A HYBRID MOVIE RECOMMENDER USING DYNAMIC FUZZY CLUSTERING

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

BY

FATİH GÜRCAN

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

JANUARY 2010

Approval of the thesis:

A HYBRID MOVIE RECOMMENDER USING DYNAMIC FUZZY CLUSTERING

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

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

Prof. Dr. Müslim Bozyiğit _____
Head of Department, **Computer Engineering**

Dr. Ayşenur Birtürk _____
Supervisor, **Computer Engineering Dept., METU**

Examining Committee Members:

Assoc. Prof. Dr. Ali Doğru
Computer Engineering Dept., METU _____

Dr. Ayşenur Birtürk
Computer Engineering Dept., METU _____

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

Asst. Prof. Dr. Pınar Şenkul
Computer Engineering Dept., METU _____

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

Date: **04.02.2010**

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

Name, Last Name: FATİH GÜRCAN

Signature :

ABSTRACT

A HYBRID MOVIE RECOMMENDER USING DYNAMIC FUZZY CLUSTERING

Gürcan, Fatih
M.Sc., Department of Computer Engineering
Supervisor : Dr. Ayşenur Birtürk

January 2010, 77 pages

Recommender systems are information retrieval tools helping users in their information seeking tasks and guiding them in a large space of possible options. Many hybrid recommender systems are proposed so far to overcome shortcomings born of pure content-based (PCB) and pure collaborative filtering (PCF) systems. Most studies on recommender systems aim to improve the accuracy and efficiency of predictions. In this thesis, we propose an online hybrid recommender strategy (CBCF_{dfc}) based on content boosted collaborative filtering algorithm which aims to improve the prediction accuracy and efficiency. CBCF_{dfc} combines content-based and collaborative characteristics to solve problems like sparsity, new item and over-specialization. CBCF_{dfc} uses fuzzy clustering to keep a certain level of prediction accuracy while decreasing online prediction time. We compare CBCF_{dfc} with PCB and PCF according to prediction accuracy metrics, and with CBCF_{onl} (online CBCF without clustering) according to online recommendation time. Test results showed that CBCF_{dfc} performs better than other approaches in most cases. We, also, evaluate the effect of user-specified parameters to the prediction accuracy and efficiency. According to test results, we determine optimal values for these parameters. In addition to experiments made on simulated data, we also perform a user study and evaluate opinions of users about recommended

movies. The results that are obtained in user evaluation are satisfactory. As a result, the proposed system can be regarded as an accurate and efficient hybrid online movie recommender.

Keywords: hybrid recommender system, content-based systems, collaborative filtering systems, fuzzy clustering

ÖZ

İÇERİKLE DESTEKLENMİŞ KOOPERATİF FİLTRELEME ALGORİTMASINA DAYALI MELEZ VE DİNAMİK BİR ÖNERİ SİSTEMİ

Gürcan, Fatih
Yüksek Lisans, Bilgisayar Mühendisliği
Tez Yöneticisi : Dr. Ayşenur Birtürk

Ocak 2010, 77 sayfa

Öneri sistemleri kullanıcıların aradıkları bilgiyi bulmalarına yardım eden bilgi erişim araçlarıdır. Saf içerik tabanlı (PCB) ve saf işbirlikçi filtreleme (PCF) metotlarının yol açtığı zorlukların üstesinden gelebilmek için şimdiye kadar pek çok melez sistem önerilmiştir. Öneri sistemleri hakkında yapılan pek çok çalışma tahminlerin doğruluğunu ve hızını arttırmayı amaçlamaktadır. Bu tezde, yapılacak tahminlerin doğruluğunu ve sürecin verimliliğini arttırmayı hedefleyen, içerikle desteklenmiş işbirlikçi filtreleme algoritmasına dayalı melez ve dinamik bir öneri sistemi (CBCF_{dfc}) önerilmektedir. CBCF_{dfc} içerik tabanlı ve kooperatif filtreleme esasına dayanan metodları birleştirerek seyreklik, yeni ürün ve aşırı uzmanlaşma gibi sorunları çözmeye çalışmaktadır. CBCF_{dfc} öneri süresini kısaltırken tahminlerin doğruluğunu belli bir düzeyin altına düşürmemek için bulanık kümeleme kullanmaktadır. Tasarlanan yaklaşım, tahmin doğruluğu ölçütlerine göre PCB ve PCF ile, önerme süresine göre CBCF_{onl} (Kümeleme kullanılmamış içerikle desteklenmiş kooperatif filtreleme) ile karşılaştırılmıştır. Test sonuçları CBCF_{dfc}'nin çoğu durumda diğer yaklaşımlara göre daha iyi bir performans gösterdiğini ortaya koymuştur. Kullanıcı tarafından belirtilen parametrelerin önerilerin doğruluğuna ve önerme hızına olan etkisi değerlendirilmiştir. Test sonuçlarına

göre bu parametreler için ideal aralıklar belirlenmiştir. Simüle edilmiş verilerle yapılan deneylere ek olarak, bir kullanıcı arayüzü geliştirilerek kullanıcıların kendilerine önerilen filmler hakkındaki görüşleri değerlendirilmiştir. Alınan sonuçlar tatmin edici çıkmıştır. Sonuç olarak geliştirilen sistemin doğru ve hızlı çalışan bir film öneri sistemi olarak değerlendirilebileceği görülmüştür.

Anahtar Kelimeler: melez öneri sistemi, içerik tabanlı öneri sistemleri, kooperatif filtrelemeye dayalı öneri sistemleri, bulanık kümeleme

kendisini bekleyen hayat için fazlasıyla saf ve temiz olan Elif Ceren'e...

ACKNOWLEDGMENTS

I would like to express my highest gratitude to my supervisor Dr. Ayşenur Birtürk for his guidance, suggestions, and encouragement to my research.

I am thankful to my jury members Prof. Dr. Mehmet R. Tolun, Assoc. Prof. Dr. Ali Doğru, Assoc. Prof. Dr. Nihan Kesim Çiçekli and Asst. Prof. Dr. Pınar Şenkul for reviewing and evaluating my thesis.

I owe special thanks to my chief Ahmet Alper Eğe and my colleagues Mehmet Yavuz and Duygu Tanrıkulu for their patience and great efforts throughout my thesis studies.

I would like to thank my parents and my sisters for their love, and endless support during my educational life.

I am also grateful to my friends for their contributions to my thesis and moral support.

Finally, I thank TUBITAK for its support troughout my master program.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVATIONS

CBCF Content-Boosted Collaborative Filtering

CBCF$_{dfc}$ Online Content-Boosted Collaborative Filtering With Clustering

CBCF$_{onl}$ Online Content-Boosted Collaborative Filtering Without Clustering

CBF Content-Based Filtering

CF Collaborative Filtering

FCM Fuzzy c-Means Clustering

MAE Mean Absolute Error

NBC Naive Bayesian Classifier

PCB Pure Content-Based Filtering

PCB$_{avg}$ Pure Content-Based Filtering Using Average Likelihood

PCB$_{max}$ Pure Content-Based Filtering Using Maximum Likelihood

PCC Pearson Correlation Coefficient

PCF Pure Collaborative Filtering

ROC Receiver Operating Characteristic

CHAPTER 1

INTRODUCTION

This chapter begins with a brief overview of recommender systems. Afterwards, major challenges and shortcomings of current recommender system technology will be introduced. Definition of our research problem and major contributions of our thesis will be given, consequently. Finally we will give the organization of thesis.

1.1 Recommender Systems Overview

The amount of opportunities and knowledge that the Internet provides is increasing day by day. Although, the Internet supplements traditional tools to gather information, finding desirable information becomes more difficult as the size and diversity of information in the Internet increases. Recommender systems have emerged to overcome this difficulty. Today, they are used intensively in most of the domains including book, movie, restaurant, news and music recommendation. Several well-known websites using recommender systems are Amazon, imdb, MovieLens and Yahoo! MUSIC.

Burke defined a recommender system as "any system that produces individualized recommendations as output, or has the effect of guiding the user in a personalized way to interesting or useful objects in a large space of possible options" [16]. Recommender systems are basically information retrieval tools helping users in their information-seeking tasks, however they differ from classic information retrieval tools from several aspects. Firstly, user profiling is a dynamic task in recommender systems in contrast to other personalization systems which always dispatch same information to the users personalized according to their profiles. Secondly, recommendations are computed

online with an interaction to the system instead of offline computation [69].

Recent recommender technology benefits from a wide range of techniques including information filtering, information retrieval, user modeling and machine learning. Recommender systems are basically composed of three components [16]:

- The information that the system necessities before the recommendation process begins
- The information that user must provide in order to generate recommendations
- An algorithm that combines first two information to arrive at its suggestions

Recommender systems are classified by different ways according to the above counterparts. Most frequently used classification is the one which is done according to the recommendation technique used: Content-based filtering (CBF) and collaborative filtering (CF) [7]. CBF is mainly based on information retrieval and information filtering techniques. This technique assumes each user as an independent entity and tries to uniquely characterize a user's profile in order to make recommendations [6, 10]. On the other side, CF does not use any kind of content data. It tries to predict preferences of a user, according to preferences of other similar users [53].

1.2 Major Difficulties in Recommender Systems Development

Recommender systems hardly tackle with some particular challenges. These challenges can be caused by domain characteristics, weakness of the technique used or inadequacy of the implementation. Some challenges occur only with CF techniques, some occur only with CBF techniques and some occur with both of them. These challenges are briefly explained below:

- **New Item:** New item, also known as "first rater", problem is a weak side of CF techniques. With CF techniques, until a new object is rated by enough number of users, the object cannot be recommended [5]. This problem can be solved by using hybrid techniques [59].

- **New User:** If the recommender does not know the preferences of a user, it cannot recommend any objects to the user. Several techniques are used to overcome this challenge. Some systems use demographics of users while some other systems use more advanced techniques such as item popularity and item entropy [65].

- **Sparsity:** In most domains, number of previously given ratings per object is generally very low [49]. This fact causes the problem of sparsity. Sparsity is not related only with the overall proportion of ratings, the distribution of the given ratings also causes sparsity. Several techniques are proposed to overcome sparsity problem. These include using demographic data of users [61], Singular Value Decomposition (SVD) [61] and EM learning algorithms [64].

- **Scalability:** In big user and object spaces, making dynamic recommendations requires large computation times. To get rid of efficiency-concerned problems, various model-based techniques are proposed. Model-based techniques predict ratings of previously unrated objects according to a created model. Frequently used model-based techniques are Bayesian networks, Clustering Algorithms and Bayesian classifiers [3].

- **Over-specialization:** In some circumstances, users may rarely receive a recommendation differing from his previously experiences. This problem can be solved by adding some serendipity to the prediction algorithm. Genetic algorithms in the context of information filtering are proposed by some researchers to overcome over-specialization [74].

- **Concept Drift:** Concept drift is used for the long term changes in the attitudes and behaviors of users while they were interacting with the recommender system. Since the profile of a user is created cumulatively, the recommender system needs to adopt itself quickly to the changes in the user's behaviors. Most popular technique that partially overcomes this problem is giving more weight to the newer observations [45].

- **Gray Sheep:** CF systems are compelled by users having extraordinary behaviors that do not fit to any category. Claypool et al. call this type of users as *gray sheep* users [19]. CBF techniques are used to overcome this problem.

- **Synonymy:** In some certain domains, objects should not be recommended if they are too similar to previously recommended objects. For instance, in news recommendation domain, different news describing same event should not be recommended several times. To overcome this problem some systems, such as DailyLearner, filter out objects those are too similar to previously recommended objects [12].

1.3 Research Problem

A rating-based movie recommender has two fundamental tasks. Firstly, it predicts ratings of previously unrated movies by using some technique. Secondly, according to actual and predicted ratings, it recommends movies to users.

In a more formal sense, when a user-movie matrix is created with the actual ratings the recommendation problem reduces to filling up this sparse matrix by using some technique. As it is formulated in [3], if the user space is denoted by U and movie space is denoted by M, our user-movie matrix can be formulated as $U \mathrm{x} M$. Let c_0 be the convenience of a movie m_0 for the user u_0, then the predictor tries to estimate c for each $m \in M$ which is not previously rated by each $u \in U$. Convenience can be expressed as $c : U \mathrm{x} M \rightarrow R$. Here R is a non-negative real number within a range, usually called as rating defined by the function. After prediction of ratings, we obtain a dense user-movie matrix which reduces the recommendation problem to simple sorting.

1.4 Brief Background and Expected Contributions

The major contribution of this thesis can be considered as the enhancement of content-boosted collaborative filtering (CBCF) algorithm proposed by Melville et al. [52]. We named our recommender as CBCF_{dfc} stands for content-boosted collaborative filtering with dynamic fuzzy clustering. Melville et al.'s work include some deficiencies. Firstly, their Bayesian classifier was only using maximum likelihood decision rule which yields inaccurate predictions compared to average likelihood decision rule. Secondly, their proposed algorithm was working only offline and did not cover online recommendation. Thirdly, in their work, the effect of parametric values like number of selected

neighbors in collaborative filtering or weight of content-based ratings to the accuracy and efficiency are not considered. Fourthly, they made their experiments only with Movielens datasets so that they did not implement a user interface in order to evaluate user preferences.

The contribution of this thesis can be considered as five-fold:

- Proposing a more accurate hybrid algorithm by employing additional features in the content-based part (i.e. adjusting value, average likelihood).
- Testing effect of user-specified parameters to the accuracy and efficiency.
- Applying fuzzy clustering to proposed algorithm in order to enable online recommendation.
- Proposing a new initialization method for fuzzy c-means clustering to increase accuracy and efficiency.
- Evaluating preferences of users in addition to evaluation of provided datasets by MovieLens.

