USER PREFERENCE BOOSTED CONTENT-BASED RECOMMENDER SYSTEM

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS INSTITUTE
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

TUĞÇE ÖZBERK YENER

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE
OF MASTER OF SCIENCE
IN
THE DEPARTMENT OF INFORMATION SYSTEMS

MAY 2013

USER PREFERENCE BOOSTED

CONTENT-BASED RECOMMENDER SYSTEM

Submitted by **Tuğçe ÖZBERK YENER** in partial fulfillment of the requirements for the degree of **Master of Science in Information Systems**, **Middle East Technical University** by,

Prof. Dr. Nazife Baykal ______________________
Director, **Informatics Institute**

Prof. Dr. Yasemin Yardımcı Çetin ______________________
Head of Department, **Information Systems**

Assoc. Prof. Dr. Sevgi Özkan ______________________
Supervisor, **Informatics Institute, METU**

Examining Committee Members:

Assist. Prof. Dr. Erhan Eren ______________________
Informatics Institute, METU

Assoc. Prof. Dr. Sevgi Özkan ______________________
Informatics Institute, METU

Assist. Prof. Dr. Murat Perit Çakır ______________________
Informatics Institute, METU

Assist. Prof. Dr. Banu Günel ______________________
Informatics Institute, METU

Dr. Yavuz İnal ______________________
YTE, TUBITAK

Date: **08/05/2013**

Name, Last Name	**:**	**Tuğçe Özberk Yener**
Signature	**:**	

ABSTRACT

USER PREFERENCE BOOSTED

CONTENT-BASED RECOMMENDER SYSTEM

Özberk Yener, Tuğçe

M.S., Department of Information Systems

Supervisor: Assoc. Prof. Dr. Sevgi Özkan

May 2013, 100 pages

In the world of information, internet becomes the most important information source. However, internet contains vast amount of information and this information is not filtered. In such an environment, the people who seek for an information is overwhelmed in the alternatives that s/he can reach via the web. Recommender systems have their real importance in this kind of situations. To overcome said overwhelming problems, recommender systems are developed to determine the people needs and to recommend suitable alternatives to them.

The current recommendation methods are classified under three main categories: collaborative filtering, content-based and hybrid approaches. Classical content-based recommendation approaches include the content information of the items. In this thesis work, we propose a user preference boosted content-based recommendation methodology. In addition to the items content information, we aimed to define a novel approach to the problem of including user's preference information to the information filtering process. The novel solution that we explained in this thesis work uses the users past like and dislike rate information related to the specific items to predict recommendation scores related to the unseen items.

The results which we obtained by implementing the proposed user preference boosted content based recommendation approach indicates that; by including the users' preference information to the items content information more accurate recommendations can be done and more reliable results can be gathered. We present the implementation details and comparative evaluation results of the proposed novel approach in this thesis.

Keywords: Recommender Systems, Content-Based Recommendation, User Preference Extraction

ÖZ

KULLANICI TERCİHİ DESTEKLİ

İÇERİK TABANLI TAVSİYE SİSTEMİ

Özberk Yener, Tuğçe

Yüksek Lisans, Bilişim Sistemleri Bölümü

Tez Yöneticisi: Doç. Dr. Sevgi Özkan

Mayıs 2013, 100 sayfa

Günümüz bilgi dünyasında, internet en önemli bilgi kaynağı haline gelmiştir. Ancak, internetteki bilgi miktarı çok büyük boyutlara ulaşmıştır ve bu bilgi filtrelenmemiştir. Böyle bir çevrede, bilgiye ulaşmak isteyen insanlar WWW aracılığıyla ulaştıkları fazla bilgi karşısında çaresiz duruma düşmektedirler. Bu tür durumlar karşısında tavsiye sistemlerinin esas önemi ortaya çıkmaktadır. Bahsedilen bilgi kirliliği karşısındaki çaresizlik problemlerinin üstesinden gelmek için, insanların ihtiyaçlarını ayırt etmek ve uygun alternatifleri kullanıcılara sunmak üzere tavsiye sistemleri geliştirilmiştir.

Mevcut tavsiye sistemi metotları üç ana kategori altında gruplanmıştır: işbirliği ile filtreleme, içerik-tabanlı ve melez yaklaşımlar. Klasik içerik tabanlı tavsiye sistemleri, tavsiye edilecek nesnelerin içerik bilgilerini kullanmaktadırlar. Bu tez çalışmasında, kullanıcı tercih bilgisi destekli içerik tabanlı bir tavsiye sistemi metodolojisi önerilmektedir. Nesnelerin içerik bilgilerine ek olarak, bilgi filtreleme sürecine kullanıcıların tercih bilgilerinin de katılacağı yeni bir yaklaşım tanımlamak hedeflenmiştir. Bu tez çalışmasında açıklanan yeni çözüm yöntemi, kullanıcının görmemiş olduğu nesnelere ait tahmin edilecek tavsiye skorlarını hesaplamak için kullanıcıların belirli nesneler ile ilgili geçmiş hoşlanma ve hoşlanmama oranı bilgilerini kullanmaktadır.

Önerilen kullanıcı tercih bilgisi destekli içerik tabanlı tavsiye yönteminin uygulanması ile elde edilen sonuçlar göstermektedir ki; kullanıcının tercih bilgisinin nesnelerin içerik bilgilerine eklenmesi yolu ile elde edilen sonuçlar, kullanıcının tercih bilgilerinin dahil edilmediği yönteme göre daha kesin tavsiyeler üretebilmektedir ve daha güvenilir sonuçlar elde edilebilmektedir. Bu tez çalışması ile önerilen yeni yaklaşımın uygulama detayları ve karşılaştırmalı değerlendirme sonuçları sunulmuştur.

Anahtar sözcükler: Tavsiye Sistemleri, İçerik Tabanlı Tavsiye, Kullanıcı Tercih Bilgisi Çıkarma

To my patient and full of love husband

ACKNOWLEDGMENTS

First of all, I would like to thank my supervisor Assoc. Prof. Dr. Sevgi Özkan for her encouraging support and guidance in this thesis study.

My biggest thanks to my husband who listens to me every day and supports me whatever I did for everything he made for me.

I would also like to thank to my mother and father who motivate me whenever I sink into despair and make the world better for me throughout my whole life.

I want to thank Burak Şimşek who is one of the "cineworm.com"s owners and who shares the dataset with me for using their data in my thesis work.

I want to thank my colleagues in TUBITAK and my friends who support me while I was studying on this thesis work.

I also would like to express my gratitude to the examining committee members of my thesis for allocating their time for reviewing my thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

WWW	World Wide Web
TV	Television
CD	Compact Disc
PC	Personal Computer
JSON	JavaScript Object Notation
TF-IDF	Term Frequency - Inverse Document Frequency
SVD	Singular Value Decomposition
GIS	Geographic Information Systems
IMDB	Internet Movie Database

CHAPTER 1

INTRODUCTION

The world became an information world via the broad access of the internet. People use internet from everywhere by wireless networks via their laptops, smart phones and even with their smart TVs to access any kind of information; such as watching videos, downloading songs, finding relevant books or finding movies to watch. This much of popularity, accessibility and the broad usage of internet carry the information overload problem with it. Everyday more and more information is uploaded to the internet and it's getting harder to find the most relevant items in the internet. People need help to find the reliable information and to extract the information which s/he is really interested in. Finding information which meet people's requirements in a timely manner becomes a challenging task in the world of growing information overload problems. Recommender systems emerged to cope with these kinds of problems by extracting the relevant information for users over the huge amount of alternatives. Some recommender system definitions from literature are given below:

- Recommender systems are software tools and techniques that are used to recommend items to users. Item is the general term which is used to define the things (movie, music, restaurant, news etc.) which are recommended to users by recommenders [65].
- Recommender systems are tools which are developed to help people for finding suitable content, product or services (such as; books, digital products,

movies, music, TV programs, web sites) which suit to their personal preference [55].

- Recommender systems are accepted as powerful tools which helps users in a personalized way to find relevant items through the huge amount of possible options [12].
- Recommender systems are the tools which help people to find relevant items from enormous number of alternatives by extracting user preferences from the information that user provides either explicitly or implicitly [4].

The roots of the recommender systems can be accepted as cognitive science [66], approximation theory [62] and information retrieval [69] fields. However, the very first papers which are specifically related to recommender systems appear in the mid-1990s [30, 64, 74] and this research area have considered as an important research area afterwards [1]. Usefulness, interestingness and personalization are the keywords which separate recommender systems from information retrieval systems [12].

Some popular web-based systems have used recommendation approaches to recommend related items to users. MovieLens [49] and Netflix [54] for recommending movies, Amazon.com [39] for recommending books - CDs and other products, VERSIFI Technologies [7] for recommending news, Last.fm [38] and Grooveshark [25] for recommending music, YouTube [80] for recommending videos, can be considered as examples of the developed applications of recommender systems. Even ScienceDirect [73] recommends articles related to the current read one to its users.

Because of the fact that people need personalized recommendations to deal with the information overload problems; there are a lot of approaches which are developed to serve as novel applications for recommending items to users [1]. There are three main recommendation approaches in the literature [31]. First one is collaborative filtering approach, second one is content-based recommendation approach and the third one is knowledge-based recommendation approach. Hybrid recommenders can be accepted as the fourth approach which combines several different techniques to generate recommendations.

In this thesis work we propose a novel content-based recommendation approach which generates recommendations related to movies. In the literature, content-based recommenders rely on to the content information of items and the user's past rating history. The content information of the items are represented via attributes. Such as; actor, director, genre and release year information for movies can construct an attribute set for each movie. Each attribute in the existing content-based systems has a weight value which reflects the attribute's importance level over the item space. In other words, the weight values of the item attributes reflect the attribute's distinctiveness over the existing item space. We believe that, in order to generate more personalized recommendations; in addition to the attribute weight values, the attribute's distinctiveness in the eyes of the user should also be considered in the recommendation process. The main contribution of this thesis work is adding the user's opinion related to the attribute's distinctiveness with a novel algorithm by mining the users past rating history and increasing the recommendation accuracy according to the classical content-based recommendation approaches.

This thesis consists of 5 chapters. The remaining 4 chapter is organized as follows.

In Chapter 2, the literature review related to recommender systems are presented. First, the formalization of recommendation problem is given. After, the input data types, application domains and user modeling approaches of recommenders are classified. Finally, main recommendation approaches and the algorithms that are used in these different approaches with their advantages and disadvantages are explained.

In Chapter 3, the proposed novel content-based recommendation approach is explained in a detailed manner. First, the motivation of the proposed approach and the overview of the proposed system is given. After that, the system architecture and the implementation details which rely on the proposed novel algorithm are described. Implementation details which are explained in this section include item and user profile generation processes and calculation of the predicted rating scores according to the proposed novel approach.

In Chapter 4, the evaluation of the proposed approach and the results of the experiments are given. First of all, the data characteristics which are used to evaluate the proposed approach is given. After that, the evaluation metrics are explained. Finally, the experiment results are presented and discussions related to the obtained results are given.

In Chapter 5, the conclusion of the thesis work is given and possible future work related to the proposed approach is mentioned.

CHAPTER 2

RECOMMENDER SYSTEMS

This chapter describes the main issues related to the recommender systems. First, the formal definition of recommendation problem is given. After that, the input data types, application domains and user modeling approaches of recommenders are classified. Finally, recommendation techniques and the algorithms that are used in these techniques are given with their advantages and disadvantages.

2.1 Formal Definition of Recommendation Problem

The recommendation problem can be degraded to the problem of predicting the possible rating score related to an item which user has not seen yet [1]. After predicting the rating scores of the unseen items, the recommender system recommends items which have the highest rating scores to the user [1].

The formal definition of the recommendation problem is [1]: Let U be the set of all users and let I be the all possible items that can be recommended to users. Let f be the satisfaction function which measures the usefulness of item i to user u:

$$f : U \times I \rightarrow R \tag{2.1}$$

where R is an ordered set which can be non-negative integers or real numbers. Then for each user u $\in$ U, the recommender system aims to find item i' $\in$ I which maximizes the user's satisfaction [1]:

$$\forall u \in \mathrm{U},\ \mathrm{i} \in \mathrm{I},\ i'_{\mathrm{u}} = \arg\max \mathrm{f}(\mathrm{u}, \mathrm{i}) \tag{2.2}$$

The users and the items in the recommendation systems can be represented with profiles. The profile which represents the users can include user characteristics such as age, gender, income, location and marital status [1]. The profile which represents the items can include item characteristics related to the item's domain [1]. For instance; if the items are movies; the profile of a movie can include the title, genre, actor, director and release year attributes of movies.

The satisfaction of the users for the items is generally represented by rating scores in recommender systems. Initially, only the rating scores which are already defined by users related to items is known by recommender systems. The recommender system should predict the unseen items' rating scores by using the previous rating scores of the already seen items. An example user-item rating matrix for movie domain is given in Table 2.1. The ratings are on the scale of 1-10 and the "-" character represents the rating related to the movie is not given by the user yet. Thus, the recommender system's task is predicting the unknown user-movie pairs' rating scores in this table.

Table 2.1: An Example Movie - User Rating Matrix

Movie/User	**User1**	**User2**	**User3**	**User4**	**User5**	**User6**
Movie1	6	-	7	-	6	7
Movie2	7	8	-	9	7	-
Movie3	-	7	6	-	-	8
Movie4	8	5	8	6	8	6
Movie5	4	-	-	-	5	-
Movie6	-	7	-	-	7	-

After predicting the rating scores which are labeled with "-" character in the given table, the recommender system will be able to recommend items to users according to the predicted rating scores. An example user-item rating matrix after predicting the unknown rating scores is given in Table 2.2.

Table 2.2: An Example Movie - User Rating Matrix After Rating Prediction

Movie/User	**User1**	**User2**	**User3**	**User4**	**User5**	**User6**
Movie1	6	**6.8**	7	**8.3**	6	7
Movie2	7	8	**7.3**	9	7	**6.6**
Movie3	**7.2**	7	6	**7.1**	**6.5**	8
Movie4	8	5	8	6	8	6
Movie5	4	**6.4**	**5.6**	**4.2**	5	**7.4**
Movie6	**5.4**	7	**6.2**	**5.6**	7	**6.8**

After generating the Table 2.2, the recommender can propose items to users by using two different techniques. One of them is recommending the items which has the highest predicted rating scores [2]. Suppose that the active user (the user who the recommendations are generated for) in our system is User4 and the threshold predicted rating score is defined as 5.0. This means that the items which have predicted rating scores more than 5.0 will be recommended to the user by the recommender. Then, the system recommends Movie1, Movie3 and Movie6 to User4. The second approach that can be used to generate recommendation list is called top-N recommendation algorithm [35, 4]. In this approach, the N represents the number of items that are going to be recommended to user which have the highest predicted rating scores. Suppose that in our example the number N is defined as 2, then the recommender represents a recommendation list which consists of Movie1 and Movie3 to User4. In this thesis work, we mainly focus on developing a novel approach to generate predicted rating scores related to unseen items rather than preparing the recommendation lists which are going to be shown to the users.

2.2 Input Data Types of Recommender Systems

The recommender systems use three types of input data to generate recommendations in any type of recommender system [65]. The output of a recommender system can be in the form of a recommendation list or a prediction score [10]. The input data types of recommender systems are described in the subsections of this section.

2.2.1. Item

Items are objects which are recommended to users by recommender systems. Items that are used in recommender systems can be divided into two categories [65].

- Items with low complexity and cost
- Items with high complexity and cost

The cost which is mentioned here can also be the time period to search the item or it can be the monetary cost to buy that item. Movies, books, CDs and news are the examples of items which have low complexity and cost. Digital cameras, mobile phones, PCs, financial investments and travels are the examples of items which have high complexity and cost [51].

2.2.2. User

The person who gets recommendations from system is called as the user of the system. Since the recommender systems are personalized systems, user information is one of the most important information that recommenders use. User information may include the user's preference information related to items and it may also include the relation information between users such as; trust levels of users [65].

2.2.3. Transactions

Transactions can be defined as the interactions between the users and the recommender system such as; the rating information related to an item given by user [65]. There are four types of transactions that are used in recommender systems [72]:

- Numerical Ratings (Such as; 1-5 stars and 1-10 scale)
- Ordinal Ratings (Such as; "strongly agree, agree, neutral, disagree, strongly disagree")
- Binary Ratings (Such as; "good, bad" and "like, dislike")
- Unary Ratings (Defines if the user has observed/purchased the item or not)

2.3 Application Domains of Recommender Systems

The application domains which are used in recommender systems are mainly grouped under four main categories [65, 51, 55] :

- Entertainment (Such as; movies, TV programs and music)
- Content (Such as; newspapers, documents and web pages)
- E-commerce (Such as; books, cameras and PCs)
- Service (Such as; traveling, consultation and house rental)

2.4 User Modeling In Recommender Systems

Recommender systems serve personalized recommendations to its users. To do this, they need to know the user preferences [65]. To gather user preferences and generate user models, recommenders can use two different techniques: implicit user modeling or explicit user modeling [66, 65, 27, 31, 41].

In explicit user modeling technique, the user needs to form his/her model himself/herself. There are three main approaches to get explicit preferences from users [41]:

- Like / Dislike
- Ratings
- Text comments

Because of the need for user effort, explicit user modeling is less preferable by users [66]. But users are more willing to express their opinions after the Web 2.0 technologies widely acceptance and this type of user modeling is more accurate than the implicit one [31]. [9] uses this type of user modeling in their content-based music recommender system.

In implicit user modeling technique, the user model is constructed by the system implicitly by interpreting the user behaviors [66, 65, 31]. Such as; if the user looks at a product's web page more than 5 seconds this means that the user is interested in this product and this information can be added to the user's preference data set [65] or if user buys a product, this behavior can be accepted as a positive rating for that item [31]. In Grundy system [66], implicit user modeling technique is used.

2.5 Recommendation Techniques

Recommender systems are classified according to the approach which they use while estimating the unknown ratings [1]. There are three main recommendation approaches [31]. These are collaborative filtering, content-based recommendation and knowledge-based recommendation. In addition to them, some hybrid approaches are also used which combines these techniques. In this section the details of recommendation techniques are presented.

2.5.1. Collaborative Filtering Recommenders

In everyday life, people take daily decisions according to the recommendations that are received from others [42, 46]. For example; before watching a movie people generally read comments related to that movie and before reading a book people generally ask their friends' comments related to that book. The first recommender systems are the automated versions of this "ask to friend" behavior [65]. This is called as collaborative filtering recommendation and this technique relies on that the people who share the same opinions in the past will share the same opinions in the future [65].

In collaborative filtering recommender systems, the satisfaction of a user related to an unseen item is predicted by using the rating history of this item which is generated by other users. More formally; the satisfaction function f(u, i) of item i for user u is predicted by using the satisfaction results of f(u_j, i) of users who also belong to the user set U and "similar" to user u [1]. The similar users are found in the user set U by comparing the rating history related to users. In collaborative filtering, the users who rated same items similarly are accepted as similar users, in other words neighbors. After finding the neighbors of the user, the items which are rated highly by neighbors and haven't seen by the user yet are found by the system and recommended to the user [1].

The term "collaborative filtering" is first used to describe Tapestry system [21] which is an electronic document filtering system. In Tapestry system the user needs to write complex queries to get recommendations. After Tapestry system, the first automatic recommender system is accepted as GroupLens [64] which recommends articles to users [4]. Other well-known collaborative filtering recommender systems can be counted as Ringo system [74] to recommend music albums/artists, Video Recommender [30] to recommend videos, Amazon.com [39] to recommend books and Jester system [22] to recommend jokes.

In collaborative filtering recommender systems the recommendations are done solely by the rating information of other users. No content information related to items are used in pure collaborative filtering systems [5]. So, the application domain is not important for this type of recommenders. Collaborative filtering recommenders can recommend any kind of item to its users including music and video [61]. The algorithms that are used in collaborative filtering recommenders are grouped under two main categories [11, 31]:

- Heuristic-Based (Memory-Based) Algorithms
- Model-Based Algorithms

The main difference between the heuristic-based approaches and model-based approaches is; model-based approaches use models that are learned by using statistical and machine learning algorithms but heuristic-based approaches use heuristic rules to calculate satisfaction scores of items according to users [1].

2.5.1.1. Heuristic-Based (Memory-Based) Algorithms

Heuristic-based algorithms are also known as memory-based algorithms and they use all of the previously rated items to predict rating scores [11, 17, 64, 74]. In other words, the whole user-item rating matrix is included to the recommendation process to generate recommendations [31]. Heuristic-based approaches are also divided into two sub categories [10]: user-based [11, 29, 33] and item-based [18, 39, 70] approaches.

a. User-Based Approach

In user-based approach, the system first finds the similar users related to a user which shares the same opinions with the target user in the past according to the user-item rating matrix. After finding the similar users, the estimated rating score is calculated via the rating scores of the similar users. This type of recommendation has made two assumptions: users who agree in the past will agree in the future and user preferences don't change over time [31].

The recommendation process of a user-based approach consists of three steps [18]:

- Create user profiles and find neighbor users
    - User profile creation can either be done by gathering implicit or explicit information related to users.
    - To find the neighbor users, k-Nearest neighbor algorithm is the most widely used approach [10, 18]. The number k indicates the number of users which are chosen as neighbor users according to their similarity scores with the target user. The analysis which are done on MovieLens dataset shows that; the neighbor set size between 20-50 is acceptable for real-world applications [28]. The number below this causes a limited prediction and decreases the accuracy. The number above this causes too much noise in the neighbor user set [31].
    - Threshold approach can also be used to find the neighbor users [10]. In threshold approach a similarity threshold value is determined and used to decide whether a user is a neighbor or not. The users who have a similarity score more than the threshold value are considered as neighbors.
- Combine the items that neighbor users have selected and calculate the predicted rating scores of those items according to the neighbor user set's rating scores.
- Recommend items to the target user whose predicted rating scores are found higher than the others and have not seen by the target user yet.

b. Item-Based Approach

In item-based approach the similarities between items are taken into account to generate recommendations instead of user similarities and indirectly the user profiles are constructed based on these item similarities [4, 10, 31]. When the number of users and items are huge, than item-based approach is preferred to user-based approach because of the computational complexity problem of user-based approach when the user/item count is large [31].

Item-Based approaches are generated to solve the sparsity and scalability problems of user-based algorithms [4]. The recommendation process of an item-based approach consists of two steps [4]:

- Calculate the similarity between items according to the users past preference history.
- Select most similar items with the particular item that the user needs a recommendation.

Item-based collaborative filtering recommendation approach's most popular implementation is Amazon.com [39] recommender system. This system first matches the items which are already purchased by the user to the similar items in the store. Then the system recommends items from those similar item set according to the generated similarity measurement [39].

c. Similarity Calculation Algorithms

In both of the user-based and item-based collaborative filtering recommender systems, the similarities between users and the similarities between items are calculated by some well-known similarity calculation algorithms. In this section, the similarity calculation algorithms which are widely used are explained.

The similarities for numerical values can be calculated by using Euclidean (Equation 2.3) and Manhattan (Equation 2.4) distance functions [4].

$$d(i,j) = \sqrt{(x_{i1} - x_{j1})^2 + \dots + (x_{in} - x_{jn})^2} \tag{2.3}$$

$$d(i,j) = |x_{i1} - x_{j1}| + \ldots + |x_{in} - x_{jn}| \tag{2.4}$$

In these distance functions the differences between attribute values are considered to find similarities. However, for the attributes whose values are not numerical such as categorical like colors, these functions cannot be used for calculating similarities [4]. To calculate similarities between items whose attributes are not numerical, the most popular approaches that are used in collaborative recommenders are Pearson correlation (Equation 2.5) and cosine-based (Equation 2.6) similarity calculations [1, 31].

$$sim(x,y) = \frac{\sum_{s \in S_{xy}} (r_{x,s} - \bar{r}_x)(r_{y,s} - \bar{r}_y)}{\sqrt{\sum_{s \in S_{xy}} (r_{x,s} - \bar{r}_x)^2 \sum_{s \in S_{xy}} (r_{y,s} - \bar{r}_y)^2}} \tag{2.5}$$

$$sim(x,y) = \cos(\vec{x},\vec{y}) = \frac{\vec{x}.\vec{y}}{||\vec{x}||_2 \times ||\vec{y}||_2} = \frac{\sum_{s \in S_{xy}} r_{x,s} r_{y,s}}{\sqrt{\sum_{s \in S_{xy}} r_{x,s}^2} \sqrt{\sum_{s \in S_{xy}} r_{y,s}^2}} \tag{2.6}$$

where S_{xy} represents the commonly rated item set by users x and y. $r_{x,s}$ represents the rating score given to item s by user x. $\bar{r}_x$ represents the average rating score given by user x to all rated items.

The result of the Pearson correlation coefficient is in the range of [-1,1], where 1 means a strong positive correlation and -1 means a strong negative correlation [31]. In Pearson correlation, the user rating scale interpretation is taken into account. Some users give relatively low ratings than some other users who give relatively high ratings. By using the user's average rating score in the equation, this interpretation difference between users is also considered while calculating the similarities [31].

It is seen from the analysis that the Pearson correlation coefficient generates better results than other techniques for user-based collaborative filtering recommender systems [29]. Some popular recommenders such as GroupLens [64] and Video Recommender [30] use Pearson correlation coefficient to calculate similarities.

