A CONTENT BASED MOVIE RECOMMENDATION SYSTEM EMPOWERED BY COLLABORATIVE MISSING DATA PREDICTION

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

BY

HİLAL KARAMAN

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

JUNE 2010

Approval of the thesis:

A CONTENT BASED MOVIE RECOMMENDATION SYSTEM EMPOWERED BY COLLABORATIVE MISSING DATA PREDICTION

submitted by **HİLAL KARAMAN** 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 _____

Dr. Ayşenur Birtürk
Computer Engineering Dept., METU _____

Asst. Prof. Dr. İlkay Ulusoy
Electrical and Electronics Engineering Dept., METU _____

Özgür Alan
ORBİM Software and Hardware Services _____

Date: 08.06.2010

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: Hilal KARAMAN

Signature :

ABSTRACT

A CONTENT BASED MOVIE RECOMMENDATION SYSTEM EMPOWERED BY COLLABORATIVE MISSING DATA PREDICTION

KARAMAN, Hilal
M.S., Department of Computer Engineering
Supervisor: Assoc. Prof. Ferda Nur ALPASLAN

June 2010, 100 pages

The evolution of the Internet has brought us into a world that represents a huge amount of information items such as music, movies, books, web pages, etc. with varying quality. As a result of this huge universe of items, people get confused and the question “Which one should I choose?” arises in their minds. Recommendation Systems address the problem of getting confused about items to choose, and filter a specific type of information with a specific information filtering technique that attempts to present information items that are likely of interest to the user. A variety of information filtering techniques have been proposed for performing recommendations, including content-based and collaborative techniques which are the most commonly used approaches in recommendation systems. This thesis work introduces *ReMovender*, a content-based movie recommendation system which is empowered by collaborative missing data prediction. The distinctive point of this study lies in the methodology used to correlate the users in the system with one

another and the usage of the content information of movies. *ReMovender* makes it possible for the users to rate movies in a scale from one to five. By using these ratings, it finds similarities among the users in a collaborative manner to predict the missing ratings data. As for the content-based part, a set of movie features are used in order to correlate the movies and produce recommendations for the users.

Keywords: recommendation system, information filtering, content-based filtering, collaborative filtering, demographic recommendation, missing data prediction, user modeling, feature weighting

ÖZ

İŞBİRLİKÇİ EKSİK VERİ TAHMİNİ YOLUYLA DESTEKLENMİŞ İÇERİK TABANLI FİLM ÖNERİ SİSTEMİ

KARAMAN, Hilal
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Ferda Nur ALPASLAN

Haziran 2010, 100 sayfa

Internet, geçirdiği evrim sonucunda bizleri farklı özelliklere sahip müzik, film, kitap, web sayfaları gibi birçok bilgi kaleminin sunulduğu bir dünyaya getirmiştir. Bu devasa öğeler evreninin bir sonucu olarak, insanların kafası karışmaya ve "Hangisini seçmeliyim?" sorusu akıllara takılmaya başlamıştır. Tavsiye Sistemleri, seçilmesi gereken öğeler hakkındaki bocalama sorununu ele almakta olup, belirli bir bilgi filtreleme tekniği ile bilgileri filtrelemekte, kullanıcıların ilgisini çekmesi muhtemel olan öğeleri onlara sunmaktadır. Öneri sistemlerinde yoğunlukla kullanılan içerik tabanlı ve işbirlikçi teknikler de içinde olmak üzere, tavsiye üretmek için kullanılabilecek çeşitli bilgi filtreleme teknikleri ortaya atılmıştır. Bu tez çalışması, işbirlikçi ve içerik tabanlı filtreleme yöntemlerinin her ikisine de dayanan bir film öneri sistemi olan *ReMovender*'ı tanıtmaktadır. Bu çalışmanın ayırt edici noktaları, kullanıcılar arasında ilişki kurma yöntemi ve filmlerin içerik bilgilerinin kullanılma şeklidir. *ReMovender*, kullanıcılara filmleri bir ve beş aralığındaki oylarla oylama fırsatı vermektedir. Eksik verileri tamamlamak amacıyla, bu oy bilgilerini işbirlikçi

yöntemle kullanıcılar arasında benzerlik bulmada kullanmaktadır. İçerik tabanlı kısımda ise, filmlerin içerik bilgileri kullanılarak bu filmler ilişkilendirilmekte ve kullanıcılara öneriler sunulmaktadır.

Anahtar Kelimeler: öneri sistemi, bilgi filtreleme, içerik filtreleme, işbirlikçi filtreleme, demografik tavsiye, eksik veri tamamlama, kullanıcı modelleme, özellik katsayısı

ACKNOWLEDGEMENTS

Firstly, I would like to express the deepest appreciation to Assoc. Prof. Ferda Nur Alpaslan for her guidance, support and encouragement throughout this study.

It is a pleasure to thank my family; my mother Hatice Karaman, my father D. Tarık Karaman and my brother M. Tuğberk Karaman, who made this thesis possible with their love and support.

I am deeply indebted to my friends Gözde Özbal, Goncagül Demirdizen, Candan Ceylan, Seda Çakıroğlu, Gizem Demir and Ertay Kaya whose support and encouragement helped me during the research and development of this thesis.

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

Finally, my special thanks go to my husband Serhat Cevahir who has made available his support in a number of ways through not only the course of this study but also any course of my life.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

RS	Recommendation System
CB	Content Based
CF	Collaborative Filtering
PC	Pearson Correlation
PCC	Pearson Correlation Coefficient
MDP	Missing Data Prediction
LS	Local Similarity
GS	Global Similarity
LU & GU	Local User Similarity & Global User Similarity
MAE	Mean Absolute Error

CHAPTER 1

INTRODUCTION

1.1 Background

Up until recently, recommendations from people have been the most helpful way to find out new movies to watch. However, these recommendations sometimes result in disappointments since everyone has different movie tastes as a result of different characteristics and life styles. In order to get more useful recommendations, the most proper way to follow is to take account of the preferences of people who are similar to us. Recommendation Systems (RSs) automate the facility of finding similarities among users for producing recommendations and provide a solution for the information overload problem.

1.2 Recommendation Systems As a Research Area

The roots of RSs go back to cognitive science, approximation theory, information retrieval and forecasting theories. Additionally, RSs have links to management science and consumer choice modeling in marketing.

RSs have become an independent research area in middle of 1990s when researchers started to focus on recommendation problems that explicitly rely on the rating structure and produced the first papers on this area [1]. In this manner, the recommendation problem was reduced to the problem of estimating ratings for the items that have not been observed by a user yet [2].

Today, RSs are built by the combination of ideas from different areas such as Artificial Intelligence, Natural Language Processing, Human-Computer Interaction, Sociology and Information Retrieval. The interest in this research area maintains its popularity because it constitutes a problem-rich research area with the need of solutions to many open-ended problems. Furthermore, there are many practical applications that help users to deal with information overload and provide personalized recommendations, content, and services to them [1].

1.3 Problem Definition

This thesis study focuses on the design, development and evaluation of a movie recommendation system which is named as *ReMovender*. The system makes the use of ratings given by users to movies, demographic data of the users and content of the movies in order to produce powerful recommendations.

1.4 Motivation

ReMovender addresses one of the main problems suffered by RSs, which is the data sparseness problem. The system manages to overcome this problem with the help of missing data prediction process that is supported by the demographic data of the users. *ReMovender* also represents a new approach for the correlation of movies by using their contents to produce CB recommendations.

1.5 Structure of This Thesis

This thesis study consists of 6 main chapters and the contents of next 5 chapters are described in this section.

Chapter 2 provides a general introduction to the recommendation problem and the recommendation systems. It gives a model of the recommendation process and describes its general concepts. The chapter also makes a brief introduction to the major recommendation techniques, gives examples of the problems suffered by RSs

and makes a comparison between the recommendation techniques in terms of problems suffered.

Chapter 3 is the related work part. This chapter represents examples of recommendation systems along with an evaluation of their positive and negative characteristics. The chapter is divided into three main subsections which discuss RSs designed for the movie domain, RSs which make use of demographic data and RSs from domains other than the movie domain respectively. The movie RSs are also categorized as the CF systems, CB systems and the hybrid systems.

Chapter 4 introduces *ReMovender* and makes a detailed description of the system. Firstly, it provides a general overview of the system which is followed by the system architecture. Design issues are presented in detail and the main parts of the recommendation process such as predicting missing data and producing CB recommendations are explained.

Chapter 5 makes the evaluation of the system. It compares the results produced by *ReMovender* with the results already produced by the existing movie recommendation systems. This chapter also makes an internal evaluation of the system and provides graphical results in order to make it clearer.

Chapter 6 concludes this thesis work and discusses the positive and negative characteristics of *ReMovender*. It explains future work that can be done to improve the performance of the system and power of the recommendations.

CHAPTER 2

RECOMMENDATION SYSTEMS

This chapter provides a general background in the area of recommendation systems. First of all, the recommendation problem and RSs are described in the following section. The next section presents a model of the RSs and then the general recommendation concepts are explained. Consequent sections define the major types of recommendation techniques and the basic problems which occur in RSs. Lastly, a comparison among recommendation techniques is made and a general conclusion is provided.

2.1 The Recommendation Problem and Recommendation Systems

Recommendations are a part of everyday life where people sometimes rely on external knowledge to make decisions about an artifact of interest or a course of action [3]. Everyone can consider some examples of recommendation from everyday life. For instance, one can read movie reviews in a magazine in order to decide which movie to see. Or in a different manner, one might visit the local bookstore and talk to the owner about his/her interests and current mood, in order to be recommended a few books he/she'd probably like. Finally, it is also possible to give a try to a cafe which is very popular and always crowded. These examples help to clarify the concept of recommendation. A person is faced with a decision which is a choice among a universe of alternatives. The universe is usually quite large and the person does not know what all the alternatives are and how to choose among them [4].

While buying a CD, the buyer can rely on the judgment of a person who shares similar tastes in music. At other times, the judgments may be based on available

information about the CD itself and the buyer's known preferences. These are some of the factors which may influence a person in making these choices. Ideally, one would like to model as many of these factors as possible in an RS in order to get more realistic recommendations [3].

An RS is an intelligent system that makes suggestions about artifacts to a user. It uses the stored preferences to suggest items of interest to the users being served [5]. For instance, they try to predict whether a user would be interested in seeing a particular movie or not [3]. An RS produces a ranked list of items on which a user might be interested, in the context of his/her current choice of an item [6]. A more specific definition of an RS is provided in [7] as being the systems which produce individualized recommendations as output or have the effect of guiding the user in a personalized way to interesting or useful items in a large space of alternatives. This definition indicates that each user in an RS will be presented with different information sources or items according to the information gained about him/her.

Although the roots of RSs can be traced back to the extensive work in the approximation, information retrieval, forecasting theories and the consumer choice modeling in marketing, RSs emerged as an independent research area in the middle of 1990's when researchers started focusing on recommendation problems that explicitly rely on the ratings structure. With a common formulation, the recommendation problem is reduced to the problem of estimating ratings for the items that have not been seen by a user yet. After estimating the ratings for the unrated items, items with the highest estimated ratings can be recommended to the user [8].

2.2 Model of the Recommendation Process

Figure 1 [4] is a general model of recommendation which summarizes the recommendation process by identifying the communication among its components.

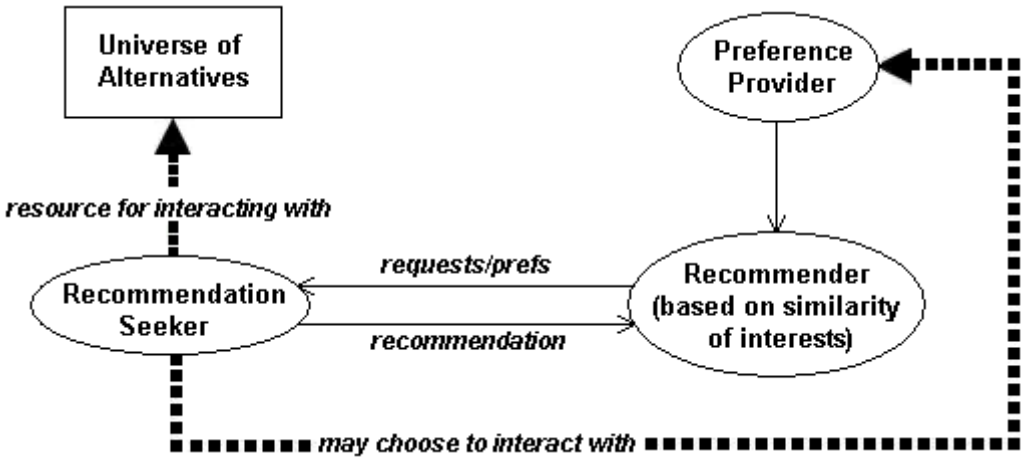

Figure 1 – Model of the Recommendation Process [4]

A recommendation seeker can be considered as a user of the system. The seeker may directly ask for a recommendation, or the recommender may produce recommendations without the seeker's demand. Seekers may get into contact with preference provider and represent some information about their preferences. If not, the recommender requests preference information from the user explicitly. Based on the known preferences of the seeker and those of other people, the recommender recommends items the seeker probably will like. The seeker may use the recommendation to select items from the universe of alternatives or to communicate with like-minded other users as the recommender also identifies people with similar interests.

This model is very general and covers only a broad range of recommendation activities. It is important to note that the real activities may vary significantly; in particular, they may not instantiate some aspects of the model.

2.3 General Recommendation Concepts

This section represents the formal formulation of a recommendation problem. Let the following be the items corresponding to the formulation:

- *C*: The set of all users.
- *S*: The set of all possible items that can be recommended, such as books, movies, or restaurants.
- *u:* A utility function that measures usefulness of a specific item $s \in S$ to user $c \in C$, i.e., $u : C \times S \to R$, where *R* is a totally ordered set.

The user space and the space of possible items can be very large, ranging in hundreds of thousands or even millions in some applications. For each user $c \in C$, choosing such item $s' \in S$ that maximizes the user's utility is desired.

More formally:

$$\forall c \in C, s_c' = \arg\max_{s \in S} u(c, s) \tag{1}$$

represents the item to be chosen [8].

RSs usually represent the utility of an item by a rating, which indicates how a particular user liked a particular item. However, in general utility can be an arbitrary function, including a profit function.

Each element in the user space *C* can be defined with a user profile that consists of various user characteristics, such as age, gender, income, marital status, etc. Similarly, each element of the item space *S* is defined with a set of characteristics that depend on the domain. For instance, in a system working on the movie domain, each movie can be represented by its title, genre, director, year of release, actors, etc.

RSs usually represent utility with ratings and the utility is initially defined only on the items which were previously rated by the users. The ratings are kept in the cells of a 2-dimensional user-item matrix having users as rows and items as columns. An example user-item rating matrix for a movie recommendation application is presented in Table 1. The ratings are specified on the scale of 1 to 5 and the "Ø" symbol means that no rating is available for the corresponding user-item pair. The recommender should be able to predict the ratings of the non-rated user-movie combinations and provide recommendations based on these predictions.

Table 1 – A Fragment of Rating Matrix in a Movie Recommendation System

	Fight Club	Avatar	Im Juli	Oldboy
James	4	3	2	4
Sophie	Ø	4	5	5
Jude	2	2	4	Ø
Mary	3	Ø	5	2

According to (1), after the estimation of the unknown ratings, actual recommendations are made to a user by selecting the highest rating among all the estimated ratings for that user. Alternatively, *n* best items can be recommended to a user.

Estimation of the ratings for the non-rated items can be done with many different methods according to which the RSs are usually classified. The following section presents a classification of recommendation approaches that are proposed in the literature.

2.4 Major Types of Recommendation Techniques

Based on the choice of reference characteristics, RSs have fallen into two main categories which are the systems using content-based (CB) methods and the systems using collaborative filtering (CF) methods [9]. As each one has both advantages and disadvantages, these two methods are combined by the hybrid approach in order to overcome the difficulties and disadvantages of both methods. Besides the two major types, demographic approach which is another recommendation technique is also analyzed in this section.

2.4.1 Collaborative Filtering

CF approach is a process of filtering information based on the collaboration of users, or the similarity between items [10]. Systems using this method make recommendations to a target user based on the opinions of other users who have similar preferences with the target user [11].

CF has been applied in many contexts. *FilmTrust* [12], *MovieLens* [13], *Recommendz* [14], *Film-Conseil* [15], PHOAKS [16] and *GroupLens* [17] are just a few of the websites that implement RSs using CF methods.

A variety of CF algorithms have been employed. Traditional algorithms compute similarity values between all pairs of users and predictions for a given user are generated by weighting other users' ratings proportionally to their similarity to the given user [9]. A variety of similarity metrics including correlation, mean-squared difference, or vector similarity are possible. Other algorithms such as Bayesian network models, dependency network models, or clustering models construct a model of underlying user preferences.

Recent work in RSs shows widespread use of CF techniques. In general, collaborative RSs follow the following steps [18]:

- Record behavior of a large number of people,
- Select a number of 'neighbors' - other people whose past behavior is similar to this person - for the current user,
- Extrapolate future behavior for the user, based on behavior of the neighbors

CF methods utilize explicit or implicit ratings from many users to recommend items to a given user. Pure collaborative systems tend to fail when little is known about a user, or when he or she has uncommon interests [9]. However, a big advantage of the CF technology is that it can offer you recommendations even though you do not know the content of recommended items [11].

2.4.2 Content-based Filtering

CB approach is based on the matching of user profile and some specific characteristics of an item [10]. The main idea behind CB RSs is that they recommend items which are similar to the ones already rated with a high rating by the target user.

CB recommendation approach has its roots in information retrieval and information filtering research [19]. Therefore, it can be considered as an outgrowth and continuation of information filtering [20]. However, the improvement of the CB recommendations over traditional information retrieval approaches comes from the fact that while information is gathered for the user, user preferences are also taken into account.

A CB system defines the items of interest by their important features. For instance, NewsWeeder [21], which is a text RS, uses the words of their texts as features, while [22], a matchmaking system, uses the characteristics and hobbies of a user as features. A CB recommender constructs a user profile based on the features present in the items the user has rated before. Many learning methods such as decision trees, neural networks, and vector-based representations can be employed to derive the user profiles.