1.5 Motivations for the Techniques Used in CBCF$_{dfc}$

The motivations for our approach are listed below:

- **Why a hybrid approach?**

  Both of CF and CBF have pluses and minuses. In our approach we created a hybrid recommender combining strong sides of CF and CBF. By employing CBF we overcome new item, sparsity and gray sheep problems. Similarly, by employing CF we overcome limited content analysis, concept drift, over-specialization and synonymy problems. All of these mentioned problems are explained in Section 1.2. Another domain-specific motivation for using CBF is the text-based and utilizable dataset we work on.

- **Why a model-based approach?**

Memory-based methods make accurate predictions, however they suffer from high online computation times in big databases [3]. To be able to make recommendations in acceptable times we developed a model-based approach.

- **Why fuzzy clustering?**

  Clustering is a widely used technique in model-based method recommender systems. In our approach we use fuzzy clustering because contrary to hard clustering; fuzzy clustering generates more proper clusters. Consequently, proper clusters of users supposed to yield more accurate predictions.

1.6 Organization of Thesis

The rest of the thesis is organized as follows. In the next chapter, we give information about works that are related to recommender systems, clustering techniques, used evaluation metrics and evaluation techniques. In chapter 3, we describe Pure Content-Based Filtering (PCB), Pure Collaborative Filtering (PCF), offline CBCF (CBCF) and online CBCF without clustering (CBCF_{onl}) algorithms in detail. In the same chapter, we introduce initialization and working mechanism of our clustering algorithm fuzzy c-means clustering and explain the interface used for user evaluation. In chapter 4, we evaluate our proposed algorithm by comparing PCB, PCF, CBCF and CBCF_{onl} and provide the detailed evaluation results. In the last chapter, we conclude the thesis and give information about the future work.

CHAPTER 2

RELATED WORK

The previous chapter provided background about recommender systems, expressed the research problem and listed possible contribution. The aim of this chapter is to present characteristics of recommender systems and prior researches. The research done on content-boosted collaborative systems will also be covered with various limitations and possible extensions of the current technology. Additionally, researches on rating systems, current neighborhood algorithms, model-based approaches, and evaluation metrics will be expressed throughout the review.

There is plenty of research on recommender systems since the mid of 90's [73, 67, 37]. As the commercial importance of them increases, the number and diversity of the researches done on this field also increase. Although it is a new research area, most of the recommenders share some common characteristics [3]. Recommenders which do not use rating-based algorithms are named as preference-based recommenders. They do not concern about individual ratings of objects; rather they try to find correct order of recommendations according to the user's preferences. There is a number of researches done on preference-based filtering systems in the literature [25, 20]. In this review we will cover only rating-based recommenders. Throughout the thesis when we say recommender; unless explicitly expressed we mean rating-based recommenders.

2.1 Classification of Recommender Systems According to Used Recommendation Technique

Recommenders are categorized into five groups according to the recommendation technique they use. These techniques are: Demographic techniques, utility-based techniques, knowledge-base techniques, content-based techniques and collaborative techniques. Burke summarized these techniques in Table 2.1 [16]. In Table 2.1, I is the set of items over which recommendations might be made, U is the set of users whose preferences are known, u is the user for whom recommendations need to be generated, and i is some item for which we would like to predict u's preference.

Even if, all of these techniques have distinctive advantages, most of recent recommenders use content-based filtering or collaborative filtering techniques.

Table 2.1: Recommendation techniques

Technique	Background	Input	Process
Demographic	Demographic information about U and their ratings of items in I.	Demographic information about u.	Identify users that are demographically similar to u, and extrapolate from their ratings of i.
Utility-based	Features of items in I.	A utility function over items in I that describes preferences of u.	Apply the function to the items and determine rank of i.
Knowledge-based	Features of items in I. Knowledge of how these items meet user needs.	A description of needs or interests of u.	Infer a match between i and need of u.
Content-based	Features of items in I.	Ratings from u of items in I.	Generate a classifier that fits rating behavior of u and use it on i.
Collaborative	Ratings from U of items in I.	Ratings from u of items in I.	Identify users in U similar to u, and extrapolate from their ratings of i.

2.1.1 Demographic Techniques

Demographic techniques use simple demographics in order to classify users [62, 54]. Major advantage of demographic systems against CF systems is the ability of recommendation in the absence of data provided by other users. In other words, demographic techniques do not suffer from sparsity-related problems. However, demographic tech-

niques are put under a strain because utilizing demographic data is difficult task for some domains. Demographic techniques are used in many commercial systems additional to CF techniques [46, 41, 86].

2.1.2 Utility-based Techniques

The idea of utility-based techniques is based on creating a utility function and calculating the utility of each object according to it. Since, utility-based systems do not base their recommendations on accumulated statistical evidence, they do not confront sparsity problem. There are several studies on utility-based systems including Tête-à-Tête and PersonaLogic [50].

2.1.3 Knowledge-based Techniques

By using knowledge-based techniques, recommenders suggest items based on users' preferences. Knowledge-based techniques use different forms of knowledge structures including declarative knowledge, procedural knowledge and structural knowledge. Semantic networks, classification schemes and thesauri are representative examples of knowledge-based techniques. Knowledge-based techniques have a certain drawback; the diversity of their recommendations is limited with their knowledge-base. Entrée and Google are representative examples of these systems which use their own knowledge-base for recommendation. Entrée uses cuisine knowledge to deduce similarity between menus [16], where Google uses popularity information of the web pages [15].

2.1.4 Content-Based Filtering Techniques

Foundations of content-based filtering techniques are found in the studies about information filtering and information retrieval [10]. Most of recent CBF systems use machine learning instead of traditional information retrieval techniques [47, 26]. Machine learning techniques which are frequently used for content-analysis are clustering [43], decision trees [57] and artificial neural networks [62].

CBF techniques, basically, use content data of users and objects. User profile of each user is created in accordance with the results of content analysis [54, 47]. Predictions are made according to these user profiles. Nkweteyim summarizes this process for CBF systems [58]:

> Content-based recommenders typically represent documents similarly to IR techniques, for example, making use of indexing and the vector space model to represent documents and the user. The model represents objects as unique fixed-length vectors of features, usually words of a document, weighted by some criteria. Recommendations are made by comparing the user profile with documents present in the system, and only those documents that are very similar to the user profile are recommended. This is analogous to comparing a user query with a document collection in a search task, and returning the documents that are most similar to the query, as the result of the search.

Because of the difficulty of utilization of content data in some domains (e.g. multimedia environments), CBF systems generally perform on only textual data [3]. The content in text-based domains is generally described by keywords. These keywords, also known as features of objects, are used to build up user profiles [19]. Features of an object may be director of a movie, author of a book or category of a song. For several domains, such as book, movie, academic article and music, we can easily find text-based datasets. However, for many domains finding text-based datasets is a difficult task.

CBF techniques have some advantages over other recommendation techniques. Firstly, since they based their recommendations on content data of objects, they do not confront sparsity-related problems. Secondly, if a user rated high number of objects, he will receive more accurate and qualified recommendations due to its improved profile [58]. On the other side, CBF techniques have some challenges also. A major challenge about CBF is the first user problem. To be able to create a user's profile, CBF needs its own previously given ratings. CBF, also, suffers from synonymy problem in some domains, such as news recommendation. In these specific domains, objects which are too similar to previously recommended objects should not be recommended. Lastly, for some specific domains implementing a convenient analyzer may be problematic.

As it is known, the accuracy of recommendations in CBF systems is relevant with the filtering capability of the employed content-based analyzer.

There are plenty of researches on CBF in the literature. One of the early researches on this area is InfoFinder which creates the user profile according to the e-mails sent by the user and notifies the user if there is something interesting for the user [17]. NewsWeeder is a content-based recommender which filters the net news according to previously given ratings. NewsWeeder uses a learning algorithm based on Minimum Description Length (MDL) principle [47]. DailyLearner, a news recommender proposed by Billsus et al., selects recommendation technique that can give the recommendation with the higher level of accuracy [12].

2.1.5 Collaborative Filtering Recommender Systems

With collaborative filtering (CF) techniques, recommendation of an object is based on previously given ratings to that object by other users. Practically, CF techniques provide a basis for recommendation of objects, regardless of whether their content can be represented in a way that is useful for recommendation. In contrary to CBF, CF does not run into some problems such as limited content analysis, synonymy, over-specialization and concept drift. On the other hand, they have some challenges including new item, sparsity, scalability and gray sheep. Neighborhood-based CF algorithm is the most frequently used CF algorithm. Mellville explains working mechanism of neighborhood-based CF in three steps [52]:

1. By using a similarity measuring method weight all users with respect to similarity with the active user.

2. Select n users that have the highest similarity with the active user. These selected users form the neighborhood of the active user.

3. Compute a prediction from a weighted combination of the selected neighbors' ratings.

Weighting Users or Objects

Similarity computation between users or objects is a critical step in CF. For item-based CF algorithms, the idea of the similarity computation between objects is first to work on the users who have rated both of these items and then to apply a similarity computation to determine the similarity between the two co-rated objects of the users. For a user-based CF algorithm, the process is similar. [77].

There are many different methods to compute similarity between users or items such as correlation-based similarity, vector cosine-based similarity and entropy based uncertainty [73, 68]. Correlation-based similarity methods are frequently used in recent CF systems. Popular variations of them are Pearson correlation, Spearman correlation and Kendall's correlation. In his research, Breese gives a comparison between these measures [14].

Selecting the Neighborhood

There are two common approaches to define the neighborhood of a user: Selecting top-N-neighbors and using a similarity threshold. In systems using top-N-neighbors technique; each user has same number of neighbors, regardless of its similarity to the users in its neighborhood. In systems using similarity threshold, two users are regarded as neighbors if their similarity value is bigger than a specified threshold. In addition, there are also studies in the literature combining these two approaches [28].

Computing the Prediction

After similarity weighting and neighborhood selection, predicted ratings are computed for each previously unrated object. Various computation approaches exist including simple average, weighted average and mean-centered weighted average (a.k.a. adjusted weighted average) [35].

Collaborative Filtering was first presented in Tapestry which is based on the idea of manually finding similar users to the active user [31]. In Tapestry, users are able to like or dislike an object which is regarded as a binary rating. In accordance with these binary ratings, objects are recommended to similar users. Since the emergence of Tapestry, extended research made on collaborative filtering. GroupLens [44], Video Recommender [37], and Ringo [73] were the first CF systems automating recommendation contrary to manuely working Tapestry. In the process of time, collaborative

filtering is performed on different domains. PHOAKS was created for helping people to find concerning information on the World Wide Web [80] and Jester was designed for recommendation of jokes [32]. Firefly was used for movie and music recommendation first, but now it covers many domains including newsgroups, books and web pages. GroupLens is an open-source movie recommender which also has been evolved since mid-90s. For the time it was first proposed it was able to use only explicit ratings, however now it can also collect implicit ratings [67].

2.1.6 Hybrid Recommender Systems

As it is stated earlier, the term hybrid is used for systems those combine CBF and CF techniques. Burke categorizes hybrid recommenders according to their combination methods, these categories are: Weighted, switching, mixed, feature combination, cascade, feature augmentation and meta-level [16]. The description of each category is given in Table 2.2.

Table 2.2: Hybridization methods

Hybridization method	Description
Weighted	The scores (or votes) of several recommendation techniques are combined together to produce a single recommendation.
Switching	The system switches between recommendation techniques depending on the current situation.
Mixed	Recommendations from several different recommenders are presented at the same time.
Feature combination	Features from different recommendation data sources are thrown together into a single recommendation algorithm.
Cascade	One recommender refines the recommendations given by another.
Feature augmentation	Output from one technique is used as an input feature to another.
Meta-level	The model learned by one recommender is used as input to another.

Besides, Tuzhilin categorizes hybrid systems into four groups according to their creation procedures [3]. The procedures followed in the creation of hybrid systems are given below.

1. Creating collaborative and content-based recommenders separately and then combining their predictions

2. Adding some content-based characteristics to a collaborative approach

3. Adding some collaborative characteristics to a content-based approach

4. Creating a complete recommender that combine content-based and collaborative characteristics

Systems categorized in the first group have different combination methods of separate recommenders' outputs. Some of them uses an immutable combination formula [19]; while some other change the weights in combination according to the accuracy of predicted ratings of CF and CBF [61]. Adding some CBF characteristics to CF systems is a popular trend in recent systems [7, 52]. These systems are also known as *collaboration via content* systems. Limited number of recent recommenders fall into third category. [76] is a characteristic example for these systems which is implemented by using dimensionality reduction techniques on user profiles. Recommenders falling into fourth category unify characteristics of CF and CBF techniques and widely used today [4, 64, 21].

Most of the recent recommenders have hybrid characteristics. Fab is a web based recommender combining CF and CBF techniques. It uses collection and selection agents to find relevant documents for a user. Relevant documents are found by collection agents and then presented to users to be rated. Eventually, every given rating changes behaviors of selection agents dynamically [7]. Basu et.al proposed a hybrid recommender based on inductive learning. Their system recommends movies by using both user preferences and ratings [8]. [71] and [9] proposed hybrid recommenders for TV viewing domain named ClixSmart and TV Scout, respectively. Melville proposed a content-boosted hybrid recommender which performs CF on the predicted ratings of pure CBF recommender. [52].

