average improvements of 1.5%, 2.3%, and 2.2%. Therefore, we can say that considering matching genres as well as matching items, we could get better predictive accuracy.

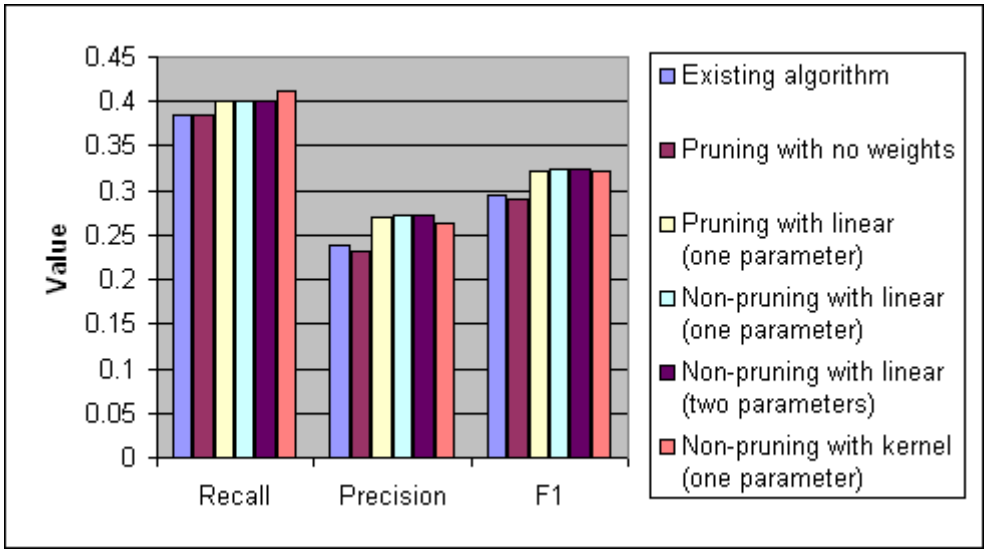

Figure 6.7: The experimental results when considering matching genres

Chapter 7
Conclusions and Future Work

The main contribution of this thesis is to present a new framework which introduces a method to apply temporal factors to current collaborative filtering systems. We have proposed and evaluated some time sensitive approaches with various combinations of time-based forgetting strategies and functions.

7.1 Summary

For the first time, we have brought up the data recency problem in collaborative filtering systems and defined the problem as the negative impact of old data on the predictive accuracy of collaborative filtering systems.

We reviewed the procedure of typical collaborative filtering systems and looked into some problems identified by researchers and commercial applications. Also, we studied some time-related research that has been published so far.

In order to overcome the data recency problem, we designed the pruning and non-pruning strategies, and employed the linear and kernel functions to apply different weights to users transaction data. Also, we demonstrated how to detect users' current preferences by classifying the data into current and past clusters by utilizing a clustering technique. Depending on it, we determined points at which interests changed.

It was achieved by implementing the Temporal Factors Collaborative Filtering (TFCF) system. The system is designed to allow control of experimentation with the techniques developed. We tested with five combinations, the pruning strategy with no weights and the linear function, the non-pruning strategy with the linear function using one and two parameters, and the kernel function using one parameter, and compared them with the existing algorithm.

Through our extensive experiments, we have obtained some improvement of accuracy predictions compared to a typical collaborative filtering algorithm. In general, using the linear function can be expected to provide the best predictive accuracy. If there are many users whose preferences have changed, we can also expect that the pruning strategy can get the best performances.

From our results it is clear that temporal factors should not be neglected when making recommendations. We believe that this thesis will pave the way for further analysis of applying temporal factors for the improvement of predictive accuracy.

7.2 Future Work

We employed the DeliBook dataset for our experimentations. Since the dataset is so sparse, we had trouble in showing good results. Hence, we would like to experiment with datasets that are non-sparse and have enough long periods of users' order dates in order to verify these apparently better recommendations.

We also want to leave the rated case to the future work since we considered only the purchase and non-purchase cases. We think that applying the rated case will be a more complex approach. Since the training data are applied

weights, the test data also should be applied weights which completely depend on their orders. But we do not know the order in which the items appear, and here lies the problem.

Lastly, the linear and kernel functions consider only the order of item observations. Thus, they do not consider how far it is between the two observations. The consequence is that the same proportions of weights are applied whether the gap is only one day or several years.