In cosine-based similarity calculation approach [11, 70], the users x and y is considered as vectors. These vectors' dimension count equals to the number of items which are rated by both of the users. After describing the vectors, the cosine of the angle between these two vectors are used to calculate the similarity between users (Equation 2.6).

The result of cosine-based approach is between [0,1] and 1 means strong similarity [31]. To take the user's rating scale interpretation into account like in the Pearson correlation, adjusted cosine similarity measurement (Equation 2.7) technique can be used [31].

$$sim(x,y) = \frac{\sum_{s \in S_{xy}} (r_{x,s} - \bar{r}_x)(r_{y,s} - \bar{r}_y)}{\sqrt{\sum_{s \in S_{xy}} (r_{x,s} - \bar{r}_x)^2} \sqrt{\sum_{s \in S_{xy}} (r_{y,s} - \bar{r}_y)^2}} \quad (2.7)$$

The results of adjusted cosine similarity measurement is between [-1,1] like in the Pearson correlation coefficient measurement [31].

In item-based recommender systems the best approach to calculate item similarities is cosine similarity measurement technique to generate most accurate recommendation results [31].

2.5.1.2. Model-Based Algorithms

To recommend items, model-based algorithms learn a predictive model by using user-item rating matrix [10, 1]. To learn the model, some offline preprocessing is done in this type of recommenders. After learning the model, this model is used to generate recommendation scores [31]. Model-based algorithms can be divided into three main categories [31]:

- **Matrix Factorization/Latent Factor Models**

In this type of approach, hidden (latent) factors are extracted from rating patterns of users. Both items and users are represented as factor vectors and the similarities between users and items are calculated according to the similarities of these vectors [37].

To extract hidden factors, Singular Value Decomposition (SVD) method can be used [16]. SVD is a matrix factorization method which is used to calculate the user-item rating matrix's best lower rank approximation values [71]. In SVD method, the highly correlated and co-occurring terms are combined together into a single factor to generate smaller vectors for items. Thus the related terms are considered as a single term and more accurate recommendations can be generated.

- **Association Rule Mining**

In this type of recommendation technique, the relationship patterns are determined to recommend items [31]. A group of items which are generally purchased together can be considered as a rule. Such as; "the people who buy a shampoo generally buy a shower-gel in the same shopping" can be found as a rule. After determining the rules, the recommendations are done by applying the rules. The results of the rules are combined and the recommendation is done from this union according to the rules' confidence level [31].

- **Probabilistic Recommendation Approaches**

In this type of recommendation technique, the prediction problem is considered as a classification problem and the recommendation task is considered as learning the classification model from historical information and using the constructed model to generate recommendations [31].

The probabilities related to all rating scores is calculated for an unseen item according to the user's past rating history. Then, the rating score which has the highest probability is accepted as the predicted rating score for that unseen item. The rating scores can be reduced to the binary scale such as; like / dislike according to a threshold rating score. One of the most known technique which is used in probabilistic recommendation approach is naive Bayes classifiers [31]. In naive Bayes classifiers the item attributes are considered as independent from each other. In addition to this, Bayesian networks and k-means clustering algorithm can also be used in probabilistic recommendation approach [31].

In clustering algorithm, the users who share the same opinions are clustered to a number of classes [11]. The class number is learned from the data itself. In this approach, the ratings of the users in the same clusters are considered as independent like in naive Bayesian model.

In Bayesian network, each node in the network has a decision tree and the edges of each tree represents a user information [79]. This approach learns a model by using a training data [79]. Also each node has a state which indicates the estimated rating score related to node [11]. The used data determines the structure of the network. The most important limitation of this approach is a user can be assigned to only one cluster [11]. This restriction limits the capability of benefiting from different clusters for users. For example; a user may belong to a class according to the preferred books for work and belongs to another class for personnel readings [11].

2.5.1.3. Limitations Of Collaborative Filtering Recommenders

The main limitations of collaborative filtering recommenders are new user, new item, sparsity and black sheep problems [5, 1].

New User Problem

If a user doesn't rate enough number of items, the preferences of that user cannot be extracted by the recommender system and relevant items won't be recommended to this user by the recommender system [1]. To solve the problem of recommending suitable items to new users, some strategies are used to determine the items which are more discriminative to a recommender system while learning a user's preference and new users are encouraged to rate those discriminative items [63, 81]. These strategies are based on item popularity, item entropy and user personalization [63, 81].

Ringo [74] is a web-based collaborative filtering recommender system which recommends music albums and artists to its users by using social information filtering. To overcome the new user problem; Ringo system asks its new users to rate the most descriptive 125 artists by sending this selected artist to new users. Descriptive artist list is prepared in two parts: most often rated artists and random selection of artists from database [74].

New Item Problem

In collaborative filtering recommender systems, the item's content attributes are not considered while making recommendations. Because of this, a new item won't be recommended to any user by the recommender system before enough number of users rate this item. To solve this problem, hybrid recommender systems are used which combines the content-based and collaborative filtering recommendation techniques [1].

Sparsity Problem

In recommender systems, the user-item rating matrix consists of predicted rating scores and user defined rating scores. In general, when we compare the user defined rating score number with the predicted rating score number, the predicted rating score number is considerably more than the number of user defined rating scores [1]. This is called sparse user-item rating matrix problem [5, 1].

Black Sheep Problem

The users whose tastes are unusual according to the users in the system, then this user cannot get useful recommendations from the collaborative recommender system [5]. Because there is no neighbor for this user who is near enough to gather accurate recommendations.

2.5.2. Content-Based Recommenders

In content-based recommendation, the recommendation scores for unseen items are predicted according to the "similar" items which are already rated by the user [1]. The items whose characteristics most suit to the user profile are chosen to recommend in content-based recommenders [31]. For example; in a movie recommender system; the recommender should find the commonalities such as common genres between the movies which are rated positively by the user and recommend movies which has high similarity with the movies that suit to the user preferences [1].

The roots of the content-based recommendation methodology reach to information retrieval [69] and information filtering areas [5]. Content-based recommendation systems generally focus on items which contain textual information. The primary reason behind this is the importance of the several text-based applications in the information retrieval and filtering communities [1]. The usage of the user profiles which keep the user needs in recommender systems can be considered as the main improvement over the traditional information retrieval approaches [1].

Grundy system [66] is accepted as the first recommender system in literature [1]. This system builds user models by using stereotypes with a small amount of information related to users for recommending books. [66] proposes that stereotypes are useful mechanisms to build user models when the information related to users are comparatively small. Stereotype means a cluster of characteristics. Because the stereotypes are cognitive information, this ancestor recommender system is considered as a content-based recommender system [66].

[43] proposes an automated message filtering system named Information Lens in a community to solve the information sharing problem. In this system, messages are characterized by their contents and then according to this content information messages are matched to its receivers. This approach can be considered as one of the ancestors of content-based recommender systems [43].

Some current recommender systems include semantic representations to their content-based recommender systems. [9] includes semantic representations of audio contents to overcome the difficulty of representing audio items with content information. By the usage of content-based approach they solve the cold-start problem of collaborative filtering approach which is the generally used approach when the domain is related to audio items. [53] also include semantics to the recommendation approach to improve the quality of predictions.

The basic process which content-based recommender systems perform is matching the attributes which are stored in user profiles to represent the user preferences with the attributes of the items which user has not seen yet to recommend new items to users [41].

In content-based recommenders, recommendation is done via three interacting modules: content analyzer, profile learner and rating predictor modules [41, 10]. (Figure 2.1) represents a basic content-based recommender system module diagram.

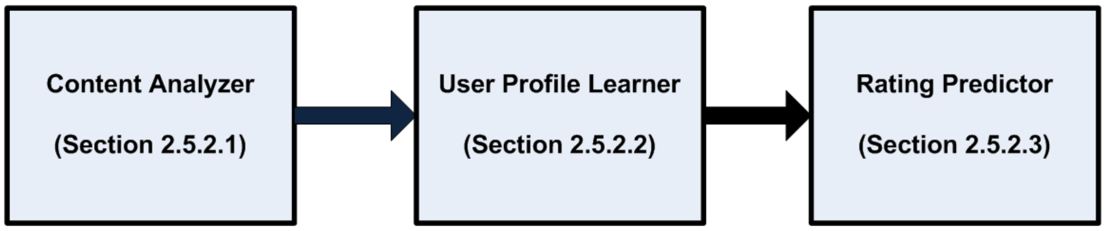


Figure 2.1: Content-Based Recommender Module Diagram

2.5.2.1. Content Analysis In Content-Based Recommenders

Content analysis step in content-based recommenders includes item representations. In content-based recommender systems, items have profiles [41]. In those item profiles, attributes related to items are kept. Items in content-based recommenders are generally stored in a database table [58]. (Table 2.3) shows an example item profile. The columns in that database table represents the item attributes. The rows represent the items. This database table is an example of structured data [58]. This means that each item in the database is represented with the same set of attributes and there is a known set of values which the attributes can hold [58].

Table 2.3 : Example Item Profile

ID	Name	Genre	Author
1001	Star Wars: Episode IV - A New Hope	Adventure	George Lucas
1002	Apocalypse Now	Drama	Joseph Conrad
1003	Armageddon	Action	Robert Roy Pool

Items can be in an unstructured data form such as news articles. A common approach to deal with unstructured data is converting unstructured data to structured data [58]. When a new item is going to be added to the content-based recommender system, the content information related to that item is analyzed by feature extraction techniques to generate the item's profile [41]. By extracting item attributes, items are converted to a suitable form which can then be used in recommendation process. In other words, items are formed into a recommender system readable form.

a. Vector Space Model

In content-based recommenders, generally the item representations are done through keyword-based Vector Space Model [41, 31]. Let items that are recommended in a recommender system be documents. In vector space model, each document is represented as a vector in a n-dimensional space [41, 31]. Each dimension in that space represents a term which belongs to the term set which consists of all terms of the document collection in the recommender system [41]. The dimensions of each vector represents the term's weight value for the document which the vector represents [41]. A weight value for a term-document pair represents the relationship between term and document. Let D = {d1, d2, ... , dN} shows the set of documents in the system, T = {t1, t2, ... , tn} represents all terms in all documents [41]. The set T is generated by using natural language processing operations such as tokenization, stop-words removal and stemming [41]. The document representation as a vector in a n-dimensional space then represented as; $d_j = \{w_{1j}, w_{2j}, ... , w_{nj}\}$, where w_{kj} shows the weight value of term t_k for document d_j [41].

In vector space model, to calculate the term weight values, the most well-known measurement technique in Information Retrieval is the Term Frequency - Inverse Document Frequency (TF-IDF) weighting technique [69, 41]. This technique is based on some assumptions [41];

- If the term occurs too frequently in a document, this shows that the term is closely related to that document. (TF assumption)
- If a term occurs in most of the documents, this means that the term is not a descriptive term for any document. (IDF assumption)
- The length of a document is not counted as a preference criteria. In other words, if a document is longer than the other one, this situation should not

affect the comparison result for these documents. (Normalization assumption)

Term frequency value is calculated by only considering the current document. This value is defined as the ratio of the term's frequency in the document to the maximum frequency value of all terms inside the document [1, 31]. More formally;

$$TF_{t_k,d_j} = \frac{\mathrm{f_{k,j}}}{\mathrm{max_z f_{z,j}}} \tag{2.8}$$

, where $\mathrm{f_{k,j}}$ is the number of times that term t_k appears in document d_j and $\mathrm{max_z f_{z,j}}$ value is found over all term frequencies of document dj.

Inverse document frequency value represents the term's occurrence frequency over all documents. Inverse document frequency value is added to the term frequency measurement to reduce the importance level of a term which appears in most of the documents [31]. This means that if a term t_k appears at fewer number of documents, this means that term t_k is more distinguishing than a term which appears at most of the documents [1]. More formally;

$$IDF_{t_k} = \frac{\log \mathrm{N}}{\mathrm{n_k}} \tag{2.9}$$

, where N represents the number of documents in the system and $\mathrm{n_k}$ shows the number of documents which includes the term t_k. So, the TF-IDF weight of a term t_k in document d_j is defined as follows:

$$TF - IDF(t_k, d_j) = w_{t_k,d_j} = TF_{t_k,d_j} \times IDF_{t_k} \tag{2.10}$$

Finally, the content of the document d_j, in other words; the profile of the document is represented as [1];

$$Content(d_j) = (w_{t_1,d_j}, \ldots, w_{t_n,d_j}) \tag{2.11}$$

To improve the vector space model some techniques are used in literature [31]:

- **Stop words**: In this technique, the so-called terms are removed from the term list to reduce vector dimensions. Such as; "a", "the", "on".

- **Stemming**: In this technique, only the roots of the terms are used as dimensions. For example the root "compute" is used for "computer", "computing" and "computation" terms and only one dimension is used for all these terms. The main goal of stemming is to find the root of the terms and count each similar term in the same term's category [58].
- **Size cutoffs**: In this technique, only the n most descriptive terms of a document is used to represent the document. For instance; Fab system [5] which recommends web pages to users uses 100 most descriptive words related to the web pages in their profiles. In Syskill & Webert system [56] the most important 128 words related to documents generates the item profiles.
- **Phrases**: In this technique, phrases such as "United Nations" are considered as single term and accepted as a single dimension. To detect phrases in a document, manuals can be used or statistical analysis techniques can be applied to documents.

Item profiles are the inputs of both the user profile learner and rating predictor modules [41].

2.5.2.2. User Profile Learning In Content-Based Recommenders

In content-based recommender systems both the users and the items have their own profiles [1]. The user profiles are constructed with the same attribute set with the item representations which the user has rated in the past and represents the user's past preference information [50, 41, 31]. The user profiles are generated by mining the user's past rating history related to the already seen items [1]. In other words, the item profiles and the user feedbacks (such as; ratings) are used to learn the user models [41, 10].

User profiles can be thought as a vector of weights (w_1,w_2,....,w_n) and each weight value indicates the user's preference information related to the content attribute which items in the recommendation process consist of [1]. Such as; title, genre and director are content attributes for movies in movie domain. There are various different techniques from information retrieval which are used to calculate the content attributes' weight values for generating user profiles [1]:

a. Relevance Feedback and Rocchio's Algorithm [5]

Relevance feedback method is a method which is inside the area of Information Retrieval field [41]. Roccio's relevance feedback method is developed for the information retrieval system named SMART [67] in which the users define queries for retrieving information. The queries are improved according to the user's feedback related to query results. The user gives feedback related to the documents which the query returns about the accordance of the documents with his/her preferences [41].

The Roccio's algorithm [67] is used commonly in content-based recommender systems. Users give ratings to the documents that the recommender system recommends and according to those ratings the user profiles are refined by the system. In Roccio's algorithm the documents are represented as vectors and each dimension of that vector represents a term in the document. The value of a dimension in the vector represents the weight of the term and calculated generally by TF-IDF term weighting scheme [5]. To assign a class to a document the similarity between the document's vector and the class's vector is found. The class's vector is generated by combining the document vectors which belong to that class. To calculate the similarity between the document vector and the class vector cosine similarity can be used [5]. After calculating the similarities, the document is assigned to a class whose vector similarity is found higher than the others.

The main disadvantage of this method is it needs user interaction during the retrieval process [31]. This approach is considered as a heuristic-based technique in content-based recommendation [1], and the PRES system [48] uses this method to recommend small articles related to home improvements.

b. Winnow algorithm [40]

This algorithm is preferred for domains which have too many possible content attributes [57]. This approach is considered as a heuristic based technique in content-based recommendation [1].

c. Machine Learning Techniques [1]

If the recommendation task is considered as a classification task, then standard machine learning techniques can be used instead of vector space model [31].

Machine learning algorithms are able to learn a model to generate user profiles. To solve the user profile generation problem; each item should be classified into the one of the two classes: Like Class and Dislike Class [41].

Machine learning algorithms can be used to learn user profiles when the items are represented as structured data [58, 41]. Structured data means that; the items in the system can be represented with the same set of attributes and the values of these attributes are from a known set [58].

- **Naive Bayes [41]**

In collaborative filtering approach, naive Bayes kind of probabilistic methods are used to determine the user's cluster. But in content-based recommenders the probabilistic methods are used for directly deciding whether an item will be liked or not liked by the user [31].

With this approach the probability of an item belonging to a class is generated by users' past rating information [41]. There are two commonly used naive Bayes classifiers in the literature [44] : multivariate Bernoulli event model and multinomial event model. In both of the classifiers the word order of the documents is lost and documents are represented as vectors of words. The values in these vectors represent whether the word exists in the document or not. In multinomial event model, the existence count of the word is also taken into account. As a result, multinomial Naive Bayes classifier performs better than the multivariate Bernoulli event model [44].

Syskill & Webert system [56] classifies unrated Web pages by using naive Bayesian classifier. In this system, user rates Web pages as "relevant" or "irrelevant". This means that the system uses two classes named "Relevant" and "Irrelevant" [56].

Some other machine learning techniques that are used in content-based recommenders are [1, 31]:

- Decision Trees
- Rule Induction
- Clustering
- Artificial Neural Networks

d. Linear Classifiers

Some linear classifiers are also used in content-based recommender systems' user profile learning process. These classifiers are [31]:

- Widrow-Hoff Algorithm
- Support Vector Machines

2.5.2.3. Rating Prediction In Content-Based Recommenders

To recommend items to a user, the user profile is compared with the item profiles which are not seen by the user yet [41]. The items whose profiles are most similar to the user's profile, in other words, the items which are similar to items that user liked in the past are recommended to users by content-based recommenders [56].

In content-based recommender systems the satisfaction function which is defined in the recommendation problem's formal definition generally defined as [1];

$$f(u,i) = score(ContentBasedProfile(u), Content(i)) \tag{2.12}$$

To calculate the satisfaction function by using the TF-IDF vectors of user and item profiles, cosine similarity measurement [69] from information retrieval literature is the most widely used metric in the literature [41]. Such as; PRES system [48] uses this metric to calculate similarities.

$$f(u,i) = cos(\overrightarrow{w_u}, \overrightarrow{w_i}) = \frac{\overrightarrow{w_u} \cdot \overrightarrow{w_i}}{||\overrightarrow{w_u}||_2 \times ||\overrightarrow{w_i}||_2} = \frac{\sum_{n=1}^{K} \mathrm{w}_{n,u} \mathrm{w}_{n,i}}{\sqrt{\sum_{n=1}^{K} \mathrm{w}_{n,u}{}^2} \sqrt{\sum_{n=1}^{K} \mathrm{w}_{n,i}{}^2}} \tag{2.13}$$

,where $\overrightarrow{w_u}$ represents the user TF-IDF vector and $\overrightarrow{w_i}$ represents the TF-IDF vector of item and K represents the total number of terms in the system [41].

If user u reads more articles which are related to the geographic information systems (GIS) field, then the content-based recommender which uses a similarity measurement algorithm such as cosine similarity measurement algorithm will calculate higher utility scores to articles which include more GIS-related keywords like; "coordinate system", "raster" and "map" [1]. Correspondingly, calculate lower utility scores to articles which don't include those keywords. This is because the user's profile defined by $\overrightarrow{w_u}$ has higher weight values for GIS-related keywords and lesser weight values for others [1]. So, the item profiles which include the GIS-related keywords are calculated as more similar to user's profile and have a high satisfaction score.

Cosine similarity measurement technique is also considered as a heuristic based technique in content-based recommendation [1]. Cosine-based similarity is used in both content-based recommenders and collaborative filtering recommenders. In content-based recommenders cosine-based similarity is used to find the similarities between the user and the items and applied to the TF-IDF weight vectors. In collaborative filtering recommenders cosine-based similarity is used to find the similarities between users and applied to the vectors which are defined by the actual rating scores of the users [1]. In content-based recommenders the k-Nearest neighbor algorithm which is explained in collaborative filtering recommenders section can also be used to generate recommendation lists to users [31].

In methods which uses TF-IDF vectors of user and item profiles (Roccio's Relevance Feedback algorithm and Winnow algorithm) the satisfaction of users are calculated according to a defined formula such as cosine similarity measurement [1]. However, in other approaches (machine learning techniques and linear classifiers); the satisfactions are predicted by using models. Models are learned with statistical learning and machine learning techniques by using the data provided via the recommender system [1].

Other similarity metrics that are used in content-based recommenders are [79]:

- **Adjusted Cosine Similarity:** Adjusted cosine similarity measurement (Equation 2.7) that we mentioned in (Section 2.5.1) is also used in content-based recommendation. In adjusted cosine similarity measurement technique the aim is to consider the different rating scales of users. Such as; the average score in a 10 point scale may be 5 points for a user and 6 points for another user.

- **Correlation-based similarity:** Pearson correlation (Equation 2.5) that we mentioned in (Section 2.5.1) is also used in content-based recommendation. [79] uses this similarity measurement technique in their hybrid recommender's content-based part and [9] also uses this technique to calculate the distance between profiles. [60] uses word-correlation factors to calculate movie content similarities for recommending movies to its users.

2.5.2.4. Limitations Of Content-Based Recommenders

The main limitations of content-based recommenders are limited-content analysis, overspecialization and new user problems [5, 1].

Limited Content Analysis

In content-based recommender systems, items that are going to be recommended should be defined by content attributes. Even these attributes can be extracted in text documents effectively by information retrieval techniques, for some other items such as multimedia data (graphical image, video stream etc.), the content attributes are harder to extract automatically by computer systems [74]. For these kinds of items, the content attributes can be assigned manually, but this is not practical due to the limited resources [74]. Another important problem related to the limited content analysis in content-based recommender systems is the problem of distinguishing items which are represented with the same set of content attributes. If two different items have the same set of content attributes, then the recommender system won't be able to distinguish these items from each other [74].

Overspecialization

In content-based recommenders only the items which are similar to the items that a user rated before can be recommended to users. This situation causes that if the user who didn't rate an item which is similar to an item that s/he supposes to like won't be recommended to that user by a content-based recommender. The user is limited to what s/he rated before. This problem is tried to be solved by [75] with including some randomness to the recommendation system via using genetic algorithms in information filtering process. In addition to this case, in some domains such as news domain, the items which are too similar to items which suits to user's preference are also shouldn't be recommended to users. If a news recommender system doesn't handle this situation, than the same news from different sources will be recommended to the same user. To solve this problem; Daily-Learner [8], eliminates not only the items which are not similar to items that the user rated but also eliminates the items which are too similar to items that the user already seen.

New User Problem

In content-based recommender systems, the recommendations are done according to the user's past rating history. If the user don't have sufficient number of ratings, then the system won't be able to define a profile to that user which indicates the user's preferences correctly. So, the user cannot get accurate recommendations from this recommender system. For instance; because of this limitation MovieLens 100k data set [52] which is generally used by recommender systems such as [23], [78] and [3] include users who have at least 20 ratings related to movies to generate recommendations.

2.5.2.5. Advantages Of Content-Based Recommenders

In content-based recommenders each user is independent from other users. In collaborative filtering recommenders, the users are strictly bind to each other. To recommend items in collaborative filtering recommenders, the neighbors of the user whose tastes match with the user should be found. In contrast, users have their own profiles in content-based recommenders and recommendations are done by these profiles without the inclusion of other users [41].

Content-based recommender systems have the ability to explain why it recommends an item according to the item attributes which match with the user's profile. This provides transparency to the recommendation process. Adversely, collaborative filtering recommender systems are black boxes, because they cannot explain why they recommend the item to the user [41].

The new item problem which exists for collaborative filtering recommender systems is not applicable for content-based recommender systems. Because the items are represented with their attributes and recommended according to these attributes, content-based recommenders doesn't need sufficient number of users to rate the new item before it can recommend the item to its users [41].

2.5.3. Knowledge-Based Recommenders

In knowledge-based recommender systems, in addition to the content information domain knowledge is used to recommend items to users [1, 65]. Knowledge-based recommenders are developed to fulfill the suggestion needs related to more complex and costly items [31]. Such as a recommendation for a car or a house which people purchase rarely. Collaborative filtering or content-based approaches are not suitable for those kinds of item recommendations [31]. Rating information related to these items are not too many and even rating information exists, people's interests change over time according to the change in life styles, income amounts or marital statuses [31]. Also, the content information related to these items are not enough to represent user preferences [31]. People need to describe more detailed preferences related to these kinds of items such as; "A house whose cost is less than $150000 and which has minimum 3 rooms."

Knowledge-based recommenders use similarities between user requirements and items or use explicit rules of recommendation instead of using rating information [31]. Users interact with the system while the recommendations are generating [31].

Knowledge-based recommenders are divided into sub categories [31, 65]: case-based [13] and constrained-based [82]. In both of the approaches users define their preferences explicitly and the recommender tries to find items which meets those preferences and the system also has an explanation related to why the presented item is recommended to a user [31]. In case-based systems; how much the user needs match with the recommendations is calculated via a similarity function [65]. In constrained-based systems; predefined knowledge bases which consist of explicit rules are used to relate user needs with item attributes [65].