2.2 Classification of Recommender Systems using Collaborative Filtering Techniques according to Method Used

As it is stated in [14], CF algorithms can be categorized into two groups: Memory-based (a.k.a. heuristic-based) algorithms and model-based algorithms. Memory-based algorithms use entire user and object space for predictions. On the other hand, model-based algorithms use a learned model representing all users and/or objects in order to

make predictions according to learned model. Many researches are done on memory-based algorithms [14, 67, 73] and model-based algorithms [12, 31, 38, 60] in the literature. In latter sections, theoretical background about and representative examples of these algorithms will be covered.

2.2.1 Collaborative Filtering using Memory-Based Algorithms

Memory-based algorithms use the entire collection of users or objects to make predictions. The working mechanism of these algorithms is given in Section 2.1.5. Memory-based algorithms predict ratings accurately in many applications, however in dynamic systems memory-based algorithms suffer from high computation times. In order to overcome inefficiency in memory-based algorithms, model-based algorithms are proposed for dynamic systems.

In order to improve accuracy of memory-based CF algorithms several techniques are proposed so far. These techniques include default voting, case amplification, voting by category, similarity updating and prediction modulation. In default voting, to overcome sparsity we give default negative or neutral ratings to previously unrated objects. As a consequence comparisons between users can be made easier, due to the increase in the number of common-rated objects [75]. In case amplification, weights of most similar users to the actual user are amplified to 1, so that their contributions in the final predictions increase [14]. Voting by category is proposed by Gokiso-cho et al. to overcome sparsity problem. With voting by category, user-object ratings matrix is divided into clusters. By transferring ratings among these clusters each object is assigned to one of the clusters. Jeong et al. proposed similarity updating and prediction modulation. Their proposed method uses an iterative message passing procedure to minimize the predictive accuracy error and evenly distribute predicted ratings over true rating scales [40].

2.2.2 Collaborative Filtering using Model-Based Algorithms

Model-based algorithms are especially used for dynamic systems running on large databases. They learn a model from the ratings space in order to use this model

to make predictions. Notable model-based algorithms proposed so far are: Neural networks [39], Latent semantic indexing [30], clustering [48] and Bayesian networks [21].

In Neural Network models, CF is seen as a classification task. This model enables classification of previously unrated objects to specified classes. Pazzani proposed a system based on neural network model [11]. Latent Semantic Indexing is an information retrieval method using Singular Value Decomposition (SVD) to identify patterns in the relationships between concepts in an unstructured collection of text. Latent Semantic Indexing is used in information discovery, text summarization, online customer support and spam filtering [85]. Bayesian Network models share some common characteristics with Neural Network models. A Bayesian network model is a probabilistic graphical model that represents a set of random variables and their conditional independencies via a directed acyclic graph [83]. Bayesian network models are used in many applications including document classification, information retrieval, data fusion and decision support. Recommender systems, described above are summarized by Adomavicius et al. [3].

Table 2.3: Classification of recommender systems research

Recommendation Approach	Memory-based	Model-based
Content-based	Commonly used techniques: Information retrieval, Clustering Representative research examples: Lang 1995, Balabanovic and Shoham 1997, Pazzani and Billsus 1997	Commonly used techniques: Bayesian classifiers, Clustering, Decision trees, Artificial neural networks Representative research examples: Pazzani and Billsus 1997, Mooney et al. 1998, Mooney and Roy 1999, Billsus and Pazzani 2000, Zhang et al. 2002
Collaborative	Commonly used techniques: Nearest neighbor (cosine, correlation), Clustering, Graph theory Representative research examples: Resnick et al. 1994, Hill et al. 1995, Shardanand and Maes 1995, Breese et al. 1998, Nakamura and Abe 1998, Aggarwal et al. 1999, Delgado and Ishii 1999, Pennock and Horwitz 1999, Sarwar et al. 2001	Commonly used techniques: Bayesian networks, Clustering, Artificial neural networks, Linear regression, Probablistic models Representative research examples: Breese et al. 1998, Chien and George 1999, Pennock and Horwitz 1999, Goldberg et al. 2001, Shani et al. 2002, Marlin 2003, Si and Jin 2003
Hybrid	Commonly used techniques: Linear combination of predicted ratings, Various voting schemes, Incorporating one component as a part of the heuristic for the other Representative research examples: Claypool et al. 1999, Good et al. 1999, Pazzani 1999, Tran and Cohen 2000, Melville et al. 2002	Commonly used techniques: Incorporating one component as a part of the model for the other, Building one unifying model Representative research examples: Basu et al. 1998, Condliff et al. 1999, Soboroff and Nicholas 1999, Ansari et al. 2000, Popescul et al. 2001, Schein et al. 2002

Clustering is a very popular approach in today's model-based applications. Below we give a summary of clustering methods.

2.2.2.1 Clustering Methods

Clustering methods group users or objects with respect to their features. Clustering methods are categorized into five groups: Iterative methods, hierarchical methods, density-based methods, grid-based methods and model-based methods [58]

- **Iterative Clustering Methods** Iterative clustering methods create an initial cluster as the first step. Afterwards by moving objects from one cluster to another iteratively, all objects are distributed into corresponding clusters. For iterative clustering methods, initial specification of the number of clusters is necessary

which is a difficult task for some datasets. K-means or fuzzy c-means clustering clustering are representative examples of iterative clustering methods.

- **Hierarchical Clustering Methods** Hierarchical clustering methods are based on the idea of decomposition of datasets. Two types of hierarchical clustering methods exist: Agglomerative methods and divisive methods. Agglomerative methods distribute every object to a separate cluster first, and then iteratively merge similar objects to same clusters. On the other side, divisive methods start from one cluster containing all objects, and then iteratively split initial cluster until a terminating condition met. Hierarchical clustering methods suffer from difficulty of reassignment of an object to a cluster after it is merged to a different cluster.

- **Density-based Clustering Methods** Density-based clustering methods are generally used in spatial databases. The clusters are represented as dense regions of data and boundaries between clusters are represented as sparse regions.

- **Grid-based Clustering Methods** Grid-based clustering methods create a number of cells and quantize the data space into these cells. Since clustering time depends on the number of cells rather than the number of data points, grid-based methods are computationally more efficient than other clustering methods.

- **Model-based Clustering Methods** Two types of model-based clustering methods exist: Statistical clustering methods and neural network clustering methods. Statistical clustering methods use probability theory to determine corresponding cluster of an object. Neural network clustering methods use a prototype vector for representation of each cluster and place each object to most suitable cluster with regard to this prototype vector.

Below we give the description and working mechanism of several popular clustering algorithms which can be regarded as representative examples for clustering algorithms.

K-means is an iterative clustering algorithm proposed by Aldendefer and Massart. In k-means clustering algorithm objects are clustered according to previously given ratings. Similarly, users are clustered according to the ratings they gave before. K-means clustering is a nondeterministic algorithm which means final status of clusters is

related with the initial assignment of users/objects to clusters. Because of that there is no guarantee for k-means to converge an optimal final set of clusters. However, since computational cost of k-means clustering algorithm is low, it is convenient to run this algorithm several times to get a better final set of clusters. Nested clustering is also possible by K-means clustering which provides a deeper categorization of objects.

K-means clustering is a hard clustering algorithm which means every user or object is member of exactly one cluster. Fuzzy version of k-means clustering algorithm is first proposed by Dunn which is called fuzzy c-means clustering [27]. In fuzzy c-means each object has a degree of membership to clusters. The membership values of users/objects to the clusters are inversely proportional with their distance to the centers of the clusters. In other words, membership values of users near the center of a cluster are bigger than membership values of users on the edge of a cluster. In fuzzy c-means clustering, center of each cluster is hypothetical in contrast to some other methods using actual users/objects as cluster centroids. Fuzzy c-means clustering is used in many applications including [13, 89].

Suryavanshi proposed a clustering algorithm named relational fuzzy subtractive clustering (RSFC) [78]. Main advantage of RSFC is its efficiency. RSFC determines number of clusters and membership threshold automatically according to given dataset. [53] uses RSFC in its hybrid recommender.

Quality Threshold (QT) clustering algorithm is used especially for gene clustering. In contrast to k-means or fuzzy c-means, QT clustering is a deterministic algorithm. In other words, for identical datasets QT clustering algorithm always generates same set of clusters. Similar to RSFC, QT clustering determines number of clusters and membership threshold automatically. As a further explanation, definition of QT algorithm is given below [84].

1. The user chooses a maximum diameter for clusters.
2. System builds a candidate cluster for each point by including the closest point, the next closest, and so on, until the diameter of the cluster surpasses the threshold.
3. System saves the candidate cluster having most points as the first true cluster,

and removes all points in the cluster from further consideration.

4. System recurses with the reduced set of points.

As it can be understood from the above algorithm, QT clustering algorithm is computationally more inefficient than K-means clustering.

Clustering methods are used in many applications so far. In [81], Ungar et al. analyzed accuracy and efficiency of clustering methods. In [53], Mojtaba et al. used a fuzzy clustering algorithm and expressed its advantages over hard clustering algorithms. In [14], Breese evaluated the efficiency of Bayesian clustering methods.

2.3 Rating Systems

Recommender systems use two types of ratings; implicit rating and explicit rating. Recommenders using explicit rating information allow users to give ratings in specified scales. On the other side, recommenders using implicit ratings derive rating information from user actions.

Explicit rating can be acquired in several ways. Simplest type is unary rating. In unary rating, user marks objects those he liked, rest of the objects stays unknown. Differently from unary rating, recommenders using binary rating enable a dislike option for users. Rating on a scale is most popular rating system used in recent recommenders.

Recommenders using implicit rating acquire information from actions of the users in order to create user profiles [33]. In most of these systems reading time spent on a page is the main indicator for the interests of a user [42]. Although reading time is an essential indicator for implicit rating, using reading time directly without normalizing may cause some shortcomings. By normalizing we mean taking difference among reading speeds of users and length of articles into consideration. The number of mouse clicks and the amount of scrolling on the page are also valuable indicators. As a representative example Nichol's research can be given. In his research, Nichols proposed a scale to evaluate the relative significance of different types of actions. Various actions of users are quantified. These quantified values of actions change profile of the user. For instance, purchase of an object is the most significant positive

action; followed by glimpse of an object. Also, when the user stops browsing the system, then system assumes the desired object has been found and a positive rating is recorded for the final object, also. [55].

Exceptionally, in some suitable domains multi-criteria rating is used. Adomavicius et al. incorporates multi-criteria rating information for restaurant recommendation in their research [2].

2.4 Metrics Used in Recommender Systems

2.4.1 Coverage Metrics

Coverage can be defined as the measure of the domain of objects in the system over which the systems can make recommendations. Different metrics are proposed on the coverage problem. The number of items for which predictions can be formed as a percentage of the total number of items, is the most prevalent measure of coverage and named as prediction coverage. Catalog coverage is a different measure from the prediction coverage which is defined as the percentage of the items in the catalog that are ever recommended to users. There is usually an inverse proportion between prediction accuracy and prediction coverage [33]. An ideal system should take the trade-off between accuracy and coverage into account and optimize the system according to user needs.

In [36], Herlocker et al. expressed the characteristics of an ideal coverage metric:

- It should measure both prediction coverage and catalog coverage
- For prediction coverage it should more heavily weight items for which the user is likely to want predictions
- There should be a way to combine the coverage measure with accuracy measures to yield an overall "practical accuracy" measure for the recommender system.

2.4.2 Accuracy Metrics

Accuracy can be defined as the closeness of agreement between a measurement result and a true value. For recommender systems, accuracy metrics measure how close a recommender system's predicted ranking of objects for a user differs from the user's true ranking of preference [36]. First experimental evaluation on accuracy metrics is done by Resnick [67]. Since then various accuracy metrics are proposed. Accuracy metrics can be categorized into three groups: Predictive accuracy metrics, classification accuracy metrics, and rank accuracy metrics.

2.4.2.1 Predictive Accuracy Metrics

Predictive Accuracy Metrics measure distance between actual ratings and predicted ratings. Mean Absolute Error (MAE) is the most widely used predictive accuracy metric because of easiness of computation and time efficiency. On the other hand, MAE is not suitable for some domains in which the granularity of true preferences is small. So far, MAE was used to evaluate the accuracy of predicted ratings for many recommender systems including [14, 73, 36]. Other predictive accuracy metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE) and Normalized Mean Absolute Error (NMAE) are also frequently used in the evaluation of recent recommenders.

2.4.2.2 Classification Accuracy Metrics

Classification Accuracy Metrics measure the frequency with which a recommendation engine makes correct or incorrect decisions about the usefulness of an object. ROC curve-based metrics, ad-hoc classification metrics and half-life utility metrics are the most common used ones. ROC curve area metric is the most popular classification accuracy metric. Broadly speaking, the ROC area metric can be defined as the measure of the diagnostic power of a filtering system. As a representative example, we list the pluses and minuses of it based on Herlocker's work [36].

Pros of ROC area metric are:

- It provides a single number representing the overall performance of an information filtering system
- It is developed from solid statistical decision theory designed for measuring the performance of tasks such as those that a recommender system performs
- It covers the performance of the system over all different recommendation list lengths

Cons of ROC area metric are:

- A large set of potentially relevant items is needed for each query
- For some tasks, users are only interested in performance at one setting, not all possible settings
- Equally distant swaps in rankings have the same effect no matter where in the ranking they occur
- It may need a large number of data points to ensure good statistical power for differentiating between two areas

2.4.2.3 Rank Accuracy Metrics

Rank accuracy metrics compare order of the recommended object list with the order of user's actual object list. In contrast to predictive accuracy metrics, rank accuracy metrics do not deal with predicted ratings of objects. Frequently used rank accuracy metrics in the literature are: Pearson correlation, Spearman correlation, Kendall's correlation and half-life utility metrics.

