A HYBRID RECOMMENDATION SYSTEM CAPTURING THE EFFECT OF TIME AND DEMOGRAPHIC DATA

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

FULYA OKTAY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

MAY 2010

Approval of the thesis:

A HYBRID RECOMMENDATION SYSTEM CAPTURING THE EFFECT OF TIME AND DEMOGRAPHIC DATA

submitted by **FULYA OKTAY** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen _____
Dean, **Graduate School of Natural and Applied Sciences**

Prof. Dr. Adnan Yazıcı _____
Head of Department, **Computer Engineering**

Assoc. Prof. Ferda Nur Alpaslan
Supervisor, **Computer Engineering Dept., METU** _____

Examining Committee Members:

Assoc. Prof. Dr. Nihan Kesim Çiçekli
Computer Engineering Dept., METU _____

Assoc. Prof. Ferda Nur Alpaslan
Computer Engineering Dept., METU _____

Prof. Dr. Mehmet Reşit Tolun
Computer Engineering Dept., Çankaya University _____

Dr. Ayşenur Birtürk
Computer Engineering Dept., METU _____

Özgür Alan
ORBIM _____

Date: 28.05.2010

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: Fulya Oktay

Signature:

ABSTRACT

A HYBRID RECOMMENDATION SYSTEM CAPTURING THE EFFECT OF TIME AND DEMOGRAPHIC DATA

Oktay, Fulya

M.S., Department of Computer Engineering

Supervisor: Assoc. Prof. Ferda Nur Alpaslan

May 2010, 71 pages

The information that World Wide Web (WWW) provides have grown up very rapidly in recent years, which resulted in new approaches for people to reach the information they need. Although web pages and search engines are indeed strong enough for us to reach what we want, it is not an efficient solution to present data and wait people to reach it. Some more creative and beneficial methods had to be developed for decreasing the time to reach the information and increase the quality of the information. Recommendation systems are one of the ways for achieving this purpose. The idea is to design a system that understands the information user wants to obtain from user actions, and to find the information similar to that. Several studies have been done in this field in order to develop a recommendation system which is capable of recommending movies, books, web sites and similar items like that. All of them are based on two main principles, which are collaborative filtering and content based recommendations.

Within this thesis work, a recommendation system approach which combines both content based (CB) and collaborative filtering (CF) approaches by capturing the

effect of time like purchase time or release time. In addition to this temporal behavior, the influence of demographic information of user on purchasing habits is also examined this system which is called “TDRS”.

Keywords: Recommender Systems, Collaborative Filtering, Content Based, Temporal and Local Differences

ÖZ

PROFİL VERİLERİNİ KULLANARAK KULANICI TERCİHLERİNDE ZAMANA BAĞLI DEĞİŞİMİ YAKALAYAN MELEZ ÖNERME SİSTEMİ

Oktay, Fulya
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Ferda Nur Alpaslan

Mayıs 2010, 71 sayfa

Dünya çapındaki ağın (WWW) sağladığı bilgi son yıllarda hızlı bir büyüme gösterdi ve bu insanların ihtiyaç duydukları veriye ulaşmaları için yeni yaklaşımların oluşmasına yol açtı. İnternet sayfaları ve arama motorları istediğimiz bilgiye ulaşmamız için yetrince güçlü olsalar dahi, bilgiyi sunmak ve insanların ona erişmesini beklemek vrimli bir çözüm değil. Bireylerin veriye ulaşma hızını düşürk ve tam anlamıyla aradıkları bilgiye ulaşmalarını sağlamak için daha yaratıcı ve faydalı çözümler geliştirilmeli. Tavsiye sistemleri bu amacı gerçekleştiren yöntemlerden biridir. Bu sistemlerin altnda yatan temel düşünce, kullanıcıların istediği şeyin ne olduğunu onların yaptığı işlemler ile anlayan ve istedikleri bilgilere benzer bilgiler bulmaya çalışan bir sistem tasarlamaktır. Bu alanda, film kitap, internet sitesi gibi ürünlr tavsiy eden pek çok çalışma yapılmıştır. Bu çalışmaların hepsi içerik tabanlı(CB) ve kolaboratif filtreleme(CF) olmak üzere iki temel prensibe dayanmaktadır.

Bu tez çalışması içinde, hem içerik tabanlı hem de işbirlikçi filtreleme yeteneği olan ve zamansal ve mekansal değişimler dikkate alınarak yapılmış bir tavsiye

sistemi tanıtılacaktır. TDRS adlı bu sistemde, CB ve CF metodularını melezleştiren sistemlere zamansal, mevsimsel değişimler ve bölgesel farklılıkların da dikkate alındıgı yeni bir metod eklenmiş ve bir iyileştirmeye gidilmiştir.

Anahtar Kelimeler: Tavsiye Sistemleri, Kolaboratif Filtreleme, İçerik Tabanlı, Zamansal ve Mekansal Farklar

To My Family and Friends

ACKNOWLEDGMENTS

I would like to express my sincere gratitude and appreciation to Assoc. Prof. Ferda Nur Alpaslan for her encouragement and support throughout this study.

I would thank the Scientific and Technological Research Council of Turkey (TÜBİTAK) for providing the financial means throughout this study.

I am indebted to my father for encouraging me to choose computer sciences to be my path for the rest of my life. I have completed this work by his prescient guidance. I know you can see me wherever you are.

I'm deeply thankful to my mother and sister for their decent countenance and grateful to Ahmet Görkem Ekmekci for being an outstanding shoulder for me.

Finally I would like to thank to my friends for their honest wishes for me to finish this work.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

CB	Content Based
CF	Collaborative Filtering
TDRS	Temporal Difference Recommendation System
MAE	Mean Absolute Error
PCC	Pearson Correlation Coefficient
EG	Exponentiated Gradient
SVM	Support Vector Machines

CHAPTER 1

INTRODUCTION

Number of internet users has been increasing so much in recent years. This also results in the growth of data amount that has been available on the internet. In January 2005, the number of pages in the publicly indexable web was exceeding 11.5 billion [8]. This resulted in an information overload problem in which users find it really difficult to locate the right information at the right time [8]. With the exponential growth of information available on the web and with the increase in the number of books, CDs, and films to choose from, a new need in technology emerged: recommender systems [1]. Recommender systems are systems that build a representation of a user's likes and dislikes and they suggest items to the user based on this representation [1]. Recommender systems have been introduced to provide a solution for navigating the huge volume of information already available and growing at an explosive rate [2]. Through the agency of those systems, it has been getting easier for internet users to reach the information they need such as movies, films, videos, books or even web pages in shorter amount of times.

According to [3], recommender systems are today's most indispensable items that are used in web pages that allow users to buy items. The reason of it is that in a

system that a recommender system is used, both the customer and the business parts get benefit. The customer benefits by receiving feedback from the system with some suggestions on items that she is likely to buy. At the same time, the business benefits with an increase in its sales [2]. In consequence of this fact, web sites that provide recommendations for their users have become popular in recent years. Amazon, IMDB, Last FM, NETFLIX, Pandora are the ones that are visited frequently.

Recommendation systems are categorized into two major classes: those in which content-based filtering and those where collaborative filtering is used [4]. Content-based approach is based on recommending items according to its content. Simply, a content-based recommendation system finds items similar to the ones that user preferred before [5, 6]. Collaborative filtering, on the other hand, is an attempt to simulate collaboration among users for sharing recommendations and reviews [1]. Collaborative filtering aims at finding the relationships among the new individual and the existing data in order to further determine the similarity and provide recommendations. How to define the similarity is an important issue [7]. Most of the CF systems apply the nearest neighbor model for computing similarities and recommendations [1]. In addition to these two main approaches, a combination of them which is called Hybrid approach has also become popular through recommender system studies. The aim of hybrid approach is to combine the benefits of content-based and collaborative recommendation to achieve more success in results.

1.1 Problem Definition

This thesis focuses on the development and evaluation of a recommendation system (TDRS) which combines content-based (CB) and collaborative filtering

(CF) approaches by considering the effect of time information on user preferences and users' demographic information.

1.2 Method and Approach

In this thesis work, a recommendation system, which is called TDRS, is implemented and evaluated. TDRS is based on combining CB and CF approaches by using time information affectively. Time information is used in order to keep track of the changes in behaviors of users according to time they do the action. Besides user demographic information such as location is also used, which results in a spatiotemporal approach hybridizing the CB and CF methods. By using item launch dates as temporal information in CB method and hybridizing it with a temporal CF algorithm which combining demographic and collaborative user similarity, a contribution is done to recommendation studies

1.3 Thesis Outline

This thesis consists of the following chapters:

Chapter 2 – Related Work introduces a detailed description of recommender systems and explanation of the methods and algorithms used in those systems. In addition to that this chapter presents popular recommender system application with their approaches to this problem. Besides, recent studies about combining CB and CF approaches and examples of time based recommender systems are explained.

Chapter 3 – TDRS describes the architecture of our solution. The algorithms and methods used in this study are explained in detail.

Chapter 4 – Evaluation explains the evaluation approach used in this work and discusses the results obtained.

Chapter 5 – Conclusion draws the conclusion of this thesis and recommends possible future work.

CHAPTER 2

BACKGROUND AND RELATED WORK

This chapter will explain the problem that recommender systems are trying to solve and current solutions introduced so far. Several recommendation techniques will be discussed with both their advantages and disadvantages. In addition, related work will be explained with corresponding studies.

2.1 Definition of Recommender Systems

Information Retrieval is the science of searching any kind of information based on different disciplines such as computer science, mathematics, statistics etc. In order to explain the utilities of information retrieval technologies, the term “information overload” should be introduced. Information overload is a term invented by Alvin Toffler which refers to an excess amount of information being provided. With the growth of World Wide Web and e-Commerce, the information on the internet has been increasing rapidly in recent years. Although the fact that users are provided with more information seems to be an advantage, unfortunately it has become more difficult for them to find the right information, which is mentioned to be called as information overload. As a result of this problem, processing and

absorbing tasks become very difficult for the users because validity of the information cannot be seen [9]. An information filtering system is a system that removes redundant or unwanted information from an information stream using (semi)automated or computerized methods prior to presentation to a human user [10].

People always confront with the problem of which book to read, which movie to watch, in which restaurant to eat and so on. Sometimes the necessary information is much more important than those, such as for education and/or business purposes people may need a related information concept. Although internet provides nearly most of the information people need, due to the problem that is mentioned in previous problem, a strategy or algorithm is needed to present internet users the exact information they need. Recommender systems are information filtering systems in which several information retrieval techniques are used. The idea behind these systems is to design a system that understands the information user wants to obtain from the actions of them, and to find the information similar to that. In recent years, recommender systems have begun to provide a technological proxy for this recommendation process, in which they are used to either predict whether a particular user will like a particular item (prediction), or to identify a set of N items that will be of interest to a certain user (top-N recommendation) [8].

Recommender systems are used in various applications such as online e-commerce sites, music players and search engines. As described in the first chapter, e-commerce sites are the most popular recommender systems since both user side and company side get benefit.

[1] stated that for a typical recommender system, there are three steps:

1. The user provides some form of input to the system. These inputs can be both explicit and implicit [11]. Ratings submitted by users are among explicit inputs whereas the URLs visited by a user and time spent reading a web site are among possible implicit inputs.
2. These inputs are brought together to form a representation of the user's likes and dislikes. This representation could be as simple as a matrix of items-ratings, or as complex as a data structure combining both content and rating information.
3. The system computes recommendations using these user profiles.

2.2 Terms and Concepts

In this part, some general terms and concepts about recommendation systems are presented to enable a better explanation of the following chapters.

Item: An item is the general name of the object that is recommended and rated by the user. This can be a book in book domain, a video in video domain, etc.