The workflow of a CB RS can be summarized with Figure 2 [19]:

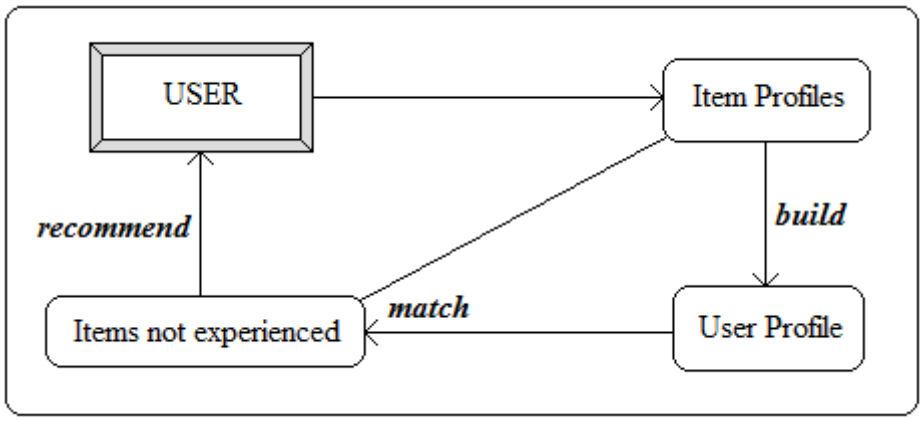

Figure 2 – Workflow of a Content-based Recommendation System

First of all, the system has a huge database consisting of the items to be recommended and the features of these items. Then the users provide some sort of information about their preferences to the system. Combining the item information with the user preferences, the system builds a profile of the users. According to the information existing in a target user's profile, the system recommends suitable items to the user.

As a shortcoming, CB systems cannot account for community endorsements. For instance, it might recommend the movie "The Mexican" to a user who likes Brad Pitt and Julia Roberts, even though many like-minded users strongly dislike the film. On the other hand, it has an advantage over CF method that CB systems can even recommend new items to users without any history in the system while collaborative systems cannot [9].

2.4.3 Hybrid Methods

Several researchers are exploring hybrid CF and CB recommenders to smooth out the disadvantages of each and gain better performance with fewer drawbacks [9]. Different ways of combining recommendation methods into a hybrid RS can be described as follows:

- CF and CB methods are implemented individually and their predictions are combined by using a weighting procedure.
- CB characteristics are incorporated into a CF approach.
- CF characteristics are incorporated into a CB approach.
- A general unifier model can be constructed to incorporate both CB and CF characteristics.

Basu et al. (1998) present a hybrid CF and CB movie recommender. Collaborative features (e.g., Bob and Mary like Titanic) are encoded as set-valued attributes. These features are combined with more typical content features (e.g., Traffic is rated R) to inductively learn a binary classifier that separates liked and disliked movies. Also in a movie recommender domain, Good et al. (1999) suggest using CB software agents to automatically generate ratings to reduce data sparseness. Claypool et al. (1999) employ separate collaborative and CB recommenders in an online newspaper domain, combining the two predictions using an adaptive weighted average: as the number of users accessing an item increases, the weight of the collaborative component tends to increase. Web hyperlinks and document citations can be thought of as implicit endorsements or ratings. Cohn and Hofmann (2001) combine document content information with this type of connectivity information to identify principle topics and authoritative documents in a collection.

2.4.4 Demographic Recommendation Methods

In demographic recommendation, users are classified based on their personal data, which they provided during the registration process themselves [23]. Alternatively, this data can be extracted from the survey responses, population census, etc. Each item is assigned to one or more classes with certain weights and the user is attracted to items from the class closest to their profile [24].

In demographic recommendation technique, users are categorized based on their personal attributes and demographic features. This technique is similar to the CF technique in the sense that it forms correlations between users of the system. However, compared to CB and CF methods, demographic recommendation technique has the advantage of not requiring a history of user ratings.

2.5 Basic Problems of Recommendation Systems

2.5.1 Cold Start Problem

Computer-based information systems involving a degree of automated data modeling face with the cold start problem. This problem is particularly related with the issue that the system cannot make any inferences for users or items about which it has not gathered sufficient information yet [25].

The cold start problem is mostly encountered in RSs. Typically, an RS compares the user's profile to some reference characteristics which can be obtained from the CB approach or the CF approach. The system cannot produce meaningful recommendations when it does not have enough information about a user or an item.

Three kind of cold start problems including new item, new user, and new system are explained in [7]. Obviously, the new user problem occurs when a user is new in the system and the system does not have enough information about this user’s preferences. Similarly, when an item is just introduced and none of the users in the system has provided a feedback about this item, the new item problem occurs. As the third one, the new system problem is a combination of new user problem and new item problem [26].

CB approach is based on matching the characteristics of an item against related features in the user's profile. For this purpose, the system must construct a detailed model of the user's tastes and preferences. The system can do this either explicitly by querying the user, or implicitly by observing the user's behavior. In both cases, the cold start problem implies that before the system can provide intelligent recommendations, the user has to use the system and contribute to the construction of user profile [25].

Similarly, CF approach is affected by the cold-start problem. In this approach, the RS identifies users who share similar preferences with the active user, and propose items which the like-minded users preferred and the active user has not seen yet. Due to the cold start problem, this approach would fail to consider the items which are not rated by any users previously [27].

2.5.2 Data Sparsity Problem

Data sparsity is one of the biggest problems that the RSs encounter. It is concluded in [28] that the quality of CF recommendations is highly dependent on the sparsity of available data. The sparsity problem is a major issue limiting the quality of recommendations and the applicability of CF in general, and it occurs when feedback data is insufficient for identifying neighbors [29].

As stated, this problem is encountered in collaborative RSs which construct recommendations based on the ratings of the other users. These systems generally construct a user-item matrix containing the ratings given by the users to the movies. As the number of users and items increase, the user-item matrix gets larger and the sparsity evolves. Several methods have been proposed to deal with the sparsity problem. Most of them succeed in providing better recommendations but a general model for dealing with sparsity cannot be introduced [30].

2.5.3 Overspecialization Problem

RSs also suffer from the over specialization problem which means that the users are restricted to getting recommendations which resemble to those already known or defined in their profiles [31]. This problem prevents users from discovering novel items and limits the originality of recommendations. Some systems make use of genetic algorithms or define semantic relations within their vocabulary in order to solve the over specialization problem. After solving the problem, systems will be able to present a wide range of options and a set of different alternatives to the users so that they will be satisfied more.

2.5.4 Scalability Problem

Scalability is a desirable property of a system which indicates its ability to either handle growing amounts of work in a graceful manner or to be readily enlarged [32].

As for the RSs with many users and many items, maintaining scalability is a challenging issue. Nearest neighbor algorithms are widely used in recommenders and these algorithms require computation that grows with both the number of users and the number of items. With millions of users and items, a typical web-based RS will suffer serious scalability problems [33].

Most RSs employ variations of CF for formulating suggestions of items relevant to users' interests. However, collaborative filtering requires expensive computations that grow polynomial with the number of users and items in the database. Methods proposed for handling this scalability problem and speeding up recommendation formulation are based on approximation mechanisms. Even if they improve performance, most of the time they result in accuracy reduction [34].

2.6 Comparison of Recommendation Techniques

Most RSs suffer from the basic recommendation problems described in the previous section. However, this is quite related with the recommendation technique that the system makes use of. For instance, CF technique suffers from all of the mentioned problems, whereas CB filtering does not suffer from the cold start problem in case of new items. Demographic methods suffer from cold start problem in case of new users, data sparsity problem, over specialization problem and lack of content information which is special to this method. These are summarized in Table 2:

Table 2 – Problems Suffered by Recommendation Techniques

Recommendation Technique	Problems Suffered
Collaborative Filtering	- Cold Start Problem for New Users - Cold Start Problem for New Items - Sparsity Problem - Over Specialization Problem - Scalability Problem
Content-based Filtering	- Cold Start Problem for New Users - Sparsity Problem - Over Specialization Problem - Lack of content information
Demographic	- Cold Start Problem for New Users - Over Specialization Problem - Demographic data collection

CHAPTER 3

RELATED WORK

Several RSs, most of which are applied to the movie domain have been developed so far. These systems can be classified according to the recommendation technique, dataset or information filtering technique they use and the domain they work on. In this chapter, the main characteristics of some existing RSs are described along with their positive and negative properties. In section 3.1, RSs working on movie domain are provided with subsections namely; demographic, CF, CB and hybrid RSs whereas section 3.2 gives examples of RSs in book domain, music domain, TV domain and e-commerce domain.

3.1 Movie Recommendation Systems

3.1.1 Systems Based on Pure Collaborative Filtering

This section focuses on movie RSs which are based on CF approach. As most of the current RSs use CF, the most related ones to *ReMovender* are provided in this section.

Firstly, the study in [35] focuses on memory-based CF algorithms which usually suffer from data sparsity. The user-item matrix is usually sparse and this leads to poor prediction quality. For this purpose, this study uses a Missing Data Prediction (MDP) algorithm which helps to increase density of user-item matrix. The proposed effective MDP algorithm takes information of both users and items into account and tries to fill the empty cells in the user-item matrix. This algorithm predicts the missing data of a user-item matrix if and only if it is thought to bring positive

influence for the recommendation of active users instead of predicting every missing data of the user-item matrix.

In order to compute similarity of users and items accurately, PCC algorithm is used. PCC is employed to define the similarity between two users based on the items they rated in common. Similarly, the basic idea in similarity computation between two items is to first isolate the users who have rated both of these items and then apply a similarity computation technique to determine the similarity between them. After predicting the missing data in the user-item matrix, the next step is to predict the ratings for the active users. This process is nearly the same as predicting the missing data for passive users. Most similar neighbors are selected with a threshold η and the predictions are done according to the ratings of these nearest neighbors. Items with the biggest predicted ratings are then recommended to the active users.

The evaluation results show that the proposed method outperforms other CF algorithms and it is more robust against data sparsity thanks to effective MDP process. A disadvantage of MDP may be that it is time consuming and the recommendation process takes longer time as the total number of users and items increase in the system. However, the recommendation process is conducted off-line and the active user will not be aware of this process.

As another example, it worth discussing [36] which provides an elegant and effective framework for CF. This study presents three main contributions which are the concepts of LU & GU, surprisal-based vector similarity and an application of the concept of maximin distance in graph theory. The intuition behind surprisal-based vector similarity is that less common ratings for a specific item provide more discriminative information than the most common ones. It expresses the relationship between any two users based on the quantities of information contained in their ratings, called surprisal.

LS among users is calculated by using PCC algorithm and two users are defined as globally similar if they can be connected through their locally similar neighbors. Due to the data sparsity problem, there exist little amount of similar neighbors and little amount of ratings for a particular item. Under the sparse data set condition, GS

of users improves the performance of the algorithm compared to using LS alone. After finding local nearest neighbors and global nearest neighbors of a target user, a prediction for a movie is made according to the nearest neighbors' ratings on that movie. Local and global predictions are given different weight values according to their impact on the effectiveness of recommendations.

This research handles data sparsity in a successful way and improves the accuracy of predictions with the help of the introduced concepts mentioned above. The overall approach provides an improvement for user-based algorithms. Therefore, this approach can be applied to other approaches which make use of both user-based and item-based algorithms to replace the traditional user-based approach so that they can achieve a higher performance.

3.1.2 Systems Based on Pure Content Based Filtering

As most of the systems combine CB filtering with CF in order to increase efficiency, there are not as many examples of pure CB systems as CF systems.

It is important to describe the work done in [37], which is based on feature weighting in CB recommendations. The study makes use of social networks and assigns different weight values to the features of a movie. Having the idea that the release date of a movie cannot affect the user's rating for the movie as much as the genre; the weights are assigned to features depending on their importance to users. The weight values are estimated from a set of linear regression equations obtained from a social network graph having items as nodes. The graph represents human judgment of similarity among items aggregated over a large population of users. The edges of the nodes in the graph are defined as the number of users who are interested in both nodes which are the movies. Solving the regression equations by using statistical data analysis methods provides estimates for the weight values of the features. It is observed that some features have stable weight values, while the features director, rating, vote, year and color have unstable or negative weights. It is also noticed that the features type, writer and company are particularly important compared with the others. These features along with their weights are used to obtain the recommendations.

As seen from the resulting weights, it is obvious that the features of a movie have different impact on the user preferences. By using these weight values, the system can make successful recommendations compared to the other CB systems. According to the experimental results measured by *recall* metric, this study outperforms other pure CB systems which consider equal weights for all features and demonstrates the effectiveness of feature weighting.

Secondly, *Movies2Go* [38] is a web-based movie RS that catches and reasons with user preferences to recommend movies. Voting based ranking procedure is combined with guaranteed properties that use syntactic features like actor/actress of movies together with a learning based approach that processes semantic features of movies.

Unlike other RSs that use social filtering, *Movies2Go* is based on individual user preferences. These preferences are obtained either as user input like initial preferences for movie dimensions, or from interactions with the user such as posing queries to learn whether the user liked the movie or not. *Movies2Go* also gives its users the chance to pose unconstrained, constrained, or instance-based queries about the movies.

The system presents two approaches to obtain user preferences. One is explicit querying and the other is passive gathering. In explicit approach, the user is interactively engaged to collect information about preferences, whereas passive gathering watches the users' browsing habits instead of asking the user for information. For instance, frequently visited or book-marked web sites are regarded as liked, but the sites that the user visits briefly and does not return to are regarded as disliked.

Using voting theory makes *Movies2Go* powerful as it holds the promise of recommending items that have the properties required to satisfy user preferences. Another key feature of the system is the incorporation of a Bayesian learning mechanism to learn frequently occurring keywords in the synopsis of recommended movies. This improves the quality of recommendation and reduces errors that might be introduced by users. The voting scheme is also robust to minor variation in

specified preferences. This is highly desirable as values from the user are likely to be only approximately correct. Besides, as stated as a future work, an explanation facility by which the user can be given more details about why a certain movie was recommended may be useful. This gives the user the chance to correct or update stored preferences.

3.1.3 Hybrid Systems

After having discussed the systems using pure CF and pure CB filtering separately, it will be useful to provide examples of hybrid systems which make use of both CB and CF methods.

[39] is a hybrid movie RS that provides improved recommendations by boosting CF approach with CB approach. The approach uses a CB predictor to enhance existing user data, and then provides personalized suggestions through CF.

For constructing the user-movie ratings matrix, the system uses *EachMovie* data set which contains rating data provided by users for various movies. The user ratings change in a scale from zero to five where zero indicates dislike for a movie and five indicates high praise. The system has a web crawler that crawls the movie URLs provided in the dataset, in order to download movie content from *IMDb* and store in the database. The provided user-ratings matrix is a matrix of users versus items, where each cell is the rating given by a user to an item. Initially, the user-ratings matrix is very sparse, since most items have not been rated by most users. The CB predictor is trained on this matrix and a pseudo user-ratings matrix is created. This pseudo matrix contains the user's actual ratings and CB predictions for the unrated items.

The CB prediction task is treated as a text-categorization problem. Movie content information is viewed as text document, and a user rating from zero to five is viewed as one of six class labels. The system implements a bag-of-words naive Bayesian text classifier (Mitchell 1997) extended to handle a vector of bags of words; where each bag-of-words corresponds to a movie feature such as title, cast, etc. The CF component is implemented by using a neighborhood based algorithm. A subset of users is chosen based on their similarity to the active user, and a weighted

combination of their ratings is used to produce predictions for the active user. Similarity between users is measured as the PC between their ratings vectors. First, all users are weighted with respect to similarity with the active user. Then *n* users who have the highest similarity to the active user are selected among them. Lastly, a prediction from a weighted combination of the selected users' ratings is computed.

In this approach the disadvantages of pure CF and CB methods are overcome. By using pure CF, a prediction cannot be made for an item which is not rated by other users previously. However, the system can make such a prediction using a CB predictor. Since a pseudo ratings matrix is used, the root of the sparsity problem is also eliminated. Rows of the pseudo user-ratings matrix contain ratings for all items; and hence all users will be considered as potential neighbors. This increases the chances of finding similar users. A drawback of the current implementation of the CB predictor is that it uses a naive Bayesian text classifier to learn a six way classification task. This approach is not ideal, since it disregards the fact that classes represent ratings on a linear scale. Using a learning algorithm that can directly produce numerical predictions can be useful to overcome this drawback.

Secondly, [3] is another hybrid movie RS that presents an inductive learning approach to recommendation using both ratings information and other forms of information about each movie in predicting user preferences. The system formalizes recommendation as a problem of learning a function that takes a user and a movie as input and produces a label indicating whether the movie would be liked or disliked as output. It does not predict an exact rating for an item and therefore the output is an unordered list of movies which are predicted to be liked by the user.

The system defines an attribute called *"users who liked movie X"* to group users like these into a single feature. It also defines a set of movies which a particular user is interested in like *"the movies which user X is interested in"*. A user is said to be interested in a movie if it is rated in the top quartile of all movies rated by that user. Working on such rules as stated above, the system uses Ripper, which is an inductive learning system able to learn from a set of rules (Cohen 1995; 1996).

All of the content features of the system are extracted from *IMDb*. In particular, these features are; actors, actresses, directors, writers, producers, production designers, production companies, editors, cinematographers, composers, costume designers, genres, genre keywords, user-submitted keywords, words in title, aka (also-known-as) titles, taglines, MPAA rating, MPAA reason for rating, language, country, locations, color, sound mix, running times, and special effects companies of movies. After all, the hybrid features combine knowledge about users who liked a set of movies with knowledge of a particular content feature associated with the movies in a set.

Using only the ratings information causes a limitation in case a new movie comes out. Since there will be a period of time when an RS will have little ratings data for this movie, the RS will initially not be able to recommend this movie reliably. By using the content information, this system will be able to make predictions for this movie even without any ratings. The system has a drawback that the provided recommendation list is unordered. It would be better to predict ratings for the movies instead of categorizing them as liked or disliked. Ordering the recommended movies according to the predicted ratings will make it easier for the users to choose better movies among all.

3.2 Recommendation Systems Using Demographic Data

Demographic RSs are similar to CB systems except that similarities are calculated using demographic data, rather than ratings on items. Some demographic systems also apply other information filtering methods such as CF or CB filtering, but they mainly focus on using the demographic data. Different systems can make use of different demographic features but the most popular features are age, gender, occupation and zip code.