2.5 Datasets for Recommender Systems

In some domains there are plenty of datasets (movie, book, article etc.); on the other hand in some domains finding suitable dataset is a challenging work. The requirements for a dataset vary according to the domain and design needs of the application. Several questions arise for a suitable dataset at this point. According to which metrics, the

evaluation is performed? Does the system make only off-line tests, or does it require on-line tests? Can evaluation be performed on simulated data, if a dataset is not currently available? Answers of previous questions point out suitable dataset for an application.

Herlocker categorized features of datasets into three groups: Domain features, inherent features and sample features. Characteristics of these features are listed below [36]:

Domain features of interest include:

- The content topic being recommended and the associated context in which recommendation takes place
- The user tasks supported by the recommender
- The novelty need
- The cost/benefit ratio of false/true positives/negatives
- The granularity of true user preferences

Inherent features include several features about ratings:

- Whether ratings are explicit, implicit, or both
- The scale on which items are rated
- The dimensions of rating
- The presence or absence of a timestamp on ratings
- The availability of user demographic information or item content information
- The biases involved in data collection

Sample features include properties commonly considered in evaluating a data set:

- The density of the ratings set overall, sometimes measured as the average percentage of items that have been rated per user; since many datasets have uneven popularity distributions, density may be artificially manipulated by including or excluding items
- The number or density of ratings from the users for whom recommendations are being made, which represents the experience of the user

in the system at the time of recommendation; ratings from users with significant experience can be withheld to simulate the condition when they were new users; and

- The general size and distribution properties of the data set - some data sets have more items than users, though most data sets have many more users than items.

In movie domain, one of the most widely used datasets is EachMovie dataset which is created for HP/Compaq Research. EachMovie was containing 2.811.983 ratings entered for 1628 movies. It was also containing timestamps for ratings and basic demographics about users. It was used for many applications including Canny's factor analysis algorithm [18], Domingos and Richardson's algorithm for computing network value [24], and Pennock's research on recommending through personality diagnosis algorithms [63]. Because of commercial reasons, EachMovie is not publicly available since 2004. MovieLens is based on EachMovie and used for most of recent movie recommendation systems [52, 66]. Jester dataset is created for joke recommendation and it has some additional features in addition to current datasets. It provides a complete training set that are rated by every single user, providing complete data on any subset of objects [31]. There are also datasets created for live systems. Schafer analyzed online user experiences with MetaLens [70], Rashid and Herlocker evaluated both quality of resulting predictions and subjective user experience in [**?**, Herlocker04, Rashid02]

2.6 User Evaluation Techniques

User evaluation techniques differ according to how data collection is performed. In some systems, ideas of users are explicitly asked. On the other side, some systems implicitly observe users behaviors. Circumstances in which the evaluations are made, also affect the success of the evaluation. Some researchers prefer to make their evaluations under controlled conditions which are called laboratory studies. Some other, especially working on context-sensitive domains, prefer field studies. A major distinction between laboratory studies and field studies is the duration of the experimentation. Laboratory studies are generally short-term studies for which the system cannot easily

reach a stable level; however field studies are experimented in long-term. Another cause of categorization within user evaluation techniques is the criterion of a successful recommendation. Some systems assume a recommendation as successful if the user purchases a recommended object. Some other systems take implicit data into consideration such as time spent on page or number of page view of the relevant object [36]. In his field study, Dahlen showed importance of user contribution to and participation in recommender systems in long term [22]. Herlocker evaluated the effect of objects' explanations to users' behaviors. He revealed that explanations have a significant effect on users' behaviors [36].

CHAPTER 3

MAIN WORK

3.1 Methodology

Our system, basically, aims to predict ratings of previously unrated movies for a sparse user-movie matrix. We develop several approaches. These approaches are pure content-based filtering (PCB), pure collaborative filtering (PCF) and content-boosted collaborative filtering (CBCF). All of these approaches work offline and incapable of online recommendation.

PCB bases its prediction process on movie content data and active user's profile. To gather information from movie content data and active user's previously given ratings PCB uses naïve Bayesian classifier (NBC). We use two decision rules in NBC named as maximum-likelihood and average-likelihood. Theoretical background and design details of PCB are explained in Section 3.2.

PCF bases its prediction process on previously given ratings to a movie by other users. Our PCF uses similarity-based neighborhood algorithm and as similarity measure it uses Pearson correlation coefficient. Details about PCF are covered in Section 3.3.

CBCF bases its prediction process on the output ratings of PCB (i.e. pseudo-ratings). CBCF performs same neighborhood-based collaborative filtering algorithm with PCF. After applying CF, differently from PCF, it combines predicted ratings of its CF part with pseudo-ratings. Design details of CBCF are explained in Section 3.4.

Clustering is applied to the final predictions of CBCF for scalability. As the clustering algorithm we use fuzzy c-means clustering. Since it is a fuzzy algorithm, each user

in the user space has memberships to a number of clusters instead of a single cluster. Fuzzy c-means clustering algorithm is an iterative algorithm trying to converge an optimal grouping of users. In order to minimize off-line cluster initialization time and uniformly distribute users, we propose a new initialization method. Fuzzy clustering algorithm and its initialization method used in our system are explained in Section 3.5.

In order to evaluate user opinions about recommended movies we implemented a graphical user interface. Our proposed interface asks users to rate at least 15 movies than according to these ratings it displays a recommendation list to users. Eventually, for each recommended movie, it asks the opinion of the user for an evaluation. Design and working mechanism of the user interface are explained in Section 3.6.

The problem space we deal with can be formulated as a matrix of users versus movies. In this matrix, each cell represents a user's rating on a specific movie. Under this formulation, our research problem reduces to predicting values for empty cells of the original matrix and listing the ones having highest predicted ratings. Original matrix is supposed to be very sparse, because each user is supposed to rate a small percentage of movies. We use a single numeric rating for each movie with high values representing interest and low values representing disinterest. We employ explicit user ratings on a scale of 1 to 5. Each approach(i.e. PCB, PCF, CBCF, CBCF$_{onl}$ and CBCF$_{dfc}$) tries to fill the matrix by its own prediction method. Throughout the thesis, when we say actual matrix we mean the original matrix composed of actual ratings. Also, when we say pseudo-matrix we mean the user-movie matrix which is the output of PCB.

3.2 Pure Content-Based Filtering Approach (PCB)

PCB analyzes movie descriptions to identify movies that are of particular interest to the user. Although using demographic data of users is possible, PCB only uses movie content data. Firstly, we discuss movie representations. Next, we explain what the user profile means for our system. We conclude with the theoretical background and design of the learning model we used. General overview and user-specified parameters of PCB are given in Figure 3.1

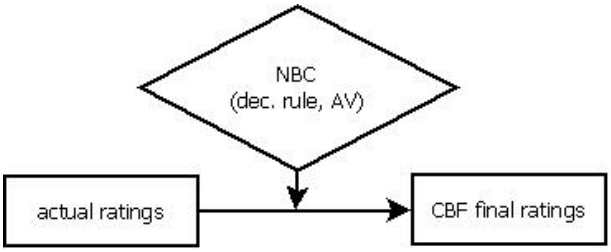


Figure 3.1: General overview of PCB

Representation of Movies

Movies those can be recommended to the user are stored in text files. Table 4.1 shows a sample of movie content file with rows describing five movies with IDs from 14 to 18. Columns indicate the attributes of movies. Each portion divided by a separator (|) corresponds to an attribute. These attributes are ID, name, release date, web address and genre of a movie, respectively.

Table 3.1: Sample movie content data

ID	Name	Release Date	URL	Genre
14	Postino,Il	01-Jan-1994	us.imdb.com/M/title-exact?Postino(1994)	0|0|0|0|0|0|0|0|1|0|0|0|0|0|1|0|0|0|0
15	MrHolland's Opus	29-Jan-1996	http://us.imdb.com/M/title-exact?Mr.Holland's Opus(1995)	0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0|
16	French Twist (Gazon maudit)	01-Jan-1995	http://us.imdb.com/M/title-exact?Gazonmaudit(1995)	0|0|0|0|0|1|0|0|0|0|0|0|0|1|0|0|0|0
17	From Dusk Till Dawn	05-Feb-1996	http://us.imdb.com/M/title-exact?FromDuskTillDawn(1996)	0|1|0|0|0|1|1|0|0|0|0|1|0|0|0|0|1|0|0
18	White Balloon, The	01-Jan-1995	http://us.imdb.com/M/title-exact?BadkonakeSefid(1995)	0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0|0

We use a movie's ID to uniquely identify the movie and genre information to classify the movies. Genre information is composed of 19 binary values each indicating membership of a movie to a genre. Details about genre information will be covered in Section 4.1

Learning a User Model

In our design we create a model of the user's preferences which is called classification learning. Our classification learning algorithm learns a function that models each user's interests. Given a previously unrated movie and the user model, this function predicts a numeric value corresponding to the degree of user's interest. We select a subset of the terms in movie content data as attributes of a movie (i.e. ID and genre information).

The algorithm we used to learn a user profile is naïve Bayes. Researchers stated naïve Bayes as an exceptionally well-performing text classification algorithm and have frequently adopted this algorithm in recent work [56, 51]. In naïve Bayes, learned profile is used to predict previously unrated movies. As it is explained before, naïve Bayesian classifier (NBC) is a simple probabilistic classifier based on Baye's Theorem. According to [82], Baye's Theorem expresses the posterior (i.e. after evidence E is observed) probability of a hypothesis H in terms of the prior probabilities of H and E, and the probability of E given H.

In its simplest form, Bayes' Theorem relates conditional and marginal probabilities of events A and B, where B has a non-zero probability.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{3.1}$$

In the above equality;

$P(A)$ is the marginal probability of A. It is marginal in the sense that it does not take into account any information about B.

$P(A|B)$ is the conditional probability of A, given B. It is derived from the specified value of B.

$P(B|A)$ is the conditional probability of B given A.

$P(B)$ is the marginal probability of B, and acts as a normalizing constant.

Bayes' theorem gives a mathematical representation of how the conditional probability of event A given B is related to the converse conditional probability of B given A [82].

For PCB we firstly create following probabilistic model:

$$p(C|F_1, \ldots, F_{19})$$

Above, C is a class variable that represents rating taking an integer value between 1 and 5. Movies are represented by a vector of features $F_1, F_2, \ldots, F_{19}$ in our model. F_i is a feature variable of a movie through 1 to 19. Features are binary values indicating membership of a movie to specified genres. We assume that all attributes are independent from each other and a movie can have membership to several genres.

We reformulate the model to make it more tractable. Using Bayes' theorem, we split this posterior distribution into a prior distribution and a likelihood:

$$p(C|F_1, \ldots, F_{19}) = \frac{p(C)p(F_1, \ldots, F_{19}|C)}{p(F_1, \ldots, F_{19})} \tag{3.2}$$

Equation 3.2 can also be written as

$$posterior = \frac{prior * likelihood}{evidence} \tag{3.3}$$

In our model for a user, prior probability for rating i corresponds:

$p(C_i)$ = (number of given is) / (total number of given ratings)

Likelihood of a movie $m = \{f_1, \ldots, f_{19}\}$ to rating i corresponds:

$$p(m|c_k) = \prod_{j=1}^{19} \frac{frequency_{kj}}{class_k} \tag{3.4}$$

In the above equation, $frequency_{kj}$ stands for number of given rating ks to movies belonging to genre j and $class_k$ stands for number of given rating ks of an active user.

Since the denominator does not depend on C and the values of the features F_i are given, we are only interested in the numerator of that fraction. If necessary operations are done the numerator can be decomposed as:

$$p(C) \prod_{i=1}^{19} p(F_i|C) = p(C)p(F_1|C)p(F_2|C) \ldots p(F_{19}|C) \tag{3.5}$$

As Equation 3.5 states, interaction between features in the same class is ignored. Although this ignorance causes false classifications, it is experimentally proved that

naïve Bayesian classifier performs very well in spite of these false classifications [23]. Another way of saying this is, naive Bayesian classifier can be a good ranker, even if it is a poor probability estimator [29].

NBC is performed for each user independently and combines proposed probability model with a decision rule. Here two decision rules can be applied to the probability model: Maximum likelihood and average likelihood. In maximum likelihood (See Equation 3.6) most probable class is picked and regarded as the prediction of NBC.

$$classify(f_1, \ldots, f_{19}) = argmax_c p(C = c) \prod_{i=1}^{19} p(F_i = f_i | C = c) \tag{3.6}$$

On the other hand, in average likelihood (See Equation 3.7) weighted sum of class' probabilities is computed and this value is regarded as the prediction of NBC.

$$classify(f_1, \ldots, f_{19}) = argavg_c p(C = c) \prod_{i=1}^{19} p(F_i = f_i | C = c) \tag{3.7}$$

NBC has a major challenge; if class or feature values do not occur together in the dataset then frequency-based probability estimate will be zero because of zero values in multiplication. To overcome this challenge we add a non-zero adjustment value (AV) to numerator and denominator of likelihoods (See Equation 3.8), a procedure which is also called *additive smoothing*. We employ AV as a user-specified parameter to be able to evaluate its effect also to the accuracy of predictions. AV is a decimal number on a scale of 0 to 1 inclusive and set to 0 by default.

$$p(m|c_k) = \prod_{j=1}^{19} \frac{frequency_{kj} + AV}{class_k + AV} \tag{3.8}$$

3.3 Pure Collaborative Filtering Approach (PCF)

As one of the most successful approaches for building recommender systems, collaborative filtering uses the known preferences of a group of users to make recommendations or predictions of the unknown preferences for other users. In other words, in collaborative filtering (CF) we try to predict how well a user will like a movie given a set of

preferences for other users [35].