User Profile: A user is the person who uses this system and provides information to a system and prefers items by giving ratings. A recommendation system works for user satisfaction. User profile is the whole information a recommendation system holds for a user. This can be the name, age, gender, location of a user and the history of how much rating user has given to an item.

Rating: The feedback that is provided by the user for an item. This may be a score in a certain range or can be extracted implicitly by following user behaviors such as interest time.

Explicit Rating: The rating submitted by users, such as rating a movie as three starts over five stars.

Implicit Rating: The rating produced by the system by following user behaviors. The system can label a rating as positive if user has spent much time with it. For

instance if a user spends more time in a web site, it may be considered as a positive rating.

Actual Rating: The rating collected explicitly or implicitly from a user for an item.

Predicted Rating: The data showing a user's expected interest for a particular item. This is calculated by the system with some algorithms.

Recommender: The entity that gives recommendations for users by considering user preferences. The results may be one item or a list of items.

Recommendation: It is the result of the recommendation process. A recommendation shows the item that the system presented for the user in order to be preferred.

User's Interest: Representation which shows how much a user prefers an item.

Prediction: The expected interest of a user in one item.

Prediction Accuracy: A measure that shows how much the predicted rating agrees with the user's actual rating. The more accurate the prediction, the better the performance of the recommendation system is.

Prediction Technique: The specific algorithm that the recommendation system will use in order to calculate the predicted rating of an item.

Recommendation System Algorithm: The algorithm used by recommendation system to produce recommendation for the user.

Active User: The user for whom a recommendation will be applied by using a recommendation system algorithm.

2.3 Recommendation Process

This part focuses on the way a recommendation system produces recommendations.

In Figure 1, the general process of a recommendation system is illustrated in very high level. Here, it can be observed that a recommendation system basically needs two inputs, which are user profiles and items, the meaning of which are explained in previous chapter. After these inputs are taken by the recommendation system, it uses the predefined recommendation algorithm and the results are obtained, which are called as recommendations.

Figure 1 General Process of Recommendation Systems

According to [21], a recommendation process consists of the following steps: information recollection, selection, transformation, structuring and presentation. The details of those phases are explained below.

2.3.1 Information Recollection

This phase includes the recollection of users' personal preferences and information about items, which is the base for the following steps. The results of this phase must truly reflect the reality; otherwise meaningful recommendations cannot be obtained.

2.3.2 Selection

It is the phase in which the items that reflect the user's preferences are selected. The method that defines similarity plays the most important role in this step.

2.3.3 Transformation

The main aim of the transformation step is to modify the items retrieved. It is an optional step and may consist of modifications such as summarization, creation of snapshots, etc.

2.3.4 Structuring

This phase defines in which way the user will navigate through recommendations. Items may be grouped or relationships may be displayed between items.

2.3.5 Presentation

This step deals with the presentation of the results to the user.

2.3.6 Feedback

Optionally, feedback may be retrieved from the users to improve the results and this step handles this problem. The kind of feedback obtained from the user could be of two kinds: implicit or explicit, which are defined in previous chapter.

2.4 Classification of Recommender Systems

In this part, recommendation systems are classified according to their recommendation techniques.

2.4.1 Collaborative Filtering(CF) Recommender Systems

Collaborative Filtering, as the name applies, is a process that filters information according to the collaboration of people that use this information. The central idea here is to base personalized recommendations for users on information obtained from other, ideally like-minded, users [12]. Collaborative filtering techniques have been successful in enabling the prediction of user preferences in the recommendation systems (Hill et al., 1995, Shardanand & Maes, 1995) [13]. According to [13], there are three major processes in the recommendation systems: object data collections and representations, similarity decisions, and recommendation computations and collaborative filtering aims at finding the relationships among the new individual and the existing data in order to further determine the similarity and provide recommendations. Although there are various CF algorithms in order to define similarity, all of them rely on the same fundamental, which is to find the similar users of the current user and recommend an item which is preferred by those similar users before. Nearest neighbor algorithm is mostly used in CF applications. In Figure 2, an illustration of CF technique is shown.

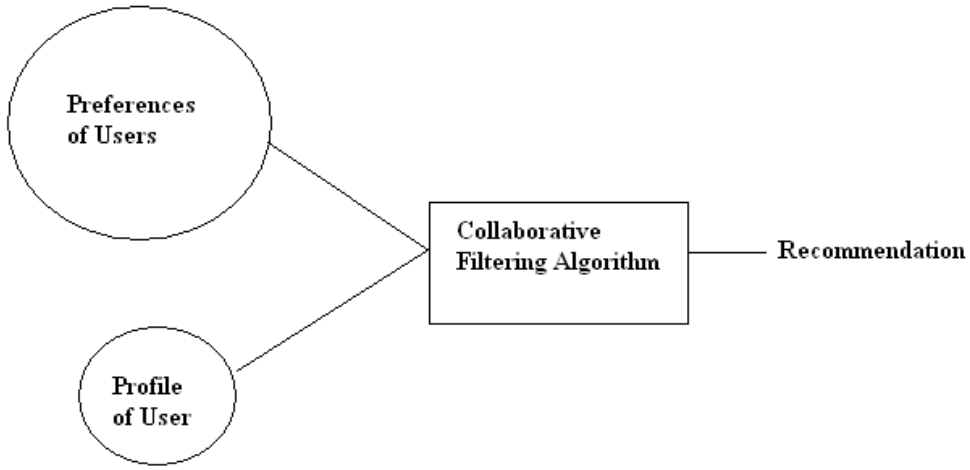

Figure 2 Illustration of CF Technique

In a typical CF scenario, there is a list of m users U = {u_1, u_2, ... u_k} and a list of n items I = {i_1, i_2, ... i_m}. Each user u_i has a list of items Iu_i , which the user has expressed his/her opinions about. This opinion may be a direct rating which denotes whether user has liked the item or not. In addition to that, an implicit feedback may be retrieved from the user such as by analyzing timing logs. Those feedbacks are employed to drive the user-item matrix, which will be the main guide through collaborative filtering recommendation process. A user-item matrix is a K x M matrix where K is the number of users and M is the number of items and each cell holds the rating that a user has given to an item. If a cell is empty, that means this user hasn't seen that item yet.

2.4.1.1 CF Approaches

There are two main categories of collaborative filtering techniques: Memory-Based and Model-Based. Memory-based approach, which makes rating predictions based on the entire collection of previously rated items by the users, is

used first by the researchers. Then, due to limitations with this approach, a model-based CF approach, which uses the collection of ratings to learn a model, is developed to make predictions. Although the model-based approach deals with some of the limitations related to memory-based CF, this approach also has its shortcomings which will be explained in detail [8].

- **Memory-Based CF:** In memory–based CF approach, a database of the ratings users have given to items is collected. To predict the rating that a user may give to an item, the ratings that other users have given to this item before is normalized according to a formula, namely the entire database is used. Memory-based CF techniques are wildly used in this area with various additions and modifications. There are two types of memory-based CF approaches:
    - **User Based:** User-based CF predicts the rating that user may give to an item based on the history of similar users to that user. First the similarities of users (rows) are calculated according to their user-item relations (ratings) and than a subset of the similar users are produced according to their similarity measurement. Finally, the ratings of these top users are averaged by taking their similarity weights into consideration. The following formula can be used to calculate the predicted rating.

$$r'_{a,i} = \frac{\sum_1^K w_{a,u_k} * r_{u_k,i}}{\sum_1^K w_{a,u_k}}$$

(1)

where,

$r'_{a,i}$ = predicted rating of user a for item i

u_k = neighbor of user a

$r_{x,i}$ = rating user x has given to item i

$w_{x,y}$ = similarity weight of user x to user y

This approach has no understanding of the content being filtered, which may cause poor recommendations. Besides, it requires a large number of ratings in order to make good predictions and it is not a realistic requirement. Huge amount of database that is needed may cause a poor response time [14].

- **Item Based:** In this approach the similarity between items is used. First the similarity between items is calculated by considering the similarity of their ratings given by users. After that, a rating can be predicted by averaging the ratings of other similar items rated by current user. Compared to user-based CF approach, it has an advantage of not requiring a huge amount of historical data belongs to many users in order to make prediction. If a different algorithm to find the similarity of items such as looking at the content of items, it doesn't even need an item to be rated by other people to take it into recommendation phase. If similarity of items is based on correlations of user ratings and there is no information about the consumption/usage of an item rated by the target user, cold start problem occurs [14]. This prevents the system from recommending new content. If similarity is based on item attributes, calculation of similarity is often highly inaccurate since it can only be as accurate as the attributes that are available, which are limited in current applications. It also requires a large number of user/item relations in order to obtain meaningful patterns [15].

- **Model-Based CF:** Model-based collaborative filtering algorithms provide item recommendation by first developing a model of user ratings which is performed by different machine learning algorithms such as Bayesian network, clustering, and rule-based approaches [16]. The Bayesian network model formulates a probabilistic model for collaborative filtering

problem [17]. Clustering model deals with collaborative filtering as a classification problem and works by clustering similar users in same class and estimating the probability that a particular user is in a particular class C, and from there computes the conditional probability of ratings [18]. The rule-based approach applies association rule discovery algorithms to find association between co-purchased items and then generates item recommendation based on the strength of the association between items [19]. The shortcomings of this approach are that it is usually time-consuming to build these systems and to make updates on it. Besides, a model-based approach cannot cover a user range as deep as the memory-based approaches [20].

2.4.1.2 Strengths and Weaknesses of CF

The main strength of CF is its independence from the recommendation domain. If a CF algorithm is implemented for a book domain, it can also be used for movie domain because a CF algorithm doesn't make use of any domain data to make prediction. Since recommended item is found by looking at the other users' tastes, a surprising item can be recommended for user and he/she can taste very different items according to his/her previous choices. This also results in a diverse user-item matrix, which enables better recommendations. In contrast to these strengths, CF approach has some weaknesses. First of all, this approach highly depends on the data provided by the users. If users do not provide enough information, the performance of the system decreases. In current systems, the percentage of the filled cells in user-item matrices to all cells is very low, which is also called as data sparsity problem.

In user-based approach, a recommendation cannot be done to a new user, who hasn't rated any item yet, since there aren't any similar users to this active user. Similarly, in item-based approach, a new item is never recommended since it

doesn’t rated by any of the users. This problem is called as cold start. Cold start problem occurs in three ways which are new user, new item and new system. New system is the combination of new user and new item. In user-based approach if two users never rated the same item before, their similarity cannot be measured although in fact they may have show great similarity. In the same way, if two similar items have never been rated by the same user, their similarity is also lost, which prevents one of them to be recommended. As a result, it can be stated that CF approach gives good results only when there is enough information provided by the users, namely if the user-item matrix is filled enough.

2.4.1.3 CF Algorithms

In this part, algorithms that are generally used in CF systems are explained. The most important CF algorithm is the one that is used in similarity calculation. In [21] the most popular CF algorithms are listed as following.

Mean Squared Differences Algorithm

This algorithm calculates the degree of dissimilarity between users by the following formula:

$$D_{xy} = \frac{\sum_{n}^{Na} C_{xn} * C_{yn} * (S_{xn} - S_{yn})^2}{\sum_{n}^{Na} C_{xn} * C_{yn}}$$

(2)

In this formula,

C_{xn} : [1, 0], depending whether item n is rated by user x or not.

C_{yn} : [1, 0], depending whether item n is rated by user y or not.

S_{xn} : rate of item n given by user x.

S_{yn} : rate of item n given by user y.

After dissimilarity between users is calculated, final prediction is calculated by taking into consideration the users below a dissimilarity threshold and using a weighted average of the ratings. These weights are inversely proportional with the dissimilarity value. In this formula we can see that result depends only on the items that are both rated by the users.