To begin with, [40] is an important recommender to be considered, which is based on both user information and item information. This study proposes a hybrid CF approach that employs the user attribute information and the item attribute information. It computes two types of similarities, namely; user based similarity and item based similarity. User based similarity is composed of user rating similarity and

the user attribute similarity. Similarly, item based similarity is composed of item rating similarity and the item attribute similarity. The item and user based similarities are then used to produce recommendations. In order to obtain demographic data about users, all users are required to provide demographic information including sex, age, profession, department, specialty, etc. during their registration to the system. The demographic information of each user can be used to classify users that like similar categories or subjects of items.

As the data set, the system uses *MovieLens* data set which provides age, gender, occupation and zip code of 943 users along with their ratings given to some of the movies existing in the domain. According to the similarity of these four demographic features of two users, the system computes an attribute similarity value between these users. The system also obtains attribute information of item from *MovieLens* data set that provides 1682 movies with features such as title, genre, release date, *IMDb* URL etc. With the values corresponding to the attributes of two items, attribute similarity of two items is calculated by using PC measurement. Rating similarity of two users and rating similarity of two items are also calculated by using PC measurement. These similarities are then combined with relative weight values and recommendations are produced according to the ratings given by the most similar users which are named as nearest neighbors. The success of this system's recommendation technique is evaluated by calculating the mean absolute error (MAE) and it is stated that as the number of nearest neighbors increases, the MAE of the system decreases.

To discuss, according to the given MAE results which change in a range between 0.78 and 0.85, the system seems very successful while making recommendations. However, these results may change depending on the relative importance weights given to user based similarity and item based similarity. Therefore, if some more results obtained by changing the weight values were provided, it would be better for us to compare the reliability of using demographic features over other methods. Another point to be considered is how the demographic features are used in the system. Unfortunately, this point is not clearly explained. Details of the techniques

those are used to categorize people according to their age, occupation and zip code are missing.

The study given in [22] is a web based RS for matchmaking, where people are able to navigate through freely in order to find the right matches for themselves in terms of both character and appearance. As can be seen, the domain of this system is the people themselves and it produces recommendations based on the demographic similarities of its users. The users are required to provide some personal information about their characteristics, appearance and hobbies during registration. This information consists of some distinguishing features such as eye color, height, gender, educational information, favorite sports, etc. This information provides some knowledge about what kind of a person this is, whereas the system needs to know the taste of this user in order to make recommendations. In order to learn the taste of a user, the main idea of the system is using previous actions of that user. However, when a new user registers to the system, there will not be any previous actions of that user. At this point, CF method is used in order to produce recommendations for the user.

With the assumption that the users with similar characteristics and hobbies may like similar persons, *Matchbook* finds a list of users who has similar properties with the target user. While computing similarity between two users, they consider each feature separately and add one to the difference between users each time a different valued attribute occurs. For instance, the distance between two users having all the same properties is zero, whereas the distance between two users having the same properties except the hair color is one. The system then calculates the distance between the target user and all other users in the system and provides a list of most similar users. The most similar users are displayed to the target user one by one with options “Add to Favorite List” or “Skip to the Other User” before having the chance to look at the remaining ones. “Add to Favorite List” provides positive evidence and “Skip to the Other User” provides negative evidence about the user’s taste. The user will also be able to look at the profiles of other users which are not recommended yet and this is treated as a positive action of the user. This process will give the system some data about the user’s taste and the latter recommendations will be

based on this data which grows each time the user selects one of these options or look at another user's profile. Since the data used for recommendation is obtained from the actions of the user itself, it can be said that the system uses CB filtering method in its recommendations except for the new users.

It can be considered as a shortcoming of the system that when the user first registers to the system, it does not have any idea about that user's taste. For this purpose, the system uses CF method; however, the assumption that a person likes other persons having similar demographic features does not sound meaningful. On the contrary, different characteristics and appearance may be more attractive for most of the people. Another shortcoming of the system is that the users do not always look at the profiles of others since they liked them but the system treats this as a positive feedback. This can provide some noisy data and there can be inconsistency in the knowledge base. Besides these shortcomings, the system has the power of having both positive and negative evidences of the user's taste. It recommends users who are similar to the ones added to the favorite list and who are less similar to the ones skipped in the past.

As a study out of movie domain, it is important to analyze [41]. This study presents a framework, whose aim is learning a profile of user interests for recommending information sources such as Web pages or news articles. The information that is used to make recommendations consists of the content of the page, the ratings of the user on other pages and the contents of these pages, the ratings given to that page by other users and the ratings of these other users on other pages and demographic information about users.

The framework explores techniques for combining recommendations from multiple approaches such as CF, CB filtering and demographic information filtering. For the CF part, the rating similarity between two users is calculated using PC measurement. The users having similar pattern of ratings are considered as similar. Content-based part works by analyzing the description of the items that have been rated by the user and the description of items to be recommended. The system represents web pages as text documents and uses the Winnow text categorization algorithm (Littlestone

and Warmuth 1994; Blum, Hellerstein and Littlestone 1995) whose success is proven by the prior experimental researches.

The demographic information used in this study is obtained from the home page of the user and is used to identify the types of users that like a certain object. For instance, the age, gender, education, etc. of people that rated a restaurant together with their rating of the restaurant helps to define the type of person that likes a certain restaurant. For categorizing the users, the system also uses text categorization. The Winnow algorithm is used to learn the characteristics of home pages associated with users that like a particular restaurant. The positive examples are the HTML home pages of users that like a particular restaurant and the negative examples are the HTML home pages of users that do not like that restaurant.

The system puts many approaches within a common framework. Thus, it makes use of all available information exploring the advantages and disadvantages of alternative approaches to making recommendations. As the strengths of these different approaches are complementary, the recommendations provided by the system will be more precise compared to the systems making use of a single approach. As for the demographic part, representing users as text documents extracted from their home pages is a successful technique. However, it would be better to assign different importance weights to the features of the user. Home pages may contain some irrelevant information which will cause the system to make wrong recommendations.

LifeStyle Finder [23] is an intelligent user profiling system that uses large-scale demographic data. The system presents a new method for user profiling that combines the benefits of the two previous approaches such as processing documents in CB manner and CF. Developers of the system have developed an original approach to user profiling namely demographic generalization. The method uses a commercially available database of demographic data that includes the interests of people all over the U.S. The database of the system consists of information extracted from surveys of more than 40,000 people, the U.S. census data, magazine subscriptions, catalog purchases, etc. The input data are used to classify users in

terms of demographic features, and these classifications are used as general characterizations of the users and their interests.

Lifestyle Finder attempts to identify one of 62 pre-existing clusters to which a user belongs. The demographic database contains information on more than 600 variables, each of which refers to a specific lifestyle characteristic, purchase, or activity. The variables include such items as owning a dog, purchasing Canadian whisky on a monthly basis, playing or watching golf, and owning a motorcycle. Each variable in a demographic cluster has an associated mean and standard deviation, indicating the likelihood of people in the cluster to have this characteristic. Given a set of input data about an individual, *Lifestyle Finder* computes the set of demographic clusters to which the user is most likely to belong. If only one cluster matches the user data, all the data available for the cluster are used as an overall profile of the user. If more than one cluster matches the user data, the demographic variables whose values are similar in all the matching clusters form a partial profile of the user. Except for the initially available user profile, the users can be asked for new information that will be helpful in further profiling. The incoming data items are used for iterative improvement of the database.

As can be understood from its title, *Lifestyle Finder* is an assertive system that has a very large scope. It is stated that by correlating the new areas with values in the input data, the users can be profiled in areas which are not directly addressed by the input data. Even if the method cannot infer all aspects of the user's profile, it infers the user's interest in areas that are very different from those covered by the input data. This approach may work fine in some related areas of life, but may sometimes result in making irrelevant recommendations. It is also declared that ninety-three percent of users surveyed agreed that *Lifestyle Finder*'s questions did not invade their privacy. This proves that the system does not prompt for any personal or private information about the users and it is a significant benefit that attention is paid to online privacy.

3.3 Recommendation Systems in Different Domains

In previous sections, movie RSs and some RSs from different domains which make use of demographic user information are presented. In this section, examples of RSs from book domain, music domain, web domain and news filtering domain will be provided.

Firstly, *LIBRA* [42] is a CB book recommender that uses learning for categorizing text. It uses information about titles extracted from Amazon and organizes the extracted data into author, title, editorial reviews, customer comments, subject terms, related authors, and related titles fields. *LIBRA* uses this extracted information in order to form "bag of words" for a set of slots including author, title, reviews and comments, subjects, related titles, and related authors. In *LIBRA*, users give rating to selected titles on a one to ten scale. A naïve Bayesian text-categorization algorithm is used to learn a profile from these rated examples. *LIBRA* interprets ratings between six and ten as positive and ratings between one and five as negative. After the user profile is learned, all other not yet recommended books are ranked based on the probability that they are rated between six and ten, meaning positive. The users are also able to provide explicit positive or negative keywords, which are used as priors to bias the role of these features in categorization.

LIBRA is an initial CB book recommender which recommends titles based on training examples supplied by an individual user. According to the conducted experiments, this approach can efficiently provide accurate recommendations without any information about other users. However, the new user problem will arise because it uses pure CB filtering method and the system will not have any training examples for the new user. At this point, CF is needed. Instead of making use of collaborative filtering approach, *LIBRA* requests explicit keywords from the new user in order to have an initial knowledge about him. Unfortunately, the users may not be pleased to be asked for some explicit information, as implicit information extraction techniques are always more preferable for the users.

Besides the book domain, there are many RSs working on the music domain. Among these, *Last.fm* [43] is a very popular music service which has over 30

million active users based in more than 200 countries [44]. The system's main aim is to learn what kind of music people love and make recommendations accordingly. *Last.fm* uses the recommender named "Audioscrobbler" and constructs a detailed profile of each user's musical taste by recording details of the songs the user listens to. *Last.fm* users can build up a musical profile using several methods such as listening to their personal music collection on a music player application on a computer or an iPod with an Audioscrobbler plug-in, or by listening to the *Last.fm* Internet radio service. All songs played are added to a log from which musical recommendations are calculated.

Last.fm generates a profile page for every user with a 'Taste-o-Meter' that gives a rating of how compatible your music taste is. Profile pages also include weekly musical neighbors, groups, events and an optional customizable playlist with tracks that the user wishes to share. Recommendations are calculated using a CF algorithm in which similarity between users is calculated based on their playlists. Users can browse and hear previews of a list of artists not listed on their own profile but which appear on those of others with similar musical tastes. *Last.fm* allows the formation of user groups between users with something in common such as being fans of an artist. *Last.fm* generates a group profile similar to the user profiles and shows the overall tastes of the group. It also permits users to manually recommend specific artists, songs or albums to the group members or other users on their friends list

Last.fm has data on millions of users listening to thousands of bands. Given its system of 'scrobbling', it has more user data than other systems because it is able to use data generated from sources other than its player. With over 15 million active users acquired without the use of marketing and only word of mouth, *Last.fm* is a powerful CF system. While listening to the recommended tracks through the web interface, the user can express his love or dislike for a track. This helps the system to refine its database for further recommendations.

The system in [45] is a TV recommender suggesting a user contents semantically related to those watched in the past. *AVATAR* analyses the information by combining CB methods with CF methods in order to meet their different advantages. CB methods compute personalized recommendations by comparing previously liked

items with descriptions of items still unknown to the user, whereas CF methods aim to recommend contents that have interested the other users who have similar preferences. As for the CB recommendation, the programs which are closely related to the programs the user liked in the past are ranked and then displayed to the user. Two factors are considered for this process. The first one is the index relevance of each program in the user profile. If a program is very appealing for the user, its relevance index takes high values. This way, the inferred programs which are related to the user preferences with highest relevance, will be ranked in top positions of the suggestion.

AVATAR uses an explicit technique in order to gather information during registration. This process provides an initial profile for each user and helps the system to make recommendations at the first step. After that, the feedback information such as watched programs or changes in preferences is collected in order to improve the recommendations. Using both CB method and CF method, *AVATAR* makes successful recommendations in the TV domain.

Lastly, it is important to mention *Amazon.com* [46] which is a wide-scope RS working on the e-commerce domain. *Amazon.com* uses CF approach and gathers information by getting explicit feedback from the users. Besides, the user's navigation history is also taken into account and this information is kept in an internal database.

Amazon.com has six main features described briefly below:

Customers who Bought feature is found on the information page of each book including two distinct recommendation lists. First list recommends books that are frequently purchased by customers who had purchased the selected book. Second one recommends authors whose books are frequently purchased by customers who had purchased the books of the selected book's author.

Eyes feature allows customers to get notifications of the newly added books. Customers can also pose queries based on several features of the books including author, title, and ISBN or publication date.

Your Recommendations feature presents a recommendation list to the users based on their rating history.

Bookstore Gift Ideas feature works online and allows customers to get recommendations from editors based on the specific categories.

Amazon.com Delivers feature is the offline version of "Bookstore Gift Ideas" where users get recommendations from editors via e-mail.

Customer Comments feature enables customers to get text recommendations from the ideas of other users.

Amazon.com aims to increase business actions by personalizing the web site according to the users' interests and needs. With the help of millions of users it has and millions of items those exist in its catalogue, *Amazon.com* create its own item-to-item CF approach in order to make powerful recommendations. The details of item-to-item CF, which form the basis of *Amazon.com*'s recommendation algorithm, is described in [47]. Firstly, a table of similar-items is built through iterative computing procedure. Then each of the user's purchased and rated items are matched to similar items and a recommendation list is created from these similar items. Building the similar-items table offline and making the recommendations online provides the system quick answering facility.

CHAPTER 4

ReMovender AS A CONTENT BASED MOVIE RECOMMENDATION SYSTEM EMPOWERED BY COLLABORATIVE MISSING DATA PREDICTION

4.1 General Overview of *ReMovender*

ReMovender is a web based recommendation system developed for the movie domain. The users are required to register in order to navigate through the system and give rating to movies.

Figure 3 shows the main page of *ReMovender* where the users can login or create a new account:

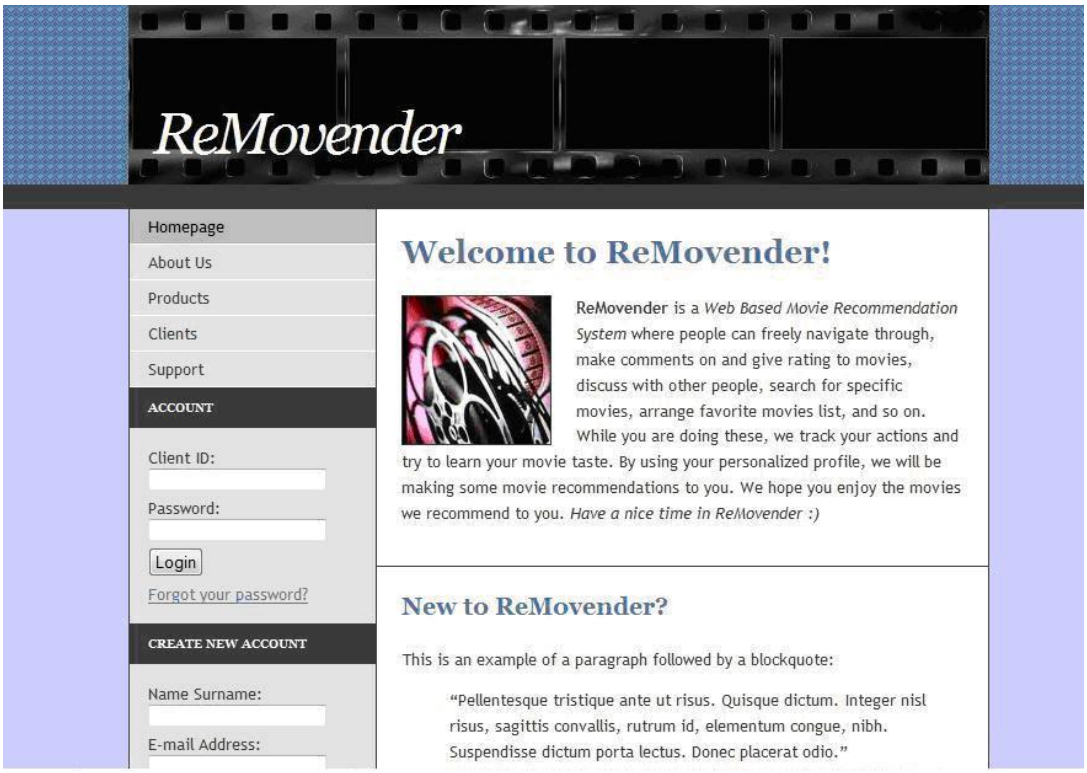

Figure 3 – Main Page of *ReMovender*

Users of *ReMovender* are also capable of making search for specific movies providing some search criteria, making discussions on the movies, displaying similar users, or displaying similar movies to a specific movie. The system also provides an interface that displays the features of a movie such as the movie's *IMDb* rating, genres, languages, runtime, director, etc. As an example, the detailed page of the movie "The Fugitive" is provided in Figure 4:

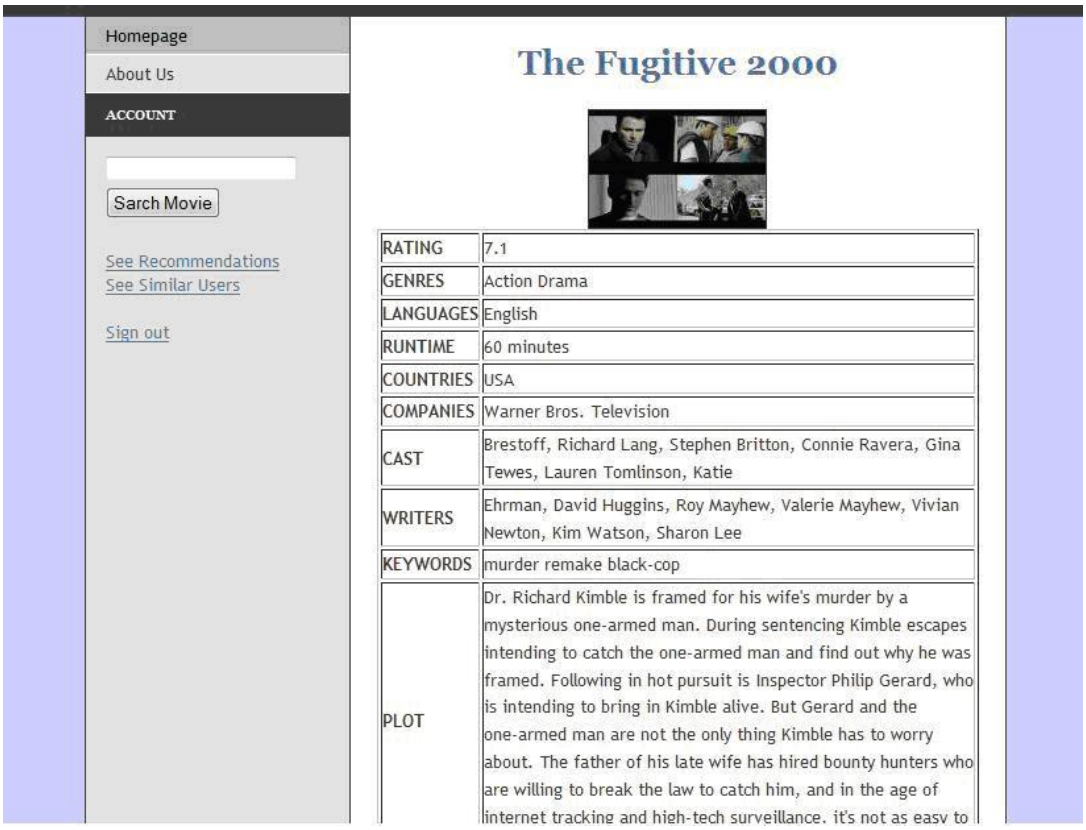

Figure 4 – Detailed Page of a Specific Movie

The users can rate movies by using the detailed movie pages. With the help of the ratings given to the movies, a user profile is created for each user. The user profile gives an idea about what kind of movies a user likes and dislikes. Each time a user gives rating to a movie, the user's profile is updated according to the yet rated movie's features.