Roughly speaking, every user provides a list of previously rated movies (i.e. user-movie ratings vector), and CF returns a list of predicted ratings for previously unrated movies. In our design, we use a neighborhood-based CF method. In our method, a subset of relevant users is chosen based on the similarity to the active user. After selection of similar users, an adjusted weighted sum of their ratings is used to calculate active user's predictions. We first discuss the weighting similarities between users. Next, we explain different neighborhood selection methods. We conclude with calculation of predictions. General overview of PCF is given in Figure 3.2

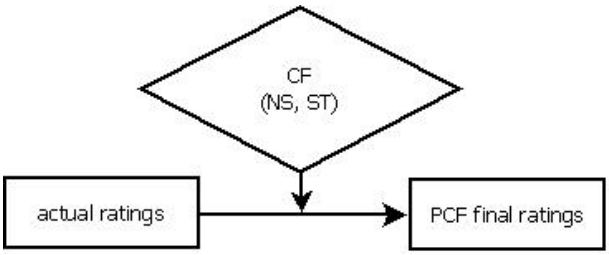


Figure 3.2: General overview of PCF

Similarity Weighting

Similarity between two users is measured by computing the Pearson correlation. Pearson correlation coefficient (PCC) is a measure of the linear dependence between two variables X and Y, giving a value between +1 and -1 inclusive. It is widely used as a measure of the strength of linear dependence between two variables.

$$r = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \overline{X})^2(Y_i - \overline{Y})^2}} \tag{3.9}$$

If we adopt above definition to our model, the Pearson correlation between users u and w becomes:

$$P_{u,w} = \frac{\sum_{i=1}^{NM}(r_{u,i} - \overline{r_u})(r_{w,i} - \overline{r_w})}{\sqrt{\sum_{i=1}^{NM}(r_{u,i} - \overline{r_u})^2(r_{w,i} - \overline{r_w})^2}} \tag{3.10}$$

where $r_{u,i}$ is the rating given to movie i by user u; $\overline{r_u}$ is the mean rating given by user u; and NM is the total number of movies. Similarity values obtained by using PCC are used for both neighborhood selection and computation of prediction.

We use two parameters to determine neighborhood of each user: Neighborhood size (NS) and similarity threshold (ST). NS is a user-specified positive integer and set to 30 by default. ST is a user-specified decimal number on a scale of 0 to 1 inclusive and set to 0.6 by default. According to our neighborhood selection algorithm, NS users with having similarities more than ST to the active user are chosen. In simple terms, NS and ST limit the size of neighborhoods. Although some systems develop different approaches, in our design each user has the same neighborhood for all movies.

Calculation of Final Predictions

Calculation of predictions is the last step in CF. So far, in many researches simple weighted average is used. However, this technique has a significant shortcoming: Rating tendencies of users differ. Some users tend to give high ratings, while some other tend to give low ratings. Because of this fact, using different users' ratings in the calculation, without taking mean ratings into consideration may be problematic. In our design, we use adjusted weighted sum (i.e. mean-centered weighted average) instead of simple weighted sum (See Equation 3.11). As it can be seen from Equation 3.12, in adjusted weighted sum we take the difference between actual rating and mean rating into consideration for each user.

$$p_{a,i} = \frac{\sum_{u=1}^{NS} r_{u,i} P_{a,u}}{\sum_{u=1}^{NS} P_{a,u}} \tag{3.11}$$

$$p_{a,i} = \overline{r_a} + \frac{\sum_{u=1}^{NS} (r_{u,i} - \overline{r_u}) P_{a,u}}{\sum_{u=1}^{NS} P_{a,u}} \tag{3.12}$$

Above, $p_{a,i}$ is the prediction for the active user a for movie i; $P_{a,u}$ is the similarity between users a and u.

3.4 Content-Boosted Collaborative Filtering Approach (CBCF)

Our CBCF approach is an effective recommendation framework that combines content-based and collaborative techniques. CBCF uses a content-based predictor to enhance existing user data, and then provides personalized suggestions through collaborative filtering [52]. The detailed description of CBCF is given below. Firstly, we perform CF on full user-movie matrix composed of pseudo-ratings. Then we combine these CF predictions with pseudo-ratings. General overview and user-specified parameters of CBCF are given in Figure 3.2.

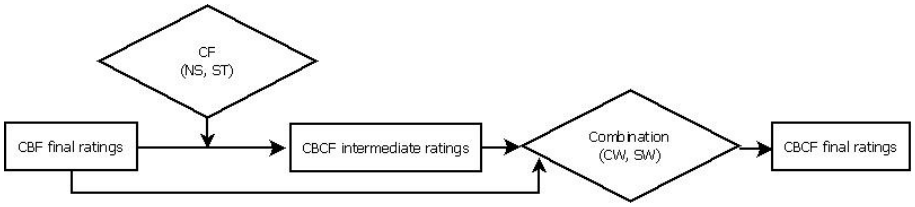


Figure 3.3: General overview of CBCF

Performing CF on Pseudo-Matrix

CBCF performs collaborative filtering on pseudo-ratings instead of actual ratings. The collaborative filtering algorithm we use is the same with PCF uses (See Section 3.3). NS and ST are user-specified parameters for also CBCF.

Combination of Predictions

In this part, we combine predicted ratings of CF with pseudo-ratings. We use a parameter named content weight (CW) which determines weight of pseudo-ratings in final predictions. At this point, we introduce an additional parameter: self weighting threshold (SW). For each user, SW indicates minimum number of rated movies for CW to be able to exactly affect the combination. There is a need for a parameter like SW because the accuracy of content-based predictions are directly proportional to the number of movies rated by the user. Usage of SW is given below:

if $nr_a < SW$ then $CW = \frac{nr_a}{SW} CW$

Above, nr_a stands for the number of movies rated by the active user

CW is a user-specified decimal number on a scale of 0 to 1 inclusive and set to 0.5 by default. SW is user-specified integer and set to 50 by default. As it can be understood, if CW is set to 1, then CBCF behaves as PCB.

The equation used to calculate final predictions of CBCF is given in Equation 3.13 [52].

$$p_{a,i} = \overline{v_a} + \frac{CW(ps_{a,i} - \overline{v_a}) + \sum_{u=1}^{NS} P_{a,u}(v_{u,i} - \overline{v_u})}{CW + \sum_{u=1}^{NS} P_{a,u}} \tag{3.13}$$

In the above equation $ps_{a,i}$ corresponds to the pseudo-rating for the active user and movie i, $v_{u,i}$ is the prediction rating of CF for user u and movie i, $\overline{v_u}$ is the mean rating of user u, $P_{a,u}$ is the Pearson correlation between users a and u, CW is the content ratio. The denominator is a normalization factor that ensures all weights sum to one.

3.5 Applying Fuzzy Clustering to CBCF

We apply fuzzy clustering to the final predictions of CBCF for scalability concerns. Our new model based strategy using dynamic fuzzy clustering is named CBCF$_{dfc}$. The structure of CBCF$_{dfc}$ is explained in detail in the following paragraphs.

General structure of fuzzy c-means

Fuzzy c-means (FCM) is a method of clustering which allows one piece of data to belong to two or more clusters. The working mechanism of FCM is explained in detail by [84]. FCM is basically based on minimization of the following objective function:

$$J_m = \sum_{i=1}^{N} \sum_{j=1}^{C} t_{ij}^m dist(u_i - c_j)^2 \tag{3.14}$$

Where m is any real number greater than 1, u_i is the ith user and c_j is the center of the jth cluster. t_{ij} is the degree of membership of u_i to cluster j, and $dist(*,*)$ is the distance between a user and the cluster center.

The degree of membership of a user is related to the inverse of the distance to the

cluster center:

$$t_{ij} = \frac{1}{dist(u_i, c_j)} \tag{3.15}$$

The memberships are normalized and fuzzyfied with a real parameter $m > 1$ so that their sum is 1. So the equation becomes:

$$t_{ij} = \frac{1}{\sum_k^{NC} (\frac{dist(x_i, c_j)}{dist(x_i, c_k)})^{\frac{2}{m-1}}} \tag{3.16}$$

In the above equation, NC is the number of clusters. m is a normalizing factor greater than 1. For m equal to 2, this is equivalent to normalizing the memberships linearly to make their sum 1. When m is close to 1, then cluster center closest to the point is given much more weight than the others, and the algorithm is similar to k-means [84].

Fuzzy partitioning is achieved through an iterative optimization of the objective function (See Equation 3.14), with the update of membership t_{ij} by Equation 3.16 and the cluster centers c_j by:

$$c_j = \frac{\sum_{i=1}^{NU} t_{ij}^m u_i}{\sum_{i=1}^{NU} t_{ij}^m} \tag{3.17}$$

This iteration will stop when $max_{ij}\{|u_{ij}^{k+1} - u_{ij}^k|\} < \epsilon$ where ϵ is a user specified value between 0 and 1, and k is the iteration steps. This procedure converges to a local minimum point of J_m. [84] summarizes fuzzy c-means in the following steps:

1. Initialize $T = [T_{ij}]$ matrix, $T^{(0)}$
2. At k-step: calculate the centers of clusters $C^{(k)} = [c_j]$ with $T^{(k)}$

$$c_j = \frac{\sum_{i=1}^{NU} t_{ij}^m u_i}{\sum_{i=1}^{NU} t_{ij}^m} \tag{3.18}$$

3. Update $T^{(k)}, T^{(k+1)}$

$$t_{ij} = \frac{1}{\sum_k^{NC} (\frac{dist(u_i, c_j)}{dist(u_i, c_k)})^{\frac{2}{m-1}}} \tag{3.19}$$

4. If $|T^{(k+1)} - T^{(k)}| < \epsilon$ then STOP; otherwise return to step 2.

To sum up, in our clustering algorithm each user has a degree of membership to clusters. The membership values of users to the clusters are inversely proportional

with their distance to the centers of the clusters. In other words, membership values of users near the center of a cluster are bigger than membership values of users on the edge of a cluster. Moreover, center of each cluster is hypothetical; contrary to some other methods taking actual users as center of clusters [53].

In fuzzy c-means some parameters are need to be set by user. These parameters are: Number of clusters (NC), degree of fuzziness (m) and sensitivity threshold (ϵ). NC is an integer value set to 10 by default. m is a decimal number bigger than 1 set to 2 by default. ϵ is used in the termination condition of iterations. It is a decimal number set to 0.01 by default.

The algorithm minimizes intra-cluster variance as well, but has the same problems as k-means, the minimum is a local minimum, and the results depend on the initial choice of weights [84]. To improve the performance of clustering and minimize offline cluster initialization time, we proposed an approach based on maximizing the distances between cluster centers.

Initialization of fuzzy c-means

Initialization is an important step of the fuzzy c-means. In our approach, we developed two methods: random initialization and max-distance initialization. In random initialization, as its name implies, we randomly select NC users as cluster centers.

On the other side, in max-distance initialization we iteratively select cluster centers according to their distance to initial cluster centers. The idea behind this method is choosing cluster centers as far as possible from each other in order to obtain separated cluster users. As distance measure we use Euclidean distance(See Equation 3.20).

$$e(u, w) = \sqrt{\sum_{i=1}^{NM} (r_{u,i} - r_{w,i})^2} \tag{3.20}$$

After selecting $k < NC$ initial cluster centers, we calculate distance function for the rest $(NU - k)$ users and select the one with the maximum distance to other cluster centers. The user u_i satisfying below condition will be the $(k+1)$th cluster center.

$$C_{k+1} = max\{e(u_i - c_1)^{pow} + \ldots e(u_i - c_k)^{pow}\}$$

Above *pow* is also a user-specified integer set to 2 by default.

In our approach, selecting initial clusters is not a trivial work. Number of initial clusters (NIC) is a user-specified integer less or equal than number of clusters (NC). It is set to 1 by default. Naïve approach for the selection of initial clusters is random selection. In addition to it, we developed another method which is supposed to select more distinguishable initial users compared to random selection. Distinguishable users are supposed to yield a better set of clusters in future iterations. The idea behind this is similar with the the idea behind max-distance initialization. We obtain these users by taking the users having maximum standard deviation values. The formula we used for the calculation of standard deviation for each user is given below.

$$st_u = \sqrt{\sum_{i=1}^{NM} (r_{u,i} - \overline{r_u})^2} \tag{3.21}$$

Above st_u is the standard deviation of user u, $\overline{r_u}$ is the mean rating of user u nad NM is the number of movies. After the calculation of standard deviation for each user we select top NIC of them as distinguishable users.

Using created clusters for prediction

The method we use is similar to Mojtaba's method in [53]. $C = c_1, c_2, \ldots, c_{NC}$ is the set of NC cluster centers representing NC clusters. The membership value t_{ij} of each user u_i to cluster C_j is proportional to its distance or dissimilarity from the cluster center c_j. The membership values of all the users are stored in matrix $T(NU\text{x}NC)$. For an active user u_a, we first find fuzzy nearest cluster, which is the cluster P to which the membership t_{pa} is maximum. Now similar users of u_a are the users having maximum membership to cluster P. Thus we simplify the computation of NS neighbors enormously. After neighborhood selection, we calculate the Pearson correlation between active user and NS users. As it is told in Section 3.4, prediction is done for only movies which have not been yet rated by u_a. The prediction formula we use is same with the one we use in PCF (See Equation 3.13).

In addition to using nearest cluster center in the selection of neighborhood, we use *top-NAC nearest* clusters. NAC is a user-specified integer representing number of active clusters in the selection of neighborhood and calculation of prediction for each online insertion. It is set to 1 by default. *Top-NAC nearest* clusters are selected according to the membership of the active user to NC clusters. Membership values

of u_a to NAC clusters determines the distribution of NS users among these clusters. Remaining work is same with the above approach.