Pearson Correlation Coefficient Algorithm

This algorithm aims to find the similarity between users and based on Pearson r correlation coefficient. The possible values of this algorithm range from -1 to +1 including 0. Values near -1 indicate a negative correlation, values close +1 indicates a positive correlation; a value of 0 shows no correlation at all. The prediction of the rating is same as the previous algorithm after Pearson coefficient has been calculated. This algorithm takes into account negative correlation as well as positive correlation

$$CFSim(u,a) = \frac{\sum_i^n (r_{u,i} - avg(r_u))(r_{a,i} - avg(r_a))}{\sqrt{\sum_1^n (r_{u,i} - avg(r_u))^2}\sqrt{\sum_1^n (r_{a,i} - avg(r_a))^2}}$$

(3)

In this formula,

u, a : users

$r_{u,i}$: rating user u has given to item i.

$avg(r_u)$: average rating of user u.

n: number of items both user u and user a has rated.

In contrast to mean squared difference method, this algorithm considers not only the users that represent more similarity to active user but all users who even have negative correlation with the active user.

Constrained Pearson r Algorithm

This algorithm is a small modification to the standard Pearson r algorithm. Instead of using a rating scale, this method considers ratings as positive or negative. As a result, two users are identified as similar if both of them rated a specific item as positively or negatively.

2.4.2 Content Based (CB) Filtering

2.4.2.1 Definition of CB Approach

Instead of looking at the choice of users and modeling user behaviors to calculate similarities and recommending items according to those similarities, there is a completely different method which can be summarized as finding the similar items that user preferred before by looking at the properties of items. The idea behind this approach is that user prefers items that are similar to the ones he/she preferred before. So this type of filtering deals only with the choices of the current user in contrast to CF approach. According to [1] there are basically four steps of content based filtering:

- The first step is to gather content information of the items belong to a specific domain, such as writer of a book, director of a movie, owner of a video, etc… Information Extraction techniques are used in most systems to extract the data of specified domain and Information Retrieval techniques are used to obtain the related information. Web crawlers are commonly used for this purpose [22].
- The second step is to retrieve feedback from users. This feedback may be either in rating format or time information can be used to measure whether user has liked the item or not.

- The third step is to extract a user profile by looking at the items that user preferred before. Different machine learning algorithms and classification techniques are used to achieve this. Item properties are taken into account with corresponding weights to classify items. Weights represent how much a property (or attribute) identifies an item.
- The last step is to find the item or items which user hasn't seen yet and which are similar like the ones user has liked before.

2.4.2.2 Content Based Filtering Algorithms

According to [25], creating a model of the user according to user history to find the items user prefers can be modeled as a classification problem. There are basically two classes of this problem which are "items user like" and "items user does not like".

Decision Trees and Rule Induction

One of the classification types for content based filtering is decision trees and induction rules method. In this approach, the features of items form the nodes of decision tree. The systems that use decision trees try to extract rules that model user preference strategy from tree constructed according to the user history. This method can be useful in domains that don't include free text properties. For instance, by asking questions to users about restaurants, a system can construct rules that can find in which restaurant user prefers to eat by using decision tree method. In contrast, this method is not appropriate for domains including free text properties since it is not easy to ask questions.

Nearest Neighbor Method

Another method used in content based filtering to classify items is nearest neighbor method. This method based on storing all the information in a database and finding neighbors of each item according to a similarity function. This function depends on the type of data. For structured data, Euclidian distance is used. If a vector space method is used, cosine similarity is preferred.

Relevance Feedback

Methods that help users to incrementally refine queries based on previous search results are commonly referred to as relevance feedback. The general principle is to allow users to rate documents returned by the retrieval system with respect to their information need. Rocchio's algorithm is a widely used relevance feedback algorithm and it is based on the modification of an initial query through differently weighted prototypes of relevant and non-relevant documents.

Linear Classifiers

There are many different linear classification algorithms, most of which can be applied to classify text items. In general, the outcome of the learning process is n x m matrix, in which n represents the predicted rating and m represents the item. This can be viewed as a numeric score prediction, which leads to linear regression approach. Widrow-Hoff rule, exponentiated gradient(EG) algorithm, support vector machines(SVM) are examples of linear classification methods

Probabilistic Method and Naïve Bayes

The naïve Bayesian classifier is one of the methods used in text classification. Researchers have recognized that this method is an exceptionally well-performing text classification algorithm and adopted frequently in recent works [25].

2.4.2.3 Strengths and Weaknesses of CB Filtering

According to [14, 21] strengths can be listed as following:

- CB filtering can recommend a completely new item, which is a definite solution to *new item* problem.
- Since content of items is static, an item is analyzed once, namely it is not analyzed again if new users are added to system.
- It is an easy and practical approach and doesn't need a wide user profile. It can produce recommendation even there is only one user in the whole system.
- People can trust this system since it is easy to explain. Therefore more people may use it.

Weaknesses of the system are listed as following in [14, 21]

- Since CB approach doesn't take into account the profiles of other users, it looses very important information to make recommendation. As explained in previous parts, the idea that if two people preferred the same items in the past they probably prefer the same items in the future is a successful approach.
- In this approach, it is likely to bother user with very similar items and stuck in that phase. Users may not be recommended with different items which that actually can prefer.
- CB approach also suffers from *new user* problem. When a new user has entered the system, since he/she doesn't have a new profile yet, an item cannot be recommended.
- Due to the fact that this approach highly depends on the content analysis and content analysis is vital problem in several domains, CB filtering may not give successful results. Domains such as video, photo, audio or text information is very difficult to make correct classifications. The

classification techniques that rely on tag information loose the relation between items that actually explain the same thing but in different words. Also some irrelevant items may be identified as similar if they have the same keywords. Namely, context information is hard to extract and this causes bad recommendations.

- Since this method is domain specific, the algorithms should be implemented from scratch for every domain, which is not a practical and effective method.

2.4.3 Hybrid Approach

2.4.3.1 Definition of Hybrid Approach

As stated in [21], although collaborative filtering has been very successful in both research and practice, it presents some major disadvantages. For instance, it can not recommend new items to the users and completely denies any information that could be extracted from contents of an item. On the other hand, content-based methods fail in providing as good recommendations as collaborative filtering does. The reason for this is that it is hard to extract really high level meaningful features of any domain. In order to avoid the problems caused by weaknesses of individual recommendation systems, researchers have come up with a new solution called hybrid recommender systems, which are techniques that combine different types of recommendation methods in order to compensate one's weaknesses with another's strengths [23]. According to researches, hybrid systems not only prevent the bad results happened because of the weaknesses of the method, but also combines the advantages of the methods that are hybridized and provide better recommendations.

2.4.3.2 Types of Hybrid Approach

According to [23], there are seven basic ways that recommender systems can be combined to build hybrids:

Mixed: Results for different recommenders are presented together either in a combined presentation or in separate lists [23].

Weighted: Scores from the recommenders are combined using weights to derive a single score. Simplest way is averaging the results of different algorithms [23].

Switching: The system uses some decision criteria to choose a recommender based on the context and uses the results from only the chosen source. For instance, if one technique is not capable of providing an adequate prediction, the other one may be used. This is a rather complex method to implement [23].

Cascade: One recommender refines the recommendations produced by another. For example, the work presented in [24] explains a system that uses this method, the system may first use content-based algorithm to find user with similar preferences and then it may use collaborative filtering techniques to make predictions for those users [23].

Feature Combination: Data from different source types are combined together and treated using one recommendation algorithm. The work presented in [5] uses this method; it attempts to create a hybrid system by first using collaborative methods to create clusters of songs and then using the features extracted from the content. The idea is to use machine learning techniques to map the content of the song to the clusters created by the collaborative techniques [23].

Feature Augmentation: The output from one technique is used as an input feature to another. One typical example of this is the use of content based techniques to create pseudo ratings of the items and then use these ratings in a collaborative technique as if they were provided by the user [23].

Meta-level: One recommender produces a model, which is then used as input for the second recommender [23].

2.4.4 Statistical Analysis

In addition to these algorithms, some statistical methods are also used to provide recommendations for users. They can be listed as follows:

Top-N: N products which received the most clicks over a certain period of time or N product which have the most average rating.

Top-N for Category: For each product category the N most frequently clicked products or the N products which have the most average rating for each category.

Sequence Patterns: Products most often succeeded other products in the same user session.

Popular: The items that have bcom popular in a certain recent tim.

Although these methods are not adequate for a recommendation system, they may be used as an addition to overcome some of th weaknesses of current recommendation systems.

2.5 Temporal Behavior

Besides the basic recommendation methods described above, a new improvement for increasing the prediction accuracy is using the time information of user profiles or item features. According to [28], user preferences change over time and therefore modeling temporal dynamics should be a key when designing recommender systems or general customer preference models. There have been some attempts to capture the effect of using time information of user choices to improve recommendation system algorithms. For instance, according to [29], more recent users' ratings on items may reflect more on users' current interests than those of long time ago. The researchers have presented a novel hybrid recommender system to overcome the interest drift problem by embedding the time-sensitive functions into the recommendation process. Another example of

applying time dependent function in recommender systems is presented in [30]. In this study, it is claimed that there are four main types of time effects in CF:

1. Time bias: the interest of whole society changes with time
2. User bias shifting: a user may change his/her rating habit over time
3. Item bias shifting: the popularity of items changes with time
4. User preference shifting: a user may change his/her attitude to some types of items

In addition to those four main types, researchers explain that there are many other time effects. For example, some season based differentiations such as a change in user's rating habit in different months of a year due to some special dates like festivals can be observed. Old users and new users may have different rating behavior. New movies and old movies may get different ratings. According to [30], the main problem is how to use these time effects to build a time-dependent predictor. Researchers have shown that previous studies have proposed many ways to use time effects. One approach views these time effects as global effects and uses them by simple models, such as linear regression. Another approach is based on neighborhood methods. This approach assumes that recent data is more important to predict users' future preferences than old data. Therefore, when calculating item-item similarity, recent rated items will be over-weighted. Moreover, in another study, an alternative method which divides rating data into bins by time and trains latent factor model in every bin has been proposed [30].

2.6 Examples of Recommender Systems

In today's world of internet, there are web sites which provide successful recommendations. In this chapter they will be summarized in order to explain source of our interest to recommender systems.

Amazon uses a series of collaborative filtering algorithms to compare user's purchasing patterns with everyone else's. Online shopping in Amazon is based on personalized recommendations. Its system favors popular, obvious items and tends to recommend less like a trusted salesman. For instance, if a user has bought the Kashmir album by Led Zeppelin, Amazon displays a list of bought items all made up of other albums of Led Zeppelin. It is a very successful online shopping system that has a recommendation system involved.

Netflix, a movie recommendation site, uses users versus item ratings matrix. It is known to be the best movie recommender system today. **MovieLens** is also a movie recommendation site using collaborative filtering. At start up the users should rate lots of movies, in order to be able to use the system effectively. Both systems offer users movies according to their preferences (explicit ratings they have given to the movies) and the other user's preferences. Each member of the system has a "neighborhood" of other like-minded users. Ratings from these neighbors are used to create personalized recommendations for the target user.

At **Pandora.com**, after typing the name of a band or song, the system gives a similar result in terms of lyrics, melody, harmony, rhythm, genre or vocal- using content-based filtering. Pandora calls these types of musical attributes as genes and its database of songs as Music Genome Project.

Last.fm is a social recommender system; it also uses collaborative filtering with the other users of the system. However, if user does not have similar tastes with the other users of the system, the system will stuck in certain styles. For the newly entered items, Pandora will more likely offer them since they may have similar "genes" with the other songs in the system. On the other side, Last.fm will be slower to recommend them due to the "cold start" problem: new items will not be popular among the users of the system.

Neither item-centric, nor user-centric approaches are the best solutions to the recommender systems. Using a hybrid approach, combining both collaborative and content-based filtering should give better results.