Semantic recommender systems are an extension of knowledge-based recommender systems [59]. Semantic recommenders are also based on a knowledge-base. That knowledge base is represented via a taxonomy or an ontology and uses Semantic Web technologies [59]. Semantic recommender systems are categorized into three different subtypes [59]:

- Ontology Based Systems
- Trust Network Based Systems
- Context Adaptable Systems

The main disadvantage of knowledge-based recommenders is acquiring knowledge [1]. Knowledge-based recommender systems can be used in domains where the domain knowledge is in a machine-readable form such as an ontology [1].

2.5.4. Hybrid Recommenders

Hybrid recommender systems are the combination of two or more recommendation approaches which aim to improve the recommendation accuracy and to cope with the pure approaches' disadvantages [12, 10]. There are seven different hybridization methods which are used in recommender systems [12]:

- **Weighted**

In this method, the predicted rating score is calculated by combining all of the recommendation techniques that are used. In the simplest case, a linear combination can be used to merge the results of different techniques.

- **Switching**

The hybrid recommender switches between the used techniques according to some criterion and uses the most suitable technique to generate recommendation scores. A switching criteria may be using content-based recommendation approach and if it cannot generate an accurate result then using collaborative filtering approach.

- **Mixed**

Recommendations from used techniques are combined into one single technique and recommendations are generated via this combined technique.

- **Feature Combination**

In this type of hybridization method, the collaborative information is accepted as another content information and collaborative information is also added to the content-based recommendation process as additional information.

- **Cascade**

In this method, one of the recommendation techniques which generate the hybrid recommender applies first, then the second approach refines the results of the first approach to produce final recommendations.

- **Feature Augmentation**

The result of one technique is included in the second technique's recommendation process. First technique's result can either be a rating score or a classification result.

- **Meta-level**

In this method, the model generated by one of the techniques is used in the second technique's recommendation process as input. This type of hybridization differs from feature augmentation by using the whole generated model as an input in the second approach. In feature augmentation, the results which are obtained by applying the learned model are used as inputs to the second recommendation approach.

The Fab system [5] is one of the well-known hybrid recommender system. It combines both content-based and collaborative recommendation approaches and generates a hybrid recommender system to recommend web pages to its users [5]. [6] recommends movies, [77] recommends news Web pages and [68] recommends materials by combining collaborative filtering with content-based recommendation technique, too. Both the rating information and content information is used in these systems' recommendation processes.

Content-boosted collaborative filtering system [47] uses a hybrid approach which includes content-based characteristics to increase user data and to generate personalized movie recommendations via collaborative filtering.

The EntreeC system [12] combines knowledge-based and collaborative filtering recommendation approaches to generate a hybrid recommender system.

CHAPTER 3

USER PREFERENCE BOOSTED CONTENT-BASED RECOMMENDATION

This chapter presents the proposed user preference boosted content-based recommendation system approach. This proposed approach includes two different methodologies related to the generation process of the user preference profiles. The proposed methodologies are described by using movie domain characteristics. First, the motivation behind the proposed approach and methodologies is given. After that, the system overview is presented with flow diagrams to clarify how the system works. Finally, the implementation details of the proposed content-based methodology is explained by describing how the movie and user profiles are generated by said methodologies and how the recommendation scores are calculated.

3.1. Motivation

In content-based recommender systems, user profiles are generated according to the item content information and the user's past rating history. The user profiles and the item profiles have the same attribute set to define user preference information and to define items, respectively. These profiles are generated by various techniques such as; TF-IDF (Term Frequency-Inverse Document Frequency) [69, 41, 77], Winnow algorithm [40] or Rocchio algorithm [67, 5]. Also the most commonly accepted and used technique is TF-IDF measurement technique [41].

We believe that; in order to generate more user oriented profiles, in addition to the item attribute characteristics, the user preference information should also be considered in user profile generation process. By combining both user preference information with the item attribute characteristics, more descriptive user profiles can be generated. As a result, more accurate recommendations can be gathered.

In the proposed approach, the item attribute characteristics are calculated by using TF-IDF measurement technique. In addition to this measurement, to generate user oriented profiles; we have used the users' past preference information and the movie attributes to calculate the user's preference ratios. We have proposed two different user preference ratio calculation techniques. The first one is calculated by considering user's preference information in the item attribute level. The second approach is a one level abstraction of the former approach. This second proposed preference ratio is calculated by considering user's preference information in the item attribute category level.

3.2. System Overview

In order to generate more user oriented profiles and obtain more accurate recommendation scores; we have developed a user preference boosted content-based recommendation system. To calculate user preference ratios, the movies in the system are represented as a list of attributes and attributes are grouped under categories. After that, users' preference ratios related to the attributes and categories can be calculated by examining user's past rating history. We have calculated the user's preference ratios for both attributes and categories.

The flow diagram of our approach related to users' preference ratio which is calculated for the attributes is shown in (Figure 3.1). Section numbers in this thesis work which refers to the steps of the proposed approach is also given in Figure 3.1. The flow diagram of our second approach whose user preference ratios are calculated for the categories is shown in (Figure 3.2). Section numbers in this thesis work which refers to the second proposed approach is also given in Figure 3.2.

Novel issues of the proposed approach are;

1. User-attribute pair's preference ratio calculation technique,
2. User-category pair's preference ratio calculation technique,
3. The generation of the user profiles by using both the attributes' TF-IDF weight values with the effect of the user's preference ratio information related to the attributes and categories which are derived from the user's past rating history,
4. The approach which is used in the generation of the predicted rating scores.

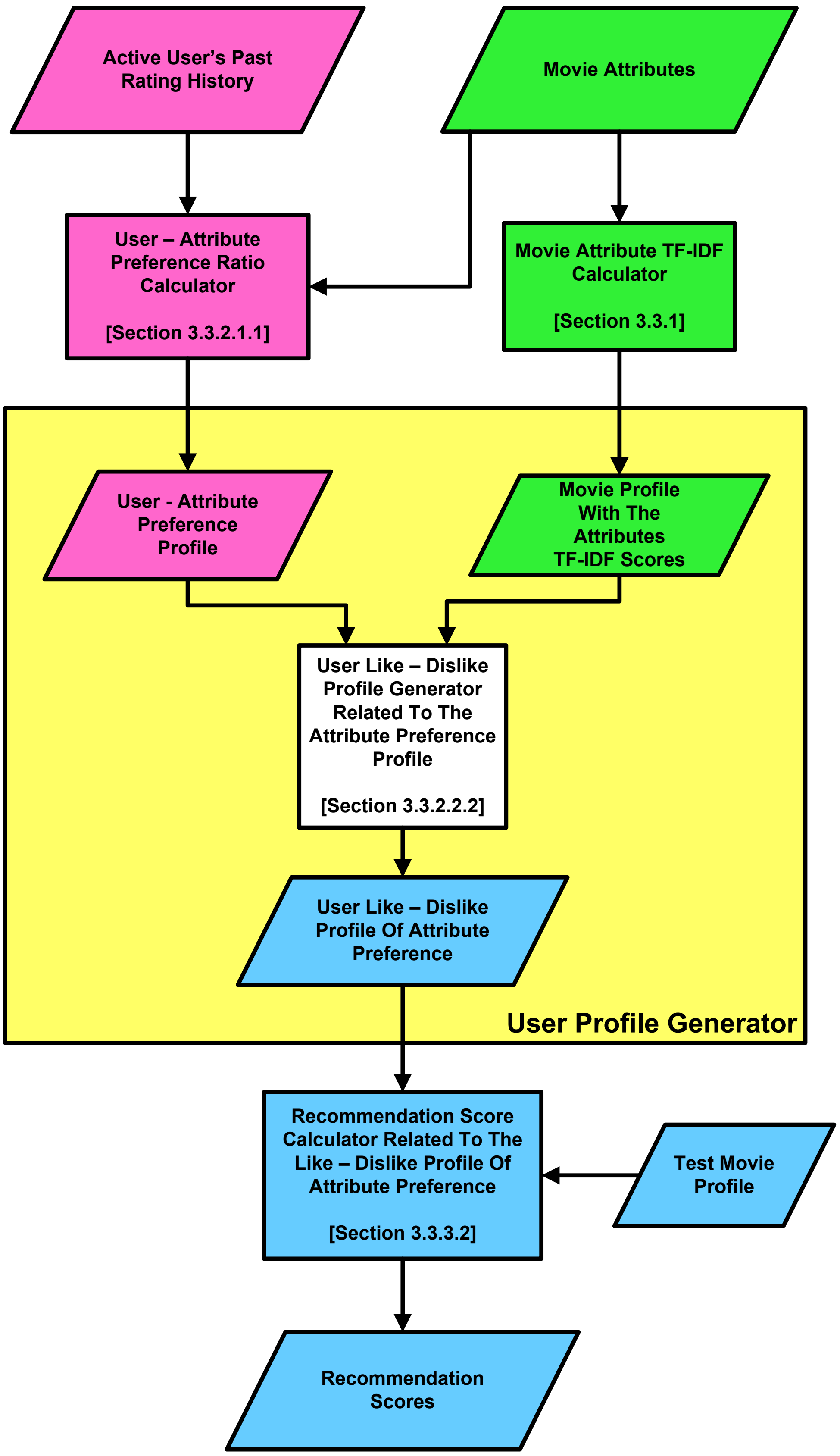

Active User's Past Rating History
Movie Attributes
User – Attribute Preference Ratio Calculator [Section 3.3.2.1.1]
Movie Attribute TF-IDF Calculator [Section 3.3.1]
User - Attribute Preference Profile
Movie Profile With The Attributes TF-IDF Scores
User Like – Dislike Profile Generator Related To The Attribute Preference Profile [Section 3.3.2.2.2]
User Like – Dislike Profile Of Attribute Preference
User Profile Generator
Recommendation Score Calculator Related To The Like – Dislike Profile Of Attribute Preference [Section 3.3.3.2]
Test Movie Profile
Recommendation Scores

Figure 3.1: Flow Diagram Of User - Attribute Preference Ratio Approach

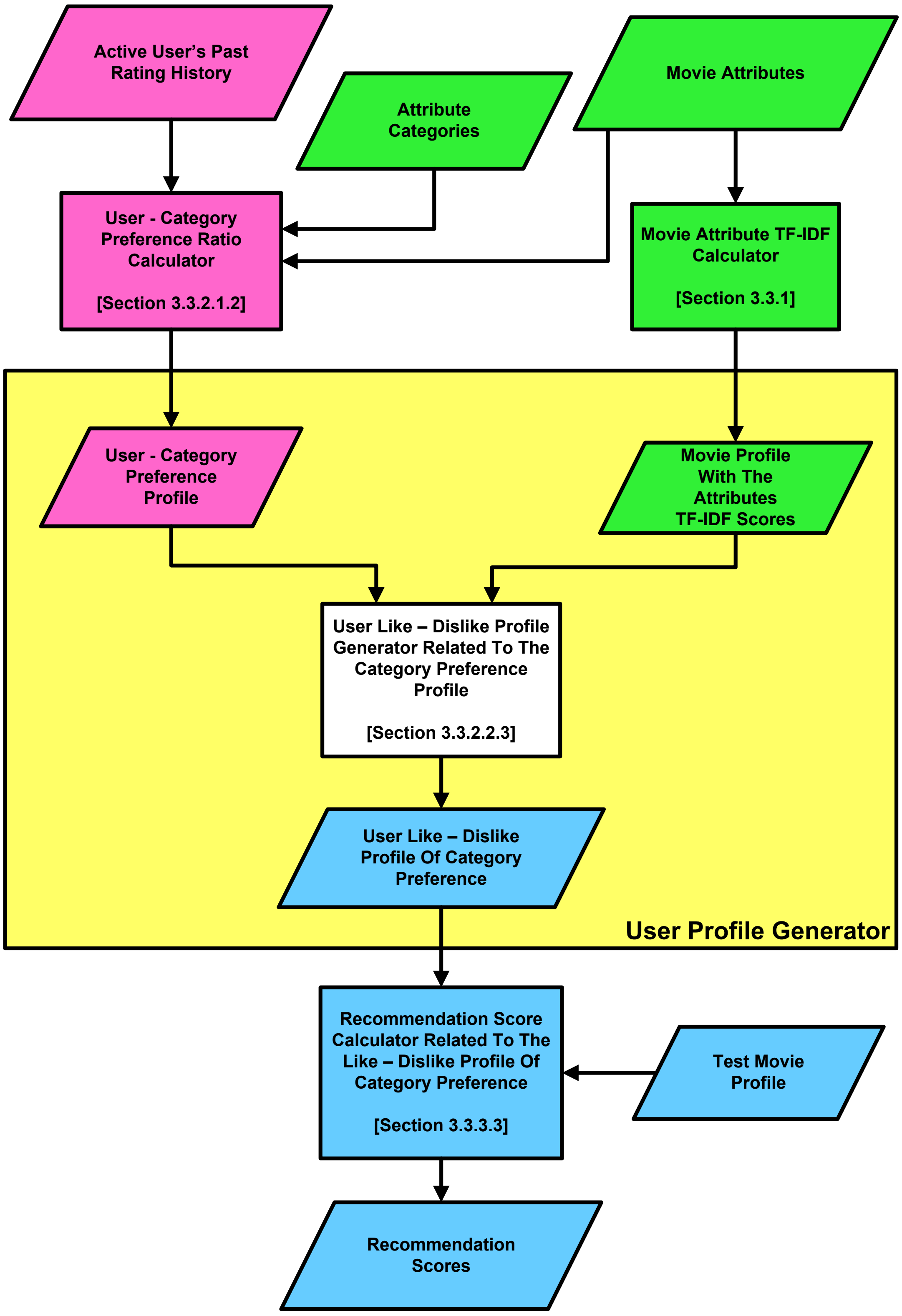

Active User's Past Rating History
Attribute Categories
Movie Attributes
User - Category Preference Ratio Calculator [Section 3.3.2.1.2]
Movie Attribute TF-IDF Calculator [Section 3.3.1]
User - Category Preference Profile
Movie Profile With The Attributes TF-IDF Scores
User Like – Dislike Profile Generator Related To The Category Preference Profile [Section 3.3.2.2.3]
User Like – Dislike Profile Of Category Preference
User Profile Generator
Recommendation Score Calculator Related To The Like – Dislike Profile Of Category Preference [Section 3.3.3.3]
Test Movie Profile
Recommendation Scores

Figure 3.2: Flow Diagram Of User - Category Preference Ratio Approach

Our system, first generates movie profile by using the content information of movies in the used movie database [15]. After generating movie profiles; the proposed system, first creates the user preference profiles by using both the movie profile and the active user's past rating history. Afterwards, the user's like-dislike profiles are generated by using active user's rating history, movie profile and the generated user preference profiles. Finally, the proposed system calculates the recommendation scores via the rating predictor module by using the user's like - dislike profile with the movie profiles of the unrated movies.

The system architecture and the details of the processes will be explained in the following subsections of this chapter.

3.3. System Architecture and Implementation Details

The proposed system is a user preference boosted content-based recommender system. The system uses the content information of the movies and the past rating history of the users to generate recommendation scores. As the system is a content based recommender; there are user and movie profiles which are generated by using the database of a web site named "cineworm.com" [15]. The called web site is a system which stores the movie information, user information and rating history of the users related to movies.

In recommendation problem the general aim is to find the relevant items which satisfy the user mostly. To find the movie which satisfies the user mostly, the system uses movie and user profiles. Said satisfaction function which indicates the satisfaction that user u gets from a movie m is defined as follows;

$$s : U \times M \rightarrow R \tag{3.1}$$

where U is the set of users, M is the set of movies and R is the rating score which is $\in$ [0,10].

As a result, the recommendation problem's solution can be formulated as;

$$\forall u \in \mathrm{U}, \mathrm{m} \in \mathrm{M}, \ m'_{\mathrm{u}} = \arg\max \mathrm{s}\,(\mathrm{u}, \mathrm{m}) \tag{3.2}$$

where U is the set of users and M is the set of movies.

The proposed user preference boosted content-based recommendation approach consists of four main phases to find the movies which maximizes the satisfaction of the users;

1. Movie Profile Generation
2. User Preference Profile Generation
3. User Like-Dislike Profile Generation
4. Recommendation Score Calculation For Unrated Movies

The details of the given phases will be explained in the following subsections.

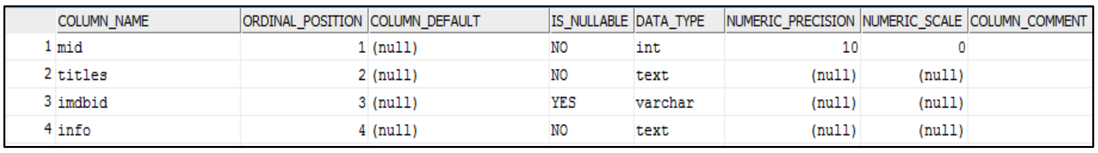
3.3.1. Movie Profile Generation

In proposed system, each movie has its own profile and has several attributes that refer to different categories which are derived from the used movie database [15].

Before generating the movie profiles the attributes are categorized according to the content knowledge of the movies which are extracted from the database [15]. For instance; "Genre" is a category in the system and "Animation" is an attribute which belongs to the category "Genre". "Actor" is another category and "Johnny Depp" is an attribute of "Actor" category. Each attribute belongs to only one category and each movie has at least one attribute which belongs to the defined categories in the system.

In the proposed system, the movie profiles are stored in four different database tables. The combination of those tables generates the profiles of the movies. The tables are "Movies", "AttributeCategory", "Attributes" and "MovieAttributePairs". The details of the tables are given below;

"Movies" table stores the general information related to movies. (Figure 3.3) shows the "Movies" table.

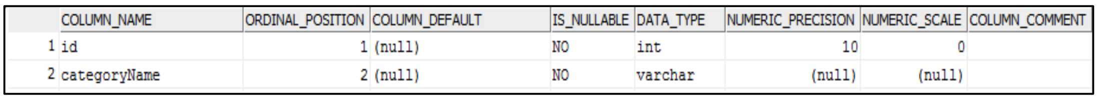


	COLUMN_NAME	ORDINAL_POSITION	COLUMN_DEFAULT	IS_NULLABLE	DATA_TYPE	NUMERIC_PRECISION	NUMERIC_SCALE	COLUMN_COMMENT
1	mid	1	(null)	NO	int	10	0	
2	titles	2	(null)	NO	text	(null)	(null)	
3	imdbid	3	(null)	YES	varchar	(null)	(null)	
4	info	4	(null)	NO	text	(null)	(null)	

Figure 3.3: Movies Table

In "Movies" table "mid" column stores the unique id of the movies. "titles" column stores the name of the movie. "imdbid" column stores the IMDB id of the movies. This column is not used in the proposed system's movie profile generation process, because the content information of the movies are already stored in this database. Also we didn't need to merge the results of the IMDB database and the "cineworm.com" web sites database. "info" column stores all the content details of the movies in the database and this column stores the content information of the movies in JavaScript Object Notation (JSON) format. JSON format is a text format which is independent from the used programming language and it is a flexible data-interchange language [34]. "info" column stores the genre, actor, director, duration, release year and author information of the movies and we extracted all of the attributes related to the movies by using this column.

"AttributeCategory" table stores the categories of the attributes which are used in the proposed system. (Figure 3.4) shows the "AttributeCategory" table.



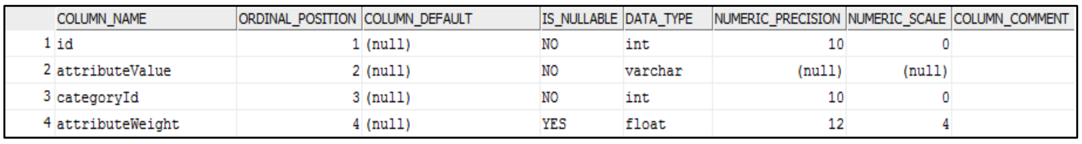
	COLUMN_NAME	ORDINAL_POSITION	COLUMN_DEFAULT	IS_NULLABLE	DATA_TYPE	NUMERIC_PRECISION	NUMERIC_SCALE	COLUMN_COMMENT
1	id	1	(null)	NO	int	10	0	
2	categoryName	2	(null)	NO	varchar	(null)	(null)	

Figure 3.4 : AttributeCategory Table

In "AttributeCategory" table "id" column stores the unique id of the attribute categories. "categoryName" column stores the name of the attribute category.

"Attributes" table stores the attributes of all movies in the system. Each attribute is inserted to this table only once, even more than one movie has the attribute.

(Figure 3.5) shows the "Attributes" table.



	COLUMN_NAME	ORDINAL_POSITION	COLUMN_DEFAULT	IS_NULLABLE	DATA_TYPE	NUMERIC_PRECISION	NUMERIC_SCALE	COLUMN_COMMENT
1	id	1	(null)	NO	int	10	0	
2	attributeValue	2	(null)	NO	varchar	(null)	(null)	
3	categoryId	3	(null)	NO	int	10	0	
4	attributeWeight	4	(null)	YES	float	12	4	

Figure 3.5 : Attributes Table

In "Attributes" table "id" column stores the unique id of the attributes. "attributeValue" column stores the value of the attribute. "categoryId" column is a foreign key to the "AttributeCategory" table which the attribute belongs to. "attributeWeight" column stores the attribute's score which is calculated by using TF-IDF measurement technique [69, 41]. This column shows the attribute's descriptiveness characteristic.

Like all other domains; in movie domain, some attributes are more descriptive than the others to distinguish a movie. We propose that, an attribute a_i is more descriptive if less number of movies in the system have the property a_i. In addition to this; we also propose that, if the category C_j has more attributes which belongs to it, it shows that this category is more descriptive while distinguishing movies. To use those assumptions, we used the TF-IDF measurement [69, 41] to calculate the "attributeWeight" column of the attributes. The TF-IDF of an attribute a_i is defined as follows;

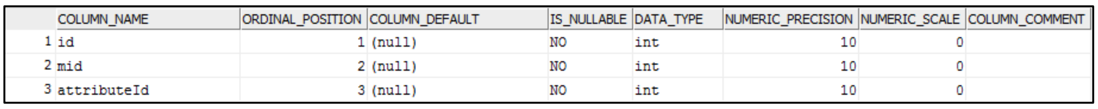
$$AS^*(a_i) = \log\left(M/m_{a,i}\right) \times \log(\#C_j) \tag{3.3}$$

*AS = Attribute Score

where M is the total movie number in the system, $m_{a,i}$ is the number of movies which has the attribute a_i. $\#C_j$ is the number of attributes under category Cj which attribute a_i belongs to.

The "attributeWeight" column is calculated after extracting the categories and theattributes of the movies which are defined in the movie database [15]. The proposed system is also capable of recalculating this attributeWeight scores of the attributes in the system if a new movie is added to the system in the future.

"MovieAttributePairs" table stores the movie - attribute pairs in the system. For each attribute which belongs to a movie, a new row is inserted to this table. (Figure 3.6) shows the "MovieAttributePairs" table.



	COLUMN_NAME	ORDINAL_POSITION	COLUMN_DEFAULT	IS_NULLABLE	DATA_TYPE	NUMERIC_PRECISION	NUMERIC_SCALE	COLUMN_COMMENT
1	id	1	(null)	NO	int	10	0	
2	mid	2	(null)	NO	int	10	0	
3	attributeId	3	(null)	NO	int	10	0	

Figure 3.6 : MovieAttributePairs Table

In "MovieAttributePairs" table "id" column stores the unique id of the table. "mid" column is a foreign key to the "Movies" table. "attributeId" column is a foreign key to the "Attributes" table. Each movie has multiple rows in this table whose count equals to the number of attributes which belongs to the movie. Also each attribute is repeated number of times which equals to the number of movies which has the attribute. Each movie and attribute is repeated more than once in this table. This means that the "mid" and "attributeId" columns together forms a unique key for this table.

Before generating the attribute's TF-IDF score, the movies in the database [15] are divided into its attributes and the results are stored in the "MovieAttributePairs" table. Actually, "MovieAttributePairs" table keeps the Movie-Attribute vector in the system and each movie is represented as a vector of attributes by the usage of this table.

3.3.2. User Profile Generation

In proposed system, each user has two preference profiles. One of them contains the user's preference ratios related to each attribute that the movies in the system consists of. Also the second preference profile of the users contains the preference ratios related to each category that the attributes belong to. After generating user preference profiles, user's like-dislike profile is generated by using the user's preference profiles. User like-dislike profile consists of user-attribute pairs whose like and dislike weights are calculated by user preference ratios. By generating such user like-dislike profile, user's recommendation scores are estimated by using the attributes that the unrated movie and the user like-dislike profile have in common.

Before generating user-attribute pairs of user like-dislike profile, user's preference ratios related to the attributes and categories which are defined in the system are calculated. To calculate those preference ratios; novel approaches are used which use the user's past rating scores and rated movies' attributes and categories. Creation of the novel user preference profiles and using these profile information while generating the user’s like-dislike profile is the main contribution of this thesis work.

In general, content-based recommender systems use user profiles which are generated by the user's past rating history and the attributes that the rated movies have. The innovative part of this proposed content-based recommender system is the generation process of the user profiles. Before generating the user's like - dislike profile, user's preference ratios related to attributes and categories are calculated. These preference ratios indicate the attribute's and category's importance level in the eyes of the user. In other words, those ratios show the attribute's and the category's affection ratio while the user is giving a rating score to a movie.