3.6 User Interface Design

For user evaluation we implemented a graphical user interface. Firstly, our user interface guides the active user to rate a number of movies. Consequently, according to given ratings it displays a number of movies to the screen as a recommendation list. As it can be guessed, the recommended movies are selected by taking top predicted movies in the created user-movie ratings matrix. Finally, user interface asks the user to evaluate movies in the recommendation list. The detailed implementation of user interface is explained with screenshots below:

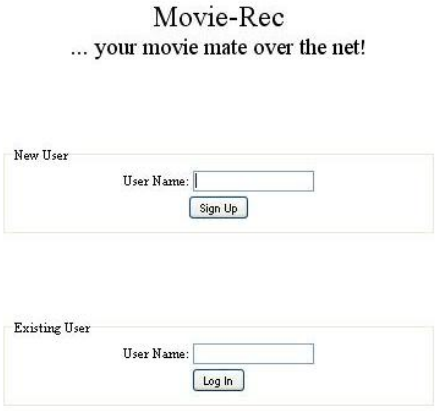


Figure 3.4: Login screen of Movie-Rec

The initial screen is given in Figure 3.4. As it can be seen from the figure, users first log in the system. After logging in, they are confronted with a screen composed of movies, each movie on a line followed by rating options from 1 to 5 (See Figure 3.5).

Figure 3.5: List of movies

Our recommender expects ratings of at least 15 movies from users. Although recommendations can be made with lower number of movies, for the sake of prediction accuracy we do not let recommendations below this threshold. When the user rates 15th movie, *Recommend* button appears at the bottom of the page. *Recommend* button directs the user to the evaluation screen (See Figure 3.6).

Figure 3.6: List of recommended movies

In final screen each user is supposed to evaluate recommended movies to himself. Here,

we present six choices to users as listed below:

- **positive attitude:** like - may like
- **negative attitude:** dislike - may not like
- **neutral attitude:** neutral - no idea

Recommended movies are divided into two groups first: Movies those are already watched and movies those are not wathced yet. Since there are hundreds of movies it is inevitable to force users to evaluate movies which they have not watced yet. To facilitate the evaluation process for users we put genre information and imdb link. For both of the situations(i.e. movies they already watched and movies they have not watched yet) users may prefer one of three attitudes; positive, negative and neutral. According to preferences of users, we evaluate the accuracy of the system. The evaluation of the system will be explained in detail in the next chapter.

CHAPTER 4

EVALUATION AND TEST RESULTS

4.1 Datasets

Our experimentations are based on MovieLens dataset that was collected by GroupLens Research Project [79]. There are currently three publicly available datasets in MovieLens database. These datasets having close sparsities are listed below:

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

MovieLens datasets have following characteristics:

- Each user has rated at least 20 movies.
- Information about movies is compatible with Internet Movie Database (imdb).
- Simple demographics about users (e.g. gender, profession, age) are available.
- Simple content data about movies (e.g. release date, genre) is available

All ratings are integer values on a scale of 1 to 5. One stands for lowest rating indicating a sharp dislike, five stands for highest rating indicating a sharp like. MovieLens datasets contain useful information about users and items. Id, name, date of release, imdb URL and genre of each movie are available. As it is explained in Section 3.2, only

Table 4.1: Sample movie content data

ID	Name	Release Date	URL	Genre
14	Postino,Il	01-Jan-1994	http://us.imdb.com/M/title-exact?Postino(1994)	0|0|0|0|0|0|0|0|1|0|0|0|0|0|1|0|0|0|0
15	MrHolland's Opus	29-Jan-1996	http://us.imdb.com/M/title-exact?Mr.Holland's Opus(1995)	0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0|
16	French Twist (Gazon maudit)	01-Jan-1995	http://us.imdb.com/M/title-exact?Gazonmaudit(1995)	0|0|0|0|0|1|0|0|0|0|0|0|0|1|0|0|0|0
17	From Dusk Till Dawn	05-Feb-1996	http://us.imdb.com/M/title-exact?FromDuskTillDawn(1996)	0|1|0|0|0|1|1|0|0|0|0|1|0|0|0|0|1|0|0
18	White Balloon, The	01-Jan-1995	http://us.imdb.com/M/title-exact?BadkonakeSefid(1995)	0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0

id and genre information of a movie is used. 18 genres are recognized by MovieLens (See Table 4.2)

Although we employ only user id in our design; age, gender, occupation and zip code of each user are also available. A sample of user content data is given in Table 4.3.

MovieLens provides training sets and test sets for experimentation. Training sets include user ID, movie ID, rating and timestamp information for a number of (*user,movie*) pairs (See Table 4.4).

4.2 Prediction Accuracy Metrics

Several metrics have been proposed for assessing the accuracy of recommender systems. There are two categories of accuracy metrics: Statistical accuracy metrics and decision-support accuracy metrics [35]. We evaluate our system with both of these metrics.

4.2.1 Mean Absolute Error

The MAE metric measures the average absolute deviation between a predicted rating and the user's actual rating. There are two major advantages of the MAE metric.

Table 4.2: Genre information

ID	Genre
0	Unknown
1	Action
2	Adventure
3	Animation
4	Children's
5	Comedy
6	Crime
7	Documentary
8	Drama
9	Fantasy
10	Film-Noir
11	Horror
12	Musical
13	Mystery
14	Romance
15	Sci-Fi
16	Thriller
17	War
18	Western

Table 4.3: Sample user demographic data

ID	Age	Gender	Occupation	Zip Code
24	21	F	artist	94533
25	39	M	engineer	55107
26	49	M	engineer	21044
27	40	F	librarian	30030
28	32	M	writer	55369
29	41	M	programmer	94043

Firstly, computation of the MAE is simple and understandable. Secondly, the MAE has well studied statistical properties that provide for testing the significance of a difference between two systems [36]. So far, the MAE (See Equation 4.1) has been used to evaluate the accuracy of many proposed systems [14, 35, 73].

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|a_i - p_i| = \frac{1}{n}\sum_{i=1}^{n}|e_i| \tag{4.1}$$

As the name suggests, the MAE is an average of the absolute errors $e_i = a_i - p_i$, where p_i is the prediction and a_i the actual value.

Lower MAE values indicates more accurate predictions.

Table 4.4: Sample training data

User ID	Movie ID	Rating	Timestamp
305	451	3	886324817
6	86	3	883603013
62	257	2	879372434
286	1014	5	879781125
200	222	5	876042340
210	40	3	891035994

4.2.2 Receiver Operating Characteristic

In our experiments as decision support accuracy measure, we use ROC area metric. ROC area metric is a measure of the diagnostic power of a filtering system. It is the area under the receiver operating characteristic (ROC) curve-a curve that plots the sensitivity and the 1-specificity of the test. Sensitivity is the probability of a randomly selected relevant object being accepted by the filter. Specificity is the probability of a randomly selected irrelevant object being rejected by the filter. The ROC curve plots sensitivity and 1 - specificity, obtaining a set of points by varying the quality threshold. The range of ROC sensitivity is between 0 and 1, where 0.5 is random and 1 is perfect. Comparing results of different systems using ROC curves may be subjective, contrary to comparing with the MAE. However, the area under a ROC curve can be used as a metric of the system's ability to discriminate relevant objects from irrelevant objects [34].

Herlocker et al. summarize advantages and disadvantages of ROC area metric [36]. Pros of the ROC area metric are:

- It provides a single number representing the overall performance of an information filtering system
- It is developed from solid statistical decision theory designed for measuring the performance of tasks such as those that a recommender system performs
- It covers the performance of the system over all different recommendation list lengths

Cons of the ROC area metric are:

- A large set of potentially relevant items is needed for each query,
- Equally distant swaps in rankings have the same effect no matter where in the ranking they occur,
- It may need a large number of data points to ensure good statistical power for differentiating between two areas.

4.3 Offline Evaluation

4.3.1 Methodology

In our offline experiments we use Mean Absolute Error (MAE) and Receiver Operating Characteristic (ROC) as statistical and decision-support accuracy metrics, respectively. Since rating scale we use ranges from 1 to 5, we select ROC-4 as decision support accuracy metric. According to ROC-4, predictions greater than or equal to 4 are supposed to be *relevant* movies and predictions less than 4 are supposed to be *irrelevant* movies for a user.

For each experiment, firstly we give brief background information. Secondly, we give the test results of experiments. Thirdly, we comment on the results and state the expected and unexpected outcomes of experiments. We perform our experiments for five distinct training sets of MovieLens (i.e. $u1$, $u2$, $u3$, $u4$, $u5$). For the sake of readability, increase on the MAE metric(i.e. decrease in prediction accuracy) is regarded as a negative change. Similarly, the increase on the ROC-4 metric(i.e. increase in classification accuracy) is regarded as a positive change.

In following sections, we evaluate experiments conducted by different parameters. Unless explicitly stated, we fix adjusting value (AV) to 0, neighborhood size (NS) to 30, similarity threshold (ST) to 0,6 and content weight (CW) to 0,5. Default values and types of these parameters are given in Table 4.5.

Before evaluating the accuracy of PCF, PCB and CBCF, we evaluate the effect of parameters to the prediction results. In Section 4.3.2 we compare decision rules of naïve Bayesian classifier: maximum likelihood and average likelihood. In Section 4.3.3

Table 4.5: Default values and types of parameters for offline evaluation

Parameter	Default Value	Type
adjusting value (AV)	0	[0,1]
neighborhood size (NS)	30	integer
similarity threshold (ST)	0,6	[0,1]
content weight (CW)	0,5	[0,1]

the effect of AV to the prediction accuracy of PCB is evaluated. In Section 4.3.4 and Section 4.3.5 we evaluate the effect of NS and ST. In Section 4.3.6 the effect of CW to the prediction accuracy of CBCF is evaluated.

Testing mechanism of offline evaluation is given in Figure 4.1.

4.3.2 Comparison of Maximum Likelihood and Average Likelihood for PCB

As it is explained in Section 3.2 we can apply two decision rules to our probability model: Maximum likelihood and average likelihood. In maximum likelihood we pick most probable class. In average likelihood we take the weighted sum of class' probabilities.

Table 4.6: Comparison of used decision rules

	MAE						ROC-4					
Approach	u1	u2	u3	u4	u5	average	u1	u2	u3	u4	u5	average
PCB_{max}	0,950	0,927	0,919	0,918	0,930	-	0,6093	0,6083	0,6138	0,6193	0,6104	-
PCB_{avg}	0,875	0,856	0,844	0,853	0,858	-	0,6088	0,6073	0,6020	0,6093	0,6050	-
% change of MAE and ROC-4 from PCB_{max} to PCB_{avg}	7,85	7,64	8,23	7,11	7,68	7,70	0,07	0,18	1,93	1,62	0,88	0,93

The results of our experiments are summarized in Table 4.6. PCB_{max} stands for PCB using maximum likelihood and PCB_{avg} stands for PCB using average likelihood. As it can be seen from Table 4.6, on the MAE metric, PCB_{avg} performs 7.7% better than PCB_{max}. On the ROC-4 metric, PCB_{avg} performs 0.94% better than PCB_{max}. This is an expected result, because when we pick most probable class as the class of

a movie, we implicitly neglect other classes' probabilities. In, PCB_{avg} we calculate the weighted sum of the numerical values of classes so that we obtain a more accurate classification of movies. In conclusion, PCB_{avg}, compared to PCB_{max} generates more accurate predictions which yields a decline on the MAE metric. Also it does a better of job of recommending relevant movies, while reducing the probability of recommending irrelevant movies to the user which yields an increase on the ROC-4 metric.

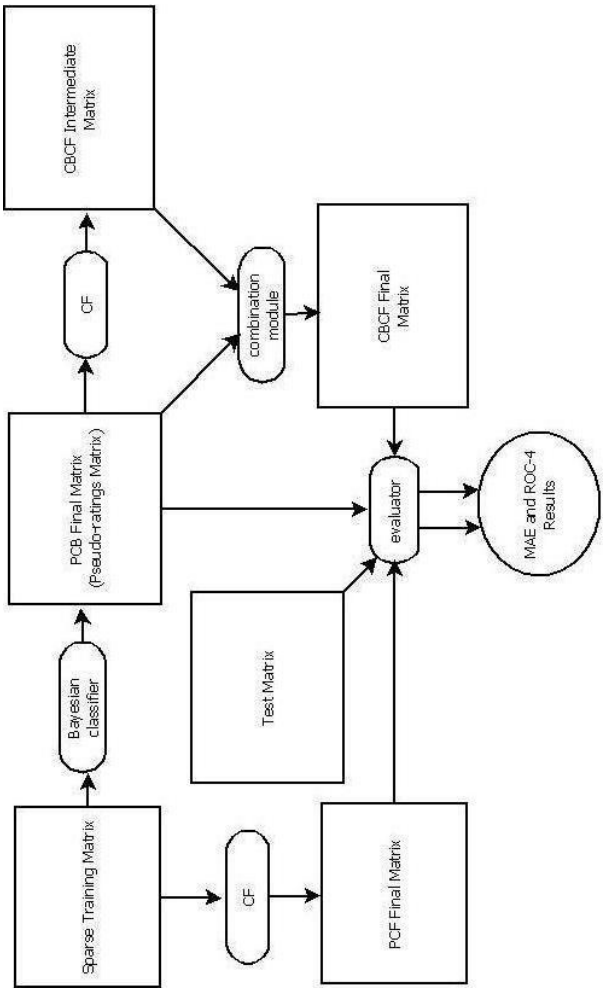


Figure 4.1: General mechanism of offline evaluation

4.3.3 Effect of Adjusting Value to the Accuracy of PCB

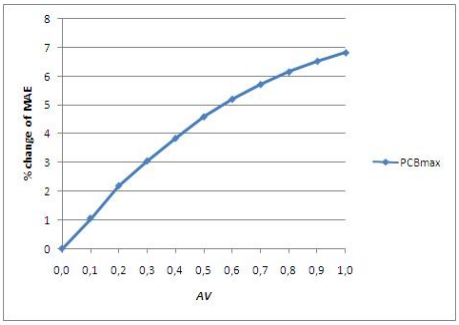
In this experiment, we tested the accuracy of of PCB_{max} and PCB_{avg} with 11 values of AV with $AV = \{0, 0.1, \ldots, 1\}$. The results of our experiments are summarized in

Table 4.7.