CHAPTER 3

TDRS: TEMPORAL DIFFERENCE RECOMMENDATION SYSTEM

Up to now, the term recommender system is introduced, the general concepts about it are explained, the recommendation process is defined and the types of recommender systems with it general algorithms and corresponding strengths and weaknesses are argued. Besides, temporal approach is introduced. In addition to that several researches which influence this study are stated. In this chapter, the work called TDRS is explained in detail with the algorithms used.

This study introduces a movie recommendation system which combines CB and CF approaches with and also makes emphasis on the effect of local and temporal differences in feedback data. In this chapter the details of our study will be explained with its algorithms and methods.

3.1 System Overview

In this thesis work, a recommendation based on combining CB and CF approaches by adding time dependent behavior is introduced. This time dependent behavior is composed of the following subgroups:

- Paying more attention to recent ratings.
- Paying more attention to the items released recently

Another addition to recommendation process is using demographic information in order to find user similarities. Besides using collaborative data, users' genders, locations and occupations are also used to improve the result quality of user similarities. By paying much more attention to the location information, a spatial differentiation in user preferences is aimed to be identified. As a result, this system works as a spatiotemporal recommendation system which hybridizes CF and CB approaches.

3.2 System Architecture

In this chapter, the architecture of the system will be explained. The overall architecture illustrated in Figure 3. The components of the system are explained below:

Database Engine: The data of items, users and the ratings users have given to items with other information such as time are all stored in database engine. This component is responsible for storing data and providing services to reach and update this data.

Recommender: The data retrieved from database engine and the user actions sent by interface component is received by recommender component. By using predefined recommendation algorithms, which will be discussed later, this component finds for all users the rating predictions for each item he/she hasn't rated before.

Evaluator: The results of the system are evaluated by this component. The details of the evaluation process are described in EVALUATION chapter.

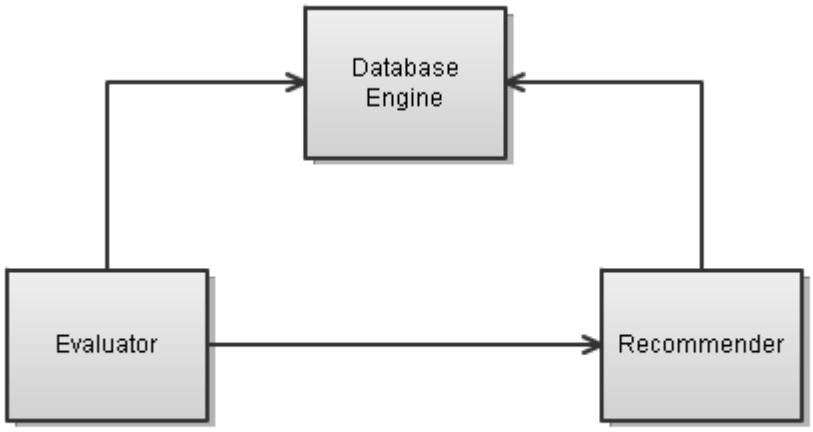

Figure 3 Recommender System Architecture

3.3 Design

In this chapter, the detailed design of the system will be explained. Firstly, the approach of this study will be described, then data modeling will be interpreted and lastly the algorithms and methods will be elaborated.

3.3.1 Description of the Approach

TDRS is a recommendation system in which content based filtering is combined with collaborative filtering method by adding temporal behavior. Before applying user based collaborative filtering, ratings are reprocessed in order to reflect temporal behaviors. Similarly, before using content based filtering to find the similarities between items, same preprocessing is applied. During this study, it is claimed that time information of the user-item pairs affect the predictions. That is

to say, the time change when a user has rated an item is important during rating prediction.

3.3.2 Data Modeling

This chapter describes the data modeling method of the system. The data used in this study is mainly divided into two groups which are user and item. Following chapters will define data modeling in detail.

3.3.2.1 User Modeling

As stated in previous chapters, CF is one of the recommendation approaches used in this study. In order to apply any CF algorithm, the information of the ratings users have given to items is needed. Therefore, user modeling is a requirement in a system that applies CF.

Given M number of users and N number of items, the system consists of M x N user item matrix R. Each entry r_{mxn} in R matrix represents the rating that user m has given to item n. This rating can be an integer value between [0, 5]. The first user modeling is a vector $[r_{m,1}, \ldots\ldots , r_{m,N}]^T$ where m = 1, …. M and the entries represent the ratings user m has given to all items.

As an addition to recommendation approach, user demographic information is also used to find the similarities between users. By doing so, an improvement to CF is intended. Below is the demographic information used in this study, which also builds up the second user modeling:

Table 1 Demographic Information of User

Property	Type	Description	Similarity Measure
Name	String	Name of the users	Not used in similarity measure
Age	Integer	Any number that can be an age of an human being	$\|A_1 - A_2\| <$ age_threshold* ? 1: 0
Gender	String	{"M", "W"}	$G_1 = G_2$? 1 : 0
Location	String	Place user lives in	$L_1 = L_2$? 1 : 0
Occupation	String	Profession of the user	O1 = O2 ? 1: 0

* age_threshold is the value used to identify if the differences in ages of two users are big enough to identify them as dissimilar according to age values. 10 is chose for this threshold in TDRS system.

3.3.2.2 Item Modeling

TDRS uses movie domain to measure the success of the introduced algorithm. Domain information is important while applying CB Filtering. The dataset for ratings users have given to movies is taken from MovieLens [6] and combined with the content information of movies retrieved from IMDB [26]. For time information of ratings the data collected by [31] is used. The features of items are used for finding similarities between them, and then this similarity information is used to find the rating predictions, which produces content based filtering. All of the features listed below are used during recommendation phase. The model of the items is described below:

Table 2 Item Modeling

Property	Type	Description	Similarity Measure
ID	String	Identity of movie	Not used in similarity measure
Name	String	Name of movie	N_1 contains N_2 or N_2 contains N_1 ? 1 : 0
Type	String	Type of the	$T_1 = T_2$? 1 : 0
Tags	List of Strings	The tags that define item content	List Similarity Function
Country	List of Strings	The list of countries movie is recorded	List Similarity Function
Cast	List of Strings	The list of actors and actresses of movie	List Similarity Function
Genre	List of Strings	The types movie is included in	List Similarity Function
Writer	List of Strings	Writers of the movie	List Similarity Function
Language	List of Strings	Languages used in the movie	List Similarity Function
Company	List of Strings	Companies of the movie	List Similarity Function

List Similarity Function used in this thesis is described below:

$$sim(L_1, L_2) = \frac{L_1 \cap L_2}{max(|L_1|, |L_2|)}$$

(4)

where,

L_1 = First list

L_2 = Second list

$L_1 \cap L_2$ = Items that are common to both lists

$|L_1|$ = Number of items in the list

3.3.2.3 Data Representation

In this chapter database design will be explained by giving information about the tables and relations of tables in TDRS database.

Users are stored in a table named as "users" in TDRS. Properties of this table are listed below.

Table 3 Table of "users"

Column Name	Data Type	Is Primary Key	Explanation
Username	String	Yes	Name of the user
Age	Integer	No	Age of the user
Gender	String	No	Gender of the user
Occupation	String	No	Profession of the user
Location	String	No	Location of the user

The other information stored in database is the content information of items. This information is hold in different tables described below:

Table 4 Tables of "Movies"

Table	Columns	Primary Key	Explanation
Title	ID, Title, Kind ID, Production Year	ID	Title information of movies
Name	ID, Name	ID	Names of actors and actresses
Cast	ID, Person ID, Movie ID, Person Role ID, Role ID	ID	Cast information of movies
Company Information	ID, kind	ID	Available company types with their fields
Type Information	ID	ID	Available movie types
Movie Information	Movie ID, Information Type ID, Information	Movie ID	Information about distribution,number of rates and rating for each movie
Keyword	Keyword, ID	ID	All possible keyword
Movie Keyword	Movie ID, Keyword ID	Movie ID, Keyword ID	Keywords of each movie

Table 4 Tables of "Movies" (cont'd)

Table	Columns	Primary Key	Explanation
Role Type	ID, Role	ID	Role information of the person
Company Name	ID, Name	ID	Names of all companies
Movie Companies	ID, Movie ID, Company ID, Company Type ID	ID	Company information of each movie
MovieLens_IMDB	MovieLens ID, IMDB ID	MovieLens ID, IMDB ID	IMDB identities of MovieLens movies
MovieLens Movies	Movie ID, Movie Name, Movie Year	Movie ID	ID, name and year information for movies in the MovieLens Dataset

In addition to these tables, some other relationships are also hold in database. "Friends" table is one of them. This table stores information showing who is whose friend according to user similarity in the system.

Table 5 Table of "friends"

Column Name	Data Type	Is Primary Key	Explanation
Username	String	Yes	Name of the user
Friend	String	Yes	Name of the friend
Similarity	Float	No	Percentage of similarity[0,1]

The other relationship stored in database is named as “relateditems”. This table holds item similarities. Below are the properties of this relationship.

Table 6 Table of “relateditems”

Column Name	Data Type	Is Primary Key	Explanation
Item1	String	Yes	Identity of the item
Item2	String	Yes	Identity of the related item
Similarity	Float	No	The value which shows the percentage of similarity. It is a float between [0, 1].

The whole recommendation process needs the information showing how much a user likes an item, namely what rating a user has given to an item. In TDRS, this information is improved with the date and time information of this evaluation. The resulted relationship is stored in “ratings” table, the properties of which are listed as following.

Table 7 Table of "ratings"

Column Name	Data Type	Is Primary Key	Explanation
Username	String	Yes	Name of the user
Item ID	String	Yes	Identity of the item
Date	DateTime	No	The time showing when user has rated the item
Rating	Integer	No	Rating user has given to item

3.3.3 Algorithms and Methods

The algorithms and methods used in TDRS will be explained in this chapter. The algorithms can be grouped as following:

- Algorithms used for finding item similarity
- Content based recommendation
- Algorithms used for finding user similarity
- Collaborative filtering based recommendation
- Algorithms for adding temporal behavior

3.3.3.1 Item Similarity Calculation

In order to find how similar am item is to another item, Euclidian distance algorithm is used. The similarity measurement of these properties is given in Item Modeling table. In addition the similarity formulas weight factor is added to the

calculation. Basically, due to the fact that not all features have the same effect on similarity calculation, their probable impact are tried to be calculated by giving them appropriate weights. These weights are taken from [27] and shown below:
The resulted algorithm is as following:

$$CBSim(i,j) = w_1 * sim(F_{1i}, F_{1j}) + w_2 * sim(F_{2i}, F_{2j}) + ... + w_n * sim(F_{ni}, F_{nj)}$$

(5)

where,

F_{ni} : n^{th} feature of item i.

w_n : the weight of feature n.

$sim(F_{ni}, F_{nj})$: value shows how similar the n^{th} feature of item I to the n^{th} feature of item j. (Calculation is given in Item Modeling table). The results of this formula are floating point values between [0, 1].

This formula is the heart of content based recommendation of TDRS. The similarities between items are calculated once after the system is started and reused whenever needed. If a new item is added to the system, this similarity calculation must be done for it too. However, during the experiments of this thesis, the item number is fixed and doesn't change.

3.3.3.2 Content Based Recommendation

After item similarities are calculated, results are used to apply content based recommendation. The aim is to find the content based predictions of ratings that user may give to items he/she hasn't seen before. To achieve this, the following formula is used:

$$CBPrediction(u,i) = avg(i) + \frac{\sum_1^N sim(i,n) * (rating(u,n) - avg(i))}{\sum_1^N sim(i,n)}$$

(6)

where,

u: user

i: item

N: number of items user u has given a rating

sim(i, n) : similarity of item i to item n

rating(u, n) : rating user has given to item n

avg(i) : average rating given for item i

In order to apply content based recommendation, for all users and items that each user hasn't rated before, a rating is predicted and stored as content based prediction.