APPENDICES

Appendix A: DDL of TFCF System

```
CREATE TABLE Training (
    ID int IDENTITY (1, 1) NOT NULL ,
    UserID varchar (30) NOT NULL ,
    ItemID varchar (10) NOT NULL ,
    GenreID varchar (10) NOT NULL ,
    OrderDate datetime NOT NULL ,
    TargetFlag char (1) NOT NULL
)

CREATE TABLE Recmnds (
    UserID varchar (30) NOT NULL ,
    Priority int NOT NULL ,
    ItemID varchar (10) NOT NULL ,
    GenreID varchar (10) NOT NULL
)
```

Appendix B: Fake Data

B1. Cases 1 and 2:

```
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Sam', '16259', '1067', '4/14/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Sam', '18236', '1067', '5/2/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Sam', '20755', '1067', '5/30/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Sam', '26843', '1067', '6/20/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Sam', '43118', '1067', '7/18/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Sam', '11928', '1129', '2/2/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Sam', '12247', '1129', '5/30/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Sam', '12776', '1129', '8/1/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Sam', '11929', '1129', '1/8/04', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Sam', '13435', '1129', '5/11/04', '1');

INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Tom', '18775', '1129', '9/16/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Tom', '19756', '1129', '9/29/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Tom', '21039', '1129', '10/12/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Tom', '28412', '1129', '11/24/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Tom', '29769', '1129', '12/1/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Tom', '49153', '1067', '4/23/03', '1');
```

```
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Tom', '7684', '1067', '6/30/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Tom', '7848', '1067', '7/17/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Tom', '22443', '1067', '12/5/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Tom', '27370', '1067', '4/19/04', '1');

INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Kim', '12376', '1425', '11/19/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Kim', '12449', '1425', '12/30/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Kim', '18441', '1425', '1/15/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Kim', '18511', '1425', '2/20/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Kim', '28904', '1425', '3/8/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Kim', '17208', '1528', '6/8/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Kim', '17524', '1528', '7/3/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Kim', '34011', '1528', '8/1/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Kim', '36994', '1528', '12/21/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Kim', '39939', '1528', '4/30/04', '1');

INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Lee', '26371', '1623', '10/7/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Lee', '26415', '1623', '10/31/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Lee', '27343', '1623', '11/27/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Lee', '27430', '1623', '12/24/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Lee', '25540', '1623', '3/8/03', '1');
```

```
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Lee', '11645', '1959', '4/19/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Lee', '12759', '1959', '6/1/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Lee', '15956', '1959', '7/13/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Lee', '26893', '1959', '12/1/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Lee', '16625', '1959', '3/10/04', '1');

INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Meg', '17208', '1528', '12/5/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Meg', '34011', '1528', '3/20/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Meg', '36994', '1528', '4/30/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Meg', '39939', '1528', '6/15/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Meg', '14991', '1528', '6/2/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Meg', '42057', '1623', '6/19/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Meg', '41545', '1623', '7/21/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Meg', '32141', '1623', '7/29/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Meg', '29364', '1623', '2/14/04', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Meg', '25540', '1623', '5/19/04', '1');

INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ann', '15951', '1959', '3/30/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ann', '23308', '1959', '6/26/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ann', '25230', '1959', '8/5/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ann', '47520', '1959', '11/29/02', '1');
```

```
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ann', '21218', '1959', '3/21/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ann', '1112', '690', '5/15/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ann', '11138', '690', '6/8/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ann', '1327', '690', '7/24/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ann', '11140', '690', '3/1/04', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ann', '1256', '690', '7/10/04', '1');

INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ken', '1595', '690', '1/8/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ken', '15950', '690', '2/11/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ken', '16025', '690', '3/10/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ken', '1605', '690', '4/16/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ken', '1476', '690', '5/5/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ken', '38110', '1425', '6/28/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ken', '7908', '1425', '7/1/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ken', '7871', '1425', '8/2/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ken', '34974', '1425', '1/7/04', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ken', '9592', '1425', '6/10/04', '1');

INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Nan', '13378', '691', '8/27/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Nan', '27357', '691', '10/2/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Nan', '29868', '691', '11/9/02', '1');
```

```
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Nan', '45991', '691', '1/1/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Nan', '6195', '691', '3/7/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Nan', '29934', '741', '4/4/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Nan', '31071', '741', '6/27/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Nan', '34304', '741', '7/2/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Nan', '31067', '741', '4/20/04', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Nan', '29193', '741', '6/17/04', '1');

INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Eve', '8718', '741', '12/19/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Eve', '31073', '741', '2/1/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Eve', '31818', '741', '3/21/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Eve', '6497', '741', '5/16/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Eve', '31067', '741', '6/11/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Eve', '10802', '859', '6/19/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Eve', '11856', '859', '6/30/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Eve', '15305', '859', '7/1/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Eve', '10801', '859', '2/28/04', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Eve', '11458', '859', '4/10/04', '1');

INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Jay', '33633', '859', '4/3/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Jay', '8128', '859', '5/7/02', '1');
```

```
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Jay', '8258', '859', '7/16/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Jay', '49232', '859', '11/10/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Jay', '17529', '859', '1/1/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Jay', '22112', '961', '2/6/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Jay', '9801', '961', '3/8/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Jay', '8717', '961', '5/15/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Jay', '10739', '961', '10/12/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Jay', '11927', '961', '3/3/04', '1');
```