To calculate the user's attribute preference ratio; suppose that; user u has rated 4 movies and all of these movies has the attribute "Comedy" in common. For instance, if the user has a rating history which is like in the (Table 3.1); this result indicates that the attribute "Comedy" has no effect on the user's preference while rating a movie. Because the user rated 2 of the movies as positive and 2 of the movies as negative. In other words, the attribute preference ratio value of the pair [u - "Comedy"] equals to 0.

Table 3.1: Example User Rating History 1

Movie Number	Rating Of User u
#1	Positive
#2	Negative
#3	Negative
#4	Positive

If the user has a rating history which is like in the (Table 3.2) or (Table 3.3); this result indicates that the attribute "Comedy" is an important criteria while giving the rating score related to a movie for that specific user. Because in (Table 3.2) user rated all of the movies as positive and in (Table 3.3) user rated all of the movies as negative. This result also shows that the user gives recommendation scores which are reliable according to this "Comedy" attribute. So, the attribute preference ratio value of the pair [u - "Comedy"] equals to 1. The calculation details of the user preference ratios related to attributes are given in the next section.

Table 3.2 : Example User Rating History 2

Movie Number	Rating Of User u
#1	Positive
#2	Positive
#3	Positive
#4	Positive

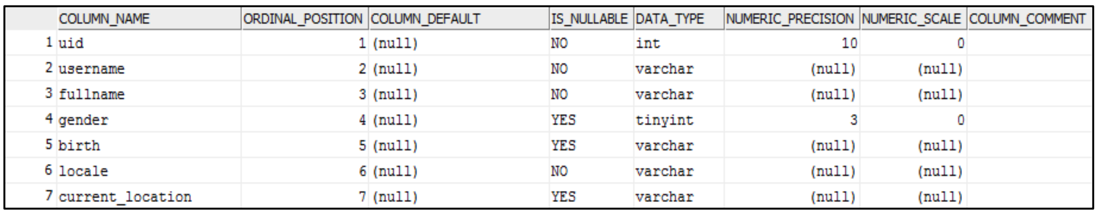
Table 3.3 : Example User Rating History 3

Movie Number	Rating Of User u
#1	Negative
#2	Negative
#3	Negative
#4	Negative

To calculate the user's category preference ratio; the proposed system calculates the average sum of the user's attribute preference ratios whose attribute values belong to the category under consideration. Suppose that; the attributes which are defined under the category "Genre" are "Drama", "Action" and "Comedy" in the system. Then suppose that; a user u has attribute preference ratios related to those attributes as 0.8, 0.6, and 0.4, respectively. Then the user preference ratio related to the category "Genre" is calculated as the average sum of those user attribute preference ratios and the [u - "Genre"] pair's category ratio value equals to 0.6.

In the proposed system, the user profiles are stored in five different database tables. The combination of those tables generates the profiles of the users. The tables are "Users", "Rating", "UserAttributePreference", "UserAttributeCategoryPreference" and "UserAttributePairs". The details of the "Users" and "Rating" tables are given below, the details of the "UserAttributePreference" table is given in the (Section 3.3.2.1.1), the details of the "UserAttributeCategoryPreference" table is given in the (Section 3.3.2.1.2) and the "UserAttributePairs" table's details are given in the (Section 3.3.2.2).

"Users" table stores the general information related to users. (Figure 3.7) shows the "Users" table.



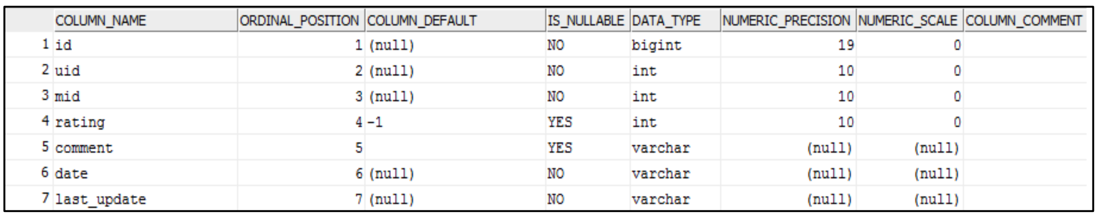
	COLUMN_NAME	ORDINAL_POSITION	COLUMN_DEFAULT	IS_NULLABLE	DATA_TYPE	NUMERIC_PRECISION	NUMERIC_SCALE	COLUMN_COMMENT
1	uid	1	(null)	NO	int	10	0	
2	username	2	(null)	NO	varchar	(null)	(null)	
3	fullname	3	(null)	NO	varchar	(null)	(null)	
4	gender	4	(null)	YES	tinyint	3	0	
5	birth	5	(null)	YES	varchar	(null)	(null)	
6	locale	6	(null)	NO	varchar	(null)	(null)	
7	current_location	7	(null)	YES	varchar	(null)	(null)	

Figure 3.7 : Users Table

In “Users" table "uid" column stores the unique id of the table. "username" column stores the username of the user. "fullname" column stores the fullname of the user. "gender" column stores the gender of the user. "birth" column stores the birthdate of the user. "locale" column stores the locale of the user and finally the "current_location" column stores the user's current location. The columns except that the "uid" column, didn't used in the proposed system. Only the "uid" column which distinguishes users from each other is used as a foreign key at other generated user profile tables.

In addition to the user's general information, the database that we use [15] includes information about the users' social network in a different database table named "Relationships" which keeps who is related to whom. This information can be gathered, because people sign in to the used web page by using their Facebook [20] or Twitter [76] accounts. At first glance, the information related to the user's social network can be seen as an important information source to generate the users' nearest neighbors by using this social network. But there is no rule related to the tastes of the people in the same social network according to the movies should match. Assuming that the people in the same social network like the same movies, not seems like a meaningful assumption to us. So, we don't use that social network information while generating the recommendation scores.

"Rating" table stores the rating history of the users related to the movies stored in the system. (Figure 3.8) shows the "Rating" table.



	COLUMN_NAME	ORDINAL_POSITION	COLUMN_DEFAULT	IS_NULLABLE	DATA_TYPE	NUMERIC_PRECISION	NUMERIC_SCALE	COLUMN_COMMENT
1	id	1	(null)	NO	bigint	19	0	
2	uid	2	(null)	NO	int	10	0	
3	mid	3	(null)	NO	int	10	0	
4	rating	4	-1	YES	int	10	0	
5	comment	5		YES	varchar	(null)	(null)	
6	date	6	(null)	NO	varchar	(null)	(null)	
7	last_update	7	(null)	NO	varchar	(null)	(null)	

Figure 3.8: Rating Table

In "Rating" table "id" column stores the unique id of the table and auto incremented by the system. "uid" column is a foreign key to the "Users" table and keeps the user id related to the rating. "mid" column is also a foreign key to the "Movies" table and keeps the movie id related to the rating. "rating" column stores the rating score related to the movie which is referred by "mid" given by the user which is referred by "uid". "rating" column's nullable information seems like it can hold null values but before running the proposed methodology in this data set, the empty rating results are removed from this table. The "rating" column stores the rating scores in the range of [0-10]. "comment" column stores the user's comment while giving the rating score and this column is not used in the proposed system. "date" column stores the information about when the user gives this rating and "last_update" column stores the information about when the user last updates this rating information and these last two columns are not used in the proposed system, either.

3.3.2.1 User Preference Profile Generation

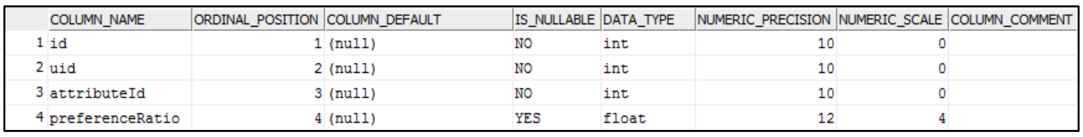
Each user in the system has two different preference profiles. One of them is related to the user's preference profile related to the attributes and the second one is related to the user's preference profile related to the categories. These user preference profiles are the pre-profiles which are generated by the proposed system before generating the user's like-dislike profile which contains the user's like and dislike weight information related to the attributes and categories that are defined in the system. Those preference profiles consists of the user, related attribute or related category and the calculated user preference ratio value. The implementation details of how those user preference profiles are created are given under this section.

3.3.2.1.1 User Attribute Preference Ratio Calculation

The "UserAttributePreference" table stores the user attribute preference profile which is derived from the user's past rating history.

The rating scale in the proposed system is [0,10], as we said before. Also the user can give rating scores which are not integer such as; "6.8". The rating precision in the system is defined as "0.1". This means that the user has 10*10 = 100 different rating scores to give a movie. In the proposed system, the ratings below 6.0 is considered as negative ratings and the ratings equal or greater than 6.0 is considered as positive ratings. Because the midpoint of the [0-10] scale is 5.0, if the user gives 5.0 to a movie, this means that this movie is an average movie for this user. So, we assume that the ratings higher than or equal to 6.0 can be considered as positive ratings.

As we said before, the user attribute preference profile of users are stored in the "UserAttributePreference" table. (Figure 3.9) shows the "UserAttributePreference" table.



	COLUMN_NAME	ORDINAL_POSITION	COLUMN_DEFAULT	IS_NULLABLE	DATA_TYPE	NUMERIC_PRECISION	NUMERIC_SCALE	COLUMN_COMMENT
1	id	1	(null)	NO	int	10	0	
2	uid	2	(null)	NO	int	10	0	
3	attributeId	3	(null)	NO	int	10	0	
4	preferenceRatio	4	(null)	YES	float	12	4	

Figure 3.9: UserAttributePreference Table

In "UserAttributePreference" table "id" column stores the unique id of the table and auto incremented by the system. "uid" column is a foreign key to the "Users" table and keeps the user id related to the user-attribute pair. "attributeId" column is a foreign key to the "Attributes" table and keeps the attribute id related to the user-attribute pair. "preferenceRatio" column stores the user's calculated preference ratio related to the attribute.

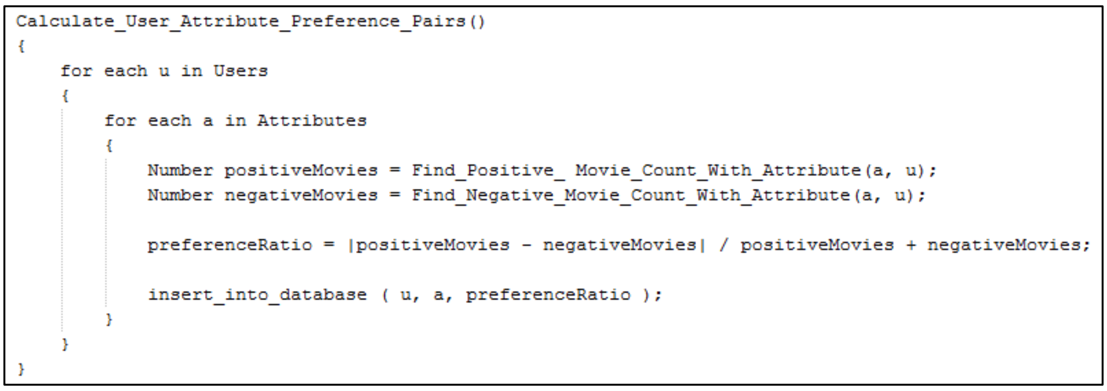
We propose that, the attribute's preference ratio according to a user can be calculated by dividing the user's past rating history into two disjoint sets: "Like Set" and "Dislike Set". After dividing the user's past rating history into these sets, the movies under each set are grouped by their attributes. We propose that, the positively rated movie count which has a specific attribute and the negatively rated movie count which has the specific attribute shows the user's preference information related to the specific attribute. If the user relies on this attribute while giving the rating scores to movies, then the difference between the positive movie count and the negative movie count should be close -in ideal case equal- to the sum of the positive movie count and the negative movie count. The attribute preference ratio's maximum value is 1, which means that the user gives all movies which has the specific attribute in the same way, either positively or negatively. If the user doesn't consider this attribute while giving the recommendation scores, this means that the positive movie count and the negative movie count values are close to each other. Than this attribute is considered as non-distinctive, and the preference ratio of the user related to this attribute converges to 0. The preference ratio of an attribute a_i for a user u_j is defined as follows;

$$\mathrm{UAPR}^{*}(a_i, u_j) = \frac{\left|\mathrm{likedMovieCount}_{a_i,u_j} - \mathrm{dislikedMovieCount}_{a_i,u_j}\right|}{\mathrm{likedMovieCount}_{a_i,u_j} + \mathrm{dislikedMovieCount}_{a_i,u_j}} \qquad (3.4)$$

*UAPR = User Attribute Preference Ratio

where $\mathrm{likedMovieCount}_{a_i,u_j}$ is the total movie number which are rated positively by the specified user u_j and have the attribute a_i. $\mathrm{dislikedMovieCount}_{a_i,u_j}$ is the total movie number which are rated negatively by the specified user u_j and have the attribute a_i.

(Figure 3.10) shows the pseudo code of how to calculate the user attribute preference ratios for all user-attribute pairs.



```
Calculate_User_Attribute_Preference_Pairs()
{
    for each u in Users
    {
        for each a in Attributes
        {
            Number positiveMovies = Find_Positive_ Movie_Count_With_Attribute(a, u);
            Number negativeMovies = Find_Negative_Movie_Count_With_Attribute(a, u);

            preferenceRatio = |positiveMovies - negativeMovies| / positiveMovies + negativeMovies;

            insert_into_database ( u, a, preferenceRatio );
        }
    }
}
```

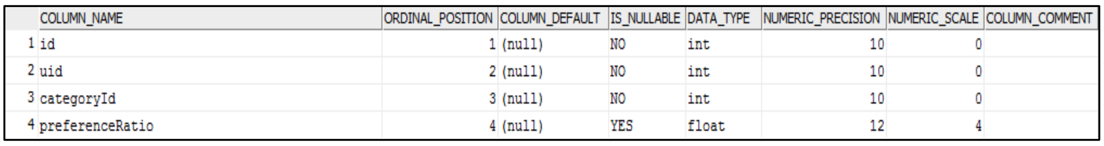
Figure 3.10 : Calculate Attribute Preference Ratio Of User Algorithm

3.3.2.1.2 User Category Preference Ratio Calculation

The "UserAttributeCategoryPreference" table stores the user category preference profile which is derived from the user's past rating history.

As we said before, this approach is a one level abstraction of the user attribute preference ratio calculation approach. User category preference ratio is considered as the user's importance level related to the categories while giving the rating scores to movies.

The user category preference profile of users are stored in the "UserAttributeCategoryPreference" table. (Figure 3.11) shows the "UserAttributeCategoryPreference" table.



	COLUMN_NAME	ORDINAL_POSITION	COLUMN_DEFAULT	IS_NULLABLE	DATA_TYPE	NUMERIC_PRECISION	NUMERIC_SCALE	COLUMN_COMMENT
1	id	1	(null)	NO	int	10	0	
2	uid	2	(null)	NO	int	10	0	
3	categoryId	3	(null)	NO	int	10	0	
4	preferenceRatio	4	(null)	YES	float	12	4	

Figure 3.11: UserAttributeCategoryPreference Table

In "UserAttributeCategoryPreference" table "id" column stores the unique id of the table and auto incremented by the system. "uid" column is a foreign key to the "Users" table and keeps the user id related to the user - category pair. "categoryId" column is a foreign key to the "AttributeCategory" table and keeps the category id related to the user - category pair. "preferenceRatio" column stores the user's calculated preference ratio related to the category.

We propose that, the category's preference ratio according to a user equals to the average sum of the user's attribute preference ratio values whose attributes are defined under the said category. The preference ratio of a category c_i for a user u_j is defined as follows;

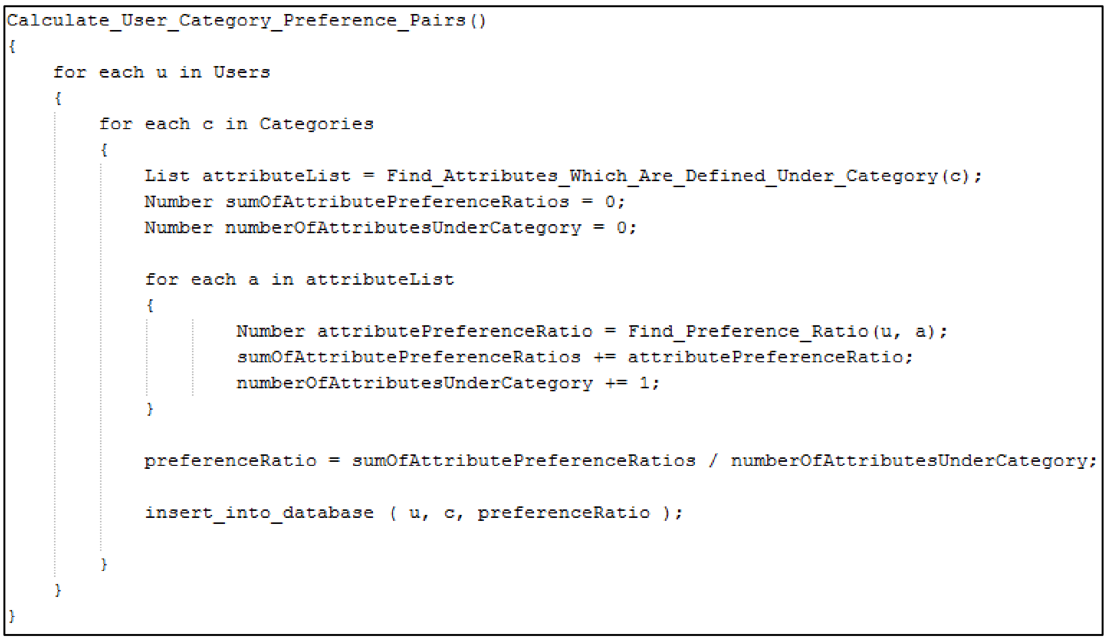
$$\mathrm{UCPR}^*(c_i, u_j) = \frac{\sum_{m=0}^{\#\mathrm{Attributes}_{c_i}} \mathrm{UAPR}^*(a_m, u_j)}{\#\mathrm{Attributes}_{c_i}} \tag{3.5}$$

*UCPR : User Category Preference Ratio

*UAPR : User Attribute Preference Ratio

where $\#\mathrm{Attributes}_{c_i}$ is the total attribute number which are defined under the category c_i. UAPR(a_m,u_j) is the user u_j's attribute preference ratio related to the attribute a_m which is defined under category c_i.

(Figure 3.12) shows the pseudo code of how to calculate the user category preference ratios for all user-attribute pairs.



```
Calculate_User_Category_Preference_Pairs()
{
    for each u in Users
    {
        for each c in Categories
        {
            List attributeList = Find_Attributes_Which_Are_Defined_Under_Category(c);
            Number sumOfAttributePreferenceRatios = 0;
            Number numberOfAttributesUnderCategory = 0;

            for each a in attributeList
            {
                    Number attributePreferenceRatio = Find_Preference_Ratio(u, a);
                    sumOfAttributePreferenceRatios += attributePreferenceRatio;
                    numberOfAttributesUnderCategory += 1;
            }

            preferenceRatio = sumOfAttributePreferenceRatios / numberOfAttributesUnderCategory;

            insert_into_database ( u, c, preferenceRatio );

        }
    }
}
```

Figure 3.12 : Calculate Category Preference Ratio Of User Algorithm

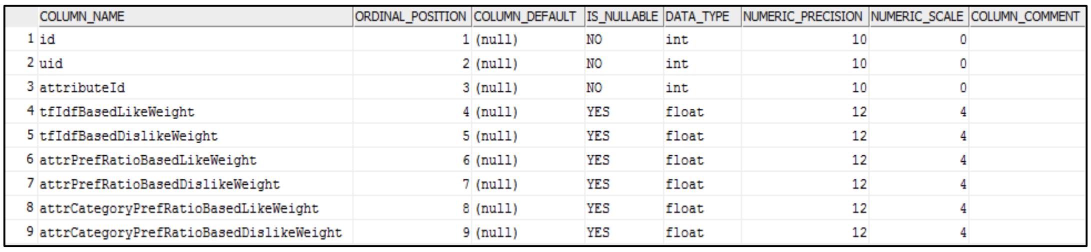
3.3.2.2 User Like-Dislike Profile Generation

The user like-dislike profile is the profile which is generated by the proposed system after calculating all attributes' TF-IDF scores and user preference ratios related to attributes and categories. This profile is the last profile of the user which is used to calculate the estimated rating scores for unrated movies.

This user profile contains the user's like and dislike weight values. In this profile, there are three different like-dislike weight pairs. One of them is calculated according to the attribute weight values of attributes which we called as TF-IDF scores. The second like-dislike weight pair is calculated by adding the effect of user attribute preference ratio to the TF-IDF scores of the attributes. Finally, the third one is calculated by appending the impact of user category preference ratio to the attribute weight value which is calculated by using TF-IDF measurement technique.

Because the main contribution of this thesis work is to show the positive impact of including user preference ratios to the user profile generation process in addition to the classical TF-IDF measurement technique; users' like and dislike weights are calculated three times with three different mentioned techniques to compare them after calculating the estimated rating scores.

The user's final profile, in other words user's like-dislike profile, is stored in the database table named "UserAttributePairs" in the proposed system. (Figure 3.13) shows the "UserAttributePairs" table.



	COLUMN_NAME	ORDINAL_POSITION	COLUMN_DEFAULT	IS_NULLABLE	DATA_TYPE	NUMERIC_PRECISION	NUMERIC_SCALE	COLUMN_COMMENT
1	id	1	(null)	NO	int	10	0	
2	uid	2	(null)	NO	int	10	0	
3	attributeId	3	(null)	NO	int	10	0	
4	tfIdfBasedLikeWeight	4	(null)	YES	float	12	4	
5	tfIdfBasedDislikeWeight	5	(null)	YES	float	12	4	
6	attrPrefRatioBasedLikeWeight	6	(null)	YES	float	12	4	
7	attrPrefRatioBasedDislikeWeight	7	(null)	YES	float	12	4	
8	attrCategoryPrefRatioBasedLikeWeight	8	(null)	YES	float	12	4	
9	attrCategoryPrefRatioBasedDislikeWeight	9	(null)	YES	float	12	4	

Figure 3.13 : UserAttributePairs Table

In "UserAttributePairs" table "id" column stores the unique id of the table and auto incremented by the system. "uid" column is a foreign key to the "Users" table and keeps the user id related to the user-attribute pair. "attributeId" column is a foreign key to the "Attributes" table and keeps the attribute id related to the user-attribute pair. "tfIdfBasedLikeWeight" column stores the like weight of the user related to the attribute, which is calculated by using the TF-IDF score and past rating history of the user. "tfIdfBasedDislikeWeight" column stores the dislike weight of the user related to the attribute, which is also calculated by using the TF-IDF score and past rating history of the user. "attrPrefRatioBasedLikeWeight" column stores the like weight of the user related to the attribute, which is calculated by the combination of the TF-IDF score and the user attribute preference ratio. "attrPrefRatioBasedDislikeWeight" column stores the dislike weight of the user related to the attribute, which is also calculated by the combination of the TF-IDF score and the user attribute preference ratio. "attrCategoryPrefRatioBasedLikeWeight" column stores the like weight of the user related to the attribute, which is calculated by the combination of the TF-IDF score and the user category preference ratio. "attrCategoryPrefRatioBasedDislikeWeight" column stores the dislike weight of the user related to the attribute category, which is also calculated by the combination of the TF-IDF score and the user category preference ratio. The details of how to calculate the like and dislike weights of the user-attribute pairs according to the said three techniques are given in the next sections.

3.3.2.2.1 TF-IDF Score Based Like-Dislike Weight Calculation

The proposed user like-dislike profile's "tfIdfBasedLikeWeight" value and the "tfIdfBasedDislikeWeight" value are calculated by using the TF-IDF measurement technique. TF-IDF measurement technique is a common and mature measurement technique in recommender systems domain. But in this thesis work, while generating the user profiles, user's like and dislike weight values are calculated in a different manner. As we explained before, the recommendation scores which are greater than or equal to 6.0 is considered as positive ratings. "tfIdfBasedLikeWeight" value is calculated by generating weighted mean of positively rated movies' rating scores according to the attribute's calculated TF-IDF score. In the same way, "tfIdfBasedDislikeWeight" value is calculated by generating weighted mean of negatively rated movies' rating scores according to the attribute's calculated TF-IDF score.