3.3.3.3 User Similarity Calculation

Collaborative User Similarity

User similarities are calculated according PCC method, which is explained in CF Algorithms chapter. As explained in previous chapters, user similarities are used to gather information form users who have similar tastes with active user.

Demographical User Similarity

In addition to collaborative similarity of users, user demographic information is also used to improve the similarity quality. The information used in this phase is user age, gender, location and occupation. In Table 1, how these features are identified as similar is described. As a result, following formula is developed:

$$DS(u,a) = w_1 * sim(F_{1u}, F_{1a}) + w_2 * sim(F_{2u}, F_{2a}) + ... + w_n * sim(F_{nu}, F_{na})$$

(7)

where,

u,a : users

F_{nu}: n^{th} feature of user u

w_n: weight of n^{th} feature to similarity

Since this study is about capturing the spatiotemporal behaviors of users, the weight of location information is increased to make it more effective in the results.

Total User Similarity

After collaborative similarity and demographical similarity are calculated, the results are combined to find the exact similarity of users. This combination is done by giving weights to the results ob above formulas. Simply,

$$Sim(u,a) = w_1 * CFSim(u,a) + w_2 * DSSim(u,a)$$

(8)

To keep collaborative similarity more effective on the results w_2 is chosen as 0.7 and w_1 is chosen as 0.3.

Due to the fact that the system holds information about many users, which in fact probably don't have many common interests, it is a better idea to match a user with a certain number of users instead of with the rest of the users. Therefore, k-nearest neighbor algorithm is used and only users who really have bigger similarities are used while applying collaborative filtering. A simple iterative approach is applied to find the suitable neighbor number. This approach can be described as below:

For each user

    For i = 0, i < 10

        For all users

            Find the most similar user

            Add it to neighbors

            Remove it from users

    For each neighbor

        If similarity < similarity_threshold

For all users

Find the most similar user

Add it to neighbors

Remove it from users

Therefore, if a neighbor of a user has a similarity below a similarity_threshold, another neighbor is added to the list. Therefore more information can be gathered for current user from his/her neighbors

3.3.3.4 Collaborative Filtering Based Recommendation

After finding the similarities of users by PCC algorithm, these similarities are used to find CF based predictions. The aim is to find the prediction of rating a user may give to items, by using the ratings users who are similar to current one have given before. Resulted formula is as below:

$$CFPrediction(u,i) = \frac{\sum_1^N sim(u,n) * rating(n,i)}{\sum_1^N sim(u,n)}$$

(9)

where,

u: user

i: item

N: number of neighbors of users who have rated item i.

sim(u, n): similarity of user u to item i

rating(n, i): rating user n has given to item i

3.3.3.5 Temporal Behavior

As stated previously, time is considered to be an affective feature for making rating prediction. In this part, how this information is used in this study is

explained in detail. At first, it will be better to explain the types of time information considered to be important while making recommendations. The work presented in [28], divides time information into three which are launch time, buying/rating time and time difference between buying/rating and launch time. The results shown in [28] have proved that while using launch time and rating time improves recommendation accuracy, time difference between rating and launch time does not have an effect on prediction. Therefore, the first important time feature used in this study is the purchase time. This is the time a user chooses an item, rates an item or buys an item. It is claimed that, more recent choices of people are more important to decide on future recommendations. Therefore less attention is paid to older ratings of users. In order to apply this idea, ratings used in collaborative filtering method are reprocessed in the way to reflect the rating time weights. Considering the time values in dataset, timestamp is divided into three parts, names as old, middle and recent. The ratings occurred in old times are multiplied by OLD_TIME_WEIGHT, the ratings given in middle times are multiplied by MIDDLE_TIME_WEIGHT and recent ratings are multiplied by RECENT_TIME_WEIGHT. This time division is done by simply dividing the total rating period into three. So, before applying classical user-based CF approach, time impact is reflected on ratings and a preprocessing is done.

The second time information considered during recommendation is launch time of the items. It is claimed that, users tend to choose items released recently instead of those produced in older times. As explained before, popularities of items decrease with time. In order to reflect this impact to recommendation process, the release time of items are preprocessed before applying content based filtering. For instance, if a rating is given to an item whose launch year is in old launch time period, then this rating is multiplied with OLD_TIME_WEIGHT. Before applying content based recommendation, all ratings are preprocessed by considering the weights shown above.

3.3.3.6 Hybridizing CB and CF with Temporal Behavior

After the ratings are preprocessed to reflect temporal differences and both CB and CF algorithms are used to make predictions, last step is to hybridize the results. The simplest method is to combine the CB and CF predictions and taking a weighted average of them. Following formula is used:

$$PredictedRating(u,i) = CB_{Weight} * CRPrediction(u,i) + CF_{Weight} * CFPrediction(u,i)$$

(10)

where,

u: user

i: item

CB_Weight : Weight of CB prediction for rating (Given in System Constants)

CF_Weight : Weight of CF prediction for rating (Given in System Constants)

and

CB_Weight + CF_Weight = 1.

3.3.4 Structural Behavior

The flow of the TDRS is shown in Figure 4. When the system is started, user data, which includes demographic information, items data composed of movie properties and ratings data with date of them are read from the database. After system constants are obtained from configuration file, the system calls CB and CF recommenders by setting them the information read. For all ratings, CB Recommender multiplies the rating value with the constant corresponds to the release time of item for which the rating was given. Similarly, CF Recommender multiplies all ratings with the constant corresponds to the date of rating. In order to avoid changing the actual rating, both recommenders make a copy of them

before doing this calculation. After time dependent ratings are found, CB Recommender finds the similarities of items while CF Recommender finds the similarities of users. User similarity calculation is composed of demographical and collaborative similarity calculations. After this operation, CF Recommender finds the neighbors of users according to similarities of them. When both CB and CF Recommenders find the predicted ratings for testing data, the results are combined to obtain a hybrid recommendation. Finally evaluator calculates the MAE of the predictions.

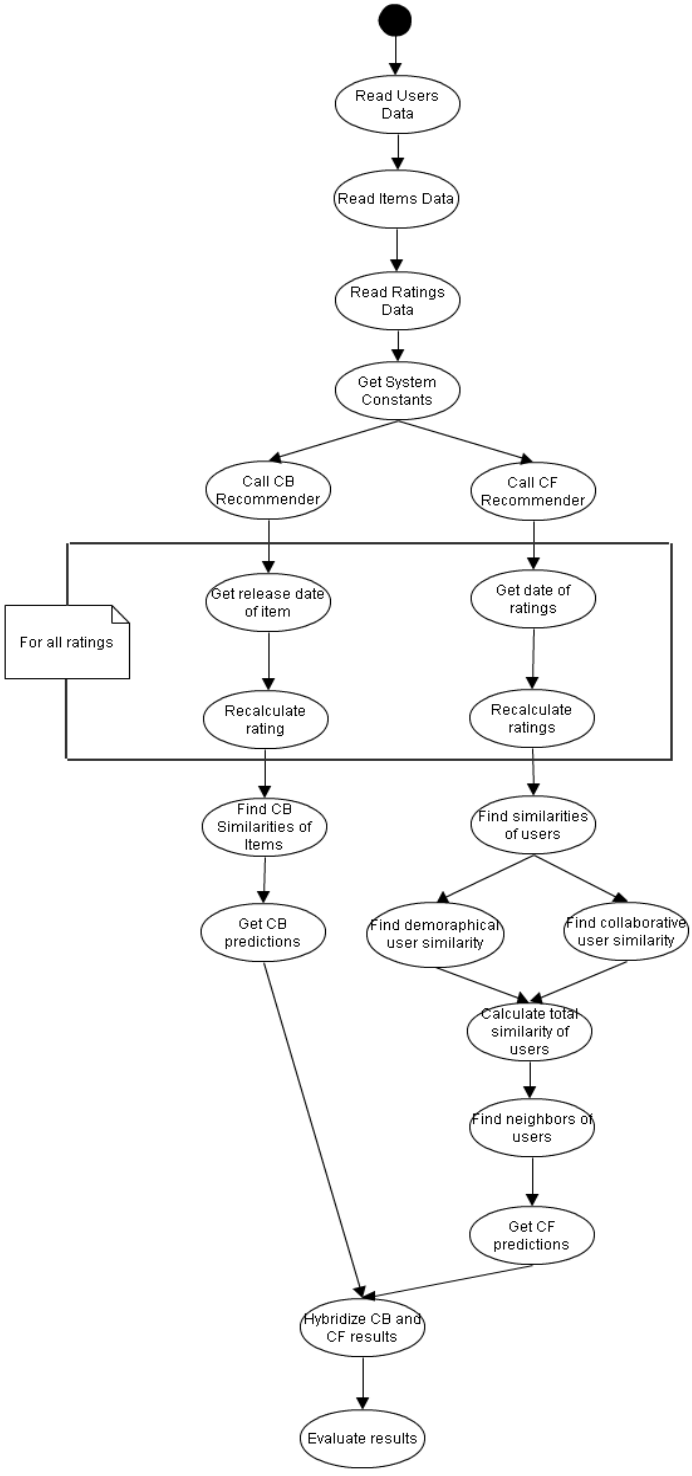

Figure 4 TDRS Flow Diagram

3.4 Implementation Details

In this chapter implementation details are listed. Here, the information about the environment, programming languages tools and methods of the components will be explained.

3.4.1 Environment and Programming Language

The system is developed in Windows XP by using Java, with Eclipse IDE. Being an object oriented language and having too many additions which improved the abilities, Java is an appropriate choice for building up environment free applications. For database related operations, MySQL is used due to its straightforward integration with Java technologies.

3.4.2 System Components

In this part, components of TDRS will be explained in detail. As stated previously, TDRS consists of four main components which are Database Engine, Recommender and Evaluator. Decomposition of TDRS is done according to the requirements of the system.

3.4.2.1 Database Engine

The data used by TDRS and defined previously is stored and managed by "Database Engine" component. As stated before, MySQL is used to manage database operations. The responsibilities of Database Engine component can be listed as below:

- Stores information about users and provides interfaces to reach and update this information
- Stores information about items and provides interfaces to reach and update this information
- Stores information about similarities between users and provides interfaces to reach and update this information
- Stores information about similarities between items and provides interfaces to reach and update this information
- Stores information of ratings and provides interfaces to reach and update this information

3.4.2.2 Recommender

The main component of TDRS is "Recommender" component. The algorithms explained in
Algorithms and Methods chapter are implemented in this component. Recommender component is divided into following sub-components:

CB Recommender

CB recommender is responsible for finding item similarities and recommending items according to this similarity. The input to CB recommender is properties of items and history of the user in terms of ratings he/she has given before. The output of CB Recommender is a list of rating-item duple containing items user hasn't rated before and corresponding predicted ratings.

CF Recommender

CF recommender is responsible for finding user similarities and recommending items according to this similarity. The input to CF recommender history of users in terms of ratings they have given before. The output of CF Recommender is a

list of rating-item duple containing items user hasn't rated before and corresponding predicted ratings.

Hybridizer

Hybridizer is responsible for combining the results of CB and CF Recommenders by using system default settings. The input of this component is two duple list of rating-item duple containing items user hasn't rated before and corresponding predicted ratings. The output is one list combined by using the method described in previous chapters, which is also the output of the Recommender component.

3.4.2.3 Evaluator

Evaluator is the component that is responsible for finding the prediction accuracy of the whole system. The metrics and method used by this component is described in CHAPTER 4. The input of this component is the output of Recommender component and the actual data retrieved from Database Engine. The output is a list of error rates gathered by different trials.

CHAPTER 4

EVALUATION

This chapter contains the description the experiments conducted for this thesis study. Firstly, dataset used by system is introduced, then evaluation approach is explained, and finally the results are presented.