The recommendations are done according to a user's profile. Movies which have the features existing in the user's profile are recommended to the user. These recommendations are calculated offline by combining several methods and several algorithms. When the user is online, the recommendations are listed with decreasing similarity order as in Figure 5:

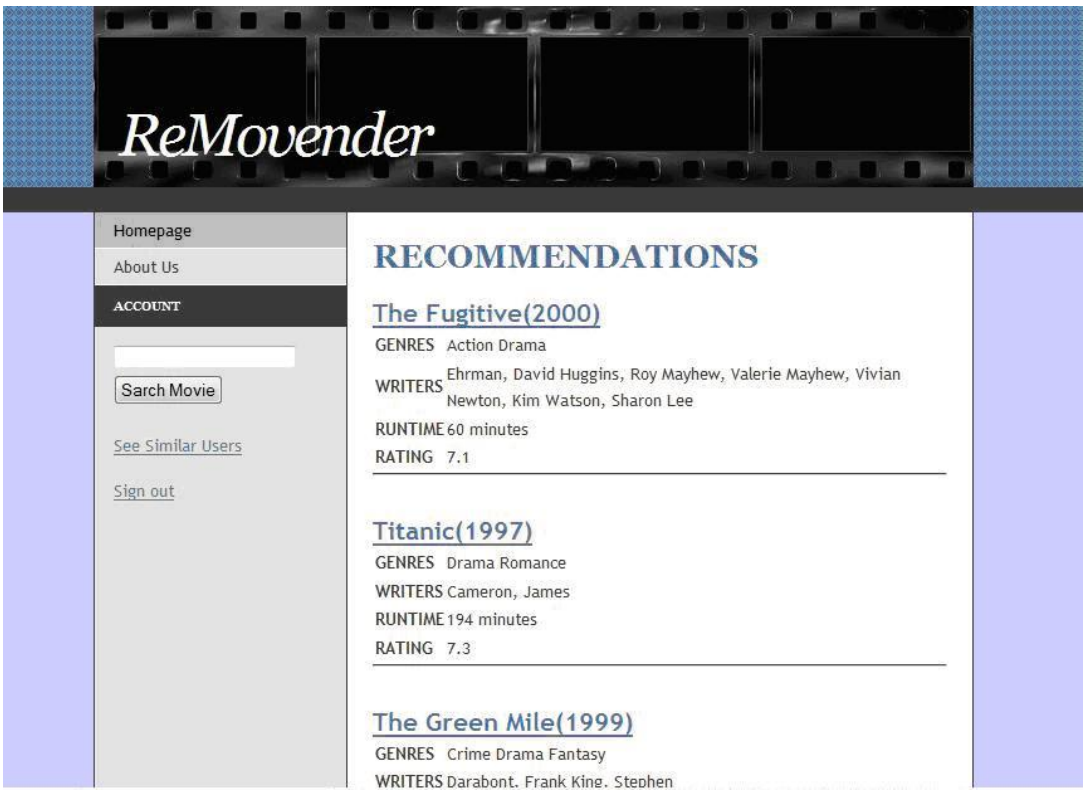

Figure 5 – Recommendations Page

ReMovender also has an administrator interface. The administrator has the ability to update the local database from time to time in order to get the new movies added to the *IMDb* database.

Figure 6 is the admin interface of *ReMovender*:

Figure 6 – Administrator Page

4.2 System Architecture

ReMovender is a combination of many components which are specialized for different phases of the recommendation process. The general architecture of *ReMovender* can be described as in Figure 7:

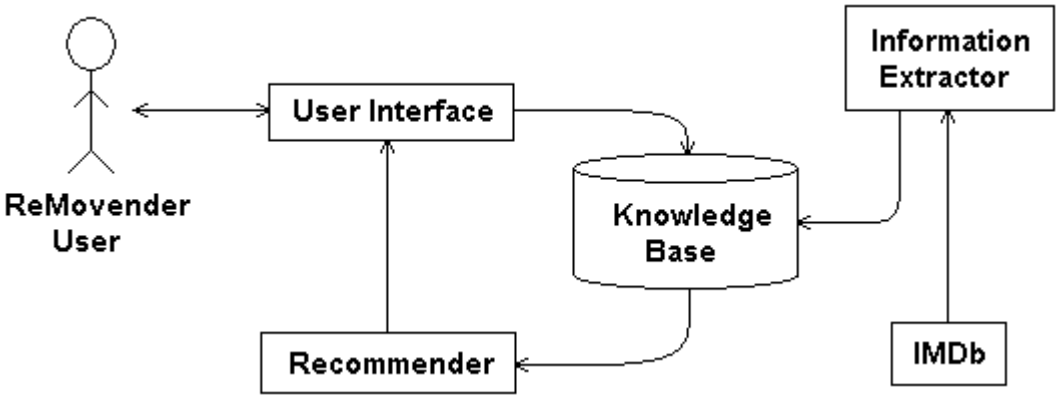

Figure 7 – General Architecture of *ReMovender*

4.2.1 User Interface

ReMovender user is directed to the user interface and relation between the user and the user interface is bidirectional. The user both provides input to the system and gets the output of the system through the user interface. Users can register or login to the system by using this interface. Besides, they can execute queries, get the results of these queries, display specific movies or display the recommendations provided by the system.

The appearance of the graphical user interface is as presented by the Figures 3-6.

4.2.2 Information Extractor

Information extractor component extracts information about the movies from the Internet Movie Database (*IMDb*). It processes all movies existing in *IMDb* and inserts this information into the local database of the system by using a Python

package called *IMDbPY*. *IMDbPY* is platform-independent and open source software that is written in Python. With the help of this package a local copy of the whole *IMDb* database can be retrieved.

In *IMDbPY*, there is a script named imdbpy2sql.py which populates the database using the plain text data files of *IMDb* [48]. In this process, *SQLObject* package which is a popular Object Relational Manager for providing an object interface to the local database is also used. As being the fastest database for the extraction process, *MySQL* is used in *ReMovender*. As a result of this extraction process, a database requiring between 2.5 and 5 GB of disc space is created.

For the purpose of having an up-to-date RS, the imdbpy2sql.py script is converted into an executable file and embedded into the graphical user interface. By this way, the system administrator has the ability to update the database from time to time through the user interface just after downloading the required plain text data files from *IMDb*. With the help of this feature, the information about the newly released movies and other new information about actors, actresses, companies, writers can be extracted.

Besides the information about the movies, their thumbnails should also be displayed to the users. However, as the images are copyrighted and not provided by the *IMDb*, they are extracted from the web via Google AJAX Search API [49] which puts Google Search in web pages with JavaScript. With the help of this API, the information extractor dynamically searches the thumbnail of a specific movie in Google images by using its name and production year as keywords. Then the first matching result is extracted and displayed on the target movie's page.

4.2.3 Recommender

Knowledge base also includes information gathered from the user actions such as giving ratings, executing queries, etc. Along with the features of the movies and the user profiles, ratings information is used for producing recommendations. Recommendations are produced by the recommender and passed to the user interface in order to be presented to the user. Details of the recommender and the recommendation process are discussed in further sections.

4.3 Design Issues

4.3.1 Local Database

The knowledge base of *ReMovender* is composed of two different databases. First one is the movies database which contains information about all movies existing in the *IMDb* whereas the second one is the ratings database. Ratings database consists of information about users and the ratings given by the users' to the movies existing in the system. These two databases are described below along with the summary of tables they include.

4.3.1.1 Movies Database

As stated above, the "movies" database includes all movies contained in *IMDb*. There are 22 tables in this database keeping the key features of the movies such as genre, director, writer, etc.

Table 3 provides a summary of the most important tables in this database along with the attribute fields they hold:

Table 3 – Table Descriptions of Movies Database

Table Name	Attributes	Summary
title	imdb_id, title, kind_id, production_year	This table stores the title, kind id and production year of a movie. Each row of this table is specific to a movie and the movies are distinguished by the field imdb_id. The kind_id field of this table refers to the id field of "kind_type" table. All of the ID fields specifying movies in other tables refer to the imdb_id field of this table.

Table 3 Continued

cast_info	id, person_id, movie_id, role_id	This table stores the cast info of a movie. person_id field refers ro the id field of "name" table and movie_id field refers to the imdb_id field of "title" table. role_id field refers to the id field of "role_type" table.
company_name	id, name	This table contains all of the available company names existing in the movie descriptions along with a unique ID.
company_type	id, kind	This table contains 4 types of companies, namely; distributors, production companies, special effects companies and miscellaneous companies with unique IDs.
info_type	id, info	This table contains all of the available info types existing in the movie descriptions assigning a unique ID to each. Genre, rating, runtimes, keyword are some examples of possible info types.
keyword	id, keyword	This table contains all of the available keywords existing in the movie descriptions and assigns a unique ID to each keyword.

Table 3 Continued

kind_type	id, kind	This table contains all of the available kind types existing in the movie descriptions. Movie, tv series, tv movie, video movie, tv miniseries, video game and episode are the possible kind types.
movie_companies	id, movie_id, company_id, company_type_id	This table stores the company information of movies. movie_id field corresponds to the imdb_id field in "title" table, company_id fiels corresponds to the id field in "company_name" table and company_type_id field corresponds to the id field of "company_type" table.
movie_info_idx	id, movie_id, info_type_id, info	This table stores the info of movies such as the ones described in "info_type" table. movie_id field refers to the imdb_id field of "title" table and info_type_id field refers to the id field of "info_type" table. The info field holds the value of the info such as "comedy", "8.6", "190 min", etc.
movie_keyword	id, movie_id, keyword_id	This table stores the keyword information of movies. movie_id corresponds to the imdb_id field of "title" table and keyword_id field corresponds to the id field of "keyword" table.

Table 3 Continued

name	id, name	This table contains all available cast names with a unique id. Directors, actors, actresses, composers, writers are all included in this table.
role_type	id, role	This table contains all of the possible role types in a movie with a unique ID. Actor, actress, writer and director are some examples of possible role types.

4.3.1.2 Ratings Database

The "ratings" database consists of two tables, namely; users and ratings. Initially, the users table includes users from *MovieLens* dataset and some sample ratings of these users are kept in ratings table. Details of these tables are provided in Table 4:

Table 4 – Table Descriptions of Ratings Database

Table Name	Attributes	Summary
users	user_id, user_name, password, name, email, gender, date_of_birth, occupation, zip	This table stores all necessary information about users.
ratings	user_id, movie_imdb_id, rating	This table stores the ratings given to the movies by the users. user_id field refers to the user_id field of "users" table and movie_imdb_id refers to the imdb_id field of the table "title" in "movies" database.

4.3.2 User Modeling and the User-Item Matrix

ReMovender models its users according to their demographic features and the ratings they have given to the movies existing in the system.

Demographic features are used in finding demographic similarities between users. The users are represented by four important features that are considered while finding demographic similarity. These features are age, gender, occupation and zip code. Formally, a user is represented as u = <age, gender, occupation, zip>. Details about this process are presented in section 4.4.3.

Ratings information is used in CF part. The input to a CF system is a matrix of users' ratings on a set of items, where each row represents ratings of a single user and each column ratings for a single item [2]. In a movie RS with M users and N items, there would be a $M \times N$ user-item matrix denoted by R. Each entry in this

matrix can be denoted by $r_{m,n}$ where *m* refers to the user index and *n* refers to the movie index. The value of each entry equals to the rating given by the m^{th} user to the n^{th} movie. Possible rating values in *ReMovender* are in a range of [1, 5] and each entry can be formally represented as $r_{m,n} = x$ where $x \in \{1,2,3,4,5\}$ [36]. An example user-item matrix R is presented in Table 5:

Table 5 – User-item Matrix

	i_1	i_2	i_3	…	i_N
u_1	5	0	3	0	4
u_2	0	1	0	2	2
u_3	4	0	5	3	0
…	0	3	0	4	1
u_M	2	0	1	0	5

The user-item matrix is composed of the user vectors and can be formally represented as $R = [u_1, u_2, \ldots, u_M]^T$. The users are therefore modeled with vectors containing the ratings given by them to all of the movies. If the user does not give rating to a specific item, that entry of the matrix is assigned a zero as in Table 5. Formally, a user is modeled in *ReMovender* as follows:

$u_M = [r_{m,1}, r_{m,2}, \ldots, r_{m,N}]^T$ where $m = 1, 2, \ldots, M$.

4.3.3 Movie Features

As can be seen in Figure 4, *ReMovender* represents a movie by a set of distinguishing features such as production year, rating, kind, genres, languages,

runtime, countries, companies, cast, writers, keywords, and plots. Some of these features are single valued, whereas some of them can have multiple values. Table 6 describes the features along with their types, domains and the number of values they can get:

Table 6 – Movie Features

Feature	Type	Domain	Value
Production year	Integer	2001, 1998, etc.	Single
Rating	Integer	7.1, 8.6, etc.	Single
Kind	String	Movie, TV Series, etc.	Single
Genre	String	Comedy, Drama, etc.	Multiple
Language	String	English, Spanish, etc.	Multiple
Runtime	Integer	95, 110 etc.	Single
Country	String	Germany, France, etc.	Single
Company	String	Avalon Films, BBC, etc.	Multiple
Cast	String	Kate Winslet, Brad Pitt, etc.	Multiple
Writer	String	Woody Allen, etc.	Multiple
Keyword	String	beach, birthday, etc.	Multiple
Plot	String	A young couple living in a Connecticut suburb, …	Multiple

Despite all of these features are presented in the user interface, some of them are not used in the recommendation process. Only kind, genre, language, country, company and writer are used while producing the recommendations.

4.4 Missing Data Prediction

Initially, *MovieLens* dataset [52] provides 100.000 ratings information in the form <user, item, rating> about 1683 movies and 943 users. The initial user-item matrix of *ReMovender* therefore has the cardinality 943×1683 with M=943 and N=1683.

Table 7 – User-Item Matrix of ReMovender

	i_1	i_2	i_3	…	i_{1683}
u_1	0	0	0		2
u_2	0	3	0		0
u_3	0	0	5		0
…					
u_{943}	1	0	0		0

As only 100.000 cells of this matrix have real values while the others have a "0" inside, the user-item matrix of *ReMovender* can be defined as a sparse matrix. The initial density of the matrix can be calculated as follows:

$$100000 \div (943 \times 1683) = 0{,}063$$

The density 0,063 will not be sufficient for making accurate predictions. Therefore, *ReMovender* uses the method "Missing Data Prediction" [35] for increasing the density of the user-item matrix in order to make more powerful recommendations. Briefly, this method predicts a rating value for a user-item pair by using the rating history of the target user and the rating histories of similar users.

Prediction of the missing data requires a method for calculating similarity between two users. At this point, *ReMovender* uses its own method which is composed of calculating LU & GU, demographic similarity and then combining these similarities in order to result in a unique similarity value between user-user pairs. Following three sections describe the calculation of each type of similarities in detail and section 4.4.4 explains how these similarities are combined to predict missing data.

4.4.1 Local User Similarity

LS between two users is computed by applying The PCC algorithm which is widely used in many CF algorithms. In statistics, the PCC is a measure of the linear dependence between two variables giving a value between +1 and −1 inclusive [50]. Applying the algorithm to *ReMovender*, the following formula is obtained with the variables being users u_1 and u_2:

$$Sim_L(u_1,u_2)=\frac{\sum_{i\in I(u1)\cap I(u2)}\left(r_{u1,i}-\bar{r}_{u1}\right)\times\left(r_{u2,i}-\bar{r}_{u2}\right)}{\sqrt{\sum_{i\in I(u1)\cap I(u2)}\left(r_{u1,i}-\bar{r}_{u1}\right)^2}\times\sqrt{\sum_{i\in I(u1)\cap I(u2)}\left(r_{u2,i}-\bar{r}_{u2}\right)^2}} \quad (2)$$

In (2), $Sim_L(u_1,u_2)$ is the LS between users u_1 and u_2, $r_{u1,i}$ is the rating given by u_1 to item i, $\bar{r}_{u1}$ is the average of the ratings given by u_1 and $I(u_1)\cap I(u_2)$ is the set of items which are rated by both u_1 and u_2.