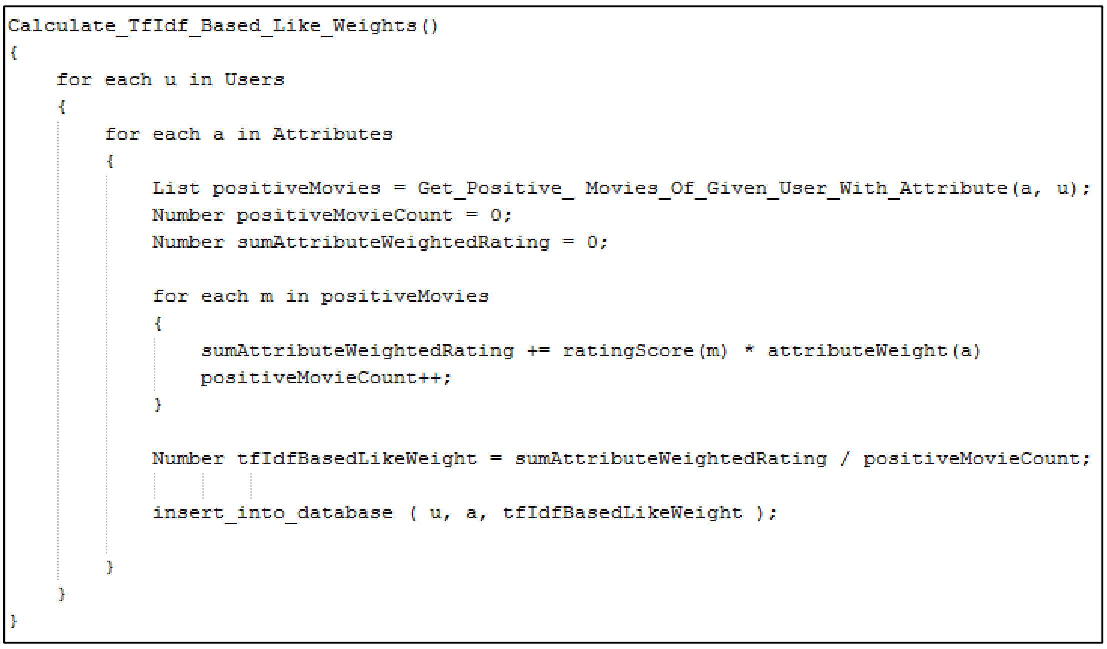
The "tfIdfBasedLikeWeight" of an attribute a_i for a user u_j is defined as follows;

$$\mathrm{TFIDFLW}^{*}(a_i, u_j) = \frac{\sum \mathrm{positiveRatingScore} * \mathrm{attributeWeight}_{a_i}}{\#\ \mathrm{likedMovieCount}_{a_i,u_j}} \qquad (3.6)$$

*TFIDFLW : TF-IDF Based Like Weight

where $\#\ \mathrm{likedMovieCount}_{a_i,u_j}$ is the total movie number which are rated positively by the given user u_j and which has the attribute a_i. positiveRatingScore is the rating score which is given by the user u_j to the positively rated movie. $\mathrm{attributeWeight}_{ai}$ is the attribute weight value of the attribute a_i which is the TF-IDF score of the said attribute a_i.

By using given formula (3.6), the like weights related to the attributes for all users are calculated by using attribute weight values which are calculated by TF-IDF measurement technique. (Figure 3.14) shows the pseudo code of how to calculate the "tfIdfBasedLikeWeight" values for all user-attribute pairs.



```
Calculate_TfIdf_Based_Like_Weights()
{
    for each u in Users
    {
        for each a in Attributes
        {
            List positiveMovies = Get_Positive_ Movies_Of_Given_User_With_Attribute(a, u);
            Number positiveMovieCount = 0;
            Number sumAttributeWeightedRating = 0;

            for each m in positiveMovies
            {
                sumAttributeWeightedRating += ratingScore(m) * attributeWeight(a)
                positiveMovieCount++;
            }

            Number tfIdfBasedLikeWeight = sumAttributeWeightedRating / positiveMovieCount;

            insert_into_database ( u, a, tfIdfBasedLikeWeight );

        }
    }
}
```

Figure 3.14 : Calculate TF-IDF Based Like Weight Algorithm

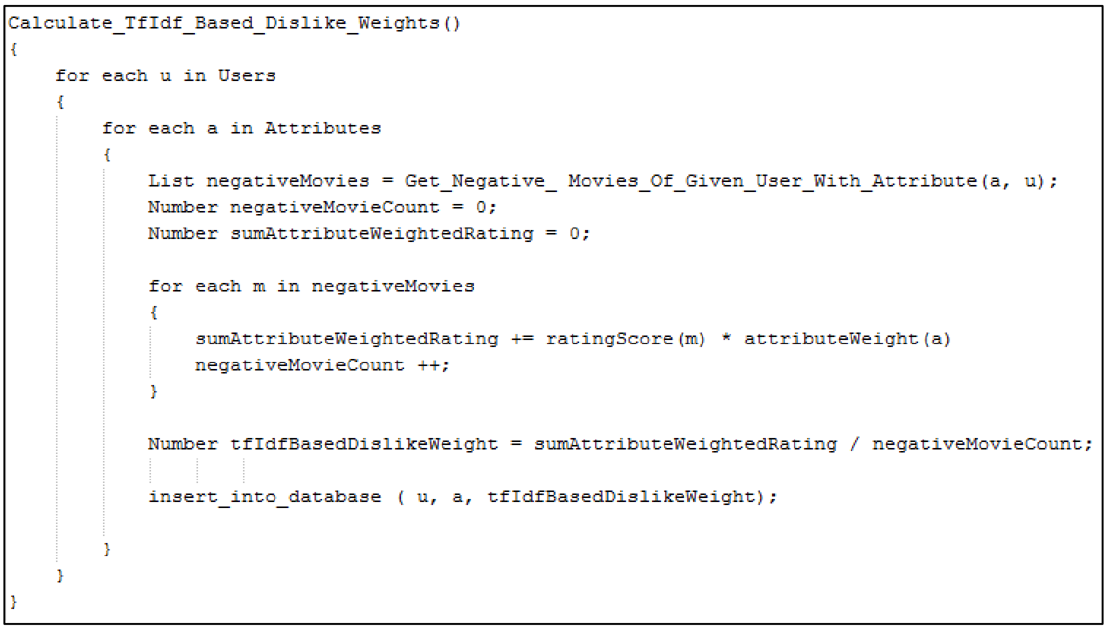
The "tfIdfBasedDislikeWeight" of an attribute a_i for a user u_j is defined as follows;

$$\mathrm{TFIDFDW}^*(a_i, u_j) = \frac{\sum \mathrm{negativeRatingScore} * \mathrm{attributeWeight}_{a_i}}{\#\, \mathrm{dislikedMovieCount}_{a_i,u_j}} \qquad (3.7)$$

*TFIDFDW : TF-IDF Based Dislike Weight

where $\#\, \mathrm{dislikedMovieCount}_{a_i,u_j}$ is the total movie number which are rated negatively by the given user u_j and which has the attribute a_i. negativeRatingScore is the rating score which is given by the user u_j to the negatively rated movie. $attributeWeight_{ai}$ is the TF-IDF score of the attribute a_i.

By using given formula (3.7), the dislike weights related to the attributes for all users are calculated by using TF-IDF scores of the attributes. (Figure 3.15) shows the pseudo code of how to calculate the "tfIdfBasedDislikeWeight" values for all user-attribute pairs.



```
Calculate_TfIdf_Based_Dislike_Weights()
{
    for each u in Users
    {
        for each a in Attributes
        {
            List negativeMovies = Get_Negative_ Movies_Of_Given_User_With_Attribute(a, u);
            Number negativeMovieCount = 0;
            Number sumAttributeWeightedRating = 0;

            for each m in negativeMovies
            {
                sumAttributeWeightedRating += ratingScore(m) * attributeWeight(a)
                negativeMovieCount ++;
            }

            Number tfIdfBasedDislikeWeight = sumAttributeWeightedRating / negativeMovieCount;

            insert_into_database ( u, a, tfIdfBasedDislikeWeight);

        }
    }
}
```

Figure 3.15 : Calculate TF-IDF Based Dislike Weight Algorithm

3.3.2.2.2 User Attribute Preference Based Like-Dislike Weight Calculation

The proposed user like-dislike profile's "attrPrefRatioBasedLikeWeight" and the "attrPrefRatioBasedDislikeWeight" values are calculated by using the combination of both proposed user attribute preference ratios and the TF-IDF scores. We propose that, by using this like and dislike weights, more accurate recommendation scores can be gathered because of the user attribute preference ratio's positive effect on generating the user profiles. "attrPrefRatioBasedLikeWeight" value is calculated by generating weighted mean of positively rated movies' rating scores according to both the attribute's TF-IDF score and the user's attribute preference ratio which is related to the attribute under consideration. Likewise, "attrPrefRatioBasedDislikeWeight" value is calculated by generating weighted mean of negatively rated movies' rating scores according to both the attribute's TF-IDF score and the user's attribute preference ratio related to the said attribute.

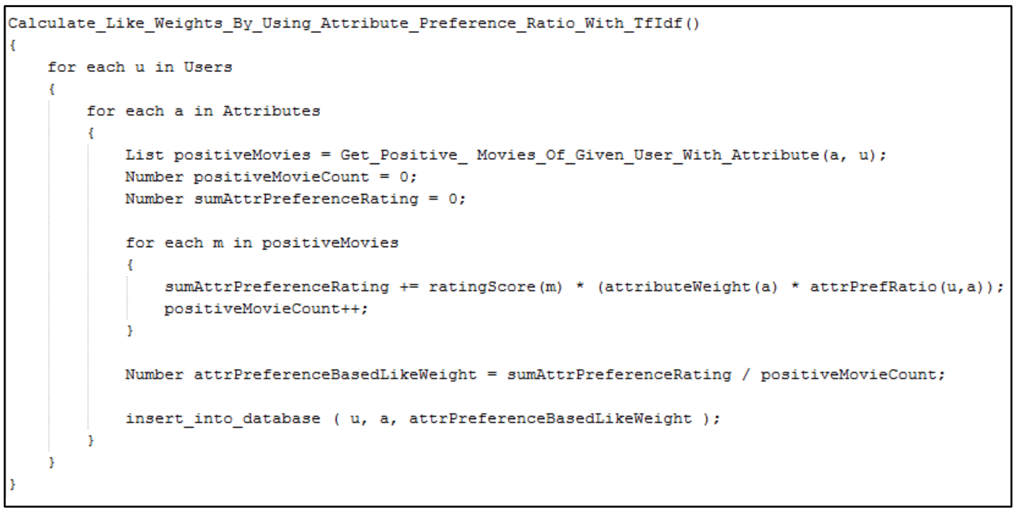
The "attrPrefRatioBasedLikeWeight" of an attribute a_i for a user u_j is defined as follows;

$$\mathrm{APRBLW}^*(a_i, u_j) = \frac{\sum \mathrm{positiveRating} * (\mathrm{attributeWeight}_{a_i} * \mathrm{attrPrefRatio}_{u_j,a_i})}{\#\,\mathrm{likedMovieCount}_{a_i,u_j}} \qquad (3.8)$$

*APRBLW : Attribute Preference Ratio Based Like Weight

where $\#\,\mathrm{likedMovieCount}_{a_i,u_j}$ is the total movie number which are rated positively by the given user u_j and which has the attribute a_i. positiveRating is the rating score which is given by the user u_j to the positively rated movie. $\mathrm{attributeWeight}_{a_i}$ is the TF-IDF score of the attribute a_i. $\mathrm{attrPrefRatio}_{u_j,a_i}$ is the attribute preference ratio that is calculated for user u_j for attribute a_i.

By using given formula (3.8) the like weights related to the attributes for all users are calculated according to the user's attribute preference ratios. (Figure 3.16) shows the pseudo code of how to calculate the "attrPrefRatioBasedLikeWeight" values for all user-attribute pairs.


```
Calculate_Like_Weights_By_Using_Attribute_Preference_Ratio_With_TfIdf()
{
    for each u in Users
    {
        for each a in Attributes
        {
            List positiveMovies = Get_Positive_ Movies_Of_Given_User_With_Attribute(a, u);
            Number positiveMovieCount = 0;
            Number sumAttrPreferenceRating = 0;

            for each m in positiveMovies
            {
                sumAttrPreferenceRating += ratingScore(m) * (attributeWeight(a) * attrPrefRatio(u,a));
                positiveMovieCount++;
            }

            Number attrPreferenceBasedLikeWeight = sumAttrPreferenceRating / positiveMovieCount;

            insert_into_database ( u, a, attrPreferenceBasedLikeWeight );
        }
    }
}
```

Figure 3.16 : Calculate Attribute Preference Based Like Weight Algorithm

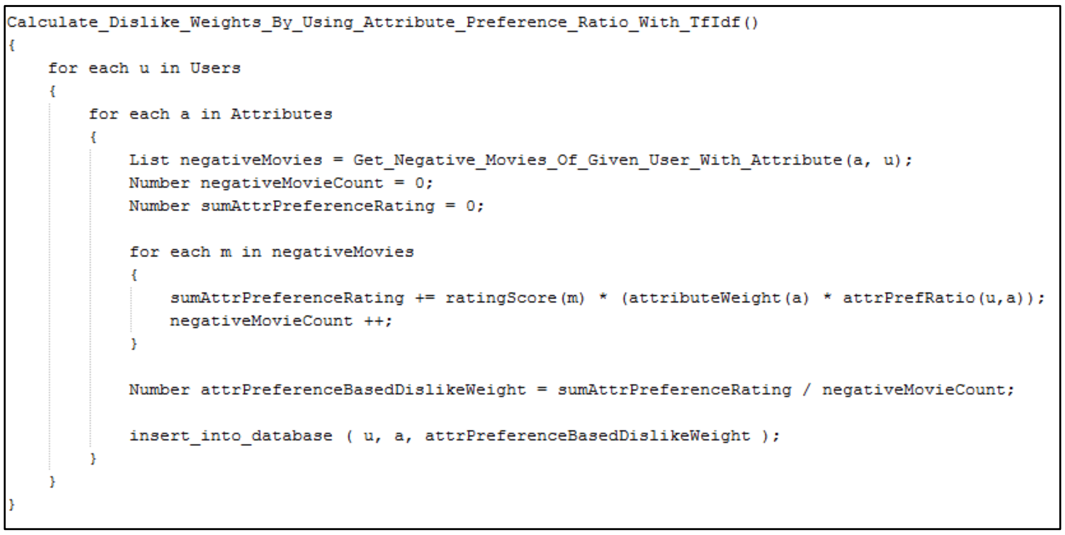
The "attrPrefRatioBasedDislikeWeight" of an attribute a_i for a user u_j is defined as follows;

$$\mathrm{APRBDW}^{*}(a_i, u_j) = \frac{\sum \mathrm{negativeRating} * (\mathrm{attributeWeight}_{a_i} * \mathrm{attrPrefRatio}_{u_j,a_i})}{\#\,\mathrm{dislikedMovieCount}_{a_i,u_j}} \tag{3.9}$$

*APRBDW : Attribute Preference Ratio Based Dislike Weight

where $\#\,\mathrm{dislikedMovieCount}_{a_i,u_j}$ is the total movie number which are rated negatively by the given user u_j and which has the attribute a_i. negativeRating is the rating score which is given by the user u_j to the negatively rated movie. $\mathrm{attributeWeight}_{a_i}$ is the TF-IDF score of the attribute a_i. $\mathrm{attrPrefRatio}_{u_j,a_i}$ is the attribute preference ratio that is calculated for user u_j for attribute a_i.

By using given formula (3.9) the dislike weights related to the attributes for all users are calculated by combining the TF-IDF scores of the attributes with the user attribute preference ratios. (Figure 3.17) shows the pseudo code of how to calculate the "attrPrefRatioBasedDislikeWeight" values for all user-attribute pairs.



```
Calculate_Dislike_Weights_By_Using_Attribute_Preference_Ratio_With_TfIdf()
{
    for each u in Users
    {
        for each a in Attributes
        {
            List negativeMovies = Get_Negative_Movies_Of_Given_User_With_Attribute(a, u);
            Number negativeMovieCount = 0;
            Number sumAttrPreferenceRating = 0;

            for each m in negativeMovies
            {
                sumAttrPreferenceRating += ratingScore(m) * (attributeWeight(a) * attrPrefRatio(u,a));
                negativeMovieCount ++;
            }

            Number attrPreferenceBasedDislikeWeight = sumAttrPreferenceRating / negativeMovieCount;

            insert_into_database ( u, a, attrPreferenceBasedDislikeWeight );
        }
    }
}
```

Figure 3.17 : Calculate Attribute Preference Based Dislike Weight Algorithm

3.3.2.2.3 User Category Preference Based Like-Dislike Weight Calculation

The proposed user like-dislike profile's "attrCategoryPrefRatioBasedLikeWeight" and the "attrCategoryPrefRatioBasedDislikeWeight" values are calculated by using the combination of both proposed user category preference ratios and the TF-IDF scores.

We propose that, by using this like and dislike weights, estimated recommendation scores should be more accurate than the pure TF-IDF measurement technique. But, we also expected that the estimated recommendation scores by using user's category preference ratios should be less accurate than the estimated recommendation scores by using the user's attribute preference ratios. The reason behind this expectation is that, user's category preference ratios are more abstract than the user's attribute preference ratios. This means that the user's attribute preference ratios describe the user preferences better than the category preference ratios. Because the category preference ratios are calculated according to the average sum of the attribute preference ratios of the user. Category preference ratios of users shows the average preference of the user related to the category and less descriptive than the user's attribute preference ratios.

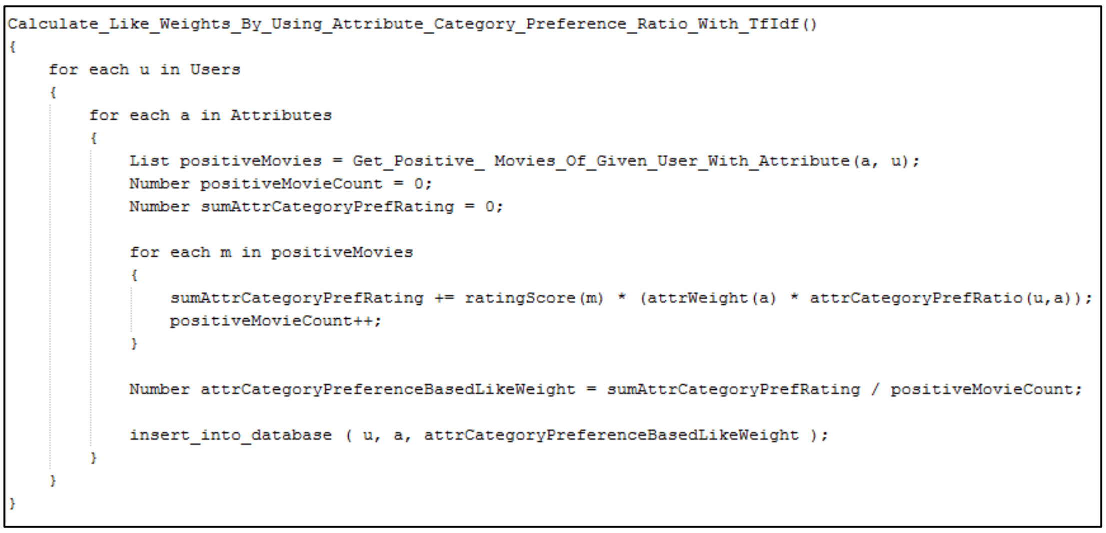
"attrCategoryPrefRatioBasedLikeWeight" value is calculated by generating weighted mean of positively rated movies' rating scores according to both the attribute's TF-IDF score and the user's category preference ratio which is related to the attribute's category. Likewise, "attrCategoryPrefRatioBasedDislikeWeight" value is calculated by generating weighted mean of negatively rated movies' rating scores according to both the attribute's TF-IDF score and the user's category preference ratio which is related to the attribute's category.

The "attrCategoryPrefRatioBasedLikeWeight" of an attribute a_i for a user u_j is defined as follows;

$$\mathrm{CPRBLW}^{*}(a_i, u_j) = \frac{\sum \mathrm{positiveRating} * (\mathrm{attrWeight}_{a_i} * \mathrm{attrCategoryPrefRatio}_{u_j, a_i})}{\#\ \mathrm{likedMovieCount}_{a_i, u_j}} \quad (3.10)$$

*CPRBLW : Category Preference Ratio Based Like Weight where $\# \text{likedMovieCount}_{a_i,u_j}$ is the total movie number which are rated positively by the given user u_j and which has the attribute a_i. positiveRating is the rating score which is given by the user u_j to the positively rated movie. attrWeight_{a_i} is the TF-IDF score of the attribute a_i. $\text{attrCategoryPrefRatio}_{u_j,a_i}$ is the category preference ratio that is calculated for user u_j for attribute a_i's category.

By using given formula (3.10) the like weights of all users are calculated according to the user's category preference ratios. (Figure 3.18) shows the pseudo code of how to calculate the "attrCategoryPrefRatioBasedLikeWeight" values for all user-attribute pairs.



```
Calculate_Like_Weights_By_Using_Attribute_Category_Preference_Ratio_With_TfIdf()
{
    for each u in Users
    {
        for each a in Attributes
        {
            List positiveMovies = Get_Positive_ Movies_Of_Given_User_With_Attribute(a, u);
            Number positiveMovieCount = 0;
            Number sumAttrCategoryPrefRating = 0;

            for each m in positiveMovies
            {
                sumAttrCategoryPrefRating += ratingScore(m) * (attrWeight(a) * attrCategoryPrefRatio(u,a));
                positiveMovieCount++;
            }

            Number attrCategoryPreferenceBasedLikeWeight = sumAttrCategoryPrefRating / positiveMovieCount;

            insert_into_database ( u, a, attrCategoryPreferenceBasedLikeWeight );
        }
    }
}
```

Figure 3.18: Calculate Category Preference Based Like Weight Algorithm

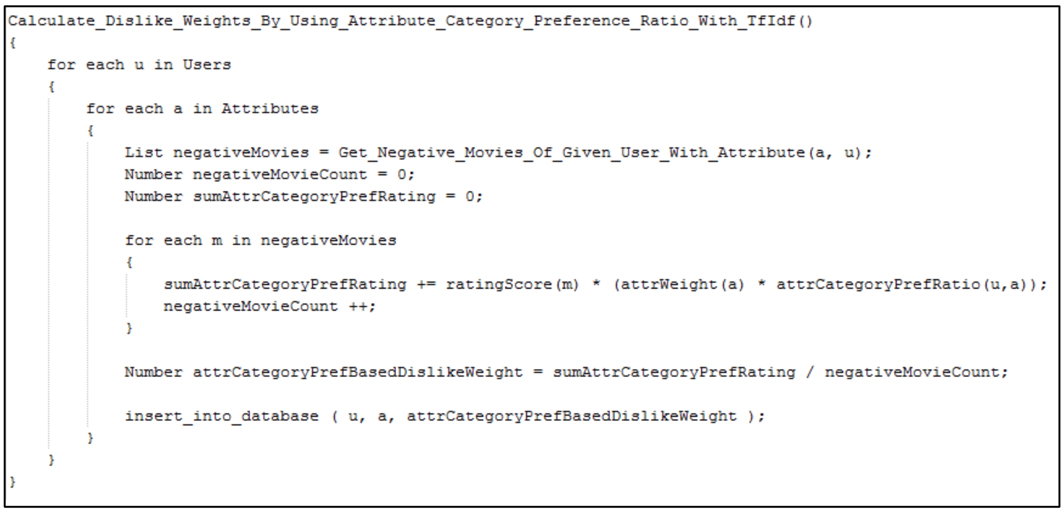
The "attrCategoryPrefRatioBasedDislikeWeight" of an attribute a_i for a user u_j is defined as follows;

$$\text{CPRBDW}^*(a_i, u_j) = \frac{\sum \text{negativeRating} * (\text{attrWeight}_{a_i} * \text{attrCategoryPrefRatio}_{u_j,a_i})}{\# \text{dislikedMovieCount}_{a_i,u_j}} \quad (3.11)$$

*CPRBDW: Category Preference Ratio Based Dislike Weight

where $\# \text{dislikedMovieCount}_{a_i,u_j}$ is the total movie number which are rated negatively by the given user u_j and which has the attribute a_i. negativeRating is the rating score which is given by the user u_j to the negatively rated movie. $\mathrm{attrWeight}_{a_i}$ is the TF-IDF score of the attribute a_i. $\mathrm{attrCategoryPrefRatio}_{u_j,a_i}$ is the category preference ratio that is calculated for user u_j for attribute a_i's attribute category.

By using given formula (3.11) the dislike weights of all users are calculated by combining the TF-IDF scores with the user category preference ratios. (Figure 3.19) shows the pseudo code of how to calculate the "attrCategoryPrefRatioBasedDislikeWeight" values for all user-attribute pairs.



```
Calculate_Dislike_Weights_By_Using_Attribute_Category_Preference_Ratio_With_TfIdf()
{
    for each u in Users
    {
        for each a in Attributes
        {
            List negativeMovies = Get_Negative_Movies_Of_Given_User_With_Attribute(a, u);
            Number negativeMovieCount = 0;
            Number sumAttrCategoryPrefRating = 0;

            for each m in negativeMovies
            {
                sumAttrCategoryPrefRating += ratingScore(m) * (attrWeight(a) * attrCategoryPrefRatio(u,a));
                negativeMovieCount ++;
            }

            Number attrCategoryPrefBasedDislikeWeight = sumAttrCategoryPrefRating / negativeMovieCount;

            insert_into_database ( u, a, attrCategoryPrefBasedDislikeWeight );
        }
    }
}
```

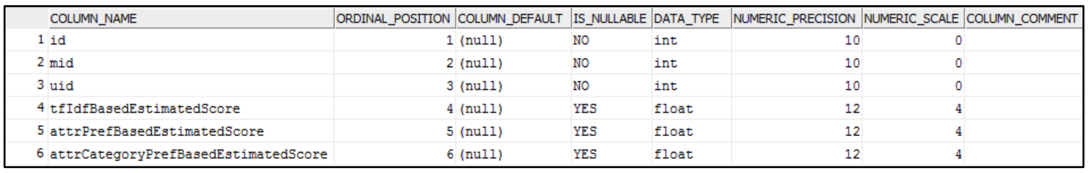
Figure 3.19: Calculate Category Preference Based Dislike Weight Algorithm

3.3.3. Recommendation Score Calculation

The proposed system presents a novel approach to predict the estimated rating scores for unseen movies. In [19], the user profile's like-dislike weights are calculated by using only the count of positive and negative movies, respectively. In spite of that; in our proposed system, the like-dislike weights are calculated by the weighted sum of given rating scores with the effects of attribute preference ratios. So, the formula which calculates the estimated score that we used is a novel approach, too. The estimated scores are calculated three times for three like-dislike pairs to show the positive influence of user preference ratios on the pure TF-IDF measurement technique and to compare the two user preference ratios positive affection degrees.