4.1 Dataset

In order to evaluate the system, a dataset which includes information about items, users and ratings with date information is needed. Besides, to compare the results with current recommendation systems, it is better to use a dataset which is popular among other recommendation projects. Therefore, the experiments of the system are conducted with MovieLens dataset prepared by GroupLens Research group at University of Minnesota. There are three different datasets provided by this group:

- 100,000 ratings for 1682 movies by 943 users
- 1 million ratings for 3900 movies by 6040 users
- 10 million ratings and 100,000 tags for 10681 movies by 71567 users

Due to the computational complexity of the others, the first one is chosen as the dataset of the system. This dataset only includes information about users and

ratings. Item content information is obtained from IMDB and the total dataset is combined with the help of [31].

The properties of this dataset are as following:

Number of users: 943

Number of items: 1682

Total ratings: 100000

Ratings scalar: [1,…, 5]

Average number of ratings a user has: 100000/943 = 106

The density of the dataset can be expressed as:

100000 / (943 * 1682) = % 6.3

4.2 Metrics

In order to measure the accuracy of recommendation method proposed in this thesis, Mean Absolute Error (MAE) metrics is used. It is a quantity used to measure how close predictions are to actual outcomes. In our system TDRS, MAE metrics is applied with the following formula:

$$MAE = \frac{\sum_1^N |r - r^a|}{N} \quad \textbf{(11)}$$

where,

r^a : equal to actual rating

r : equal to predicted rating

N: number of total ratings.

Smaller the MAE, larger the accuracy of the recommendation system's accuracy is. The reason for choosing this metrics is that it is appropriate for TDRS since it tries to find the actual rating a user may give to an item and how well this process is done can be tested by MAE.

4.3 Evaluation

Throughout this thesis study, a movie dataset consisting of users, movies with their features and ratings users have given to movies with date and location information is needed. As proposed in previous chapters, our main aim in this thesis study is to measure the effect of date and time information on recommendation process. During these experiments the following questions are tried to be answered.

1. Does paying much more attention to the recent activity of users than to the previous activity of them improve the results?
2. Does paying much more attention to the items released recently improve the results?
3. Does using demographic information of users and paying much more attention to locale of user improve the results?

In order to answer these questions, TDRS is evaluated by k-fold cross validation method. The dataset is divided into two which are training and testing sets. The aim is to use some of the data for finding similarities of users and items, then to use the rest of it for determining the accuracy of the recommendation system. 500 users are chosen randomly from dataset. After that, 300, 200 and 100 users with 20, 10 and 5 ratings are chosen as training dataset and the rest of the dataset is used as testing dataset. This procedure is repeated k times and the error is calculated by taking the average of results. K is chosen to be 10.

The results of the algorithm are both compared with traditional method by explaining the effects of constants and other experiments described before in this thesis.

4.4 System Constants

As described in previous chapters, there are some values used within formulas to give weights to attributes. In this chapter these values are introduced.

CB_WEIGHT: The weight of content based prediction
CF_WEIGHT: The weight of collaborative filtering based prediction
NEIGHBOR_COUNT: Number of neighbors of a user
OLD_LAUNCH_TIME_WEIGHT: The value by which the rating for an item released in old times is multiplied
MIDDLE_LAUNCH_TIME_WEIGHT: The value by which the rating for an item released in middle times is multiplied
RECENT_LAUNCH_TIME_WEIGHT: The value by which the rating for an item released in recent times is multiplied
OLD_BUYING_TIME_WEIGHT: The value by which the rating for made in old times is multiplied
MIDDLE_BUYING_TIME_WEIGHT: The value by which the rating for made in middle times is multiplied
RECENT_BUYING_TIME_WEIGHT: The value by which the rating for made in recent times is multiplied
TYPE_WEIGTH: The weight of movie type similarity used in item similarity calculation (0.18)
WRITER_WEIGHT: The weight of writer similarity used in item similarity calculation (0.36)
GENRE_WEIGHT: The weight of genre similarity used in item similarity calculation (0.04)

KEYWORD_WEIGHT: The weight of keywords similarity used in item similarity calculation (0.03)

CAST_WEIGHT: The weight of cast similarity used in item similarity calculation (0.01)

COUNTRY_WEIGHT: The weight of country similarity used in item similarity calculation (0.07)

LANGUAGE_WEIGHT: The weight of language similarity used in item similarity calculation (0.09)

COMPANY_WEIGHT: The weight of company similarity used in item similarity calculation (0.21)

GENDER_WEIGHT: The weight of gender similarity used in demographic user similarity calculation

LOCATION_WEIGHT: The weight of location similarity used in demographic user similarity calculation

OCCUPATION_WEIGHT: The weight of occupation similarity used in demographic user similarity calculation

AGE_WEIGHT: The weight of age similarity used in demographic user similarity calculation

AGE_THRESHOLD: Threshold value for age to identify two ages as similar or not

DEMOGRAPHIC_SIM_WEIGHT: The weight of demographic similarity used in user similarity calculation

COLLABORATIVE_SIM_WEIGHT: The weight of demographic similarity used in user similarity calculation

4.5 Results

4.5.1 Experiment Results

In this part, the experiments are explained in detail and corresponding results are given. During the experiments, different number of users and ratings is chosen while system constants are also being modified to obtain the best results. In Table 8, the results of the algorithms used in TDRS are shown. The first column of the table represents the number of users used as training dataset. The second column shows the method evaluated. The other three columns represent the number of ratings per user used as training dataset. The last column shows the mean absolute error obtained by using the rest of the dataset as testing dataset. The results shown in Table 8 are the best values obtained by changing system constants the effect of which is explained in following chapters. As described in previous chapters, different combinations of user and rating numbers are tested and the results of changing this factor are also shown in following graphics. A fix configuration is chosen for comparing the algorithms to make reasoning.

As can be observed below, using the purchase information increased the accuracy of the recommendation. The difference can be seen by comparing the Pure CF with Temporal CF results. This result shows that the interests of users tend to change over time, therefore paying much more attention to their recent choices increase the recommendation quality.

Another result that can be inferred is that using the launch time of items while applying content based recommendation has also increased the accuracy of content based recommendation. This can be observed by comparing the results of Pure CB and Temporal CB methods. According to this improvement, it can be said that users tend to prefer newer items more than the older ones.

Another improvement is achieved by using the demographical information of users to find the similarities between them. The results can be seen by comparing

Pure CF without demographic and Pure CF without demographic information. How this demographic information is used is explained in following chapters.

Although recommendation accuracy has increased after temporal information is used, when the dataset consists of fewer users and fewer ratings, MAE of the system increases for all algorithms. As a result, temporal information is useful when there are more users and rating history is full enough to make sense.

Table 8 Algorithm Results

Training Users	Method	Given 5	Given 10	Given 20
100	Pure CF (Without demographic)	0,9345	0,9121	0,8939
	Pure CF (With demographic)	0,9001	0,8561	0,8421
	Pure CB	0,8902	0,8899	0,8876
	Hybrid CFCB	0,8495	0,8214	0,8126
	Temporal CF	0,8973	0,8450	0,8198
	Temporal CB	0,8879	0,8761	0,8654
	Temporal Hybrid CFCB	0,8187	0,8031	0,7997
200	Pure CF (Without demographic)	0,9115	0,9071	0,8565
	Pure CF (With demographic)	0,8911	0,8715	0,8143
	Pure CB	0,8869	0,8719	0,8501
	Hybrid CFCB	0,8417	0,8201	0,8066
	Temporal CF	0,8901	0,8356	0,8091
	Temporal CB	0,8798	0,8610	0,8444

Table 8 Algorithm Results (cont'd)

Training Users	Method	Given 5	Given 10	Given 20
	Temporal Hybrid CFCB	0,8007	0,7992	0,7812
300	Pure CF (Without demographic)	0,9032	0,8987	0,8343
	Pure CF (With demographic)	0,8711	0,8661	0,8045
	Pure CB	0,8776	0,8675	0,8587
	Hybrid CFCB	0,8311	0,8198	0,7923
	Temporal CF	0,8898	0,8221	0,7921
	Temporal CB	0,8504	0,8442	0,8332
	Temporal Hybrid CFCB	0,7819	0,7721	0,7614

4.5.2 Impact of Temporal CB

The results shown in Table 8 are explained in previous part. In this chapter, the effect of system constants used in temporal CB algorithm will be shown in detail. These constants are:

- OLD_LAUNCH_TIME_WEIGHT
- MIDDLE_LAUNCH_TIME_WEIGHT
- RECENT_LAUNCH_TIME_WEIGHT

The values above are the weights that are given to launch time and used during CB rating prediction. How these constants are used is explained in previous chapters. To apply this algorithm, the time period is divided into three as explained in previous chapters. It is an important issue to make this division logically for reflecting the effect of time correctly to rating predictions. Since the tests are made on different subsets of the dataset, every time the periods are

updated. For instance, if the launch times of items in a subset differ from 1970 to 2010, then the difference of this minimum and maximum years are calculated, which is 40 in this case. Then, time is simply divided into 3, approximately 13 for these values.

During the experiments, several combinations of those weights are tested with dataset consisting of 300 testing users with 20 ratings per user. First, only recently launched items are taken into consideration by setting OLD_LAUNCH_TIME_WEIGHT and MIDDLE_LAUNCH_TIME_WEIGHT to 1. The results are shown in Figure 5. It is observed that the error decreases while the value of this weight increases. Therefore the optimum value for this constant is set to 1.

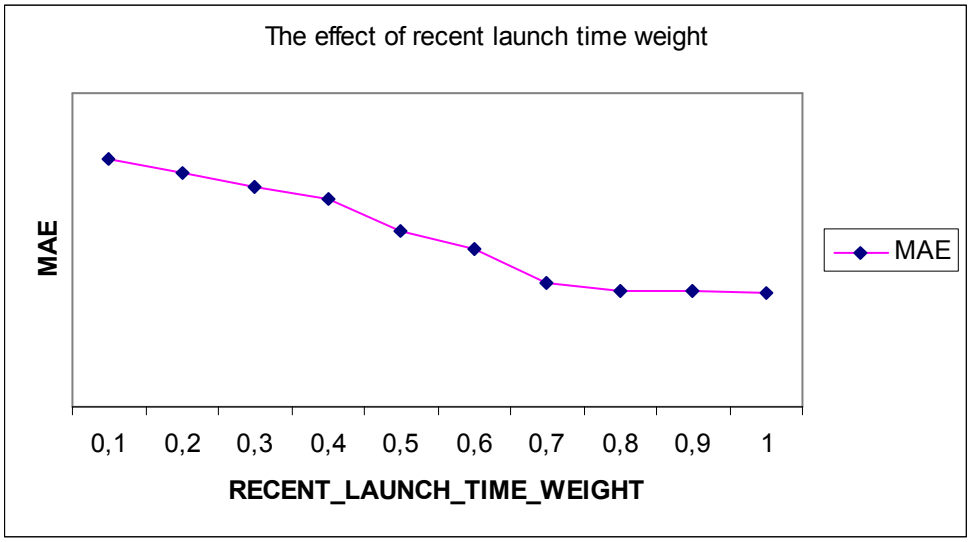

Figure 5 The Effect of RECENT_LAUNCH_TIME_WEIGHT for CB

The same procedure is followed to find the optimum weights of OLD_LAUNCH_TIME_WEIGHT and MIDDLE_LAUNCH_TIME_WEIGHT. It can be seen from Figure 6 and Figure 7 that error is in its minimum when MIDDLE_LAUNCH_TIME_WEIGHT is about 0.6 or 0.7 and

OLD_LAUNCH_TIME_WEIGHT is between 0.3 or 0.4. During the experiments they are set to 0.6 and 0.3 respectively.