There are many reasons for applying the PCC algorithm in computing LS of users. In the PCC algorithm, if two users give an item the same rating, it means that these two users are similar to each other. However, the ratings of a specific item are usually centralized around an average value and if the ratings are close to this

average value, the rating only represents that these two users act like most other people. Therefore, it cannot be concluded that these two users are similar or dissimilar based on this rating. On the other hand, if the rating is totally different from the average value, it will provide more discriminative information to determine whether these users' preferences are similar. Intuitively, a rarely given rating for an item will be extremely useful to help us distinguish the user which gives the unexpected rating from other users. For instance, the movie "*The Shawshank Redemption*" is highly favored by lots of people. The fact that a user likes the movie tells us almost nothing about his/her preference. In contrast, if a user dislikes the movie and gives a very low rating for it, he/she can be easily distinguished from others.

Although PCC algorithm is very powerful in computing LS of users, it can sometimes give unreliable results. For instance, in the case that two users rate only one movie in common, and the difference $r_{u1,i} - \bar{r}_{u1}$ is equal to the difference $r_{u2,i} - \bar{r}_{u2}$, the LS between these users equals to 1. However, as they do not have enough number of commonly rated items, it is not meaningful to consider these users as similar to each other. Therefore, a correlation significance weighting factor γ is needed to devalue similarity weights in case of small number of co-rated items. Instead of Sim_L which is defined above, the devalued $Sim_L^{'}$ defined with the following equation is used as the LS between two users:

$$Sim_L^{'}(u_1, u_2) = \frac{Min(|I_{u1} \cap I_{u2}| \times \gamma)}{\gamma} \times Sim_L(u_1, u_2) \quad (3)$$

According to (3), if the number of co-rated items is smaller than the factor γ, the LS of these users will be devalued. This process prevents the systems from overestimating the similarity of users who may not have similar overall preferences although they have rated a few items identically.

4.4.2 Global User Similarity

The LS helps to find similar neighbors of the target user. However, in some cases, it cannot be possible to find enough number of neighbors as the user does not have co-

rated items with enough number of users. To solve this problem, global user similarity is defined. The main idea behind GS is that neighbors of neighbors are also neighbors.

With the help of GS, more neighbors of a target user can be found even when he/she has few local neighbors. For this purpose, firstly a user graph is constructed by using the users as nodes and LS values as the weight of edges. GS between the users is defined as the maximin distance of these users in the graph:

$$Sim_G(u_1, u_2) = maximin\_distance(u_1, u_2) \quad (4)$$

In *ReMovender*, the Floyd-Warshall algorithm which is an efficient algorithm to find all-pairs shortest paths on a graph [51] is applied to recursively compute all-pairs of maximin distances. For the Floyd-Warshall recurrence, c_{ij}^{k} can be defined as the maximin distance between two nodes *i* and *j* with intermediate vertices belonging to the set {1, 2, ... , k}.

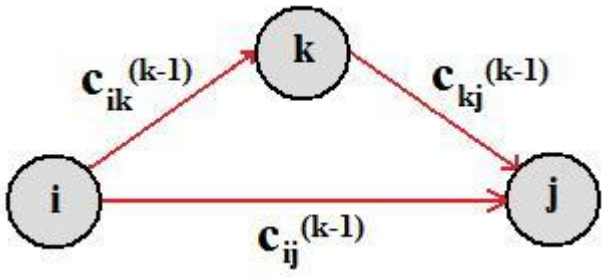

Figure 8 – Floyd Warshall Algorithm

As a result, the recurrence relation can be defined as the following:

$$c_{ij}^{k} = \max^{k}\left\{c_{ij}^{k-1}, \min(c_{ik}^{k-1}, c_{kj}^{k-1})\right\} \quad (5)$$

As the algorithm finishes and the edge values of the graph becomes updated, the new edge values turn out to be global similarities. It is important to note that some of the edge values can increase while some of them remain unchanged.

4.4.3 Combining Local & Global User Similarities

Taking both LU & GU into account, the following CF framework is proposed. To predict an active user's (u_a) rating on a particular item, under LU & GU, firstly his *k* nearest neighbors are found both for the *k* local nearest neighbors ($nn_L^k(u_a)$) and the *k* global nearest neighbors ($nn_G^k(u_a)$). Then both $nn_L^k(u_a)$ and $nn_G^k(u_a)$ are employed to predict the user's rating

$$\hat{r}_{LU\&GU}(a,i) = (1-\alpha) \times \frac{\sum_{u_k \in nn_L^k(u_a)} sim_L'(u_k, u_a) \times r_{k,i}}{\sum_{u_k \in nn_L^k(u_a)} sim_L'(u_k, u_a)} + \alpha \times \frac{\sum_{u_k \in nn_G^k(u_a)} sim_G(u_k, u_a) \times r_{k,i}}{\sum_{u_k \in nn_G^k(u_a)} sim_G(u_k, u_a)} \quad (6)$$

The parameter α determines the extent to which the prediction relies on LU & GU. With α = 0, it indicates that the prediction depends completely on LS and with α = 1, it states that the prediction depends completely on GS. α can be determined experimentally by using cross-validation.

4.4.4 Demographic Similarity

The main idea behind making predictions using demographic data is the assumption that people with similar characteristics enjoy similar movies. It is believed that age, gender, occupation and hometown play an important role on movie preferences and a set of users who have a high level of demographic similarity with the target user is found. Then the rating of the target user is predicted based on the ratings of most similar people.

The general formula of the demographic prediction for a user *a* and a movie *i* is given in (7) where $\bar{a}$ is the average movie rating of the user *a*, a_k is a user from the set of nearest demographic neighbors of user *a* denoted by *D(a)*, $r_{ak,i}$ is the rating given by user a_k to movie *i*, and $\bar{a}_k$ is the average of ratings given by the user a_k:

$$\hat{r}_D(a,i) = \bar{a} + \frac{\sum_{a_k \in D(a)} Sim_D(a, a_k) \times (r_{ak,i} - a_k)}{\sum_{a_k \in D(a)} Sim_D(a, a_k)} \quad (7)$$

The system uses *MovieLens* dataset [52] at first step and treats *MovieLens* users as its own users. It has 943 users whose demographic information is taken from [52]. Following are some details about the features which are used as demographic data:

age: This feature is used with the assumption that people having the same age have similar movie tastes. For instance, old people enjoy watching documentary films whereas children would like to watch cartoons.

gender: Gender plays an important role on movie preferences. Men would like to watch action films while women would prefer watching romance.

occupation: Occupation can also give an opinion about the movie taste of a user. For instance, soldiers enjoy watching war films while engineers like science fictions.

zip code: Zip code may have an impact on the movie preference of a user. Users living in the same region may have similar interests.

The demographic similarity between two users u_1 and u_2 is given by;

$$Sim_D(u_1, u_2) = \frac{\sum_{f \in F} w_f \times Sim(u_{1f}, u_{2f})}{\sum_{f \in F} w_f} \quad (8)$$

where *f* represents a feature of the user from the set of all demographic features *F*, w_f represents the relative weight of feature *f*, u_{1f} and u_{2f} represent the values of *f* for u_1 and u_2, and $Sim(u_{1f}, u_{2f})$ represents the similarity between values of *f* for u_1 and u_2.

4.4.4.1 Determining Feature Weights

In order to measure demographic similarity between users, it is necessary to find generic weights for age, gender, occupation and zip code. The variables of the system are described in Table 8:

Table 8 – Description of the Variables

Feature	Type	Domain	Distance
Age	Integer	$[age_{min}, age_{max}]$	$1 - [(A_1 - A_2) / (age_{max} - age_{min})]$
Gender	String	'M', 'F'	$G_1 = G_2 \; ? \; 1:0$
Occupation	String	doctor, student, etc.	$O_1 = O_2 \; ? \; 1:0$
Zip code	String	61820, 23227, etc.	$Z_1 = Z_2 \; ? \; 1:0$

As understood from Table 8, the difference between feature values is normalized to a value in the interval [0, 1]. The difference between two users' gender, occupation or zip code is one if the values corresponding to them are the same, and zero otherwise. The difference measure of age is different from the others. Users having the same age have an age similarity of one and the users having the minimum and maximum ages in the system have an age similarity of zero. For other users, age similarity is normalized between zero and one according to the difference between their ages. Note that the youngest user of the system is 7 years old, where the oldest user is 73.

Let's assume that the system has following two users:

u_1 = <23, 'F', 'Student', '61820'> u_2 = <36, 'F', 'Teacher', '23227'>

Table 9 – Feature Values of u_1 and u_2 With Corresponding Distances

Feature	Type	Domain	Distance
Age	23	36	1 – (| 23 – 60 | / | 73 – 7 |) = 0.803
Gender	F	F	1
Occupation	Student	Teacher	0
Zip code	61820	23227	0

As seen in Table 9, each user profile is supported by a vector of four values. Euclidean or cosine similarity between these vectors could have been used. However, this would not be a reliable approach since movie preferences may be affected by these features with different extents. So, having defined the feature similarities, it is needed to determine the feature weights of the equation. By using (8) and providing that the sum of feature weights equals one as in (9), the equation can be rewritten as in (10):

$$w_{age} + w_{gender} + w_{occupation} + w_{zip} = 1 \tag{9}$$

$$Sim_D(u_1, u_2) = \left(w_{age} \times sim_{age}\right) + \left(w_{gender} \times sim_{gender}\right) + \left(w_{occupation} \times sim_{occupation}\right) + \left(w_{zip} \times sim_{zip}\right) \tag{10}$$

The feature weights are estimated by using a graph which is constructed from the ratings given by users. This graph represents the rating similarity between users who are the nodes of it. The edges of the graph are the number of common and same rated items by these users. If two users rated n items with the same rating, this will add n to the edge value which is initially 0 for each user pair. In order to have more reliable results, the ratings having a difference of one are also considered but each one is counted as 0.25 instead of 1. Table 10 is an example of a ratings table and the corresponding network graph:

Table 10 – An Example Ratings Table with an Imaginary User and an Imaginary Item

user_id	item_id	rating
1	1	4
1	2	3
1	3	4
2	1	4
2	2	2
3	3	4

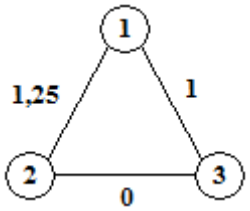

Figure 9 – The network graph that corresponds to Table 10

As the users 2 and 3 did not rate any items in common, the edge weight is 0. Users 1 and 3 rated one movie with the same rating, so their distance is 1. Users 1 and 2 rated one item with the same rating and one item with close ratings, and their distance is 1 + 0.25.

A linear regression framework for determining the optimal feature weights is described. Let two users be denoted as u_1 and u_2, and the edge weight between them as $E(u_1, u_2)$. According to previous definition of the edge weights, $E(u_1, u_2)$ equals to the number of items which are rated with the same rating by u1 and u2.

Recall that demographic similarity between u_1 and u_2 has been defined as $Sim_D(u_1, u_2)$ in (10). Equating $E(u_1, u_2)$ with $Demographic\_sim(u_1, u_2)$ leads to the following set of regression equations:

$$Sim_D(u_1,u_2) = (w_{age} \times sim_{age}) + (w_{gender} \times sim_{gender}) + (w_{occupation} \times sim_{occupation}) + (w_{zip} \times sim_{zip}) = E(u_1,u_2) \quad (11)$$

By using random samples from [52], enough number of regression equations of the above form are obtained. Solving these equations by using multiple linear regression analysis technique with dependent variable as 'common rated item number' and independent variables as age, gender, occupation and zip, the statistical data analysis package SPSS provided estimates for the values of wage, w_{gender}, $w_{occupation}$ and w_{zip}. Normalizing the standardized coefficients of the independent variables, the weights corresponding to the features are obtained. It is observed that zip code has an unstable weight value while the other features have stable weight values. So, zip code is removed from the equations.

Table 11 shows the produced weight values for each feature:

Table 11 – Resulting Feature Weights

Feature	Weight
Age	0,56
Gender	0,41
Occupation	0,03
Zip code	0,00

Since the zip has zero weight, it is not included while calculating the demographic similarity between users. This means that zip code similarity is not directly

proportional to rating similarity, therefore has no effect on the movie preferences. Also note that the movie preference has a high reliability on age and gender compared with occupation. The meaning of this is that the rating similarity between users is highly proportional to age and gender similarities.

4.4.5 Predicting Missing Ratings

As stated before, prediction of the missing ratings is done by using the local user similarity, global user similarity and demographic user similarity. The details of calculating these 3 types of similarities are explained above. This section describes how to combine them for producing a single prediction for an active user.

With the combination of LU and GU, the prediction in (6), which is denoted as $\hat{r}_{LU\&GU}(a,i)$, is obtained. On the other hand, (7) shows the demographic similarity denoted by $\hat{r}_{D}(a,i)$. These 2 predictions are combined with a significance weighting factor β as follows:

$$\hat{r}(a,i) = \beta \times \hat{r}_{LU\&GU}(a,i) + (1-\beta) \times \hat{r}_{D}(a,i) \quad (12)$$

After applying this formula, a resulting prediction for a user-movie pair is made. With the help of this missing data prediction process, sparsity of the user-item matrix is decreased. The numeric details about the density of user-item matrix before and after applying missing data prediction are explained in evaluation part.

4.5 Content Based Recommendation

This part of the thesis explains in detail, how the content information of movies is included in the prediction process. As already explained in former sections, the values in the cells of user-item matrix are recalculated and updated with the help of collaborative methods applied. The sparseness of the matrix is therefore reduced. However, there may still be some empty cells due to the inadequate number of nearest neighbors for that user-item pair. For this reason, content information is used to make user-item matrix full so that a prediction can be made for each user-item pair.

4.5.1 The Movie Features

A movie can be described with a large number of features. Some of these features help so much to distinguish a movie from others, whereas some features do not provide much information about that movie. Having this in mind and in order to be able to assign reasonable weight values to the features according to the results produced by [37], *ReMovender* uses a subset of the movie features that is composed of kind, genre, writer, country, language and company. The detailed descriptions of these features are provided below:

Kind: The items in *ReMovender* are considered as movie throughout this work. However, the items can belong to one of 7 different groups and movie is only one of these groups. Kind of an item is a numeric value between 1 and 7 and it represents the type of the item. An item can be a movie, an episode, a video game, etc. Table 12 shows the 7 types with corresponding kind IDs:

Table 12 – Item Kinds

Kind ID	Kind Name
1	Movie
2	TV Series
3	TV Movie
4	Video Movie
5	TV Mini Series
6	Video Game
7	Episode

Genre: Genre feature gives opinion about what a movie is about. This feature can have a subset of the following 19 predefined values:

Adult, Drama, Short, Biography, Action, Comedy, Family, Musical, Adventure, Animation, Documentary, Thriller, Sci-Fi, Talk-Show, Fantasy, Crime, Music, Sport, Romance, History, Horror, Mystery, War, Game-Show, Western, Reality-TV, News, Film-Noir

Writer: Writer feature can have multiple values as a movie can be written by more than one writer. The writer names are not predefined. A new movie is inserted into the database along with its writer.

Country: Country feature stores information about the country in which the movie is produced. It is usually single-valued, but it can sometimes have more than one value.

Language: Language feature holds information about the language which the movie is written in. Like country feature, language is usually single-valued, but it can sometimes have multiple values.

Company: There are 4 different company types in IMDb database. These are production companies, distributors, special effects companies and miscellaneous companies. In *ReMovender*, only production companies are taken into account. A movie is usually produced by multiple production companies; therefore this feature is a multiple-valued feature.

4.5.2 Item Profile

As *ReMovender* works in the movie domain, its movies are considered as its items. The profile of a movie in *ReMovender* is constructed by using the features which are explained in the previous section. As five features are used to describe movie content, the profile of a movie turns out to be a vector of size 5. The members of this vector correspond to kind, genre, writer, country, language and company in turn. As

genre, writer and company may also be composed of more than one member, the profile of a movie can be regarded as a vector of vectors.

$V_{kind} = < kind >$
$V_{genre} = < genre_1, genre_2, \ldots , genre_n >$
$V_{writer} = < writer_1, writer_2, \ldots , writer_n >$
$V_{country} = < country_1, country_2, \ldots , country_n >$
$V_{language} = < language >$
$V_{company} = < company_1, company_2, \ldots , company_n >$
$V_{movie} = < V_{kind} , V_{genre} , V_{writer} , V_{country}, V_{language} , V_{company} >$

4.5.3 User Profile

The profile of a user is composed of the ratings given to the movies along with the content information of these movies.

4.5.4 Content-based Prediction of the Ratings

In this work, the rating that could be given by a user to a movie is predicted by using two different kinds of predictions. One of these predictions is related with the occurrence frequency of a specific feature value in the set of rated movies, while the other is related with the average of ratings given to that specific feature value. Combining these two predictions by using a relativity factor δ, an overall prediction is obtained as follows:

$$pred\_overall(u,i) = \delta \times pred\_overall_{count}(u,i) + (1-\delta) \times pred\_overall_{rating}(u,i) \quad (13)$$

In order to calculate $pred\_overall_{count}(u,i)$ and $pred\_overall_{rating}(u,i)$ separately, six initial predictions are made for each. Each one of these initial predictions corresponds to one feature of the movie. Then these predictions are combined by using some weight values to alter the effect of the features, as each feature has a different impact on the given ratings.

Note that the weight values given in [37] are provided in Table 13:

Table 13 – Feature Weights

Feature	Weight
Kind	0.19
Genre	0.04
Writer	0.38
Country	0.07
Language	0.10
Company	0.22

Following sections describe the prediction process in detail.

4.5.4.1 Count Based Prediction

Prediction in terms of count takes the occurrence frequency of a specific feature value into account. The idea behind this kind of prediction is that if a value occurs with a high frequency in the rated items list, this means that the user is highly interested in that value. For instance, if all movies rated by a user have genre "romance", this gives us a clue that the user is fond of romantic movies. In the light of this idea, the rating prediction for genre feature is formulated as follows:

$$pred\_count_{romance}(u,m) = \frac{|RM_{romance}| - \min\_count_{genre}}{\max\_count_{genre} - \min\_count_{genre}} \times 5 \quad (14)$$

In the formula, $RM_{romance}$ represents the set of movies rated by user u, which have "romance" in their genre list, and $|RM_{romance}|$ is the cardinality of this set. On the other hand, min_count_{genre} is the occurrence number of the sparsest genre, while max_count_{genre} is the occurrence number of the most common genre in rated items list of user u. Calculation of these values is done through the database with the help of appropriate SQL statements.

Suppose that Table 14 shows the occurrence number of genres which exist in a user's rated movies list and we are trying to predict a rating based on a specific genre such as "drama".