B2. Case 3:

```
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Sam', '16259', '1067', '4/14/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Sam', '18236', '1067', '5/2/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Sam', '20755', '1067', '5/30/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Sam', '26843', '1067', '6/20/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Sam', '11928', '1129', '7/18/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Sam', '43118', '1067', '2/2/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Sam', '12247', '1129', '5/30/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Sam', '12776', '1129', '8/1/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Sam', '11929', '1129', '1/8/04', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Sam', '11930', '1129', '5/11/04', '1');
```

```
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Tom', '18775', '1129', '9/16/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Tom', '19756', '1129', '9/29/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Tom', '21039', '1129', '10/12/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Tom', '49153', '1067', '11/24/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Tom', '28412', '1129', '12/1/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Tom', '7684', '1067', '4/23/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Tom', '29769', '1129', '6/30/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Tom', '7848', '1067', '7/17/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Tom', '22443', '1067', '12/5/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Tom', '27370', '1067', '4/19/04', '1');

INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Kim', '12376', '1425', '6/19/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Kim', '12449', '1425', '11/30/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Kim', '18441', '1425', '1/15/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Kim', '18511', '1425', '2/20/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Kim', '17208', '1528', '3/8/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Kim', '28904', '1425', '6/8/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Kim', '17524', '1528', '7/3/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Kim', '34011', '1528', '8/1/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Kim', '36994', '1528', '12/21/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Kim', '39939', '1528', '4/30/04', '1');
```

```
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Lee', '26371', '1623', '7/7/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Lee', '26415', '1623', '7/31/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Lee', '27343', '1623', '11/27/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Lee', '27430', '1623', '2/24/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Lee', '11645', '1959', '3/8/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Lee', '25540', '1623', '4/19/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Lee', '12759', '1959', '6/1/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Lee', '15956', '1959', '7/13/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Lee', '26893', '1959', '12/1/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Lee', '16625', '1959', '3/10/04', '1');

INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Meg', '17208', '1528', '6/5/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Meg', '34011', '1528', '11/20/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Meg', '36994', '1528', '4/30/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Meg', '42057', '1623', '6/2/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Meg', '39939', '1528', '6/15/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Meg', '41545', '1623', '6/19/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Meg', '14991', '1528', '7/21/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Meg', '32141', '1623', '7/29/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Meg', '29364', '1623', '2/14/04', '1');
```

```
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Meg', '25540', '1623', '5/19/04', '1');

INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ann', '15951', '1959', '3/30/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ann', '23308', '1959', '6/26/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ann', '25230', '1959', '7/5/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ann', '47520', '1959', '11/29/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ann', '1112', '690', '1/21/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ann', '21218', '1959', '2/15/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ann', '11138', '690', '6/8/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ann', '1327', '690', '7/24/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ann', '11140', '690', '3/1/04', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ann', '1256', '690', '7/10/04', '1');

INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ken', '1595', '690', '6/8/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ken', '15950', '690', '12/11/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ken', '16025', '690', '3/10/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ken', '1605', '690', '4/16/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ken', '38110', '1425', '5/5/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ken', '1476', '690', '6/28/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ken', '7908', '1425', '7/1/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ken', '7871', '1425', '8/2/03', '1');
```

```
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ken', '34974', '1425', '1/7/04', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ken', '9592', '1425', '6/10/04', '1');

INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Nan', '13378', '691', '8/27/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Nan', '27357', '691', '10/2/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Nan', '29868', '691', '11/9/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Nan', '29934', '741', '1/1/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Nan', '45991', '691', '3/7/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Nan', '31071', '741', '4/4/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Nan', '6195', '691', '6/27/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Nan', '34304', '741', '7/2/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Nan', '31067', '741', '4/20/04', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Nan', '29193', '741', '6/17/04', '1');

INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Eve', '8718', '741', '9/19/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Eve', '31073', '741', '12/1/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Eve', '31818', '741', '3/21/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Eve', '6497', '741', '5/16/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Eve', '10802', '859', '6/11/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Eve', '31067', '741', '6/19/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Eve', '11856', '859', '6/30/03', '1');
```

```
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Eve', '15305', '859', '7/1/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Eve', '10801', '859', '2/28/04', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Eve', '11458', '859', '4/10/04', '1');

INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Jay', '33633', '859', '4/3/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Jay', '8128', '859', '5/7/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Jay', '8258', '859', '7/16/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Jay', '49232', '859', '11/10/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Jay', '22112', '961', '1/1/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Jay', '17529', '859', '2/6/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Jay', '9801', '961', '3/8/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Jay', '8717', '961', '5/15/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Jay', '10739', '961', '10/12/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Jay', '11927', '961', '3/3/04', '1');
```