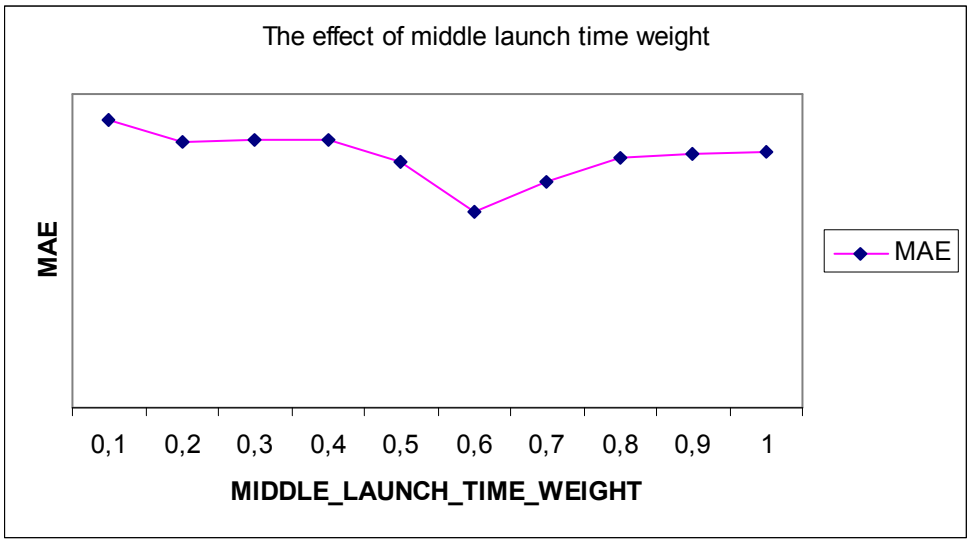

Figure 6 The Effect of MIDDLE_LAUNCH_TIME_WEIGHT for CB

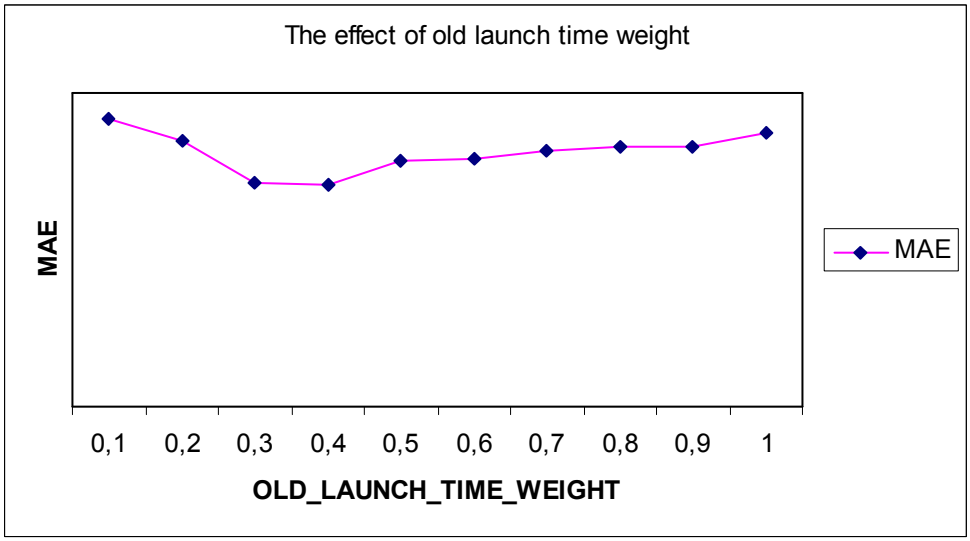

Figure 7 The Effect of OLD_LAUNCH_TIME_WEIGHT for CB

4.5.3 Impact of Temporal CF

Temporal information is also used in CF algorithm. As stated in previous chapters, the idea is to reflect the change in user preferences over time to prediction process. In order to apply this idea, the ratings are multiplied by a weight according to rating time. These weights are:

- OLD_BUYING_TIME_WEIGHT
- MIDDLE_BUYING_TIME_WEIGHT
- RECENT_BUYING_TIME_WEIGHT

After user similarities are found, the ratings are rearranged according to their dates. For instance if a rating's date corresponds to and old buying time, then this rating is multiplied by OLD_BUYING_TIME_WEIGHT. In order to divide the time period, a similar procedure tried in temporal CB algorithm is applied. The total time period of ratings is calculated and this period is simply divided into three.

Several tests are conducted to find the optimum temporal CF weights with training dataset consisting of 300 users with 20 rating per each user. In order to designate three weights defined in previous chapters, a controlled experiment is done. Namely, to find the optimum value for OLD_BUYING_TIME_WEIGHT, the other ones are adjusted as 1. Values differing from 0 to 1 are given to OLD_BUYING_TIME_WEIGHT to capture the effect of it on prediction accuracy. The same method is applied to other constants ant their optimum values are obtained. The results of these experiments are shown in Figure 8, Figure 9 and Figure 10. It is observed that the optimum value for OLD_BUYING_TIME_WEIGHT is 0.1, for MIDDLE_BUYING_TIME_WEIGHT is 0.5 and for RECENT_BUYING_TIME_WEIGHT is 1, since the MAE is smaller for these values.

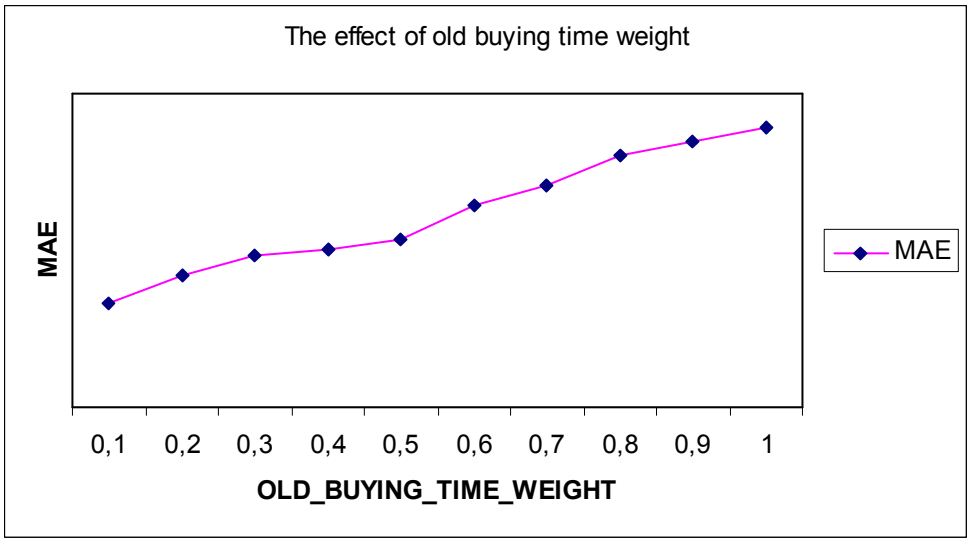

Figure 8 The Effect of OLD_BUYING_TIME_WEIGHT for CF

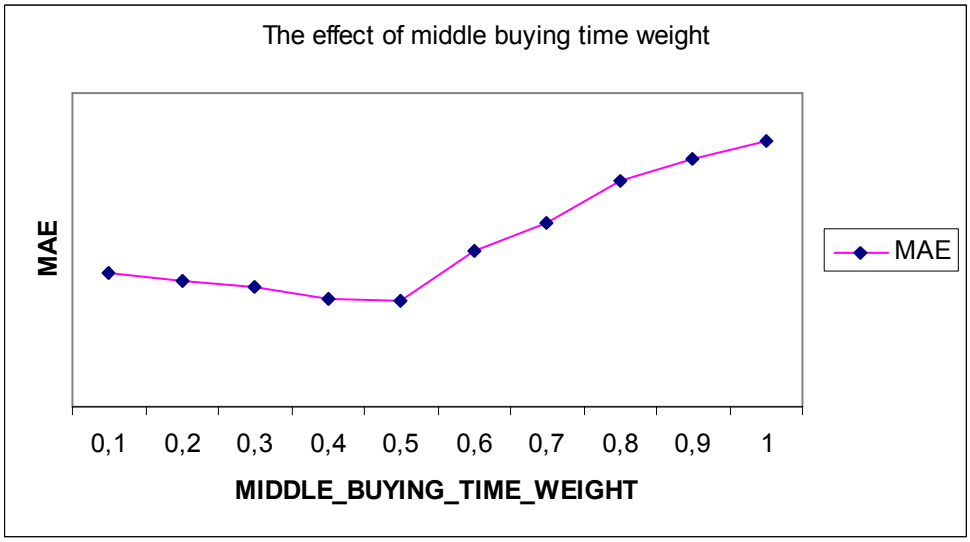

Figure 9 The Effect of MIDDLE_BUYING_TIME_WEIGHT for CF

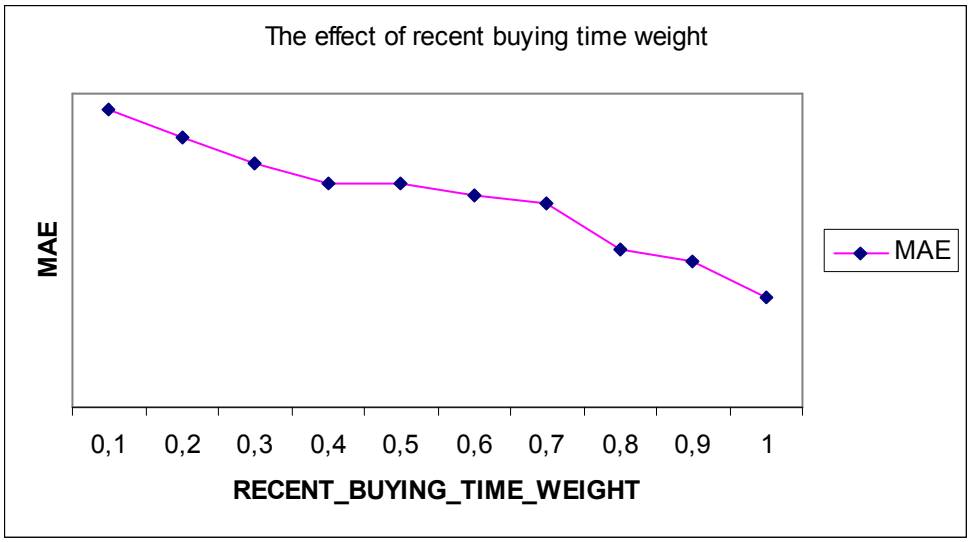

Figure 10 The Effect of RECENT_BUYING_TIME_WEIGHT for CF

4.5.4 Impact of Demographic Information

Another improvement for recommendation process was using the demographic information of users. The first aim of this process is adding a spatial approach to temporal recommendation and having a spatiotemporal recommendation system. Thus, while using the demographic information of users, it was decided to pay much more attention to place information. However, the results showed that age information affects the result more than the place information. To find the best weight values for age, gender, occupation and location information, a controlled experiment is conducted. Sequentially, one value is multiplied with 1 while the others are multiplied with 0. After most effective property is found, several weights are tried for these four properties and results are shown in Table 9. It is observed that, using age and location information affects the results in contrast to gender and occupation which seem not to be very effective on this domain set. MAE is in its minimum when occupation and gender is set to 0, age is set to 0.6 and location is set to 0.4. It can be stated that using location information improves the results.

Table 9 Effect of Demopraphic Features

Age	Gender	Location	Occupation	MAE
1	0	0	0	0,8987
0	1	0	0	1,003
0	0	1	0	0,9389
0	0	0	1	1,2376
0,5	0	0,5	0	0,8123
0,6	0	0,4	0	0,7823
0,7	0	0,3	0	0,8145

As expressed in **User Similarity Calculation** chapter, to find user age similarity of users a threshold value is defined. This threshold is used to decide whether two ages are similar or not. The amount of this value affects the demographic similarity results. In Figure 11 , it is observed that optimum value for this threshold is 15, since the error is in its minimum.