Table 14 – A Sample Genre Occurrence List

Genre	Occurrence Number
Adventure	2
Comedy	16
Crime	4
Drama	7
Horror	8
Mystery	12
Romance	9
War	1

From Table 14, RM_{drama} is obtained as 7, min_count_{genre} is obtained as 1 from "war" and max_count_{genre} is obtained as 16 from "comedy". Therefore;

$$pred\_count_{drama}(u,m) = \frac{7-1}{16-1} \times 5 = 2$$

- **Prediction of Kind**

Every movie has exactly 1 kind, which can be a movie, TV series, or etc. Representing the specific kind that is currently being analyzed with *k*, the following prediction can be made for that kind:

$$pred\_count_{k}(u,m) = \frac{|RM_{k}| - \min\_count_{kind}}{\max\_count_{kind} - \min\_count_{kind}} \times 5 \qquad (15)$$

SQL statements that return the required numbers are provided below:

$|RM_k|$ → `SELECT COUNT(*) AS a FROM imdb.title where id in ( . . . ) and kind_id = k;`

min_count_{kind} → `SELECT MIN(a) FROM (SELECT COUNT(*) AS a FROM imdb.title WHERE id IN ( . . . ) GROUP BY kind_id) AS b;`

max_count_{kind} → `SELECT MAX(a) FROM (SELECT COUNT(*) AS a FROM imdb.title WHERE id IN ( . . . ) GROUP BY kind_id) AS b;`

As a movie has exactly 1 kind, the result found by analyzing the specific value k is also the result for the kind feature.

$$pred\_count_{kind}(u,m) = pred\_count_{k}(u,m)$$

- **Prediction of Genre**

Prediction of genre is a little different from kind since one movie can have more than one genre. A movie's genre is usually represented by a set consisting of different genres. In this case, a prediction is made for each genre separately, and then the average of these results is calculated to find the genre prediction of the

movie. Representing the specific genre that is currently being analyzed with *g*, the following prediction can be made for that genre:

$$pred\_count_g(u,m) = \frac{\left|RM_g\right| - \min\_count_{genre}}{\max\_count_{genre} - \min\_count_{genre}} \times 5 \quad (16)$$

SQL statements that return the required numbers are provided below:

$|RM_g|$ → `SELECT COUNT(*) AS a FROM imdb.movie_info WHERE info_type_id = 3 AND movie_id IN( . . . ) AND info = g;`

min_count_{genre} → `SELECT MIN(a) FROM (SELECT COUNT(*) AS a, info FROM imdb.movie_info WHERE info_type_id = 3 AND movie_id IN ( . . . ) GROUP BY info) AS b;`

max_count_{genre} → `SELECT MAX(a) FROM (SELECT COUNT(*) AS a, info FROM imdb.movie_info WHERE info_type_id = 3 AND movie_id IN ( . . . ) GROUP BY info) AS b;`

As a movie has more than one genre, the prediction result is the average of all genres of a movie.

$$pred\_count_{genre}(u,m) = avg(pred\_count_{g_1}(u,m),..., pred\_count_{g_n}(u,m))$$

- **Prediction of Writer**

Like the genre, writer field of a movie is also a set consisting of more than one value. Likewise, a prediction is made according to each writer separately and the results are combined so that an average value is calculated. Representing the specific writer that is currently being analyzed with *w*, the following prediction can be made for that writer:

$$pred\_count_w(u,m) = \frac{\left|RM_w\right| - \min\_count_{writer}}{\max\_count_{writer} - \min\_count_{writer}} \times 5 \quad (17)$$

SQL statements that return the required numbers are provided below:

$|RM_w|$ → `SELECT COUNT(*) AS a FROM ( SELECT c.id FROM cast_info c, name n WHERE n.name = w AND c.role_id = 4 AND c.person_id = n.id AND c.movie_id IN( . . . )) AS b;`

min_count_{writer} → `SELECT MIN(a) FROM ( SELECT COUNT(*) AS a, n.name FROM cast_info c, name n WHERE c.role_id = 4 AND c.person_id = n.id AND c.movie_id IN ( . . . ) GROUP BY n.name) AS b;`

max_count_{writer} → `SELECT MAX(a) FROM ( SELECT COUNT(*) AS a, n.name FROM cast_info c, name n WHERE c.role_id = 4 AND c.person_id = n.id AND c.movie_id IN ( . . . ) GROUP BY n.name) AS b;`

As a movie has more than one writer, the prediction result is the average of all writers of a movie.

$$pred\_count_{writer}(u,m) = avg(pred\_count_{w_1}(u,m),..., pred\_count_{w_n}(u,m))$$

- **Prediction of Country**

The country field of a movie is usually a single value. However, in the movies having more than one country, the same method as in genre and writer is applied. Representing the specific country that is currently being analyzed with *c*, the following prediction can be made for that country:

$$pred\_count_c(u,m) = \frac{|RM_c| - \min\_count_{country}}{\max\_count_{country} - \min\_count_{country}} \times 5 \quad (18)$$

SQL statements that return the required numbers are provided below:

$|RM_c|$ → `SELECT COUNT(*) AS a FROM imdb.movie_info WHERE info_type_id = 8 AND movie_id IN( . . . ) AND info = c;`

$min\_count_{country}$ → `SELECT MIN(a) FROM (SELECT COUNT(*) AS a, info FROM imdb.movie_info WHERE info_type_id = 8 AND movie_id IN ( . . . ) GROUP BY info) AS b;`

$max\_count_{country}$ → `SELECT MAX(a) FROM (SELECT COUNT(*) AS a, info FROM imdb.movie_info WHERE info_type_id = 8 AND movie_id IN ( . . . ) GROUP BY info) AS b;`

As a movie can have more than one country, the prediction result is the average of all countries of a movie.

$$pred\_count_{country}(u,m) = avg(pred\_count_{c_1}(u,m),..., pred\_count_{c_n}(u,m))$$

- **Prediction of Language**

Usually, the language is also a single valued field. Like the country, in the movies having more than one language, the same method as in genre and writer is applied. Representing the specific language that is currently being analyzed with *l*, the following prediction can be made for that language:

$$pred\_count_{l}(u,m) = \frac{|RM_l| - \min\_count_{language}}{\max\_count_{language} - \min\_count_{language}} \times 5 \quad (19)$$

SQL statements that return the required numbers are provided below:

$|RM_l|$ → `SELECT COUNT(*) AS a FROM imdb.movie_info WHERE info_type_id = 4 AND movie_id IN( . . . ) AND info = l;`

$min\_count_{language}$ → `SELECT MIN(a) FROM (SELECT COUNT(*) AS a, info FROM imdb.movie_info WHERE info_type_id = 4 AND movie_id IN ( . . . ) GROUP BY info) AS b;`

$max\_count_{language}$ → `SELECT MAX(a) FROM (SELECT COUNT(*) AS a, info FROM imdb.movie_info WHERE info_type_id = 4 AND movie_id IN (. . . ) GROUP BY info) AS b;`

As a movie can have more than one language, the prediction result is the average of all languages of a movie.

$$pred\_count_{language}(u,m) = avg(pred\_count_{l_1}(u,m),..., pred\_count_{l_n}(u,m))$$

- **Prediction of Company**

A movie is usually produced by a number of different companies. Therefore, a prediction is made for each company separately and the results are combined in order to get the average value of them. Representing the specific company that is currently being analyzed with *c*, the following prediction can be made for that company:

$$pred\_count_c(u,m) = \frac{|RM_c| - \min\_count_{company}}{\max\_count_{company} - \min\_count_{company}} \times 5 \quad (20)$$

SQL statements that return the required numbers are provided below:

$|RM_c|$ → `SELECT COUNT(*) AS a FROM (select cn.name FROM movie_companies mc, company_name cn WHERE cn.name = c AND mc.company_id = cn.id AND mc.movie_id IN( . . . ) AND mc.company_type_id = 2 GROUP BY cn.name) AS b;`

$min\_count_{company}$ → `SELECT MIN(a) FROM (SELECT COUNT(*) AS a, cn.name FROM movie_companies mc, company_name cn WHERE mc.company_id = cn.id AND mc.movie_id IN ( . . . ) AND mc.company_type_id = 2 GROUP BY cn.name) AS b;`

$max\_count_{company}$ → `SELECT MAX(a) FROM (SELECT COUNT(*) AS a, cn.name FROM movie_companies mc, company_name cn WHERE mc.company_id = cn.id AND mc.movie_id IN ( . . . ) AND mc.company_type_id = 2 GROUP BY cn.name) AS b;`

As a movie has more than one company, the prediction result is the average of all companies of a movie.

$$pred\_count_{company}(u,m) = avg(pred\_count_{c_1}(u,m),..., pred\_count_{c_n}(u,m))$$

- **Combining the Count Based Predictions**

Up to this part, the details of making count based predictions for different features are provided. This part combines these predictions with different weight values and produces a resulting prediction based on the count data.

$$pred\_count_{overall}(u,i) = w_{kind} \times pred\_count_{kind} + w_{genre} \times pred\_count_{genre} + w_{writer} \times pred\_count_{twriter} + w_{country} \times pred\_count_{country} + w_{language} \times pred\_count_{tlanguage} + w_{company} \times pred\_count_{company} \quad (21)$$

4.5.4.2 Rating Based Prediction

As the second prediction type, this prediction method takes previous ratings into account. It is important to know the rating given by the user to similar valued items. For instance if a user rated N movies having the genre "romance", then the predicted rating is calculated as follows:

$$pred\_rating_{romance}(u,m) = \frac{\sum_{i \in RM_{romance}} r(u,i)}{|RM_{romance}|} \quad (22)$$

In this formula, $RM_{romance}$ represents the set of rated movies which have "romance" in their genre list, and $|RM_{romance}|$ is the cardinality of this set. The formula can be rewritten as follows:

$$pred\_rating_{romance}(u,m) = \frac{r(u,i_1) + r(u,i_2) + \ldots + r(u,i_N)}{N}$$

Following parts examine each feature separately and explain how the prediction is made for that feature.

- **Prediction of Kind**

Every movie has exactly one kind, which can be a movie, TV series, or etc. Representing the specific kind that is currently being analyzed with *k*, the following prediction can be made for that kind:

$$pred\_rating_{k}(u,m) = \frac{\sum_{i \in RM_{k}} r(u,i)}{|RM_{k}|} \quad (23)$$

$pred\_rating_k(u,m)$ = `SELECT AVG(rating) FROM (SELECT * FROM movielens.ratings WHERE user_id = 9 AND movieimdb_id IN (SELECT id FROM title WHERE kind_id = k)) AS b;`

As a movie has exactly one kind, the result found by analyzing the specific value k is also the result for the kind feature.

$$pred\_rating_{kind}(u,m) = pred\_rating_k(u,m)$$

- **Prediction of Genre**

Prediction of genre is a little different from kind since one movie can have more than one genre. A movie's genre is usually represented by a set consisting of different genres. In this case, a prediction is made for each genre separately, and then the average of these results is calculated to find the genre prediction of the movie. Representing the specific genre that is currently being analyzed with *g*, the following prediction can be made for that genre:

$$pred\_rating_g(u,m) = \frac{\sum_{i \in RM_g} r(u,i)}{\left|RM_g\right|} \quad (24)$$

$pred\_rating_g(u,m)$ = `SELECT AVG(rating) FROM (SELECT * FROM movielens.ratings WHERE user_id = 9 AND movieimdb_id IN (SELECT movie_id FROM movie_info WHERE info_type_id = 3 AND info = g)) AS a";`

As a movie has more than one genre, the prediction result is the average of all genres of a movie.

$$pred\_rating_{genre}(u,m) = avg(pred\_rating_{g_1}(u,m),..., pred\_rating_{g_n}(u,m))$$

- **Prediction of Writer**

Like the genre, writer field of a movie is also a set consisting of more than one value. A prediction is made according to each writer separately and the results are

combined so that an average value is calculated. Representing the specific writer that is currently being analyzed with *w*, the following prediction can be made for that writer:

$$pred\_rating_w(u,m) = \frac{\sum_{i \in RM_w} r(u,i)}{|RM_w|} \quad (25)$$

$pred\_rating_w(u,m)$ = `SELECT AVG(rating) FROM (SELECT * FROM movielens.ratings WHERE user_id = 9 AND movieimdb_id IN (SELECT c.movie_id FROM cast_info c, name n WHERE c.role_id = 4 AND c.person_id = n.id AND n.name = w)) AS a;`

The rating based prediction result for writer feature is the average of all writers of the movie.

$$pred\_rating_{writer}(u,m) = avg(pred\_rating_{w_1}(u,m),..., pred\_rating_{w_n}(u,m))$$

- **Prediction of Country**

As already stated, country field of a movie is usually a single value. However, in the movies having more than one country, the same method as in genre and writer is applied. Representing the specific country that is currently being analyzed with *c*, the following prediction can be made for that country:

$$pred\_rating_c(u,m) = \frac{\sum_{i \in RM_c} r(u,i)}{|RM_c|} \quad (26)$$

$pred\_rating_c(u,m)$ = `SELECT AVG(rating) FROM (SELECT * FROM movielens.ratings WHERE user_id = 9 AND movieimdb_id IN (SELECT movie_id FROM movie_info WHERE info_type_id = 8 AND info = c)) AS a";`

For movies with more than one country, the prediction result is the average of all countries.

$$pred\_rating_{country}(u,m) = avg(pred\_rating_{c_1}(u,m),..., pred\_rating_{c_n}(u,m))$$

- **Prediction of Language**

As stated above, language is usually a single valued field. For the movies having more than one language, the same method as in genre and writer is applied. Representing the specific language that is currently being analyzed with *l*, the following prediction can be made for that language:

$$pred\_rating_l(u,m) = \frac{\sum_{i \in RM_l} r(u,i)}{|RM_l|} \quad (27)$$

$pred\_rating_l(u,m)$ = `SELECT AVG(rating) FROM (SELECT * FROM movielens.ratings where user_id = 9 AND movieimdb_id IN (SELECT movie_id FROM movie_info WHERE info_type_id = 4 AND info = l)) AS a;`

For more than one language, the prediction result is the average of all languages.

$$pred\_rating_{language}(u,m) = avg(pred\_rating_{l_1}(u,m),..., pred\_rating_{l_n}(u,m))$$

- **Prediction of Company**

As stated before, a movie is usually produced by a number of different companies. Therefore, a prediction is made for each company separately and the results are combined in order to get the average value of them. Representing the specific company that is currently being analyzed with c, the following prediction can be made for that company:

$$pred\_rating_c(u,m) = \frac{\sum_{i \in RM_c} r(u,i)}{|RM_c|} \quad (28)$$

$pred\_rating_c(u,m)$ = `SELECT AVG(rating) FROM (SELECT * FROM movielens.ratings WHERE user_id = 9 AND movieimdb_id IN`

```
(SELECT mc.movie_id FROM movie_companies mc, company_name cn
WHERE cn.name = c AND mc.company_id = cn.id AND
mc.company_type_id = 2)) AS b;
```

As a movie has more than one production companies, the prediction result is the average of all companies.

$$pred\_rating_{company}(u,m) = avg(pred\_rating_{c_1}(u,m),..., pred\_rating_{c_n}(u,m))$$

- **Combining the Rating Based Predictions**

Up to this part, the details of making rating based predictions for different features are provided. This part combines these predictions with different weight values and produces a resulting prediction based on the rating data.

$$\begin{aligned} pred\_rating_{overall}(u,i) &= w_{type} \times pred\_rating_{type} + w_{genre} \times pred\_rating_{genre} \\ &+ w_{writer} \times pred\_rating_{twriter} + w_{country} \times pred\_rating_{country} \\ &+ w_{language} \times pred\_rating_{tlanguage} + w_{company} \times pred\_rating_{company} \end{aligned} \quad (29)$$

4.5.4.3 Overall Prediction

As stated above, overall prediction of missing ratings is made by combining the count based prediction with rating based prediction. As these prediction types may affect the rating pattern of users in different levels from each other, they are combined with a relativity factor δ. With a user denoted by *u* and a movie denoted by *m*, the overall prediction is calculated as follows:

$$pred_{overall}(u,m) = \delta \times pred\_count_{overall}(u,m) + (1-\delta) \times pred\_rating_{overall}(u,m) \quad (30)$$

In the evaluation of the system, which is included in chapter 5, the effect of the factor δ is analyzed and the most suitable δ value is determined.

As an illustrative example, consider a movie M having the following feature values:

Kind: k

Genre: <g_1, g_2, g_3>

Writer: <w_1, w_2>

Country: <coun>

Language: <lang>

Company: <$comp_1$, $comp_2$>

The steps to predict rating given by a user U to movie M are as follows:

Count Based Prediction

$$pred\_count_{kind}(U,M) = pred\_count_{k}(U,M) = a_1$$

$$pred\_count_{genre}(U,M) = avg(pred\_count_{g_1}(U,M), pred\_count_{g_2}(U,M), pred\_count_{g_3}(U,M)) = b_1$$

$$pred\_count_{writer}(U,M) = avg(pred\_count_{w_1}(U,M), pred\_count_{w_2}(U,M)) = c_1$$

$$pred\_count_{country}(U,M) = pred\_count_{coun}(U,M) = d_1$$

$$pred\_count_{language}(U,M) = pred\_count_{lang}(U,M) = e_1$$

$$pred\_count_{company}(U,M) = avg(pred\_count_{comp_1}(U,M), pred\_count_{comp_2}(U,M)) = f_1$$

$$pred\_count_{overall}(U,M) = a_1 \times w_{kind} + b_1 \times w_{genre} + c_1 \times w_{writer} + d_1 \times w_{country} + e_1 \times w_{language} + f_1 \times w_{company}$$

Rating Based Prediction

$$pred\_rating_{kind}(U,M) = pred\_rating_{k}(U,M) = a_2$$

$$pred\_rating_{genre}(U,M) = avg(pred\_rating_{g_1}(U,M), pred\_rating_{g_2}(U,M), pred\_rating_{g_3}(U,M)) = b_2$$

$$pred\_rating_{writer}(U,M) = avg(pred\_rating_{w_1}(U,M), pred\_rating_{w_2}(U,M)) = c_2$$

$$pred\_rating_{country}(U,M) = pred\_rating_{coun}(U,M) = d_2$$

$$pred\_rating_{language}(U,M) = pred\_rating_{lang}(U,M) = e_2$$

$$pred\_rating_{company}(U,M) = avg(pred\_rating_{comp_1}(U,M),$$
$$pred\_rating_{comp_2}(U,M)) = f_2$$

$$pred\_rating_{overall}(U,M) = a_2 \times w_{kind} + b_2 \times w_{genre} + c_2 \times w_{writer} + d_2 \times w_{country}$$
$$+ e_2 \times w_{language} + f_2 \times w_{company}$$

Overall Prediction

$$pred_{overall}(U,M) = \delta \times pred\_count_{overall}(U,M) + (1-\delta) \times pred\_rating_{overall}(U,M)$$

CHAPTER 5

EVALUATION OF THE SYSTEM

In this section, experimental evaluation of the system is provided. First of all, the data set which is used in the evaluation process is defined. Secondly, evaluation metrics are described and then the evaluation of the demographic feature weights is made. Lastly, the impacts of the parameters α, β and δ are analyzed providing some graphical and numerical results.