The implementation details of how to calculate estimated rating scores according to said three techniques are given in the following subsections.

The estimated ratings for unseen movies are stored in the database table named "EstimatedRating" in the proposed system. (Figure 3.20) shows the "EstimatedRating" table.



	COLUMN_NAME	ORDINAL_POSITION	COLUMN_DEFAULT	IS_NULLABLE	DATA_TYPE	NUMERIC_PRECISION	NUMERIC_SCALE	COLUMN_COMMENT
1	id	1	(null)	NO	int	10	0	
2	mid	2	(null)	NO	int	10	0	
3	uid	3	(null)	NO	int	10	0	
4	tfIdfBasedEstimatedScore	4	(null)	YES	float	12	4	
5	attrPrefBasedEstimatedScore	5	(null)	YES	float	12	4	
6	attrCategoryPrefBasedEstimatedScore	6	(null)	YES	float	12	4	

Figure 3.20 : EstimatedRating Table

In "EstimatedRating" table "id" column stores the unique id of the table and auto incremented by the system. "mid" column is a foreign key to the "Movies" table and keeps the movie which is related to the estimated rating score. "uid" column is a foreign key to the "Users" table and keeps the user id related to the estimated rating score. "tfIdfBasedEstimatedScore" column stores the estimated rating value which is calculated by using "tfIdfBasedlLikeWeight" and "tfIdfBasedDislikeWeight" values from user like-dislike profile and relies on the pure TF-IDF measurement technique. "attrPrefBasedEstimatedScore" column stores the estimated rating value which is calculated by using "attrPrefRatioBasedLikeWeight" and "attrPrefRatioBasedDislikeWeight" values from user like-dislike profile and relies on the user's attribute preference ratio values. "attrCategoryPrefBasedEstimatedScore" column stores the estimated rating value which is calculated by using "attrCategoryPrefRatioBasedLikeWeight" and "attrCategoryPrefRatioBasedDislikeWeight" values of the user like-dislike profile and relies on the user's category preference ratio values.

3.3.3.1 Recommendation Score Calculation by Using TF-IDF Based Like-Dislike Weight Scores

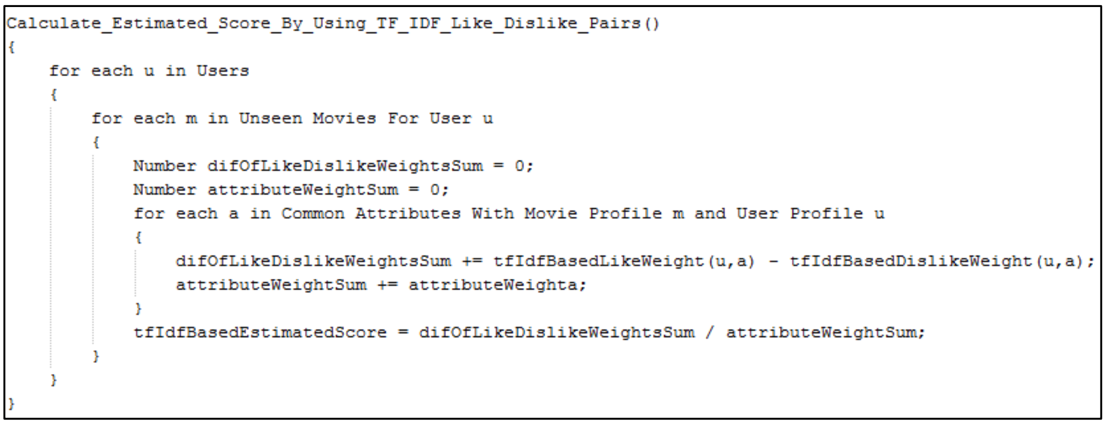
The estimated rating score for an unseen movie m_m, for user u_j by using the TF-IDF based like-dislike pairs is defined as follows;

$$\mathrm{ES}^*{}_{\mathrm{TF-IDF}}(\mathrm{u_j}, \mathrm{m_m}) = \frac{\sum_{\mathrm{i=1}}^{\mathrm{commonAttr}} \mathrm{tfIdfBasedLikeWeight}_{\mathrm{u_j a_i}} - \mathrm{tfIdfBasedDislikeWeight}_{\mathrm{u_j a_i}}}{\sum_{\mathrm{i=1}}^{\mathrm{commonAttr}} \mathrm{attributeWeight}_{\mathrm{a_i}}} \quad (3.12)$$

*$\mathrm{ES}_{\mathrm{TF\text{-}IDF}}$: Estimated Score Calculated By TF-IDF Based Like-Dislike Weights

where commonAttr represents the common attributes between movie profile of movie m_m and user like-dislike profile of user u_j. $\mathrm{tfIdfBasedLikeWeight}_{\mathrm{u_j a_i}}$ is the TF-IDF based like weight of user u_j related to attribute a_i. $\mathrm{tfIdfBasedDislikeWeight}_{\mathrm{u_j a_i}}$ is the TF-IDF based dislike weight of user u_j related to attribute a_i and $\mathrm{attributeWeight}_{\mathrm{a_i}}$ is the attribute weight of attribute a_i which is calculated by TF-IDF measurement technique.

The estimated scores related to TF-IDF measurement technique are calculated by using given formula (3.12). The results are then stored in the database. (Figure 3.21) shows the pseudo code of how to calculate the TF-IDF based estimated scores for unseen movie-user pairs.



```
Calculate_Estimated_Score_By_Using_TF_IDF_Like_Dislike_Pairs()
{
    for each u in Users
    {
        for each m in Unseen Movies For User u
        {
            Number difOfLikeDislikeWeightsSum = 0;
            Number attributeWeightSum = 0;
            for each a in Common Attributes With Movie Profile m and User Profile u
            {
                difOfLikeDislikeWeightsSum += tfIdfBasedLikeWeight(u,a) - tfIdfBasedDislikeWeight(u,a);
                attributeWeightSum += attributeWeighta;
            }
            tfIdfBasedEstimatedScore = difOfLikeDislikeWeightsSum / attributeWeightSum;
        }
    }
}
```

Figure 3.21: Calculate Estimated Score By TF-IDF Based Like Dislike Weights Algorithm

After calculating each estimated score for movie-user pair, a new row is inserted to the "EstimatedRating" table and the row's "tfIdfBasedEstimatedScore" column is filled with the predicted rating score.

3.3.3.2 Rating Prediction by Using User Attribute Preference Ratio Boosted Like-Dislike Weight Scores

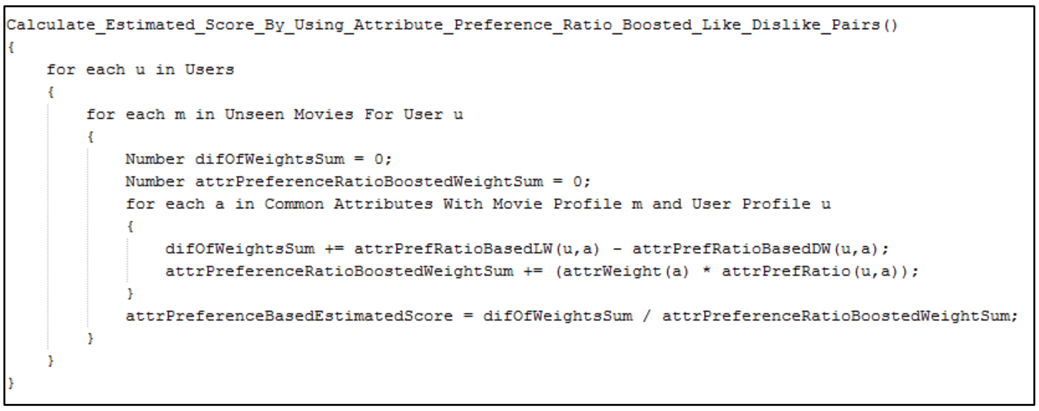
The estimated rating score for an unseen movie m_m, for user u_j by using the proposed user attribute preference ratio measurement technique based like-dislike pairs is defined as follows;

$$ES^{*}_{attrPref}(u_j, m_m) = \frac{\sum_{i=1}^{commonAtt} attrPrefRatioBasedLW_{u_j a_i} - attrPrefRatioBasedDW_{u_j a_i}}{\sum_{i=1}^{commonAttributes} (attributeWeight_{a_i} * attrPreferenceRatio_{u_j a_i})} \quad (3.13)$$

*$ES_{attrPref}$: Estimated Score Calculated By Attribute Preference Ratio Based Like-Dislike Weights

where commonAtt represents the common attributes between movie profile of movie m_m and user profile of user u_j. $attrPrefRatioBasedLW_{u_j a_i}$ is the attribute preference ratio based like weight of user u_j related to attribute a_i. $attrPrefRatioBasedDW_{u_j a_i}$ is the attribute preference ratio based dislike weight of user u_j related to attribute a_i. $attributeWeight_{a_i}$ is the attribute weight of attribute a_i which is calculated by TF-IDF measurement technique and $attrPreferenceRatio_{u_j a_i}$ is the attribute preference ratio of user u_j for attribute a_i.

The estimated scores related to attribute preference ratio measurement technique are calculated by using given formula (3.13). The results are then stored in the database. (Figure 3.22) shows the pseudo code of how to calculate the user attribute preference ratio boosted estimated scores for unseen movie-user pairs.



```
Calculate_Estimated_Score_By_Using_Attribute_Preference_Ratio_Boosted_Like_Dislike_Pairs()
{
    for each u in Users
    {
        for each m in Unseen Movies For User u
        {
            Number difOfWeightsSum = 0;
            Number attrPreferenceRatioBoostedWeightSum = 0;
            for each a in Common Attributes With Movie Profile m and User Profile u
            {
                difOfWeightsSum += attrPrefRatioBasedLW(u,a) - attrPrefRatioBasedDW(u,a);
                attrPreferenceRatioBoostedWeightSum += (attrWeight(a) * attrPrefRatio(u,a));
            }
            attrPreferenceBasedEstimatedScore = difOfWeightsSum / attrPreferenceRatioBoostedWeightSum;
        }
    }
}
```

Figure 3.22: Calculate Estimated Score By Attribute Preference Boosted Like-Dislike Weights Algorithm

After calculating each estimated score for movie-user pair, the row related to the user-movie pair from "EstimatedRating" table is found from the database and the row's "attrPrefBasedEstimatedScore" column is filled with the predicted rating score.

3.3.3.3 Rating Prediction by Using User Category Preference Ratio Boosted Like-Dislike Weight Scores

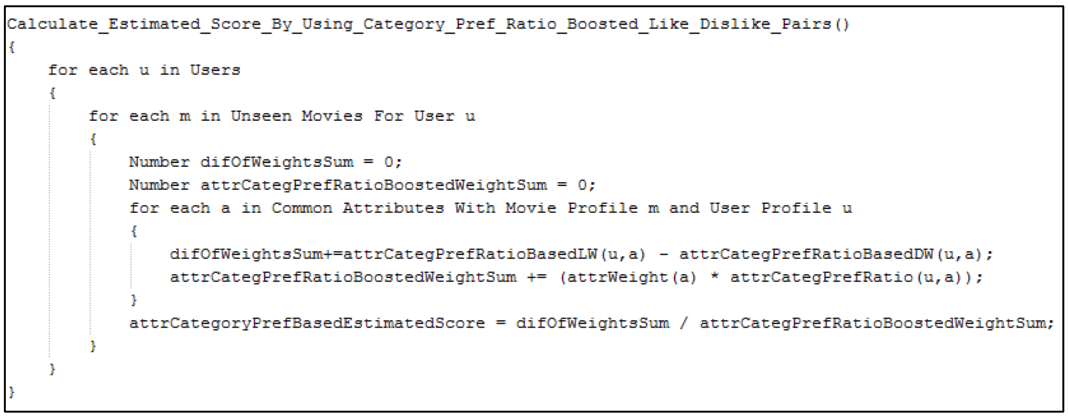
The estimated rating score for an unseen movie m_m, for user u_j by using the proposed user category preference ratio measurement technique based like-dislike pairs is defined as follows;

$$\mathrm{ES}^*{}_{\mathrm{categoryPref}}\left(u_j, m_m\right) = \frac{\sum_{i=1}^{\mathrm{cmmAttr}} \mathrm{attrCategPrefRatioBasedLW}_{u_j a_i} - \mathrm{attrCategPrefRatioBasedDW}_{u_j a_i}}{\sum_{i=1}^{\mathrm{cmmAttr}} (\mathrm{attributeWeight}_{a_i} * \mathrm{attrCategoryPrefRatio}_{u_j a_i})} \quad (3.14)$$

*$\mathrm{ES}_{\mathrm{categoryPref}}$: Estimated Score Calculated By Category Preference Ratio Based Like-Dislike Weights

where $\mathrm{cmmAttr}$ represents the common attributes between movie profile of movie m_m and user profile of user u_j. $\mathrm{attrCategPrefRatioBasedLW}_{u_j a_i}$ is the category preference ratio based like weight of user u_j related to attribute a_i's category. $\mathrm{attrCategPrefRatioBasedDW}_{u_j a_i}$ is the category preference ratio based dislike weight of user u_j related to attribute a_i's category. $\mathrm{attributeWeight}_{a_i}$ is the attribute weight of attribute a_i which is calculated by TF-IDF measurement technique and $\mathrm{attrCategoryPrefRatio}_{u_j a_i}$ is the category preference ratio of user u_j for attribute a_i.

The estimated scores related to category preference ratio measurement technique are calculated by using given formula (3.14). The results are then stored in the database at the category preference ratio related estimated column field. (Figure 3.23) shows the pseudo code of how to calculate the user category preference ratio boosted estimated scores for unseen movie-user pairs.



```
Calculate_Estimated_Score_By_Using_Category_Pref_Ratio_Boosted_Like_Dislike_Pairs()
{
    for each u in Users
    {
        for each m in Unseen Movies For User u
        {
            Number difOfWeightsSum = 0;
            Number attrCategPrefRatioBoostedWeightSum = 0;
            for each a in Common Attributes With Movie Profile m and User Profile u
            {
                difOfWeightsSum+=attrCategPrefRatioBasedLW(u,a) - attrCategPrefRatioBasedDW(u,a);
                attrCategPrefRatioBoostedWeightSum += (attrWeight(a) * attrCategPrefRatio(u,a));
            }
            attrCategoryPrefBasedEstimatedScore = difOfWeightsSum / attrCategPrefRatioBoostedWeightSum;
        }
    }
}
```

Figure 3.23: Calculate Estimated Score By Category Preference Boosted Like-Dislike Weights Algorithm

After calculating each estimated score for movie-user pair, the row related to the user-movie pair from "EstimatedRating" table is found from the database and the row's "attrCategoryPrefBasedEstimatedScore" column is filled with the predicted rating score.

At the end of the rating prediction process, for all user-unseen movie pairs, the rating scores are estimated by three different techniques.

CHAPTER 4

EVALUATION & RESULTS

This chapter presents the evaluation phase of the proposed user preference boosted content-based recommendation system approach and shows the results which are obtained from the testing phase of the system. First of all, the data characteristics which are used to evaluate the proposed system are explained in the first subsection. After that, the evaluation metrics that are used to appraise the proposed system are given in the next subsection. Then, the experiments with the given data set are given in the third subsection. In the experiments phase, the same data set is used with both TF-IDF measurement technique and the proposed approaches to consider the results in an objective manner and see the user preference ratio metrics' impact on results. Finally, the comparison between the proposed approaches and the TF-IDF measurement technique, the comparison between two proposed approaches and the comparison between the proposed approaches and some other known techniques from the literature are given in the last subsection.

4.1 Data Set

In the evaluation phase of the proposed user preference boosted content-based recommendation system, the dataset which belongs to a web page named cineworm.com [15] is used. People can sign in to this web page by using their Facebook [20] or Twitter [76] accounts, can rate and add comment to movies that they were already watched, can keep a will watch list, can share the movies that they watch with their friends and can see their friends' movie selections and ratings.

This web page is chosen to evaluate this proposed system because, the recommender systems in the literature generally uses the same data sets such as MovieLens [52], EachMovie (not available any more) or JesterJoke [32]. We wanted to use a new and a real dataset to get reliable results for recommendation process and for these reasons we contact with the developers of the cineworm.com [15]. After telling our intent related to their dataset, they were agreed to share their dataset with us.

In the dataset that we have used; initially 1428 users, 102491 movies and 26270 ratings exist. After removing the users which has no rating, removing the movies which are not related to any rating and removing the ratings which has no rating score from the database; the user count decrease to 473, the movie count decrease to 2226 and the rating count decreases to 17505.

The sparsity of a dataset is computed with the formula given below [31]:

$$Sparsity = 1 - \left(\frac{|R|}{|I| \times |U|}\right) \tag{4.1}$$

where R is the total rating count, I is the total item count and U is the total user count.

The popular datasets which are used in movie domain related recommender systems and their sparsity values are given in (Table 4.1):

Table 4.1 : Popular Datasets Sparsity Values

Dataset	Sparsity
EachMovie	0.9763
MovieLens 100K	0.978
MovieLens 1M	0.9575
MovieLens 10M	0.9869

The sparsity of cineworm.com dataset which we have used equals to 0.9548 (1 - 15781 / 2226 x 157) which is nearly as sparse as the known and used popular datasets.

The movie count related to rating count shows that a number of movies are rated by more than one users. (Figure 4.1) shows the movie statistics according to the given rating count.

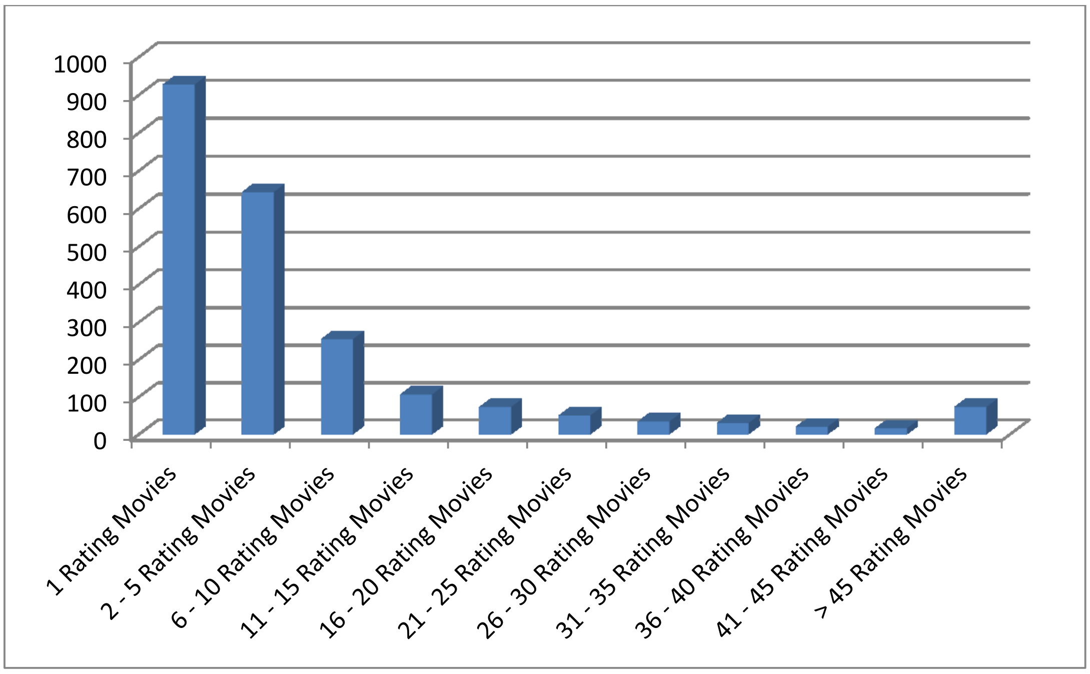


Figure 4.1: Movie-Rating Count Statistic

Because the proposed recommender system is a content-based recommender system and in content-based recommender systems, the users' past rating history plays the most important role in the creation of the user profiles, we analyze the user's in the dataset according to their rating counts. (Figure 4.2) shows the user statistics according to the rating counts.

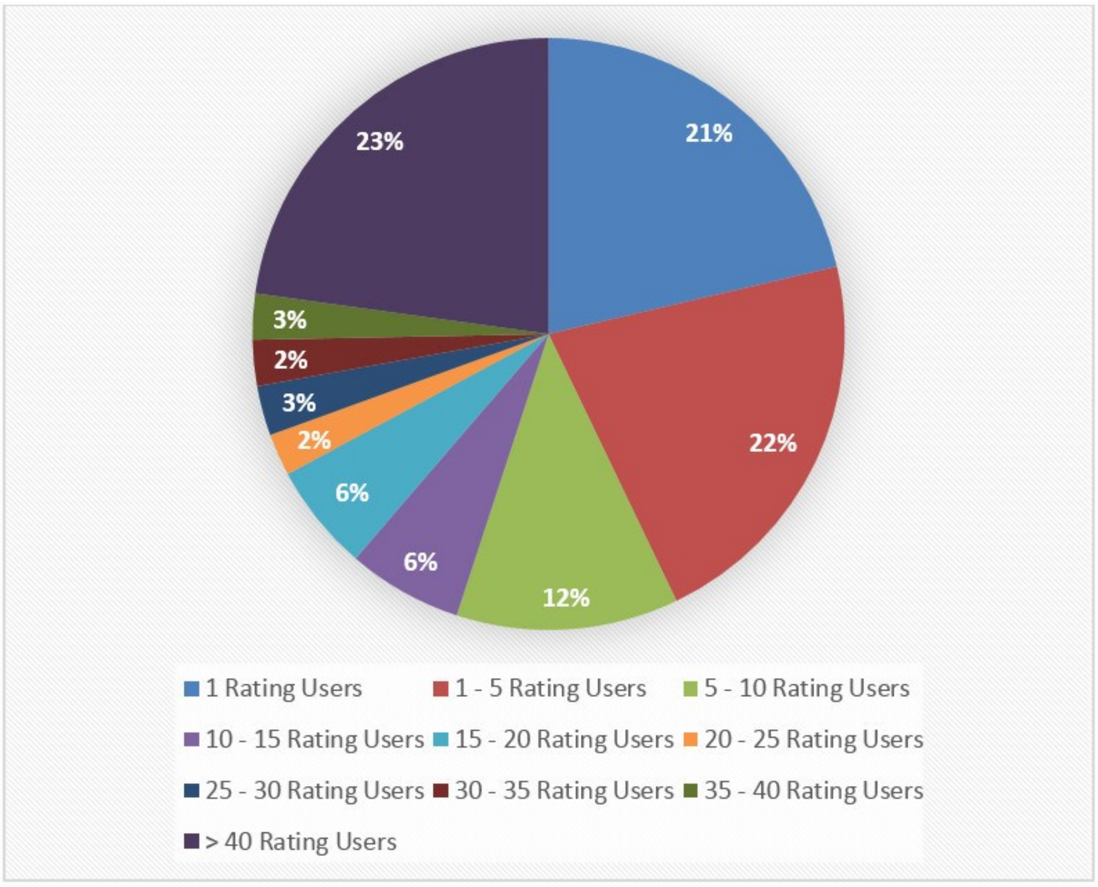


Figure 4.2: User-Rating Count Statistic

The above figure shows us that some users in the database don't have enough rating information to create reliable user profiles for them to recommend movies. In content based recommendation systems, to generate user profiles which reflects the user preferences, user should rate a number of items. For instance; in MovieLens [52] and EachMovie (not available any more) datasets; users have at least 20 ratings related to movies. These datasets are the most commonly used datasets in literature [79, 83, 60, 47]. So, we also removed the users who have less than 20 ratings from the database. As a result; after cleaning the database, 157 users with 15781 ratings exists in our database. The rating count decreases from 17505 to 15781. This means that approximately 10% of the all rating information in the database is removed from the database. This result indicates that the users which has less than 20 ratings don't have significant number of ratings.

The attribute categories related to movies in the cineworm.com data set are given in (Table 4.2). The attributes related to the "Genre", "Duration", "Actor", "Director" and "Release Year" categories are extracted from the cineworm.com data set. But the "Author" category's value exists for only 349 of the 2226 movies in the database. This means that only approximately 15% of the movies has the author information in the database. So; we discard the "Author" category from the attribute category list that we used in our proposed recommender system.