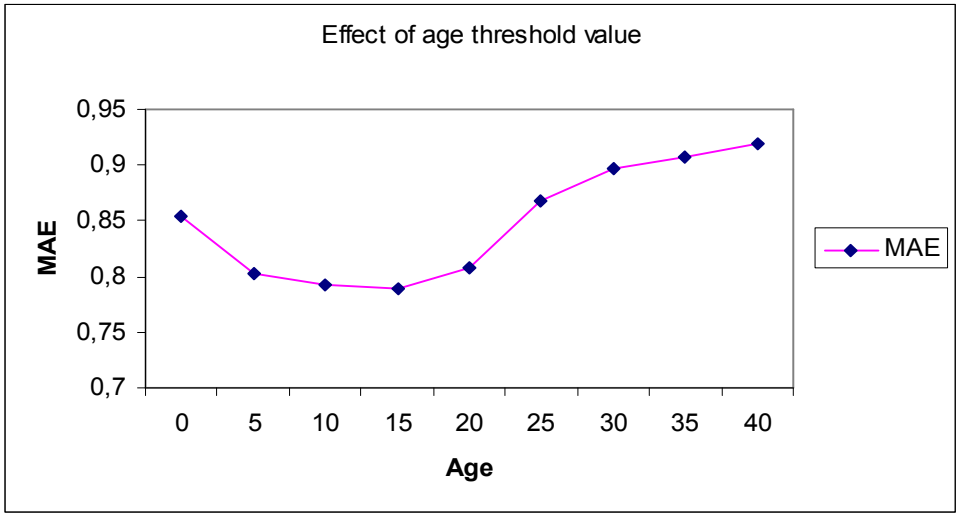

Figure 11 Effect of Age Threshold Value

After demographic similarities are calculated, the way this value is combined with collaborative similarity affects the result. This corresponds to the affect of DEMOPRAGHIC_SIMILARITY_WEIGHT. As shown in, best results are obtained when DEMOPRAGHIC_SIMILARITY_WEIGHT is 0.4.

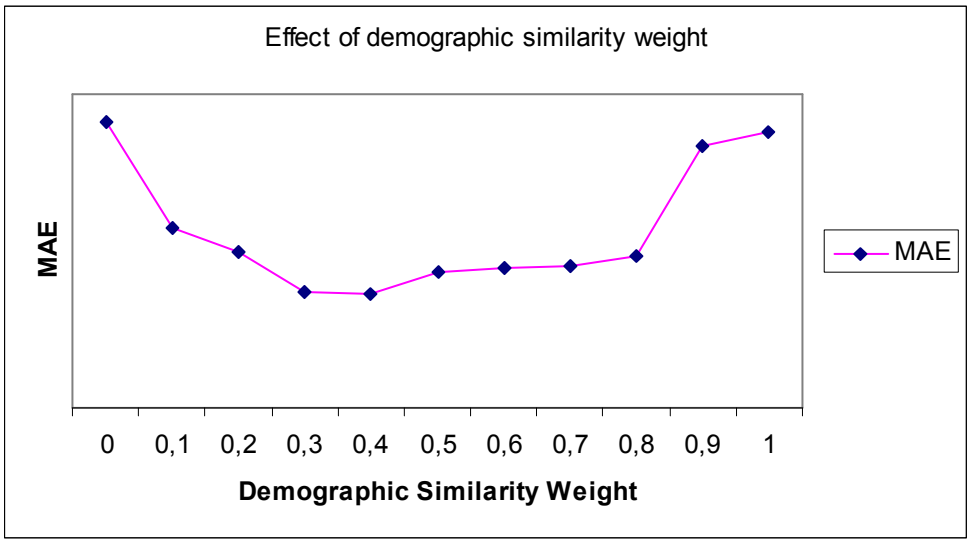

Figure 12 Effect of Demographic Similarity Weight

4.5.5 Impact of CB and CF weight

The other evaluation must be done on CB and CF weights of the total recommendation algorithm. As explained before, a hybrid method is used in this study. The results of CB algorithm is combined with the results of CF algorithm. This combination is done by multiplying the ratings of each algorithm with a constant and then adding them. These constants must be equal to 1 after addition. For instance, if the CB weight is 0,3, then CF weight must be 0,7. Below are results on different combinations of those weights. As can be seen from Figure 13, optimum results are obtained when the weights of CB and CF algorithms are equal.

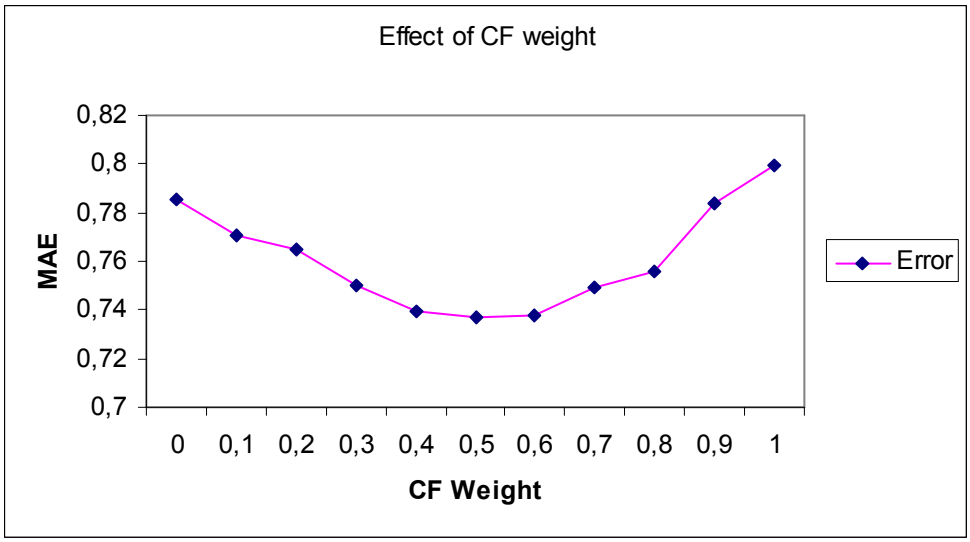

Figure 13 Effect of CF Weight

CHAPTER 5

CONCLUSION AND FUTURE WORK

Within this thesis study, a recommendation system algorithm emphasized on the effect of temporal information to rating prediction is introduced. TDRS uses both CB recommendation and CF recommendation algorithms by considering the launch time of items and date of ratings. In addition to time dependent information, user demographic information is also used during user similarity calculation process of CF recommendation.

First, an introduction to the study is made and then recommendation systems are explained in detail. Current approaches and their relation with this study are presented. After that, system architecture of TDRS is examined by also covering the algorithms used. Finally, the evaluation approach used to find the performance of TDRS is explained. The results and improvements are determined and the effects of system constants on recommendation process are discussed.

The analysis done through this study shows that using temporal information on both CB prediction and CF prediction improves the accuracy of the results. In addition to that, demographic information is also useful to find the similarities of users, which results in again an improvement on recommendation process. Using item launch dates as temporal information in CB method and hybridizing it with a

temporal CF algorithm which combining demographic and collaborative user similarity is a contribution to recommendation studies.

As a future work, more efficient ways of using temporal information may be considered. Instead of using a discrete approach such as dividing the time into three major periods, a continuous function should be determined to catch the relationship between time and interest drifting more effectively. Another improvement can be applying a personal weight mechanism which allows a dynamic control of the effect of weights on user preferences. This dynamic weight approach can be used on weights used to reflect temporal differences and/or weights of CB and CF algorithms.

Temporal information may consist of seasonal or daily changes in other domains. Local festivals or weather conditions may affect user choices. By considering these situations, algorithms can be tested on other domains.

REFERENCES

[1] Mustafa Bilgic, Explanation for Recommender Systems: Satisfaction vs. Promotion. May 19, 2004.

[2] Rosario Sotomayor, Joe Carthy and John Dunnion, The Design and Implementation of an Intelligent Online Recommender System.

[3] MSearchGroove, www.msearchgroove.com, last accessed date: May 2010.

[4] G. Adomavicius, A.Tuzhilin. (2005) "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions", IEEE Transaction on knowledge and data engineering, VOL 17, NO.16, JUNE 2005.

[5] Improving content-based similarity measures by training a collaborative model. Richard Stenzel, Thomas Kamps. ISMIR 2005, 6th International Conference on Music Information Retrieval, pp 264-271, London, UK, 11-15 September 2005, Proceedings.

[6] Grouplens, www.grouplens.org, last accessed date: May 2010.

[7] Anne Yun-An Chen, Dennis McLeod. Collaborative Filtering for Information Recommendation Systems.

[8] Magnus Mortensen. Design and Evaluation of a Recommender System.

[9] Yang, C.C.; Chen, Hsinchun; Honga, Kay (2003), "Visualization of large category map for Internet browsing", Decision Support Systems 35 (1): 89-102.

[10] Wikipedia, en.wikipedia.org/wiki/Information_filtering, last accessed date: May 2010.

[11] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work, pages 175{186. ACM Press, 1994.

[12] Daniel Billsus, Michael J. Pazzani . Learning Collaborative Information Filters.

[13] Anne Yun-An Chen and Dennis McLeod. Collaborative Filtering for Information Recommendation Systems.

[14] Gözde Özbal, A content boosted collaborative filtering approach for movie recommendation based on local & global user similarity and missing data prediction.

[15] Review of Personalization Technologies: Collaborative Filtering vs. ChoiceStream's Attributized Bayesian Choice Modeling.

[16] Item-based Collaborative Filtering Recommendation Algorithms Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl.

[17] Brachman, R., J., Khabaza, T., Kloesgen, W., Piatetsky-Shapiro, G., and Simoudis, E. 1996. Mining Business Databases. Communications of the ACM, 39(11), pp. 42-48, November.

[18] Ungar, L. H., and Foster, D. P. (1998) Clustering Methods for Collaborative Filtering. In Workshop on Recommender Systems at the 15th National Conference on Artificial Intelligence.

[19] Sarwar, B. M., Karypis, G., Konstan, J. A., and Riedl, J. (2000). Analysis of Recommendation Algorithms for E-Commerce. In Proceedings of the ACM EC'00 Conference. Minneapolis,MN. pp. 158-167.

[20] G.-R. Xue, C. Lin, Q. Yang, W. Xi, H.-J. Zeng, Y. Yu, and Z. Chen. Scalable collaborative filtering using cluster-based smoothing. In Proc. of SIGIR, 2005.

[21] Hugo Siles Del Castillo. November 6, 2007. Hybrid Content-Based Collaborative-Filtering Music Recommendations.

[22] Raymond J. Mooney and Loriene Roy. Content-based book recommending using learning for text categorization. In Proceedings of the Fifth ACM Conference on Digital Libraries, pages 195{204. ACM Press, 2000.

[23] Robin Burke. Hybrid Systems for Personalized Recommendations.

[24] Clustering approach for hybrid recommender system. Qing Li and Byeong Man Kim. 2003 IEEE / WIC International Conference on Web Intelligence, pp 33-38, Halifax, Canada, 13-17 October 2003.

[25] Michael J. Pazzani, Daniel Billsus. Content Based Recommendation Systems.

[26] IMDB, http://www.imdb.com/, last accessed date: May 2010.

[27] Souvik Debnath, Niloy Ganguly, Pabitra Mitra. Feature Weighting in Content Based Recommendation System Using Social Network Analysis.

[28] Yehuda Koren. Collaborative Filtering with Temporal Dynamics.

[29] Shanle Ma, Xue Li, Yi Ding, and Maria E. Orlowska, A Recommender System with Interest-Drifting.

[30] Liang Xiang, Qing Yang, Time-dependent Models in Collaborative Filtering based Recommender System.

[31] Mustafa Azak, Crossing: A Framework to Develop Knowledge-Based Recommenders In Cross Domains.