5.1 Data Set

Evaluation of *ReMovender* is done with one of the *MovieLens* datasets [52] which is maintained by the *GroupLens* Research Group [17] at University of Minnesota. The group has three available datasets currently and statistics about these three data sets are as follows:

Table 15 – MovieLens Data Sets

	Number of Movies	Number of Users	Number of Ratings
1st Data Set	1682	943	100 000
2nd Data Set	3900	6040	1 000 000
3rd Data Set	10681	71567	10 000 000

In the evaluation of *ReMovender*, the 1st data set which contains 100.000 ratings in a one to five scale, given by 943 distinct users to 1682 distinct movies is used. The reason for using this data set is to compare the results with the studies [35] and [36] which have already made the evaluation of their systems on the same data set. It is guaranteed by the dataset that each user has rated at least 20 items in the system.

For the content based part, the contents of the movies are retrieved from IMDb. For each movie existing in the *MovieLens* data set, the content information is retrieved by using the title and production year of that movie. The <title, production year> pair is assumed to be unique for each movie as it is a low possibility that two movies with the same name can be produced in the same year. Most of the movies could be correlated by using this information. However, nearly 400 movies could not be correlated due to some inconsistencies such as language differences, usage of 'and' instead of '&' or vice versa, capital letter inconsistencies, etc. It was also a problem that the titles of some movies in the *MovieLens* data set use the aka (also known as) titles in IMDb. After correcting these inconsistencies manually, all of the movie IDs in the *MovieLens* dataset could be correlated with their IMDb IDs. With the help of this correlation, the system had the chance to obtain all the content information required while making the content based prediction.

The proposed approach is applied on three different sets of users namely; T100, T200 and T300. The first set is composed of 100 training users, the second one is composed of 200 training users and the third set is composed of 300 training users. These sets are also tested under three different circumstances namely G5, G10 and G20. In G5, the test user is assumed to have rated 5 movies, in G10 the test user is assumed to have rated 10 movies and G20 assumes that the test user rated 20 movies.

5.2 Evaluation Metrics

The evaluation of *ReMovender* is conducted by using the metrics Mean Absolute Error (MAE), *precision* and *recall*.

In statistics, MAE is a quantity which is used to measure how close the predictions are to the actual values. The MAE is given by the following formula:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|f_i - y_i| = \frac{1}{n}\sum_{i=1}^{n}|e_i| \quad (31)$$

As its name suggests, the MAE is an average of the absolute errors $e_i = f_i - y_i$, where f_i is the prediction and y_i is the true value. In *ReMovender*, MAE is used in the collaborative part which is the prediction of missing data [55].

As an illustrative example, Table 16 provides some actual ratings given by user *u* to different movies along with the ratings predicted by the system:

Table 16 – Actual Ratings vs. Predicted Ratings

Movie	Predicted Rating	Actual Rating
M_1	1	3
M_2	4	4
M_3	2	3
M_4	3	5

By using the numeric values provided in Table 16, MAE of the ratings produced by the system can be calculated as follows:

$$MAE = \frac{1}{4}\left[|1-3| + |4-4| + |2-3| + |3-5|\right] = \frac{5}{4} = 1.25$$

Precision and *recall*, which are the most popular metrics for evaluating information retrieval systems, are used in the evaluation of content-based prediction part. In the evaluation of some earlier recommendation systems, these metrics have been used by Billsus and Pazzani [53], Basu et al. [3], and Sarwar et al. [54].

Precision represents the probability that a selected item is really relevant for the user [56]. It is defined as the ratio of relevant items selected correctly to the number of items selected, as shown in (32):

$$\Pr ecision = \frac{N_{rs}}{N_s} \tag{32}$$

On the other hand, *recall* represents the probability that a relevant item will be selected [56]. It is defined as the ratio of relevant items selected to the total number of relevant items available, as shown in (33):

$$\mathrm{Re}\, call = \frac{N_{rs}}{N_r} \tag{33}$$

There are also several approaches which combine *precision* and *recall* into a single metric. One approach is the *F-measure*, whose formula is presented in (34):

$$F - measure = \frac{2 \times \Pr ecision \times \mathrm{Re}\, call}{\Pr ecision + \mathrm{Re}\, call} \tag{34}$$

In the definition of these metrics, the term *'relevant'* is often used. Thus, it is important to decide what is relevant to the user and what is not. As stated throughout this work, *ReMovender* users rate the movies in a scale from 1 to 5. Average of the all ratings given by a user constitutes a threshold value for that user and rating values above the threshold are considered as relevant, while the remaining ones are considered as irrelevant.

5.3 Evaluation of the Demographic Feature Weights

Recall from (7) that the demographic similarity of users is used in demographic rating prediction.

$$\hat{r}_D(a,i) = \bar{a} + \frac{\sum_{a_k \in D(a)} Sim_D(a,a_k) \times (r_{ak,i} - a_k)}{\sum_{a_k \in D(u)} Sim_D(a,a_k)}$$

As already stated, *D(a)* represents the set of nearest neighbors of user *a*, which means the users who have similar demographic features with the target user *a*. However, the similarity should be clearly defined by using a threshold value in order to distinguish similar users from dissimilar ones. For instance let λ be the threshold value. Therefore, the users with a demographic similarity lower than λ are regarded as dissimilar, while the others are regarded as similar.

5.3.1 Determining the Threshold Value

In order to determine the finest threshold value, experiments are done using the MAE metric. As the rating information is not taken into account G5, G10 and G20 are not used in this experiment. 300 users are taken as training users and 200 users are taken as test users. This means that T300 is used in the experiment. Table 17 shows the resulting MAE values with threshold values 0.2, 0.4, 0.6 and 0.8 respectively.

Table 17 – Produced MAE's with different threshold values

Threshold Value (λ)	MAE
0.2	1.823707
0.4	1.199572
0.6	1.563328
0.8	1.734125

The values are put into a chart in order to make the impact of different threshold values easier to understand. As can be seen from Table 17 and the chart in Figure 10, the most suitable threshold value is 0.4 that produces the smallest MAE. In the consequent sections of the evaluation, the threshold value is taken as 0.4 so that λ=0.4.

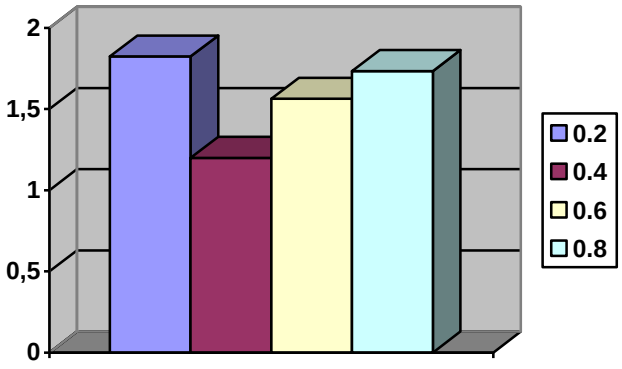

Figure 10 – Threshold Value vs. MAE

5.3.2 Analysis of Feature Weights

Section 4.4.4.1 gives the weight values assigned to the features. To recall, age is assigned 0.56, gender is assigned 0.41 and occupation is assigned 0.03. Besides, zip code is assigned 0, therefore not taken into account. In this section, the MAE metric is used to calculate the error produced by using these weight values for making demographic predictions. For now, the effects of local user similarity and global user similarity are not taken into account. Only the demographic prediction is used in the calculation of MAE.

As the rating information is not used in demographic prediction, this part of the evaluation is not divided into subsets G5, G10 and G20. However; T100, T200 and T300 are used and result for each one is provided separately.

Table 18 – Resulting mean absolute errors with produced feature weights.

Number of Training Users	Number of Predicted Items	MAE
T100	12688	1.423707
T200	9155	1.323456
T300	6313	**1.199572**

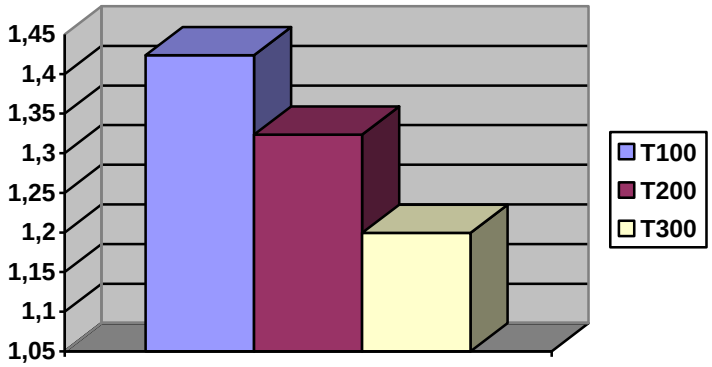

Figure 11 – MAE values for different sets of users

As the chart in Figure 11 indicates, the smallest MAE is obtained with T300, where the number of training users is the biggest of all. On the other hand, the biggest MAE is obtained with T100 where the smallest number of training users is used. This gives the idea that as the number of training users increase, the MAE of the system decreases. This is because when the number of training users is big, the system has the chance to train itself better.

It would be more meaningful to compare the results with different combination of weight values. This will help to determine whether the produced weight values are effective or not. Table 19 shows MAE values produced by using six different weight combinations:

Table 19 – Resulting MAE values on T300

Age	Gender	Occupation	Zip	MAE
0.56	0.61	0.03	0.00	**1.199572**
0.25	0.25	0.25	0.25	1.229572
1	0	0	0	1.563567
0	1	0	0	1.453729
0	0	1	0	1.834218
0	0	0	1	**1.878356**

Note that, the smallest MAE is produced in the first row of Table 19 which uses the weight values produced within this study. Note that, according to the produced weight values, the impact of the features on the ratings can be ordered as follows:

gender(0.61) > age(0.56) > occupation(0.03) > zip(0.00)

Figure 12 proves that this ordering is correct since MAE increases in the direction gender → age → occupation → zip.

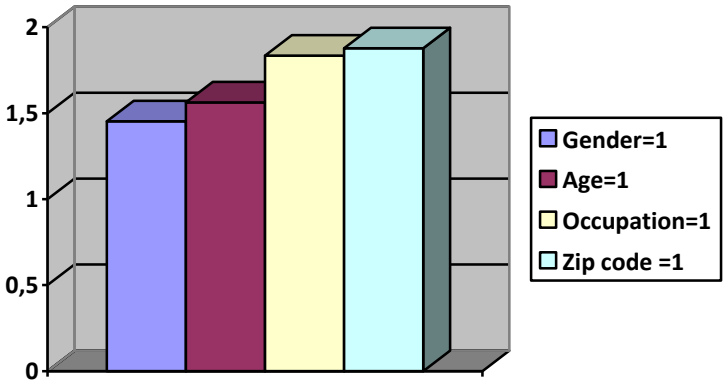

Figure 12 – MAE values produced by using different feature weights

5.4 Evaluation of Missing Rating Prediction

Recall from (12) that prediction based on LU & GU is combined with prediction based on demographic data with a significance weighting factor β, and a general rating prediction is made.

$$\hat{r}(a,i) = \beta \times \hat{r}_{LU\&GU}(a,i) + (1-\beta) \times \hat{r}_{D}(a,i)$$

In this part of the evaluation process, firstly the introduced method is evaluated by using β=0.5 and the results are compared with existing algorithms. Then the impact of parameter β is examined.

5.4.1 Comparative Results with β=0.5

In order to test the performance of *ReMovender*'s missing rating prediction approach, the MAE values obtained for the 9 configurations were compared with those of the state-of-the-arts algorithms. The values are obtained by using the *MovieLens* dataset.

The parameters or thresholds that are used throughout the prediction process are empirically set as $\gamma = 30$, $\eta = 0.6$, NumberOfNeighbors = 35 and $\alpha = 0.5$ just like in the experimental setup of [36]. With the help of this, the results could be compared to those of [36]. As the rating information is taken into account in LU & GU part, G5, G10 and G20 are used in this experiment along with T100, T200 and T300. Different from other systems, another parameter β is set to 0.5 in *ReMovender*. It would be useful to state that after predicting missing data according to (12) with β=0.5, the density of the user-item matrix became ~0.8.

In Table 20, MAE comparison of the collaborative missing data prediction (LU&GU+D) with state-of-the-arts algorithms including user-based using PCC (UPCC) [57], item-based using PCC (IPCC) [58], Similarity Fusion (SF) [59], Effective Missing Data Prediction (EMDP) [35], and Local & Global User Similarity (LU&GU) [36] is provided. It can be easily observed from this table that addition of the demographic features makes the MAE results worse.

Table 20 – MAE comparison with state-of-the-arts algorithms on MovieLens

Training Users	Methodology	G5	G10	G20
100	**LU&GU+D**	**1.3483**	**1.2127**	**1.0846**
	CBCF	0.7889	0.7653	0.7561
	LU&GU	0.791	0.7681	0.7565
	EMDP	0.7896	0.7668	0.7806
	SF	0.8446	0.7807	0.7717
	UPCC	0.8377	0.8044	0.7943
	IPCC	0.9639	0.8922	0.8577

Table 20 Continued

	LU&GU+D	**1.2984**	**1.2875**	**0.9827**
	CBCF	0.7816	0.7648	0.7533
	LU&GU	0.7937	0.7733	0.7719
200	EMDP	0.7997	0.7953	0.7908
	SF	0.8507	0.8012	0.7862
	UPCC	0.8185	0.8067	0.796
	IPCC	0.955	0.9135	0.871
	LU&GU+D	**1.1765**	**1.0393**	**0.9632**
	CBCF	0.7637	0.7562	0.7384
	LU&GU	0.7718	0.7704	0.7444
300	EMDP	0.7925	0.7951	0.7552
	SF	0.8062	0.7971	0.7527
	UPCC	0.8055	0.7910	0.7805
	IPCC	0.9862	0.9266	0.8573

Figure 13 shows the resulting MAE values with a chart. It can be concluded from the chart that MAE decreases as the number of training users and the number of given ratings for a user increase.

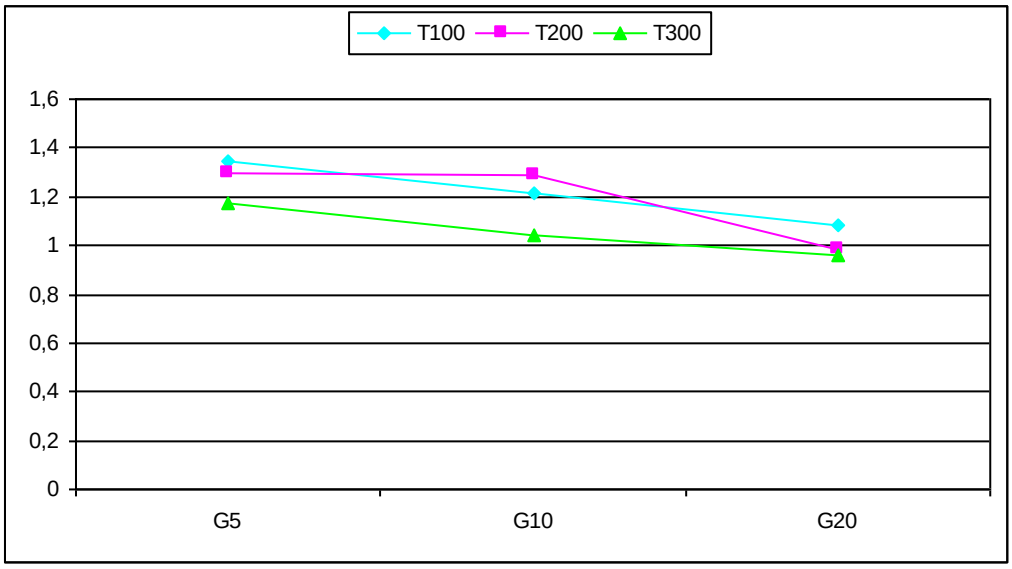

Figure 13 – MAE Results with Different Sets of Users and Ratings

5.4.2 Impact of β

In order to determine the finest value for β, experiments are conducted by using the MAE metric. Table 21 shows the resulting MAE values with β values 0.00, 0.25, 0.50, 0.75 and 1.00 respectively which are obtained by using T300 and G20.

Table 21 – Produced MAE's with different β values

β	MAE
0.00	1.199572
0.25	1.178707
0.50	0.963200
0.75	0.923639
1.00	0.723075

The values are put into a chart in order to make the impact of different β values easier to understand. As can be seen from Table 21 and the chart in Figure 14, the most suitable β value is 1 which means that the demographic information is not taken into account and the prediction is made based on only LU & GU. For this reason, demographic prediction is not taken into account while predicting missing data in order to make content based recommendation.

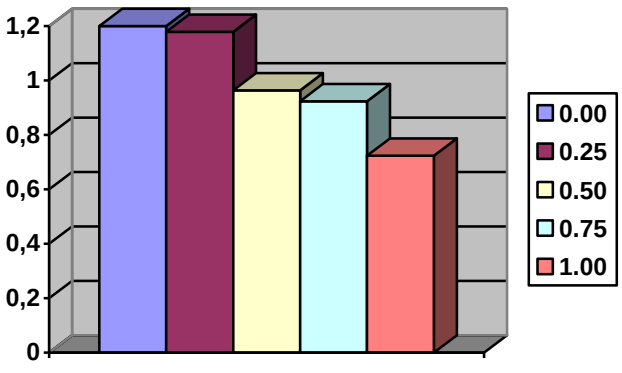

Figure 14 – β vs. MAE on T300

It would be useful to state that after predicting missing data according to (12) with β=1, the density of the user-item matrix became ~0.65.