B3. Case 4:

```
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Sam', '16259', '1067', '4/14/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Sam', '11928', '1129', '4/14/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Sam', '18236', '1067', '5/2/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Sam', '12247', '1129', '5/30/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Sam', '20755', '1067', '6/20/02', '1');
```

```
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Sam', '12776', '1129', '5/30/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Sam', '26843', '1067', '8/1/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Sam', '11929', '1129', '8/1/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Sam', '43118', '1067', '3/14/04', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Sam', '11930', '1129', '5/11/04', '1');

INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Tom', '18775', '1129', '9/16/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Tom', '49153', '1067', '9/16/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Tom', '19756', '1129', '10/12/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Tom', '7684', '1067', '9/29/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Tom', '21039', '1129', '11/24/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Tom', '7848', '1067', '6/30/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Tom', '28412', '1129', '7/17/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Tom', '22443', '1067', '7/17/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Tom', '29769', '1129', '2/19/04', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Tom', '27370', '1067', '4/19/04', '1');

INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Kim', '12376', '1425', '11/19/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Kim', '17208', '1528', '11/19/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Kim', '12449', '1425', '12/30/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Kim', '17524', '1528', '1/15/03', '1');
```

```
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Kim', '18441', '1425', '2/20/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Kim', '34011', '1528', '7/3/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Kim', '18511', '1425', '8/1/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Kim', '36994', '1528', '8/1/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Kim', '28904', '1425', '2/17/04', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Kim', '39939', '1528', '4/30/04', '1');

INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Lee', '26371', '1623', '10/7/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Lee', '11645', '1959', '10/7/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Lee', '26415', '1623', '10/31/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Lee', '12759', '1959', '11/27/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Lee', '27343', '1623', '12/24/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Lee', '15956', '1959', '6/1/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Lee', '27430', '1623', '7/13/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Lee', '26893', '1959', '7/13/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Lee', '25540', '1623', '1/10/04', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Lee', '16625', '1959', '3/10/04', '1');

INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Meg', '17208', '1528', '12/5/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Meg', '42057', '1623', '12/5/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Meg', '34011', '1528', '3/20/03', '1');
```

```
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Meg', '41545', '1623', '4/30/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Meg', '36994', '1528', '6/15/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Meg', '32141', '1623', '7/21/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Meg', '39939', '1528', '7/29/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Meg', '29364', '1623', '7/29/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Meg', '14991', '1528', '4/1/04', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Meg', '25540', '1623', '5/19/04', '1');

INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ann', '15951', '1959', '3/30/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ann', '1112', '690', '3/30/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ann', '23308', '1959', '6/26/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ann', '11138', '690', '8/5/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ann', '25230', '1959', '11/29/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ann', '1327', '690', '6/8/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ann', '47520', '1959', '7/24/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ann', '11140', '690', '7/24/0', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ann', '21218', '1959', '5/2/04', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ann', '1256', '690', '7/10/04', '1');

INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ken', '1595', '690', '1/8/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ken', '38110', '1425', '1/8/03', '1');
```

```
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ken', '15950', '690', '2/11/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ken', '7908', '1425', '3/10/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ken', '16025', '690', '4/16/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ken', '7871', '1425', '7/1/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ken', '1605', '690', '8/2/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ken', '34974', '1425', '8/2/034', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ken', '1476', '690', '1/10/04', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Ken', '9592', '1425', '6/10/04', '1');

INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Nan', '13378', '691', '8/27/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Nan', '29934', '741', '8/27/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Nan', '27357', '691', '10/2/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Nan', '31071', '741', '11/9/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Nan', '29868', '691', '1/1/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Nan', '34304', '741', '6/27/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Nan', '45991', '691', '7/2/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Nan', '31067', '741', '7/2/034', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Nan', '6195', '691', '2/10/04', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Nan', '29193', '741', '6/17/04', '1');

INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Eve', '8718', '741', '12/19/02', '1');
```

```
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Eve', '10802', '859', '12/19/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Eve', '31073', '741', '2/1/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Eve', '11856', '859', '3/21/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Eve', '31818', '741', '5/16/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Eve', '15305', '859', '6/30/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Eve', '6497', '741', '7/1/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Eve', '10801', '859', '7/1/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Eve', '31067', '741', '1/5/04', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Eve', '11458', '859', '4/10/04', '1');

INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Jay', '33633', '859', '4/3/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Jay', '22112', '961', '4/3/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Jay', '8128', '859', '5/7/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Jay', '9801', '961', '7/16/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Jay', '8258', '859', '11/10/02', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Jay', '8717', '961', '3/8/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Jay', '49232', '859', '5/15/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Jay', '10739', '961', '5/15/03', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Jay', '17529', '859', '1/30/04', '1');
INSERT INTO Training (UserID, ItemID, GenreID, OrderDate, TargetFlag)
    VALUES ('Jay', '11927', '961', '3/3/04', '1');
```