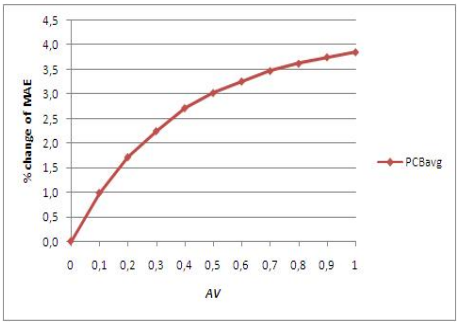
Table 4.7: Effect of AV to the accuracy of PCB_{max} and PCB_{avg}

		MAE						ROC-4					
Approach	AV	u1	u2	u3	u4	u5	% change of MAE from AV=0	u1	u2	u3	u4	u5	% change of ROC-4 from AV=0
PCB_{max}	0	0,950	0,927	0,919	0,918	0,930	-	0,6093	0,6083	0,6138	0,6193	0,6104	-
PCB_{max}	0,1	0,939	0,916	0,914	0,906	0,920	1,1	0,6105	0,6091	0,6135	0,6193	0,6102	0,05
PCB_{max}	0,2	0,928	0,908	0,903	0,897	0,909	2,2	0,6120	0,6100	0,6137	0,6208	0,6115	0,23
PCB_{max}	0,3	0,920	0,899	0,894	0,891	0,902	3,1	0,6130	0,6116	0,6150	0,6209	0,6114	0,35
PCB_{max}	0,4	0,913	0,894	0,886	0,883	0,894	3,8	0,6135	0,6117	0,6169	0,6216	0,6123	0,49
PCB_{max}	0,5	0,907	0,888	0,878	0,877	0,888	4,6	0,6153	0,6117	0,6178	0,6214	0,6133	0,60
PCB_{max}	0,6	0,900	0,885	0,872	0,871	0,884	5,2	0,6170	0,6115	0,6195	0,6218	0,6132	0,72
PCB_{max}	0,7	0,895	0,880	0,868	0,867	0,880	5,7	0,6179	0,6119	0,6196	0,6225	0,6139	0,81
PCB_{max}	0,8	0,891	0,876	0,864	0,863	0,877	6,2	0,6181	0,6124	0,6202	0,6230	0,6131	0,84
PCB_{max}	0,9	0,887	0,874	0,860	0,861	0,873	6,5	0,6186	0,6134	0,6207	0,6229	0,6136	0,92
PCB_{max}	1	0,885	0,873	0,857	0,859	0,868	6,8	0,6180	0,6127	0,6208	0,6224	0,6152	0,92
PCB_{avg}	0	0,875	0,856	0,844	0,853	0,858	-	0,6088	0,6073	0,6020	0,6093	0,6050	-
PCB_{avg}	0,1	0,867	0,848	0,836	0,845	0,848	1,0	0,6110	0,6061	0,6014	0,6081	0,6041	- 0,05
PCB_{avg}	0,2	0,860	0,842	0,831	0,838	0,842	1,7	0,6123	0,6079	0,6004	0,6091	0,6050	0,08
PCB_{avg}	0,3	0,856	0,838	0,827	0,834	0,837	2,2	0,6122	0,6107	0,6015	0,6103	0,6070	0,31
PCB_{avg}	0,4	0,852	0,834	0,823	0,830	0,833	2,7	0,6142	0,6121	0,6023	0,6116	0,6085	0,54
PCB_{avg}	0,5	0,849	0,831	0,821	0,828	0,830	3,0	0,6157	0,6143	0,6022	0,6112	0,6089	0,66
PCB_{avg}	0,6	0,847	0,829	0,819	0,826	0,828	3,3	0,6167	0,6153	0,6028	0,6122	0,6094	0,80
PCB_{avg}	0,7	0,844	0,828	0,817	0,825	0,827	3,5	0,6172	0,6153	0,6042	0,6132	0,6097	0,90
PCB_{avg}	0,8	0,843	0,827	0,816	0,824	0,825	3,6	0,6169	0,6165	0,6040	0,6137	0,6111	0,98
PCB_{avg}	0,9	0,841	0,826	0,815	0,822	0,824	3,8	0,6165	0,6171	0,6053	0,6143	0,6124	1,10
PCB_{avg}	1	0,841	0,825	0,815	0,821	0,823	3,9	0,6152	0,6175	0,6062	0,6142	0,6133	1,12

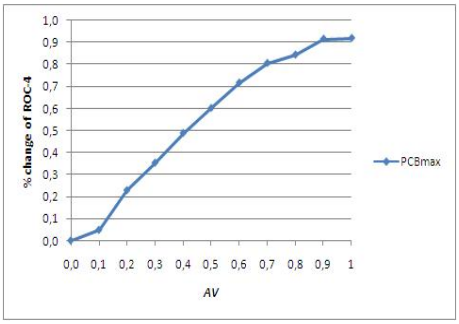
Change of accuracy with increasing values of AV is shown in Figure 4.2. Figure 4.2(a) and Figure 4.2(b) show the percentage change of the MAE metric from AV=0 to $AV = \{0.1, 0.2, \ldots, 1\}$ for PCB_{max} and PCB_{avg}, respectively. Similarly, Figure 4.2(c) and Figure 4.2(d) show the percentage change of ROC-4 metric from AV=0 to $AV = \{0.1, 0.2, \ldots, 1\}$.



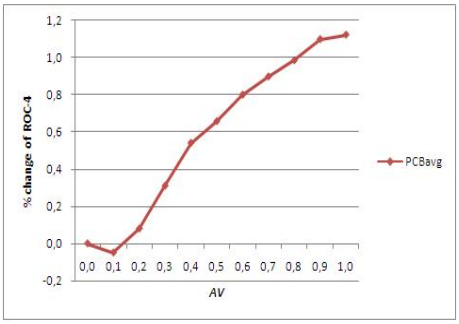
(a) percentage change of MAE from AV=0 to $AV = \{0, 0.1, \ldots, 1\}$ for PCB_{max}



(b) percentage change of MAE from AV=0 to $AV = \{0, 0.1, \ldots, 1\}$ for PCB_{avg}



(c) percentage change of ROC-4 from AV=0 to $AV = \{0, 0.1, \ldots, 1\}$ for PCB_{max}



(d) percentage change of ROC-4 from AV=0 to $AV = \{0, 0.1, \ldots, 1\}$ for PCB_{avg}

Figure 4.2: Effect of AV to the accuracy of PCB_{max} and PCB_{avg}

Results show that, as AV increases and gets closer to 1, PCB_{max} and PCB_{avg} perform a significant improvement on MAE and ROC-4 metrics. The reason behind the improvement in the prediction accuracy is the decrease of the effect of zero-valued probability estimates. However, the improvement diminishes as AV increases because of the addition of AV with both the numerator and denominator of the probability estimates (See Equation 3.8). To conclude, employing a sizeable parameter like AV is rational.

4.3.4 Effect of Neighborhood Size to the Accuracies of PCF and CBCF

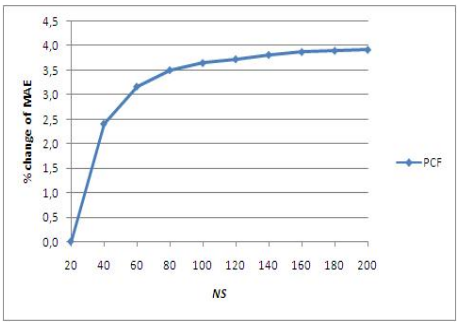
In this experiment, we tested accuracy of PCF and CBCF with 10 values of NS with $NS = \{20, 40, \ldots, 200\}$. The results of our experiments are summarized in Table 4.8.

Table 4.8: Effect of NS to the accuracy of PCF and CBCF

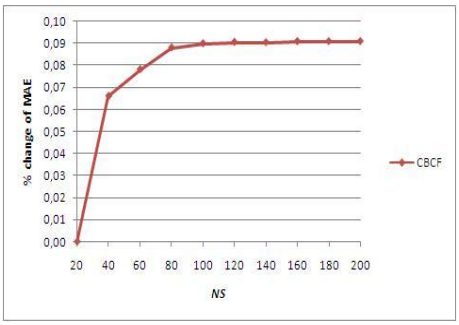
		MAE						ROC-4					
Approach	NS	u1	u2	u3	u4	u5	% change of MAE from NS = 20	u1	u2	u3	u4	u5	% change of ROC-4 from NS = 20
PCF	20	0,929	0,907	0,903	0,899	0,913	-	0,5837	0,6003	0,5974	0,5979	0,5923	-
PCF	40	0,911	0,886	0,879	0,879	0,888	2,41	0,5846	0,6017	0,5993	0,5986	0,5961	0,29
PCF	60	0,903	0,879	0,872	0,871	0,881	3,17	0,5862	0,6021	0,5999	0,5996	0,5962	0,42
PCF	80	0,900	0,875	0,869	0,868	0,879	3,51	0,5870	0,6022	0,6004	0,5999	0,5965	0,49
PCF	100	0,899	0,873	0,867	0,866	0,879	3,66	0,5870	0,6023	0,6002	0,6000	0,5965	0,49
PCF	120	0,898	0,873	0,867	0,865	0,878	3,74	0,5869	0,6028	0,6001	0,5999	0,5959	0,47
PCF	140	0,897	0,871	0,866	0,864	0,878	3,82	0,5871	0,6030	0,6004	0,5999	0,5957	0,49
PCF	160	0,897	0,871	0,866	0,865	0,876	3,89	0,5872	0,6032	0,6004	0,5998	0,5959	0,50
PCF	180	0,897	0,871	0,865	0,864	0,876	3,90	0,5874	0,6029	0,6005	0,5998	0,5961	0,50
PCF	200	0,897	0,870	0,865	0,864	0,875	3,93	0,5872	0,6031	0,6006	0,5996	0,5960	0,50
CBCF	20	0,842	0,825	0,816	0,820	0,823	-	0,6167	0,6206	0,6088	0,6180	0,6154	-
CBCF	40	0,841	0,825	0,815	0,820	0,822	0,066	0,6162	0,6207	0,6091	0,6177	0,6152	-0,018
CBCF	60	0,841	0,825	0,815	0,820	0,822	0,078	0,6162	0,6204	0,6090	0,6177	0,6154	-0,026
CBCF	80	0,841	0,824	0,815	0,820	0,822	0,088	0,6164	0,6203	0,6092	0,6175	0,6152	-0,030
CBCF	100	0,841	0,824	0,815	0,820	0,822	0,090	0,6163	0,6202	0,6091	0,6176	0,6151	-0,037
CBCF	120	0,841	0,824	0,815	0,820	0,822	0,091	0,6163	0,6203	0,6091	0,6176	0,6152	-0,034
CBCF	140	0,841	0,824	0,815	0,820	0,822	0,090	0,6163	0,6203	0,6091	0,6176	0,6152	-0,034
CBCF	160	0,841	0,824	0,815	0,820	0,822	0,091	0,6163	0,6203	0,6091	0,6176	0,6152	-0,034
CBCF	180	0,841	0,824	0,815	0,820	0,822	0,091	0,6163	0,6203	0,6091	0,6176	0,6152	-0,034
CBCF	200	0,841	0,824	0,815	0,820	0,822	0,091	0,6163	0,6203	0,6091	0,6176	0,6152	-0,034

Change of accuracy with increasing values of NS is shown in Figure 4.3. Figure 4.3(a) and Figure 4.3(b) show the percentage change of the MAE metric from NS=20 to $NS = \{40, 60, \ldots, 200\}$ for PCF and CBCF, respectively. Similarly, Figure 4.3(c) and Figure 4.3(d) show the percentage change of ROC-4 metric.