Table 4.2: Attribute Categories

Attribute Category Number	Attribute Category Name
1	Genre
2	Duration
3	Actor
4	Director
5	Release Year
6	Author

To extract the attributes related to the categories; we have developed a data extraction program in Java Programming Language and the information which is kept in JSON format in cineworm.com database is transferred to our database. While transferring the attributes to our database; a grouping is done related to the attributes which belong to "Duration" and "Release Year" categories. This grouping is done because of the fact that, a movie with 120 minutes duration should be considered in the same group with the movie with 115 minutes duration. The duration amounts are divided into the groups with 30 minute intervals, because 30 minute is the least meaningful time interval for movie domain according to us. In the same way, a movie whose release year is 2002 should be considered in the same group with a movie whose release year is 2003. The grouping related to the release year category is done by 10 year intervals up to 2000s. Because in movie domain, the movies are called as "A movie which belongs to 70s" or "A movie which belongs to 80s". After 2000, we grouped the release year category by 5 years intervals because of the fact that a movie which belongs to a year which is greater than 2000 is a movie which is recorded in near past and more people are possible to watch that movies. So, using 5 year intervals instead of 10 year intervals is more suitable to distinguish movies which belong to near past.

The grouping which is done related to "Duration" attribute category is given in (Table 4.3) and the grouping which is done related to "Release Year" attribute category is given in (Table 4.4).

Table 4.3: Duration Attribute Values

Duration Attribute Value
Less Than 30
30-59
60-89
90-119
120-149
More Than 149

Table 4.4: Release Year Attribute Values

Release Year Attribute Value
Before 1960
1960-1969
1970-1979
1980-1989
1990-1994
1995-1999
2000-2004
2005-2009
After 2009

The genres that movies in the cineworm.com dataset can have are listed in (Table 4.5) and the movies in the database can have more than one genres. In addition to this, movies can have multiple directors and actors. The genres, actors and directors are not grouped and each distinct attribute is added to our database related to these categories.

Table 4.5: Genre Attribute Values

Genre Attribute Value	Genre Attribute Value	Genre Attribute Value
Action	Eastern	Indie
Adventure	Erotic	Music
Animation	Family	Musical
Comedy	Fantasy	Mystery
Crime	Foreign	Neo-noir
Disaster	History	Noir
Documentary	Holiday	Road
Drama	Horror	Romance
Science Fiction	Sports	TV
Short	Suspense	War
Sport	Thriller	Western

4.2 Evaluation Metrics

In the proposed recommender system; the database includes the movies, the users and the ratings that users give to movies in the system. To evaluate the proposed system, the rating table in the database needs to be divided into 2 parts. One part is needed for training phase of the proposed approach, the second part is needed for the testing phase of the proposed approach. So, we divided the rating table in our database into the test and train table pairs 3 times with the 10% and 90% of the rating information, respectively. As a result, we have 3 randomly divided test-train rating table pairs. With the train part of the rating table, the proposed system is trained and with the test part of the rating table the proposed system is evaluated. This training and testing phases are repeated for 3 times for 3 randomly divided train-test rating table pairs.

To evaluate the proposed recommender system, we used the decision support metrics: precision, recall, F-measure and accuracy metrics [31]. We choose those metrics in evaluation phase because most of the recommender systems in the literature [6, 1, 77, 68] use those evaluation metrics in their evaluation process.

To evaluate recommender systems, ROC curves can also be used. This curve represents the precision and recall metric results combination. The F-Measure metric is another metric which summarizes the precision and recall of ROC curve and can be used in place of ROC curve [26]. We prefer to use F-Measure instead of ROC curve.

Mean Absolute Error (MAE) metric is another approach which can be used to test the prediction accuracy and this metric calculates the average deviation between predicted rating score and the actual rating score [31]. According to [45] and [14]; mean absolute error metric cannot reflect the real user experience. Because user's in the real world are interested in whether an item is recommended or not. But, mean absolute error metric measures the accuracy by only relying to the rating scores. The evaluated rating score related to that item is not as important as the predicted classification of the item: liked / not liked. Because of these reasons; we rely on the precision, recall, F-measure and accuracy metrics to evaluate the proposed recommendation approach.

To use those metrics; the predicted rating scores and the actual rating scores should be converted to the binary scale: like and dislike. As we mentioned before; the rating scores less than 6.0 is considered as negative ratings and the rating scores equal to or more than 6.0 is considered as positive ratings. We categorize the recommendation results as shown in the (Table 4.6) to calculate the evaluation metrics.

Table 4.6: Categorization of recommendation results

	Actual Like	**Actual Dislike**
Predicted as Like	TP	FP
Predicted as Dislike	FN	TN

The movies whose ratings were marked as liked (actual like) and recommended to the users by the proposed system (predicted as like) are considered as true positives

(TP). The movies whose ratings were marked as liked (actual like) but not recommended to the users by the proposed system (predicted as dislike) are considered as false negatives (FN). The movies whose ratings were marked as disliked (actual dislike) but recommended to the users by the proposed system (predicted as like) are considered as false positives (FP). The movies whose ratings were marked as disliked (actual dislike) and not recommended to the users by the proposed system (predicted as dislike) are considered as true negatives (TN).

After categorizing the recommendation results; then we calculated the precision, recall, F-measure and accuracy metrics for each train-test rating table pairs with the experiments that are done for the proposed recommender system by using the formulas (Equation 4.2), (Equation 4.3), (Equation 4.4) and (Equation 4.5), respectively.

$$Precision = \frac{TP}{TP+FP} \tag{4.2}$$

Precision can be defined as the ratio of number of predicted as liked and defined as liked movies -in other words predicted as liked correctly- to the number of movies that are predicted as liked. This ratio shows the probability of a recommended movie is really liked by the user.

$$Recall = \frac{TP}{TP+FN} \tag{4.3}$$

Recall can be defined as the ratio of the number of predicted as liked and defined as liked movies to the number of movies which are defined as liked in the system. This ratio shows the probability of a liked movie is really recommended by the system to the user.

$$F - measure = \frac{(\beta^2 + 1) * Precision * Recall}{\beta^2 * Precision + Recall} \tag{4.4}$$

F-measure is a combination of precision and recall metrics. The F-measure's result is balanced when β = 1. If β is defined as β > 1, then it gives more importance to precision metric; and recall metric otherwise. In our experiments, we define β as 1, because there is no reliable reason to give more importance to one of the metrics. So we use the balanced F-measure score.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \tag{4.5}$$

Accuracy can be defined as the ratio of the sum of predicted as liked and defined as liked movies and predicted as disliked and defined as disliked movies, in other words the number of movies which are recommended correctly, to the total possible recommendations.

4.3 Experiments

In this section, the experiments done through this thesis work will be described. To evaluate the proposed recommender system, we divided the rating table into 3 randomly selected test and train rating table pairs as we said before and did experiments 3 times for those pairs. In each experiment; we generated 3 different recommendation scores for each movie in the test part. One of them is generated according to the TF-IDF measurement technique which is a mature content based recommendation technique. The second recommendation score is generated according to the proposed user category preference boosted content based recommendation approach. Finally, the last recommendation score is generated according to the proposed user attribute preference boosted content based recommendation approach. The recommendation score related to TF-IDF measurement technique is calculated to compare the proposed approaches with this mature technique with the same data set and gather reliable comparison results.

The experiments were done at a personal computer which has 8 GB RAM and 1.7 GHz CPU with i5 processor. The database management operations were done via PHP-MyAdmin and SQLDeveloper tools. MySQL was used as a database while doing experiments. The proposed recommender system's application part was developed at JAVA programming language and developed program was run at Eclipse Juno tool.

4.3.1. Experiments for the First Train-Test Rating Pair

The experiment results related to the first train-test rating table pair according to the proposed approaches are given under this subsection. The proposed recommender system was trained with the first train-test rating table pair's train part and the experiment results related to the test part are given below:

(Table 4.7) shows the experiment results for the TF-IDF measurement technique for the first test rating table.

Table 4.7: Experiment Results for TF-IDF Measurement with First Train-Test Pair

Evaluation Metric	Value
Precision	0.94
Recall	0.73
F-Measure	0.82
Accuracy	0.71

(Table 4.8) shows the experiment results for the proposed user category preference boosted content based recommendation system approach for the first test rating table.

Table 4.8: Experiment Results for User Category Preference Boosted Content Based RS with First Train-Test Pair

Evaluation Metric	Value
Precision	0.94
Recall	0.75
F-Measure	0.83
Accuracy	0.73

(Table 4.9) shows the experiment results for the proposed user attribute preference boosted content based recommendation system approach for the first test rating table.

Table 4.9: Experiment Results for User Attribute Preference Boosted Content Based RS with First Train-Test Pair

Evaluation Metric	Value
Precision	0.94
Recall	0.78
F-Measure	0.85
Accuracy	0.75

4.3.2. Experiments for the Second Train-Test Rating Pair

The experiment results related to the second train-test rating table pair according to the proposed approaches are given under this subsection. The proposed recommender system was trained with the second train-test rating table pair's train part and the experiment results related to the test part are given below:

(Table 4.10) shows the experiment results for the TF-IDF measurement technique for the second test rating table.

Table 4.10: Experiment Results for TF-IDF Measurement with Second Train-Test Pair

Evaluation Metric	Value
Precision	0.94
Recall	0.75
F-Measure	0.83
Accuracy	0.73

(Table 4.11) shows the experiment results for the proposed user category preference boosted content based recommendation system approach for the second test rating table.

Table 4.11: Experiment Results for User Category Preference Boosted Content Based RS with Second Train-Test Pair

Evaluation Metric	**Value**
Precision	0.94
Recall	0.77
F-Measure	0.85
Accuracy	0.74

(Table 4.12) shows the experiment results for the proposed user attribute preference boosted content based recommendation system approach for the second test rating table.

Table 4.12: Experiment Results for User Attribute Preference Boosted Content Based RS with Second Train-Test Pair

Evaluation Metric	**Value**
Precision	0.94
Recall	0.80
F-Measure	0.86
Accuracy	0.77

4.3.3. Experiments for The Third Train-Test Rating Pair

The experiment results related to the third train-test rating table pair according to the proposed approaches are given under this subsection. The proposed recommender system was trained with the third train-test rating table pair's train part and the experiment results related to the test part are given below:

(Table 4.13) shows the experiment results for the TF-IDF measurement technique for the third test rating table.

Table 4.13: Experiment Results for TF-IDF Measurement with Third Train-Test Pair

Evaluation Metric	**Value**
Precision	0.94
Recall	0.75
F-Measure	0.83
Accuracy	0.73

(Table 4.14) shows the experiment results for the proposed user category preference boosted content based recommendation system approach for the third test rating table.

Table 4.14: Experiment Results for User Category Preference Boosted Content Based RS with Third Train-Test Pair

Evaluation Metric	**Value**
Precision	0.95
Recall	0.76
F-Measure	0.85
Accuracy	0.74

(Table 4.15) shows the experiment results for the proposed user attribute preference boosted content based recommendation system approach for the third test rating table.

Table 4.15: Experiment Results for User Attribute Preference Boosted Content Based RS with Third Train-Test Pair

Evaluation Metric	Value
Precision	0.95
Recall	0.79
F-Measure	0.86
Accuracy	0.77

After implementing all the experiments, then the categories' importance levels are calculated according to the attributes' attribute weights under each category. The category weights which are calculated by dividing the sum of attribute weights to the number of attributes are as follows:

• Actor Category Weight	= 11,30
• Director Category Weight	= 9,95
• Genre Category Weight	= 4,50
• Release Year Category Weight	= 1,07
• Duration Category Weight	= 0,91

This category weights indicates that while calculating the predicted rating scores the actor category affects the results more than all other categories. So, the category importance order can be given like that: actor, director, genre, release year and duration.

4.4 Comparison

In this section, experiment results of the proposed approaches are compared with the results of the TF-IDF measurement technique. In addition to this comparison; the proposed approaches' experiment results are compared with each other. Finally, the experiment results are compared with some other known techniques from the literature.

4.4.1. Comparison of Proposed Approaches with TF-IDF Measurement Technique

(Table 4.16) shows the average precision, recall, F-measure and accuracy metric results for both proposed approaches and TF-IDF measurement technique.

Table 4.16: Comparison of the Proposed Approaches with the TF-IDF Based Recommendation Approach

Approach	**Precision (%)**	**Recall (%)**	**F-Measure (%)**	**Accuracy (%)**
TF-IDF Based Recommendation	94	74.3	82.6	72.3
Proposed User Category Preference Boosted Content Based Recommendation	94.3	76	84.3	73.6
Proposed User Attribute Preference Boosted Content Based Recommendation	94.3	79	85.6	76.3

The comparison between the proposed user preference boosted content based recommendation approaches with the TF-IDF based recommendation approach related to evaluation metrics is illustrated as Figure 4.3.

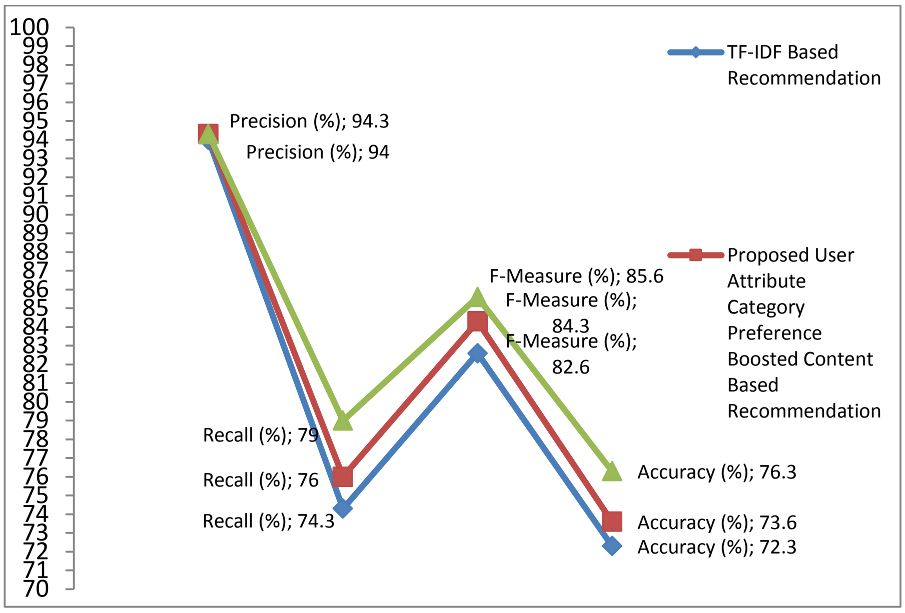


Figure 4.3: Evaluation Metrics Related Comparison

It is seen from the (Table 4.16) and (Figure 4.3); both of the two proposed approaches results are better than the classical TF-IDF measurement technique based recommendation approach, for all evaluation metrics. This is because of the fact that; the proposed approaches consider the users' preference information in addition to the attributes' TF-IDF measurement score. We expected to get more accurate recommendation scores by adding the users' preference information to the movies' attribute characteristics which TF-IDF scores reflect. The results which are seen from (Table 4.16) and (Figure 4.3) indicate that; by adding users' preference information, more reliable recommendation results can be gathered.

The comparison between the proposed user preference boosted content based recommendation approaches with the TF-IDF measurement technique based recommendation approach related to precision, recall, F-Measure and accuracy evaluation metrics are illustrated as (Figure 4.4), (Figure 4.5), (Figure 4.6) and (Figure 4.7), respectively.

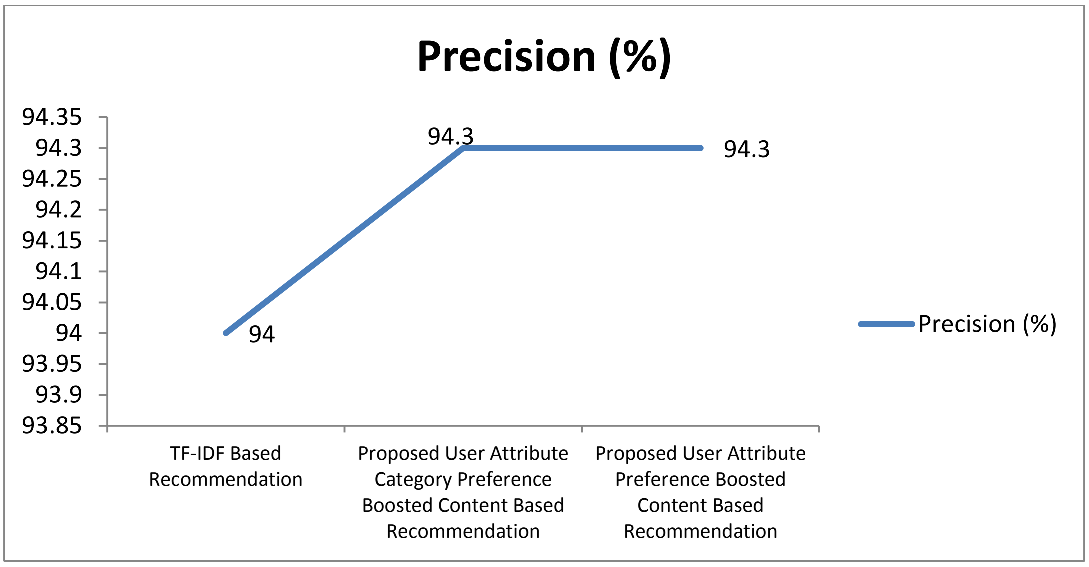


Figure 4.4: Precision Evaluation Metric Related Comparison

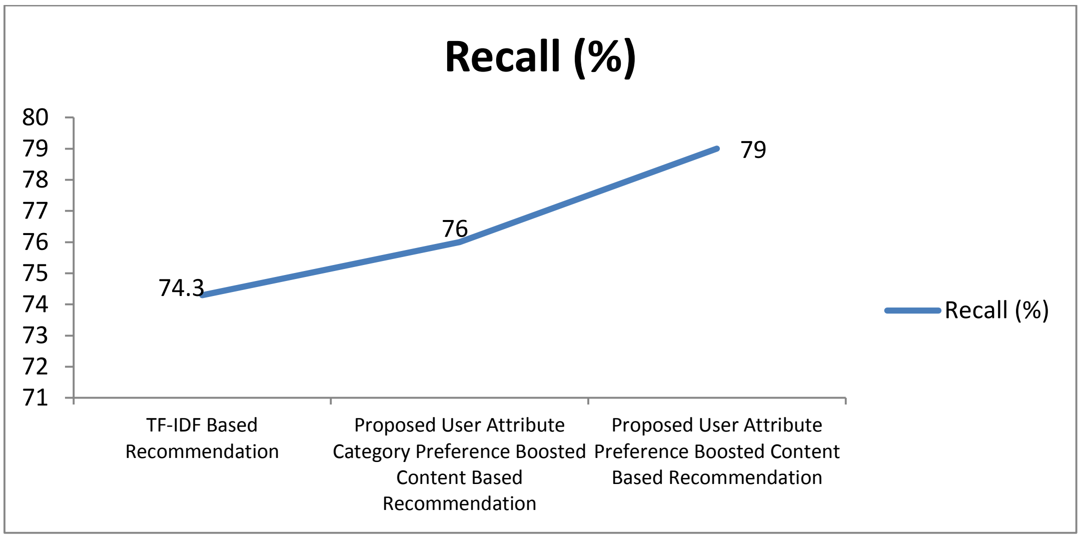


Figure 4.5: Recall Evaluation Metric Related Comparison

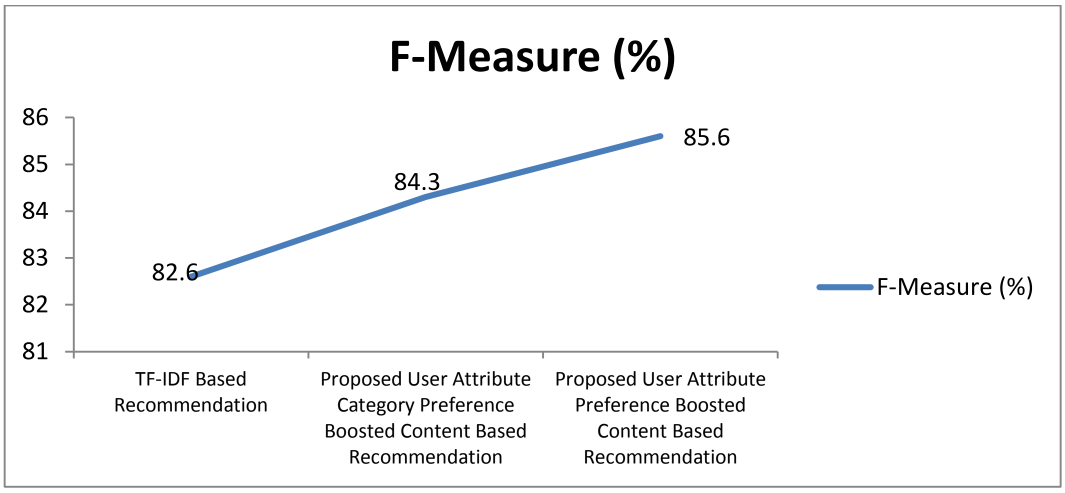


Figure 4.6: F-Measure Evaluation Metric Related Comparison

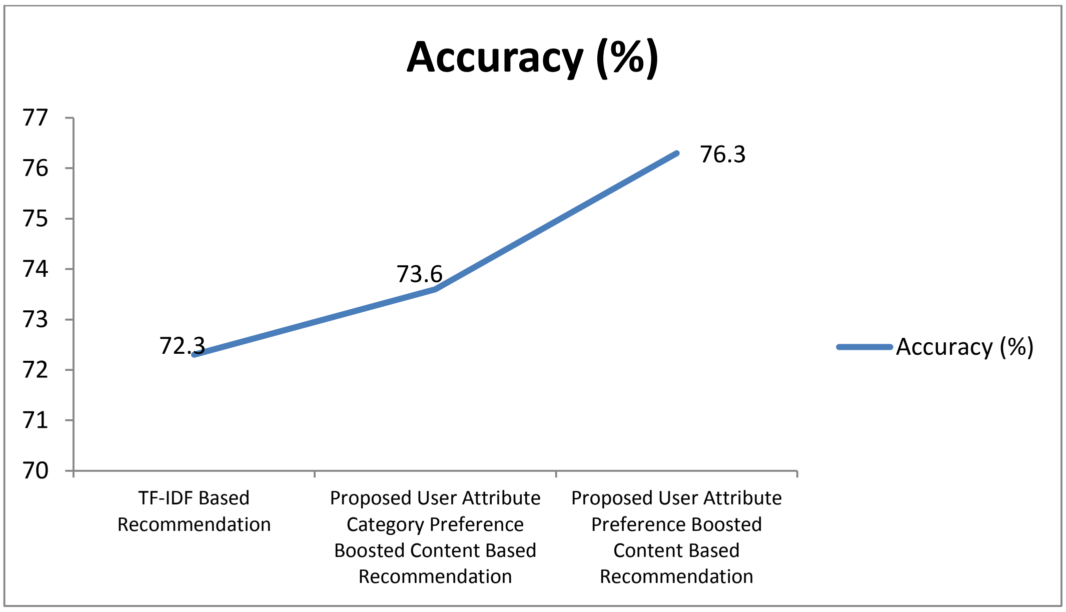


Figure 4.7: Accuracy Evaluation Metric Related Comparison

The lowest difference between the said three techniques is at the precision metric's results as seen from (Table 4.16) and (Figure 4.4). This evaluation metric shows the ratio of the number of correctly recommended movies to the number of recommended movies to the users. So, by finding the really relevant movies related to users, all three recommendation approaches generate similar results. Contrary to this, the highest difference between the proposed approaches with the TF-IDF measurement approach is at the recall metric's results as seen from (Table 4.16) and (Figure 4.5). This evaluation metric shows the ratio of the number of correctly recommended movies to the number of marked as liked movies by the users. So, the proposed approaches generate better results for recommending the liked movies to the users. This means that, the proposed approaches recommend liked movies more correct than the TF-IDF measurement approach. In addition to this, the recommended approaches generate better results for both F-Measure and accuracy evaluation metrics as seen from (Table 4.16), (Figure 4.6) and (Figure 4.7). The F-Measure metric is a combination of precision and recall metrics and to calculate this metric; as we said before, we give equal weights to precision and recall metrics. Even if the three approaches have similar results for precision metric, by getting better results from F-Measure evaluation metric, it can be said that the proposed approaches generate more reliable recommendation scores than the TF-IDF measurement approach. Accuracy metric shows the number of movies which are recommended correctly to the total possible recommendations, as we said before. By getting better results from this evaluation metric, it can be said that the proposed approaches generate more accurate recommendations than the TF-IDF measurement technique approach.

4.4.2. Comparison of Proposed Approaches with Each Other

It is also seen from the (Table 4.16) and (Figure 4.3); for each evaluation metric, the proposed user attribute preference boosted content based recommendation approach's results are better than the proposed user category preference boosted content based recommendation approach's results. The reason behind this result is the user's attribute preference information includes more detailed information related to user preferences than the user's category preference information. Because; in user's category preference related approach, the user's preferences related to the attributes under a specific category is summarized and a final category preference ratio is calculated. So, this generalization causes getting less accurate results than the user's attribute preference approach. As we said before, user's category preference ratio is a one level abstraction of the user's attribute preference ratio. Correspondingly, the recommendation scores which are calculated according to user category preference approach are generated with less detailed information. As a result, user category preference based approach's results are less accurate than the user attribute preference based approach's results.