5.5 Evaluation of the Content-based Recommendation

As stated in 5.2, evaluation of content-based recommendation part is done by using the precision and recall metrics. Remember from (13) that content-based recommendation is composed of two parts which are count-based prediction and rating-based prediction:

$$pred\_overall(u,i) = \delta \times pred\_overall_{count}(u,i) + (1-\delta) \times pred\_overall_{rating}(u,i)$$

In this section, first an internal evaluation is made to determine the finest δ value. Then the results produced by using the best δ value are compared with the existing studies.

5.5.1 Impact of δ

Precision, recall and F-measure values are calculated by setting δ to 0, 0.25, 0.5, 0.75 and 1 respectively. This experiment is done on 5000 ratings given by some number of users to movies. Table 22 shows the resulting values for precision, recall and F-measure:

Table 22 – Precision, Recall and F-measure for Different δ Values

Significance Weighting (δ)	Precision(%)	Recall(%)	F-measure
0.00	68	71	69.4
0.25	**72**	**78**	**74.9**
0.50	62	59	60.4
0.75	34	32	33.0
1.00	21	16	18.1

On the other hand, Figure 15 displays the results graphically so that the impact of parameter δ can be observed better:

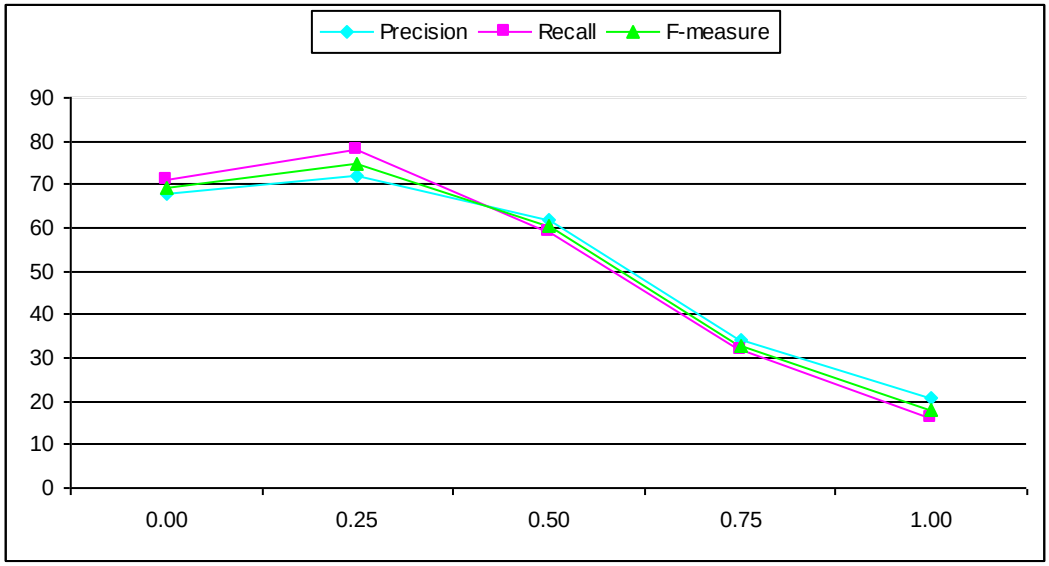

Figure 15 – δ vs. Precision, Recall and F-measure

It can be easily observed from Table 22 and Figure 15 that the finest values for precision, recall and F-measure are obtained by setting δ to 0.25. For this reason, while making comparison with existing systems in the following section, δ=0.25 is used.

5.5.2 Comparative Results with δ=0.25

Table 23 gives the precision, recall and F-measure results from some existing studies which are obtained from [19]:

Table 23 – Comparison with Existing Systems

Methodology	Precision(%)	Recall(%)	F-measure
ReMovender	**72**	**78**	**74.9**
MovieLens	66	74	69.7
MovieMagician Feature-Based	61	75	67.3
MovieMagician Clique-Based	74	73	73.5
MovieMagician Hybrid	73	56	63.3
OPENMORE	62.02	91.7	74

The results are put into a chart in Figure 16 to see the different results obtained from different methodologies graphically:

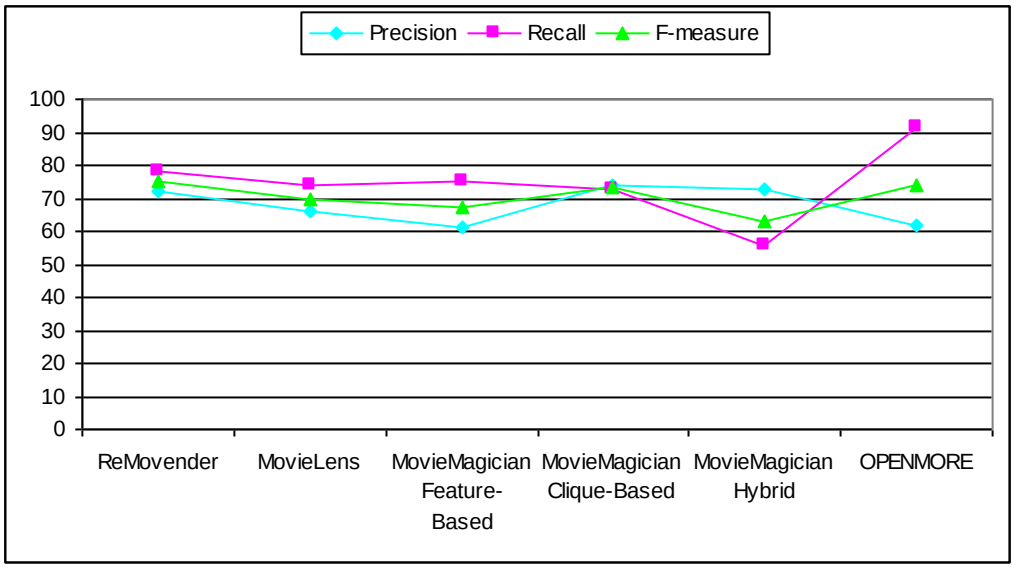

Figure 16 – Precision, Recall and F-measure for Different Methodologies

It can be seen from Table 23 and Figure 16 that precision of *ReMovender* has an average value among existing systems, while its recall is the second highest value among others. For the F-measure metric, which is the combination of precision and recall, it can be concluded that the methodology of *ReMovender* outperforms the other methodologies in the F-measure metric.

CHAPTER 6

CONCLUSION AND FUTURE WORK

To conclude, this thesis study represents a web based movie recommendation system which is named as *ReMovender*. As explained throughout this work, *ReMovender* is composed of many different approaches and algorithms, each of which help to increase the capacity and success of the system.

As the first improvement, demographic data is embedded into the user based similarity part. With the addition of demographic data, the system is aimed to make more powerful recommendations. However, the distance measure of some of the demographic features could have been formulated in a different way so that the demographic similarity between two users is calculated more accurately. For the gender feature, the distance between genders of two users is taken as 1 if the users have the same gender and 0 otherwise. This formula seems to be reasonable since this cannot be done in a different way. Similarly, the distance between the ages of two users is calculated with a formula which can be considered as a rational way of comparing ages. However, the distance formula between occupations and zip codes may be replaced with a better formula. As a future work, distance between zip codes can be calculated by using the geographical distance between the corresponding locations. Besides, the occupations which are related with each other can be grouped together. By this way, dissimilar but related occupations will be assigned a distance value between 0 and 1 instead of 0.

As for the content based part, the number of features which are used to correlate movies can be increased in the future. As stated before, relying on the results of an existing study [37], features having consistent weight values are used in

ReMovender. In the future, as *ReMovender* grows and owns more number of users, these weight values can be altered and new features having their own weight values can be added. This can be done in different ways such as requesting feedback from users or analyzing the ratings given by them.

Some improvements can also be done in the evaluation part as a future work. In the evaluation part, the weight values assigned to movie features are assumed to be the values producing the best results. As a future work, this assumption can be supported by adding some numerical analysis and providing some graphical results.

REFERENCES

[1] Gediminas Adomavicius, Alexander Tuzhilin, "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions," IEEE Transactions on Knowledge and Data Engineering, VOL. 17, NO.6, June 2005.

[2] Oznur Kirmemis, Aysenur Birturk, "A Content-Based User Model Generation and Optimization Approach for Movie Recommendation," 6th Workshop on Intelligent Techniques for Web Personalization & Recommender Systems, 2008.

[3] C. Basu, H. Hirsh, W. Cohen, "Recommendation as Classification: Using Social and Content-Based Information in Recommendation," Recommender Systems Papers from 1998 Workshop, Technical Report WS-98-08, AAAI Press, 1998.

[4] L. Terveen, W. Hill, "Beyond Recommender Systems: Helping People Help Each Other", In Carroll, J., (Ed.), HCI in the New Millennium. Addison Wesley, pp. 475-486, 2001.

[5] Dipankaj G. Medhi, Juri Dakua, "MovieReco: A Recommendation System", World Academy of Science, Engineering and Technology, vol. 4, pp. 70-74, April 2005.

[6] Souvik Debnath, "Machine Learning Based Recommendation System", Master Thesis, Indian Institute of Technology, Dept. of Comp. Science and Engineering, May 2008.

[7] Mark Van Setten, "Supporting people in finding information, Hybrid recommender systems and goal-based structuring", Telematica Instituut Fundamental Research Series, vol. 016. Enschede, the Netherlands: Telematica Instituut, 2005.

[8] Gediminas Adomavicius, Alex Tuzhilin, “Recommendation Technologies: Survey of Current Methods and Possible Extensions”, Stern School of Business, New York University, 2004.

[9] Alexandrin Popescul, Lyle Ungar, David Pennock, “Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments”, In Proceedings of 17th Conference in Uncertainty in Artificial Intelligence, pp 437-444, University of Washington, Seattle, Washington, USA, August 2-5, 2001.

[10] Zheng Wen, “Recommendation System Based on Collaborative Filtering”, December 12, 2008.

[11] Q. Chen, U. Aickelin, “Movie Recommendation Systems Using an Artificial Immune System”, In Poster Proceedings of ACDM 2004 Engineers’ House, Bristol, UK, 2004.

[12] Jennifer Golbeck, James Hendler, “FilmTrust: Movie Recommendations using Trust in Web-based Social Networks”.

[13] MovieLens, www.movielens.umn.edu, Last accessed on 8 May 2010.

[14] Garden, Matthew, Gregory Dudek, “Semantic feedback for hybrid recommendations in Recommendz”, In Proceedings of the IEEE International Conference on e-Technology, e-Commerce, and e-Service (EEE05), Hong Kong, China, March 2005.

[15] P. Perny, J. D. Zucker, “Preference-based Search and Machine Learning for Collaborative Filtering: the ``Film-Conseil'' recommender system”, Information, Interaction, Intelligence, 1(1):9-48, 2001.

[16] L. Terveen, W. Hill, Amento, B., McDonald, D., Creter, J., “PHOAKS: A System for Sharing Recommendations”, CACM 40(3) pp.59-62, 1997.

[17] GroupLens Research, www.grouplens.org, Last accessed on 8 May 2010.

[18] Rajatish Mukherjee, Neelima Sajja, Sandip Sen, "A Movie Recommendation System - An Application of Voting Theory in User Modeling", User Modeling and User-Adapted Interaction 13: 5-33, 2003.

[19] Oznur Kirmemis, "OPENMORE: A Content-based Movie Recommendation System", Master Thesis, Middle East Technical University, Department of Computer Engineering, May 2008.

[20] N. J. Belkin, W. B. Croft, "Information filtering and information retrieval: Two sides of the same coin?", Communications of the ACM 35, 29–39, December 1992.

[21] K. Lang, "Newsweeder: Learning to filter news", In Proceedings of the 12th International Conference on Machine Learning, Lake Tahoe, CA, pp. 331-339, 1995.

[22] Gozde Ozbal, Hilal Karaman, "Matchbook, A Web Based Recommendation System for Matchmaking", International Symposium on Computer and Information Sciences (ISCIS'08), Istanbul, Turkey, October 2008.

[23] Bruce Krulwich, "Lifestyle Finder: Intelligent User Profiling Using Large-Scale Demographic Data", AI Magazine, Vol 18, No 2, June 22, 1997.

[24] Przemyslaw Kazienko, Pawel Kolodziejski, "Personalized Integration of Recommendation Methods for E-commerce", International Journal of Computer Science & Applications Vol. 3 Issue 3, pp 12-26, 2006.

[25] Cold Start, http://en.wikipedia.org/wiki/Cold_start, Last accessed on 8 May 2010.

[26] Hugo Siles Del Castillo, "Hybrid Content-Based Collaborative-Filtering Music Recommendations", Master Thesis, University of Twente, November 2007.

[27] Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, David M. Pennock, "Methods and Metrics for Cold-Start Recommendations", In Proceedings of SIGIR-2002 Workshop, Tampere, Finland, August 11-15, 2002.

[28] Miha Grcar, Dunja Miadenic, Blaz Fortuna, Marko Grobelnik, "Data sparsity issues in the collaborative filtering framework", 7th International Workshop on Knowledge Discovery on the Web, WebKDD 2005, Chicago, IL, USA, August 21, 2005.

[29] Zan Huang, Daniel Zeng, "A Link Analysis Approach to Recommendation under Sparse Data", In Proceedings of the Tenth Americas Conference on Information Systems, New York, New York, August 2004.

[30] Manos Papagelis, Dimitris Plexousakis, Themistoklis Kutsuras, "Alleviating the Sparsity Problem of Collaborative Filtering Using Trust Inferences".

[31] Yiwen Wang, Natalia Stash, Lora Aroyo, Laura Hollink, Guus Schreiber, "Semantic Relations in Content-based Recommender Systems".

[32] Scalability, http://en.wikipedia.org/wiki/Scalability, Last accessed on 8 May 2010.

[33] Badrul M. Sarwar, George Karypis, Joseph Konstan, John Riedl, "Recommender Systems for Large-scale E-Commerce: Scalable Neighborhood Formation Using Clustering".

[34] Manos Papagelis, Ioannis Rousidis, Dimitris Plexousakis, Elias Theoharopoulos, "Incremental Collaborative Filtering for Highly-Scalable Recommendation Algorithms", ISMIS 2005, LNAI 3488, pp. 553-561, 2005.

[35] H. Ma, I. King, M. R. Lyu, "Effective missing data prediction for collaborative filtering", In Proceedings of 30th annual international ACM SIGIR conference on research and development in information retrieval, pp. 39-46, New York, 2007.

[36] Heng Luo, Changyong Niu, Ruimin Shen, Carsten Ullrich, "A collaborative filtering framework based on both local user similarity and global user similarity", In Proceedings of 2008 European Conference on Machine Learning and Knowledge Discovery in Databases, Part I, 2008.

[37] Souvik Debnath, Niloy Ganguly, Pabitra Mitra, "Feature weighting in content based recommendation system using social network analysis", WWW, 2008.

[38] Rajatish Mukherjee, Partha Sarthi Dutta, Sandip Sen, "MOVIES2GO – A new approach to online movie recommendation", In Proceedings of IJCAI Workshop on Intelligent Techniques for Web Personalization, Seattle, WA, USA, August 2001.

[39] Prem Melville, R. Mooney, and R. Nagarajan, "Content-boosted collaborative filtering", In Proceedings of SIGIR-2001 Workshop on Recommender Systems, New Orleans, LA, September 2001.

[40] SongJie Gong, "A collaborative recommender based on user information and item information", In Proceedings of the 2009 International Symposium on Information Processing (ISIP'09), Huangshan, P. R. China, August 21-23, 2009.

[41] Michael J. Pazzani, "A Framework for Collaborative, Content-Based and Demographic Filtering".

[42] R.J. Mooney, P.N. Bennett, and L. Roy, "Content-Based Book Recommending Using Learning for Text Categorization", In Proceedings of Recommender Systems Papers from 1998 Workshop, Technical Report WS-98-08, 1998.

[43] last.fm, www.last.fm, Last accessed on 8 May 2010.

[44] Last fm, http://en.wikipedia.org/wiki/Last.fm, Last accessed on 8 May 2010.

[45] Yolanda Blanco Fernández, Jose J. Pazos Arias, Alberto Gil Solla, Manuel Ramos Cabrer, "A Flexible Approach to Improve the Personalized TV by Semantic Inference", In Proceedings of Workshop on Web Personalization, Recommender Systems and Intelligent User Interfaces (WPRSIUI-05), Reading, pp. 69-79, U.K., 2005.

[46] Amazon.com, www.amazon.com, Last accessed on 8 May 2010.

[47] Linden G, Smith B, York J, "Amazon.com recommendations: item-to-item collaborative filtering", Internet Computing, IEEE, Vol. 7, No. 1, pp. 76-80, 2003.

[48] IMDb Alternate Interfaces, http://www.imdb.com/interfaces#plain, Last accessed on 8 May 2010.

[49] Google AJAX Search API, http://code.google.com/intl/tr-TR/apis/ajaxsearch/, Last accessed on 8 May 2010.

[50] Pearson product-moment correlation coefficient, http://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient, Last accessed on 8 May 2010.

[51] Floyd-Warshall's Algorithm, http://www.algorithmist.com/index.php/Floyd-Warshall's_Algorithm, Last accessed on 8 May 2010.

[52] MovieLens Data Sets, http://www.grouplens.org/node/73, Last accessed on 8 May 2010.

[53] D. Billsus, M.J.Pazzani, "Learning collaborative information filters", In Proceedings of the 15th National Conference on Artificial Intelligence, 1998.

[54] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, "Analysis of recommendation algorithms fore-commerce", In Proceedings of ACM E-Commerce, 2000.

[55] Mean Absolute Error, http://en.wikipedia.org/wiki/Mean_absolute_error, Last accessed on 8 May 2010.

[56] Precision and Recall, http://en.wikipedia.org/wiki/Precision_and_recall, Last accessed on 8 May 2010.

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

[58] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, "Item-based collaborative filtering recommendation algorithms", In Proceedings of WWW Conference, 2001.

[59] J. Wang, A. P. de Vries, M. J. Reinders, "Unifying user-based and item-based collaborative filtering approaches by similarity fusion", In Proceedings of SIGIR, 2006.