REFERENCES

Billsus, D. and Pazzani, J. (1999). A Hybrid User Model for News Story Classification. In *Proceedings of the 17th International Conference on User Modeling*, Springer Verlag, pp. 99-108.

Breese, J. S., Heckerman, D., and Kadie, C. (1998). Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pp. 43-52.

Chalmers. M., Rodden. K., and Brodbeck. D. (1998*).* The Order of Things: Activity-Centred Information Access. In *Proceedings of the 7th International Conference on the World Wide Web*, Brisbane, April 1998, pp. 359-367.

Claypool, M., Le, P., Waseda, M., and Brown, D. C. (2001). Implicit Interest Indicators. In *Proceedings of the 6th International Conference on Intelligent User Interfaces*, pp. 33-40.

Crabtree, I. B. and Soltysiak, S. J. (1998). Identifying and Tracking Changing Interests. *International Journal of Digital Libraries*, vol. 2: 38-53.

Han, J. and Kamber, M. (2001). *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers.

Herlocker, J., Konstan, J., Borchers, A., and Riedl, J. (1999). An Algorithmic Framework for Performing Collaborative Filtering, In Preceedings of ACM SIGIR'99.

Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T. (2004). Evaluating Collaborative Filtering Recommender Systems. In *Proceedings of the ACM Transactions on Information Systems*, vol. 22, no. 1, pp. 5-53.

Hofmann, T. (2003). Collaborative Filtering via Gaussian Probabilistic Latent Semantic Analysis. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 259-266.

Huang, Z., Chen, H., and Zeng, D. (2004). Applying Associative Retrieval Techniques to Alleviate the Sparsity Problem in Collaborative Filtering. In *Proceedings of the ACM Transactions on Information Systems*, vol. 22, no. 1, pp. 116-142.

Karypis, G. (2000). Evaluation of Item-Based Top-N Recommendation Algorithms. *Technical Report #00-046*, Dept. of Computer Science, University of Minnesota, Minneapolis, MN.

Koychev, I. (2000). Gradual Forgetting for Adaptation to Concept Drift. In *Proceedings of ECAI 2000 Workshop: Current Issues in Spatio-Temporal Reasoning*, pp. 101-106, Berlin, Germany.

Koychev, I. and Schwab, I. (2000). Adaptation to Drifting User's Interests. In *Proceedings of ECML2000/MLnet Workshop: Machine Learning in the New Information Age*, pp. 39-45, Barcelona, Spain.

Kukar, M. (2003). Drifting Concepts as Hidden Factors in Clinical Stud*ies*. In *Proceedings of 9th Conference on Artificial Intelligence in Medicine in Europe*, AIME 2003, Protaras, Cyprus.

Maneeroj, S., Kanai, H., and Hakozaki, K. (2002). An Improved Recommendation Method for Better Filtering Information out of Database. *IPSJ Transactions on Databases Abstract*, vol. 43, no. SIG05-007.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. (1994). GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, pp. 175-186, Chapel Hill, NC.

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2000a). Analysis of Recommendation Algorithms for E-Commerce. In *Proceedings of ACM Conference on Electronic Commerce*, Minneapolis, MN.

Sarwar, B. M., Konstan, J. A., Borchers, A., Herlocker, J., Miller, B., and Riedl, J. (1998). Using Filtering Agents to Improve Prediction Quality in the GroupLens Research Collaborative Filtering System. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work,* Seattle, WA.

Schwab, I., Pohl, W., and Koychev, I. (2000). Learning to Recommend from Positive Evidence. In *Proceedings of Intelligent User Interfaces*, pp. 241-247, New Orleans, LA.

Yoo, H. (2004). *DeliBook On-line Bookstore Dataset*. YoungJin E-Commerce. http://www.delibook.com.