4.4.3. Comparison of Proposed Approaches with Approaches from Literature

(Table 4.17) shows the precision, recall and F-measure metric results for both proposed approaches and some known approaches from literature.

Table 4.17: Comparison of the Proposed Approaches with Other Approaches

Approach	**Precision (%)**	**Recall (%)**	**F-Measure (%)**
HybridMovieRecommender	83	34	-
MovieMagician Feature-Based	61	75	67.3
MovieMagician Clique-Based	74	73	73.5
MovieMagician Hybrid	73	56	63.4
MovieLens	66	74	69.8
OPENMORE	62	91.7	74.1
AttributeBasedRecommender Collaborative Filtering	67.4	34.2	45.4
AttributeBasedRecommender Content-Based	69.8	35.7	47.2
TF-IDF Based Recommendation	94	74.3	82.6
Proposed User Category Preference Boosted Content Based Recommendation	**94.3**	**76**	**84.3**
Proposed User Attribute Preference Boosted Content Based Recommendation	**94.3**	**79**	**85.6**

HybridMovieRecommender [6] is a movie recommender system which combines collaborative filtering with content-based recommendation and uses a dataset which is obtained from 260 users and includes 45,000 movie ratings. MovieMagician [24] is a recommender system which uses hybrid recommendation approach and uses two different datasets to evaluate the system: a locally collected dataset at University of Saskatchewan and a movie dataset which is reachable from DEC Research Systems Center. In this recommender system; kinds, directors and actors are defined as features of a movie. MovieLens [49] uses collaborative filtering approach and OPENMORE [36] uses content-based recommendation approach to recommend movies by using MovieLens dataset [52]. AttributeBasedRecommender [68] is a material recommender system which uses an attribute-based multidimensional approach for recommendation by using a real-world dataset which is obtained from the usage data of a course management system named Moodle. In addition to them, TF-IDF approach is implemented by using our data set to make a fair comparison.

It can be seen from the (Table 4.17) that, proposed approaches outperform these approaches according to precision and F-measure metrics. Only OPENMORE [36] approach outperforms the proposed approaches at recall metric. But, the F-measure metric is the combination of precision and recall metrics. So, by using the F-measure metric as the base metric, we can say that the proposed approaches generate better results than the said approaches at (Table 4.17).

CHAPTER 5

CONCLUSION AND FUTURE WORK

This thesis presents a novel content-based recommendation approach which uses the user preference information that is extracted via a new approach. The item attributes' and the item attribute categories' importance levels in the eyes of users are calculated by a statistical point of view. Then, this importance levels are used while the user profiles are generating. User's past preferences are divided into two sets according to the given rating scores to calculate the importance levels: Like Set and Dislike Set. Each attribute's importance level are then calculated by mining the like set and dislike set.

The main contribution of our approach is that by including the item attribute's and item attribute category's importance levels according to each user while creating the user profiles will increase the accuracy of the user profiles and consequently increase the accuracy of predicted rating scores.

Our first hypothesis was, if an item attribute is discriminative for a user, then the user will rate each item which has this specific attribute in the same way, either positively or negatively. By looking at the rating scores in this point of view, we extract the hidden item attributes which effect the user while user gives a rating score to an item. We propose that; by extracting the item attribute's distinctiveness according to each user, more user oriented profiles can be generated. Generating more user oriented profiles is one of the main goals of recommender systems. Because, the main aim of recommender systems is to generate more personalized recommendations. Also to generate more personalized recommendations, user preferences should be well understood by the system and user profiles should reflect the user preferences correctly.

Our second hypothesis was, including more discriminative item attributes' weight values to the rating prediction process more than less discriminative one's weights in proportion to the distinctiveness scores will increase the accuracy of predicted rating scores. After calculating the distinctiveness of each item attribute and item attribute category, we include the discriminative item attributes and item attribute categories weights to the rating prediction process more than less discriminative one's weights by a novel rating score prediction algorithm.

To the best of our knowledge, in the literature there is not any similar work which distinguishes the item attributes' and item attribute categories' importance levels according to each user by a statistical approach which is similar to the proposed approach. Because of this reason; to evaluate the proposed approach in a fair way, we also implement the traditional content-based recommendation approach which uses TF-IDF measurement with our data set to compare the results.

In the evaluation phase of our proposed approach, we have used a new and a real dataset which is obtained from the website named "cineworm.com" [15]. We choose to use this dataset, because we wanted to use a real dataset which is extracted from a real world application to test our approach. To use this dataset, we clean the original dataset and convert the website's database tables into a different form which is suitable for our proposed approach. Then, we divided our dataset three times to the two randomly divided parts: train part and test part. We apply our recommendation approach to these train-test parts three times. We used the most common evaluation metrics like precision, recall, f-measure and accuracy to evaluate the proposed approaches.

We evaluate our proposed two approaches separately. Our first approach uses item attribute's importance levels while generating user profiles and calculating predicted scores. Our second approach is a one level abstraction of the former one. In the second approach, item attribute categories' importance levels are used instead of item attribute importance levels. When we compare our two proposed approaches, the first approach which considers item attribute importance levels outperforms the second approach in all metrics. Because, with the first approach more detailed user profiles are generated. In the second approach user profiles are less detailed because of theusage of the item attribute categories instead of item attributes themselves which are more detailed than categories.

In the experiments, proposed approaches generate better scores for all metrics than the traditional content-based recommendation approaches which uses TF-IDF scores of item attributes. In addition to traditional content-based recommendation approach, our approaches outperform the some known approaches from literature in f-measure metric which is the combination of both precision and recall metrics. As a future work; to make more fair comparisons with the approaches from the literature, same datasets that are used in their approaches can be used to implement the proposed approaches; such as the MovieLens dataset [52].

As a future work; to validate the proposed approaches in a different way, we could also compare the automatically generated preference weights with those that could be collected from the users directly. A set of users could be asked to give weights to the categories and attributes. After collecting the answers, a comparison could be done between the generated results and the collected ones.

In the implementation phase of our proposed approach, we used a database which consists of movies as items. But, the proposed approach is a domain independent approach. Any item which can be represented via attributes and attribute categories can be used with our proposed approach. As a future work, the proposed approach can be applied to any other domain which is related to books, news or products. For instance; if the proposed approach is implemented for the book domain, the categories that the proposed approaches will use may be: author, genre, publication year, title, subject and publisher.

Because the proposed approach is a content-based approach, it suffers from the overspecialization problem. To include some randomness to the proposed approach, a new hybrid recommendation approach which combines collaborative filtering approach with our proposed approach can be developed to recommend items to users. In addition to this, users' demographic information such as age, gender and nationality information can be used to categorize the users and a different hybrid recommendation approach which combines demographic recommendation approach with the content-based approach could be developed to gather more accurate recommendations.

In the experiments phase, we accepted the ratings below 6 as negative ratings and the ratings above 6 as positive ratings. As a future work; instead of this assumption, a normalization of user ratings to equalize the results between users who are slow to like or quick to like could be done while implementing the proposed approaches.

Interaction design for recommenders is another topic in recommender systems domain. As a future work, an interface which users can interact with the proposed system effectively by considering the small mobile device screens can be integrated to the proposed system.

REFERENCES

[1] Adomavicius, G. & Tuzhilin, A. (2005). 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 17 (6), 734--749.

[2] Adomavicius, G. & Tuzhilin, A. (2001). Expert-Driven Validation of Rule-Based User Models in Personalization Applications. Data Min. Knowl. Discov. 5 (1-2), 33--58. (Doi: 10.1023/A:1009839827683.)

[3] Aksel, F. & Birtürk, A. (2010). Enhancing Accuracy of Hybrid Recommender Systems through Adapting the Domain Trends.

[4] Almazro, D., Shahatah, G., Albdulkarim, L., Kherees, M., Martinez, R. & Nzoukou, W. (2010). A Survey Paper on Recommender Systems. CoRR abs/1006.5278.

[5] Balabanović, M. & Shoham, Y. (1997). Fab: content-based, collaborative recommendation. Commun. ACM 40 (3), 66--72. (Doi: 10.1145/245108.245124.)

[6] Basu, C., Hirsh, H. & Cohen, W. (1998). Recommendation as Classification: Using Social and Content-Based Information in Recommendation. In In Proceedings of the Fifteenth National Conference on Artificial Intelligence (pp. 714--720). AAAI Press.

[7] Billsus, D., Brunk, C. A., Evans, C., Gladish, B. & Pazzani, M. (2002). Adaptive interfaces for ubiquitous web access. Commun. ACM 45 (5), 34--38. (Doi: 10.1145/506218.506240.)

[8] Billsus, D. & Pazzani, M. J. (2000). User Modeling for Adaptive News Access. User Modeling and User-Adapted Interaction 10 (2-3), 147--180. (Doi: 10.1023/A:1026501525781.)

[9] Bogdanov, D., Haro, M., Fuhrmann, F., XambÃ, A., GÃmez, E. & Herrera, P. (2013). Semantic audio content-based music recommendation and visualization based on user preference examples. Information Processing & Management 49 (1), 13 - 33. (Doi: 10.1016/j.ipm.2012.06.004.)

[10] Brahim, H. F. F. B. O. (2012). A comparison study of some algorithms in Recommender Systems. , 130-135.

[11] Breese, J. S., Heckerman, D. & Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. In Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence (pp. 43--52). Morgan Kaufmann Publishers Inc.. (ISBN: 1-55860-555-X.)

[12] Burke, R. (2002). Hybrid Recommender Systems: Survey and Experiments. User Modeling and User-Adapted Interaction 12 (4), 331--370. (Doi: 10.1023/A:1021240730564.)

[13] Burke, R. (2000). Knowledge-Based Recommender Systems. In ENCYCLOPEDIA OF LIBRARY AND INFORMATION SYSTEMS (pp. 2000). Marcel Dekker.

[14] Carenini, G. & Sharma, R. (2004). Exploring more realistic evaluation measures for collaborative filtering. In Proceedings of the 19th national conference on Artifical intelligence (pp. 749--754). AAAI Press. (ISBN: 0-262-51183-5

[15] Cineworm.com: http://cineworm.com/, accessed at 17.04.2013

[16] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K. & Harshman, R. (1990). Indexing by latent semantic analysis. JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE 41 (6), 391--407.

[17] Delgado, J. & Ishii, N. (1999). Memory-Based Weighted-Majority Prediction for Recommender Systems.

[18] Deshpande, M. & Karypis, G. (2004). Item Based Top-N Recommendation Algorithms. ACM Transactions on Information Systems 22, 143--177.

[19] Düzgün, S. & Birtürk, A. (2012). A Web-Based Book Recommendation Tool for Reading Groups .

[20] Facebook: http://www.facebook.com/, accessed at 17.04.2013

[21] Goldberg, D., Nichols, D., Oki, B. M. & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. Commun. ACM 35 (12), 61--70. (Doi: 10.1145/138859.138867.)

[22] Goldberg, K., Roeder, T., Gupta, D. & Perkins, C. (2001). Eigentaste: A Constant Time Collaborative Filtering Algorithm. Inf. Retr. 4 (2), 133--151. (Doi: 10.1023/A:1011419012209.)

[23] Gori, M. & Pucci, A. (2007). ItemRank: a random-walk based scoring algorithm for recommender engines. In Proceedings of the 20th international joint conference on Artifical intelligence (pp. 2766--2771). Morgan Kaufmann Publishers Inc.

[24] Grant, S. & McCalla, G. I. (2001). A Hybrid Approach to Making Recommendations and Its Application to the Movie Domain. In Proceedings of the 14th Biennial Conference of the Canadian Society on Computational Studies of Intelligence: Advances in Artificial Intelligence (pp. 257--266). Springer-Verlag. (ISBN: 3-540-42144-0.)

[25] Grooveshark: http://grooveshark.com/, accessed at 17.04.2013

[26] Gunawardana, A. & Shani, G. (2009). A Survey of Accuracy Evaluation Metrics of Recommendation Tasks. J. Mach. Learn. Res. 10, 2935--2962.

[27] Hanani, U., Shapira, B. & Shoval, P. (2001). Information Filtering: Overview of Issues, Research and Systems. User Modeling and User-Adapted Interaction 11 (3), 203--259. (Doi: 10.1023/A:1011196000674.)

[28] Herlocker, J., Konstan, J. A. & Riedl, J. (2002). An Empirical Analysis of Design Choices in Neighborhood-Based Collaborative Filtering Algorithms. Inf. Retr. 5 (4), 287--310. (Doi: 10.1023/A:1020443909834.)

[29] Herlocker, J. L., Konstan, J. A., Borchers, A. & Riedl, J. (1999). An algorithmic framework for performing collaborative filtering. In Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval (pp. 230--237). ACM. (ISBN: 1-58113-096-1.)

[30] Hill, W., Stead, L., Rosenstein, M. & Furnas, G. (1995). Recommending and evaluating choices in a virtual community of use. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (pp. 194--201). ACM Press/Addison-Wesley Publishing Co.. (ISBN: 0-201-84705-1.)

[31] Jannach, D., Zanker, M., Felfernig, A. & Friedrich, G. (2011). Recommender Systems An Introduction. Cambridge University Press.

[32] JesterJoke: http://eigentaste.berkeley.edu/dataset/, accessed at 17.04.2013

[33] Jin, R., Chai, J. Y. & Si, L. (2004). An automatic weighting scheme for collaborative filtering. In Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval (pp. 337--344). ACM. (ISBN: 1-58113-881-4.)

[34] JSON: http://en.wikipedia.org/wiki/JSON, accessed at 17.04.2013

[35] Kim, H. K., Kim, J. K. & Ryu, Y. U. (2009). Personalized Recommendation over a Customer Network for Ubiquitous Shopping. IEEE Trans. Serv. Comput. 2 (2), 140--151. (Doi: 10.1109/TSC.2009.7.)

[36] Kirmemiş, O. (2008). Openmore: A content-based movie recommendation system.Unpublished master's thesis, Computer Engineering Department of Middle East Technical University.

[37] Koren, Y., Bell, R. & Volinsky, C. (2009). Matrix Factorization Techniques for Recommender Systems. Computer 42 (8), 30--37. (Doi: 10.1109/MC.2009.263.)

[38] Last.fm: http://www.lastfm.com.tr/, accessed at 17.04.2013

[39] Linden, G., Smith, B. & York, J. (2003). Amazon.com Recommendations: Item-to-Item Collaborative Filtering. IEEE Internet Computing 7 (1), 76--80. (Doi: 10.1109/MIC.2003.1167344.)

[40] Littlestone, N. & Warmuth, M. K. (1994). The weighted majority algorithm. Inf. Comput. 108 (2), 212--261. (Doi: 10.1006/inco.1994.1009.)

[41] Lops, P., Gemmis, M. & Semeraro, G. (2011). Content-based Recommender Systems: State of the Art and Trends. In F. Ricci, L. Rokach, B. Shapira & P. B. Kantor (ed.),Recommender Systems Handbook (pp. 73-105). Springer US. (ISBN: 978-0-387-85819-7.)

[42] Mahmood, T. & Ricci, F. (2009). Improving recommender systems with adaptive conversational strategies. In Proceedings of the 20th ACM conference on Hypertext and hypermedia (pp. 73--82). ACM. (ISBN: 978-1-60558-486-7.)

[43] Malone, T. W., Grant, K. R., Turbak, F. A., Brobst, S. A. & Cohen, M. D. (1987). Intelligent information-sharing systems. Commun. ACM 30 (5), 390--402. (Doi: 10.1145/22899.22903.)

[44] McCallum, A. & Nigam, K. (1998). A comparison of event models for Naive Bayes text classification .

[45] McLaughlin, M. R. & Herlocker, J. L. (2004). A collaborative filtering algorithm and evaluation metric that accurately model the user experience. In Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval (pp. 329--336). ACM. (ISBN: 1-58113-881-4.)

[46] McSherry, F. & Mironov, I. (2009). Differentially private recommender systems: building privacy into the net. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 627--636). ACM. (ISBN: 978-1-60558-495-9.)

[47] Melville, P., Mooney, R. J. & Nagarajan, R. (2002). Content-boosted collaborative filtering for improved recommendations. In Eighteenth national conference on Artificial intelligence (pp. 187--192). American Association for Artificial Intelligence. (ISBN: 0-262-51129-0.)

[48] van Meteren, R. & van Someren, M. (2000). Using Content-Based Filtering for Recommendation .

[49] Miller, B. N., Albert, I., Lam, S. K., Konstan, J. A. & Riedl, J. (2003). MovieLens unplugged: experiences with an occasionally connected recommender system. In Proceedings of the 8th international conference on Intelligent user interfaces (pp. 263--266). ACM. (ISBN: 1-58113-586-6.)

[50] Mladenic, D. (1999). Text-Learning and Related Intelligent Agents: A Survey. IEEE Intelligent Systems 14 (4), 44--54. (Doi: 10.1109/5254.784084.)

[51] Montaner, M., López, B. & De La Rosa, J. L. (2003). A Taxonomy of Recommender Agents on theInternet. Artif. Intell. Rev. 19 (4), 285--330. (Doi: 10.1023/A:1022850703159.)

[52] MovieLens Dataset: http://www.grouplens.org/taxonomy/term/14, accessed at 17.04.2013

[53] Nadschläger, S., Kosorus, H., Bögl, A. & Küng, J. (2012). Content-Based Recommendations within a QA System Using the Hierarchical Structure of a Domain-Specific Taxonomy.. In A. Hameurlain, A. M. Tjoa & R. Wagner

(ed.),DEXA Workshops (pp. 88-92). IEEE Computer Society. (ISBN: 978-1-4673-2621-6.)

[54] Netflix: http://www.netflix.com/, accessed at 17.04.2013

[55] Park, D. H., Kim, H. K., Choi, I. Y. & Kim, J. K. (2012). A literature review and classification of recommender systems research. Expert Systems with Applications 39 (11), 10059 - 10072. (Doi: 10.1016/j.eswa.2012.02.038.)

[56] Pazzani, M. & Billsus, D. (1997). Learning and Revising User Profiles: The Identification ofInteresting Web Sites. Mach. Learn. 27 (3), 313--331. (Doi: 10.1023/A:1007369909943.)

[57] Pazzani, M. J. (1999). A Framework for Collaborative, Content-Based and Demographic Filtering. Artif. Intell. Rev. 13 (5-6), 393--408. (Doi: 10.1023/A:1006544522159.)

[58] Pazzani, M. J. & Billsus, D. (2007). The adaptive web. In P. Brusilovsky, A. Kobsa & W. Nejdl (ed.), (pp. 325--341). Springer-Verlag. (ISBN: 978-3-540-72078-2.)

[59] Peis, E., del Castillo, J. M. M. & Delgado-López, J. A. (2008). Semantic Recommender Systems. Analysis of the state of the topic.. Hipertext.net 6, (online).

[60] Pera, M. S. & Ng, Y.-K. (2013). A group recommender for movies based on content similarity and popularity. Inf. Process. Manage. 49 (3), 673--687. (Doi: 10.1016/j.ipm.2012.07.007.)

[61] Ping, H. Q. & Ming, X. (2012). Research on Several Recommendation Algorithms. Procedia Engineering 29 (0), 2427 - 2431. (Doi: 10.1016/j.proeng.2012.01.326.)

[62] Powell, M. (1981). Approximation Theory and Methods. Cambridge Univ. Press.

[63] Rashid, A. M., Albert, I., Cosley, D., Lam, S. K., McNee, S. M., Konstan, J. A. & Riedl, J. (2002). Getting to know you: learning new user preferences in recommender systems. In Proceedings of the 7th international conference on Intelligent user interfaces (pp. 127--134). ACM. (ISBN: 1-58113-459-2.)

[64] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P. & Riedl, J. (1994). GroupLens: an open architecture for collaborative filtering of netnews. In

Proceedings of the 1994 ACM conference on Computer supported cooperative work (pp. 175--186). ACM. (ISBN: 0-89791-689-1.)

[65] Ricci, F., Rokach, L. & Shapira, B. (2011). Introduction to Recommender Systems Handbook. In F. Ricci, L. Rokach, B. Shapira, P. B. Kantor, F. Ricci, L. Rokach, B. Shapira & P. B. Kantor (ed.),Recommender Systems Handbook (pp. 1--35). Springer. (ISBN: 978-0-387-85819-7.)

[66] Rich, E. (1979). User Modeling via Stereotypes. In Cognitive Science: A Multidisciplinary Journal, Vol. Vol. 3, No. 4 (pp. 329--354).

[67] Rocchio, J. J. (1971). Relevance feedback in information retrieval. In G. Salton (ed.),The Smart retrieval system - experiments in automatic document processing (pp. 313--323). Englewood Cliffs, NJ: Prentice-Hall.

[68] Salehi, M. & Kmalabadi, I. N. (2012). A Hybrid Attribute-based Recommender System for E-learning Material Recommendation. IERI Procedia 2 (0), 565 - 570. (Doi: 10.1016/j.ieri.2012.06.135.)

[69] Salton, G. (1989). Automatic text processing: the transformation, analysis, and retrieval of information by computer. Addison-Wesley Longman Publishing Co., Inc.. (ISBN: 0-201-12227-8.)

[70] Sarwar, B., Karypis, G., Konstan, J. & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th international conference on World Wide Web (pp. 285--295). ACM. (ISBN: 1-58113-348-0.)

[71] Sarwar, B. M., Karypis, G., Konstan, J. A. & Riedl, J. T. (2000). Application of Dimensionality Reduction in Recommender Systems: A case study. In WebKDD Workshop at the ACM SIGKKD.

[72] Schafer, J. B., Frankowski, D., Herlocker, J. & Sen, S. (2007). The adaptive web. In P. Brusilovsky, A. Kobsa & W. Nejdl (ed.), (pp. 291--324). Springer-Verlag. (ISBN: 978-3-540-72078-2.)

[73] ScienceDirect: http://www.sciencedirect.com/, accessed at 17.04.2013

[74] Shardanand, U. & Maes, P. (1995). Social Information Filtering: Algorithms for Automating "Word of Mouth". In (pp. 210--217). ACM Press.

[75] Sheth, B. & Maes, P. (1993). Evolving agents for personalized information filtering. In (pp. 345--352).

[76] Twitter: https://twitter.com/, accessed at 17.04.2013

[77] Wen, H., Fang, L. & Guan, L. (2012). A hybrid approach for personalized recommendation of news on the Web. Expert Systems with Applications 39 (5), 5806 - 5814. (Doi: 10.1016/j.eswa.2011.11.087.)

[78] Xu, B., Bu, J., Chen, C. & Cai, D. (2012). An exploration of improving collaborative recommender systems via user-item subgroups. In Proceedings of the 21st international conference on World Wide Web (pp. 21--30). ACM. (ISBN: 978-1-4503-1229-5.)

[79] Xu, B., Zhang, M., Pan, Z. & Yang, H. (2005). Content-based recommendation in e-commerce. In Proceedings of the 2005 international conference on Computational Science and Its Applications - Volume Part II (pp. 946--955). Springer-Verlag. (ISBN: 3-540-25861-2, 978-3-540-25861-2.)

[80] YouTube: http://www.youtube.com/, accessed at 17.04.2013

[81] Yu, K., Schwaighofer, A., Tresp, V., Xu, X. & Kriegel, H.-P. (2004). Probabilistic Memory-Based Collaborative Filtering. IEEE Trans. on Knowl. and Data Eng. 16 (1), 56--69. (Doi: 10.1109/TKDE.2004.1264822.)

[82] Zanker, M., Jessenitschnig, M. & Schmid, W. (2010). Preference reasoning with soft constraints in constraint-based recommender systems. Constraints 15 (4), 574--595. (Doi: 10.1007/s10601-010-9098-8.)

[83] Zigkolis, C., Karagiannidis, S., Koumarelas, I. & Vakali, A. (2013). Integrating similarity and dissimilarity notions in recommenders. *Expert Systems with Applications* (0). (Doi: 10.1016/j.eswa.2013.03.018.)

TEZ FOTOKOPİSİ İZİN FORMU

ENSTİTÜ

Fen Bilimleri Enstitüsü	☐
Sosyal Bilimler Enstitüsü	☐
Uygulamalı Matematik Enstitüsü	☐
Enformatik Enstitüsü	☑
Deniz Bilimleri Enstitüsü	☐

YAZARIN

Soyadı	: Özberk Yener
Adı	: Tuğçe
Bölümü	: Bilişim Sistemleri

TEZİN ADI (İngilizce) : USER PREFERENCE BOOSTED CONTENT-BASED RECOMMENDER SYSTEM

TEZİN TÜRÜ : Yüksek Lisans ☑ Doktora ☐

1. Tezimin tamamından kaynak gösterilmek şartıyla fotokopi alınabilir. ☐
2. Tezimin içindekiler sayfası, özet, indeks sayfalarından ve/veya bir bölümünden kaynak gösterilmek şartıyla fotokopi alınabilir. ☐
3. Tezimden bir (1) yıl süreyle fotokopi alınamaz. ☑

TEZİN KÜTÜPHANEYE TESLİM TARİHİ : …………………….