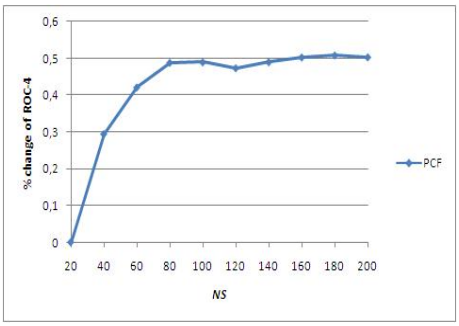
Results show that, the the accuracy of PCF, on both MAE and ROC-4 metrics, improves with increase of NS and remains almost constant after some point (See Figure 4.3(a) and Figure 4.3(c)). It is an expected result, because after a point additional neighbors are getting to have small similarities with the active user so that they do not have a significant effect in the calculation of final predictions. Although, CBCF follows a similar pattern with PCF on the MAE(See Figure 4.3(b)), comparative to PCF, CBCF is insensitive to change of NS on both metrics. This, also, is an expected result,



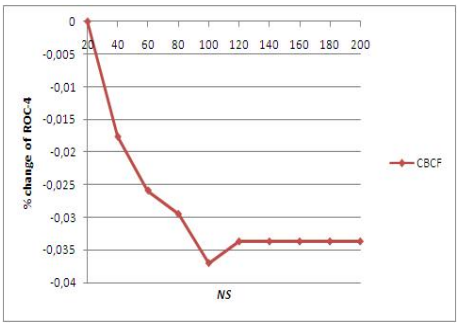
(a) percentage change of MAE from NS=20 to $NS = \{40, 60, \ldots, 200\}$ for PCF



(b) percentage change of MAE from NS=20 to $NS = \{40, 60, \ldots, 200\}$ for CBCF



(c) percentage change of ROC-4 from NS=20 to $NS = \{40, 60, \ldots, 200\}$ for PCF



(d) percentage change of ROC-4 from NS=20 to $NS = \{40, 60, \ldots, 200\}$ for CBCF

Figure 4.3: Effect of NS to the accuracy of PCF and CBCF

because CBCF performs CF on a dense matrix in contrast to PCF which performs CF on a very sparse matrix. Another reason is the weight of pseudo-ratings, which also decreases the effect of NS in prediction accuracy. Roughly speaking, increase of NS has a positive impact on the accuracy to some extent.

4.3.5 Effect of Similarity Threshold to the accuracy of PCF and CBCF

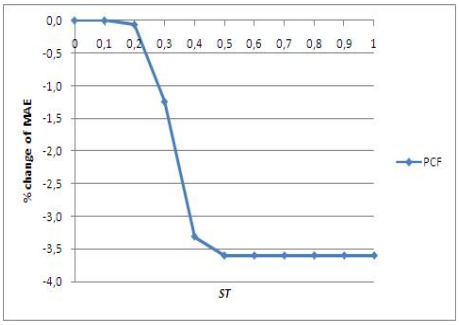
In this experiment, we tested the accuracy of PCF and CBCF with 11 values of ST with $ST = \{0, 0.1, \ldots, 1\}$. The results of our experiments are summarized in Table 4.9.

Table 4.9: Effect of ST to the accuracy of PCF and CBCF

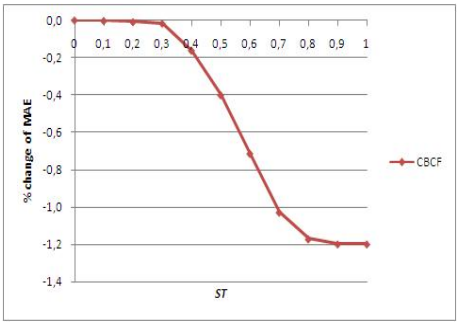
		MAE						ROC-4					
Approach	ST	u1	u2	u3	u4	u5	% change of MAE from ST = 0	u1	u2	u3	u4	u5	% change of ROC-4 from ST = 0
PCF	0	0,889	0,862	0,852	0,853	0,864	-	0,5760	0,5959	0,5945	0,5944	0,5968	-
PCF	0,1	0,889	0,862	0,852	0,853	0,865	0,00	0,5760	0,5959	0,5945	0,5944	0,5968	0,00
PCF	0,2	0,890	0,862	0,852	0,854	0,865	-0,06	0,5767	0,5966	0,5949	0,5943	0,5968	0,06
PCF	0,3	0,903	0,870	0,860	0,865	0,877	-1,25	0,5838	0,5998	0,5994	0,5968	0,5979	0,69
PCF	0,4	0,917	0,892	0,882	0,882	0,892	-3,31	0,5850	0,6026	0,5997	0,5993	0,5961	0,86
PCF	0,5	0,917	0,894	0,884	0,886	0,896	-3,60	0,5851	0,6021	0,5996	0,5979	0,5949	0,75
PCF	0,6	0,917	0,894	0,884	0,886	0,896	-3,60	0,5852	0,6021	0,5996	0,5979	0,5949	0,75
PCF	0,7	0,917	0,894	0,884	0,886	0,896	-3,60	0,5852	0,6021	0,5996	0,5979	0,5949	0,75
PCF	0,8	0,917	0,894	0,884	0,886	0,896	-3,60	0,5852	0,6021	0,5996	0,5979	0,5949	0,75
PCF	0,9	0,917	0,894	0,884	0,886	0,896	-3,60	0,5852	0,6021	0,5996	0,5979	0,5949	0,75
PCF	1	0,917	0,894	0,884	0,886	0,896	-3,60	0,5852	0,6021	0,5996	0,5979	0,5949	0,75
CBCF	0	0,836	0,820	0,808	0,814	0,817	-	0,6198	0,6194	0,6085	0,6163	0,6164	-
CBCF	0,1	0,836	0,820	0,808	0,814	0,817	0,00	0,6196	0,6194	0,6085	0,6163	0,6164	-0,01
CBCF	0,2	0,836	0,820	0,808	0,814	0,817	-0,01	0,6196	0,6196	0,6084	0,6161	0,6167	0,00
CBCF	0,3	0,836	0,821	0,809	0,814	0,817	-0,02	0,6200	0,6199	0,6083	0,6178	0,6171	0,09
CBCF	0,4	0,838	0,822	0,810	0,815	0,818	-0,16	0,6187	0,6208	0,6077	0,6178	0,6171	0,05
CBCF	0,5	0,839	0,823	0,812	0,818	0,819	-0,40	0,6172	0,6215	0,6080	0,6183	0,6160	0,02
CBCF	0,6	0,842	0,825	0,816	0,820	0,823	-0,71	0,6163	0,6206	0,6087	0,6180	0,6153	-0,05
CBCF	0,7	0,843	0,827	0,818	0,823	0,826	-1,03	0,6165	0,6196	0,6074	0,6167	0,6150	-0,17
CBCF	0,8	0,844	0,828	0,820	0,824	0,827	-1,17	0,6167	0,6197	0,6077	0,6160	0,6149	-0,17
CBCF	0,9	0,845	0,828	0,820	0,824	0,827	-1,20	0,6168	0,6196	0,6072	0,6159	0,6146	-0,21
CBCF	1	0,845	0,828	0,820	0,824	0,827	-1,20	0,6168	0,6196	0,6072	0,6160	0,6147	-0,20

Change of the accuracy with increasing values of ST is shown in Figure 4.4. Figure

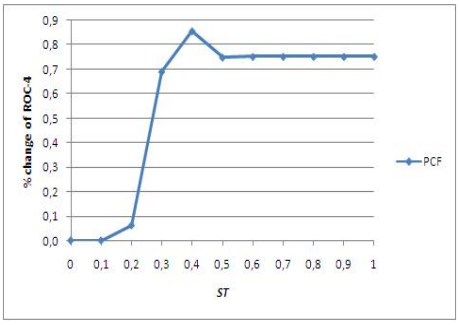
4.4(a) and Figure 4.4(b) show the percentage change of the MAE metric from ST=0 to $ST = \{0.1, 0.2, \ldots, 1\}$ for PCF and CBCF, respectively. Similarly, Figure 4.4(c) and Figure 4.4(d) show the percentage change of ROC-4 metric.



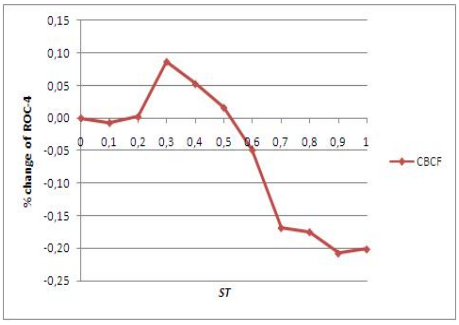
(a) percentage change of MAE from ST=0 to $ST = \{0, 0.1, \ldots, 1\}$ for PCF



(b) percentage change of MAE from ST=0 to $ST = \{0, 0.1, \ldots, 1\}$ for CBCF



(c) percentage change of ROC-4 from ST=0 to $ST = \{0, 0.1, \ldots, 1\}$ for PCF



(d) percentage change of ROC-4 from ST=0 to $ST = \{0, 0.1, \ldots, 1\}$ for CBCF

Figure 4.4: Effect of ST to the accuracy of PCF and CBCF

Results show that, the accuracy of PCF and CBCF decrease on the MAE metric with the increase in ST and remains almost constant after some point (See Figure 4.4(a) and Figure 4.4(b)). It is an expected outcome, because after a point, ST starts to limit neighborhood of the active user. Consequently, neighborhood becomes a smaller value than NS. As it is explained in Section 4.3.4 if number of neighbors decreases the accuracy also decreases to some extent. Additionally, we can infer that, effect of ST increases comparatively in smaller values of NS. On the ROC-4 metric PCF and CBCF follow similar patterns (See Figure 4.4(c) and Figure 4.4(d)), however relative

to PCF, CBCF is insensitive to change in NS. This, also, is also an expected result, because CBCF performs CF on a dense matrix in contrast to PCF which performs CF on a very sparse matrix. Another reason is the weight of pseudo-ratings, which also decreases the effect of ST in the accuracy.

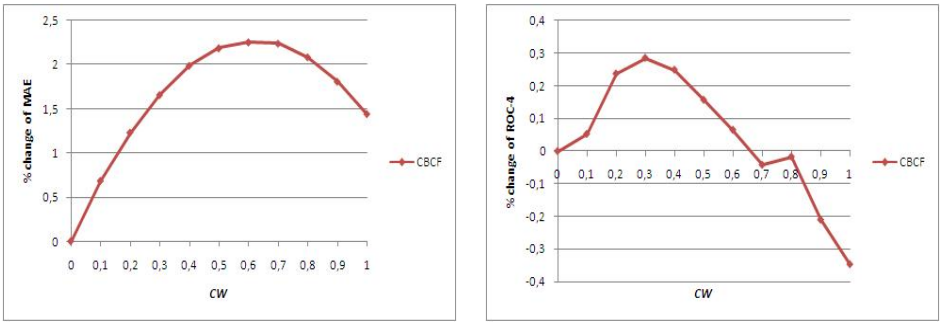
4.3.6 Effect of Content Weight to the Accuracy of CBCF

In this experiment, we tested the accuracy of CBCF with 11 values of CW with $CW = \{0, 0.1, \ldots, 1\}$. The results of our experiment are summarized in Table 4.10.As it is mentioned before, content weight(CW) indicates maximum weight of pseudo-ratings in the combination of CBCF's final predictions. CW is applied with self weighting (SW) threshold. SW indicates minimum number of rated movies for CW to totally affect the combination. For these experiments we fixed SW to 50. The results of our experiment are summarized in Table 4.10.

Table 4.10: Effect of CW to the accuracy of CBCF

		MAE						ROC-4					
Approach	CW	u1	u2	u3	u4	u5	% change of MAE from CW = 0	u1	u2	u3	u4	u5	% change of ROC-4 from CW = 0
CBCF	0	0,878	0,870	0,851	0,865	0,869	-	0,6110	0,6132	0,6053	0,6063	0,6036	-
CBCF	0,1	0,873	0,863	0,846	0,859	0,862	0,68	0,6120	0,6128	0,6046	0,6065	0,6050	0,05
CBCF	0,2	0,869	0,857	0,842	0,854	0,858	1,23	0,6126	0,6136	0,6052	0,6076	0,6076	0,24
CBCF	0,3	0,867	0,853	0,838	0,850	0,854	1,66	0,6135	0,6134	0,6051	0,6089	0,6072	0,28
CBCF	0,4	0,865	0,850	0,836	0,847	0,850	1,99	0,6126	0,6124	0,6051	0,6096	0,6073	0,25
CBCF	0,5	0,864	0,848	0,834	0,845	0,848	2,19	0,6126	0,6121	0,6044	0,6094	0,6055	0,16
CBCF	0,6	0,864	0,847	0,833	0,844	0,847	2,25	0,6120	0,6097	0,6039	0,6091	0,6067	0,06
CBCF	0,7	0,864	0,847	0,834	0,844	0,847	2,24	0,6108	0,6090	0,6028	0,6086	0,6068	-0,04
CBCF	0,8	0,866	0,848	0,835	0,845	0,848	2,09	0,6121	0,6091	0,6029	0,6094	0,6054	-0,02
CBCF	0,9	0,869	0,851	0,837	0,847	0,851	1,81	0,6094	0,6077	0,6034	0,6081	0,6044	-0,21
CBCF	1	0,872	0,854	0,841	0,850	0,854	1,44	0,6095	0,6076	0,6017	0,6073	0,6028	-0,35

Change of accuracy with increasing values of CW is shown in 4.5. 4.5(a) shows the percentage change of the MAE metric from CW=0 to $ST = \{0.1, 0.2, \ldots, 1\}$. Similarly, 4.5(b) shows the percentage change of the ROC-4 metric from CW=0 to $CW = \{0.1, 0.2, \ldots, 1\}$.



(a) percentage change of the MAE from CW=0 to $CW = \{0, 0.1, \ldots, 1\}$

(b) percentage change of ROC-4 from CW=0 to $CW = \{0, 0.1, \ldots, 1\}$

Figure 4.5: Effect of CW to the accuracy

Results show that, the accuracy of CBCF, on both MAE and ROC-4 metrics increases until some point and decreases after that point. As it can be seen in Figure 4.5, there is an optimal interval for CW around 0,5 on both MAE and ROC-4 metrics. It is understandable because predictions get worse as CW gets closer to endpoints consequently CBCF gets closer to PCF and PCB. As a result, choosing CW somewhere around 0,5 is rational.

4.3.7 Comparing Accuracies of PCB, PCF and CBCF

For default values of our parameters (See Table 4.5) we compare PCB, PCF and CBCF. The results of our experiments are summarized in Table 4.11.

Table 4.11: Comparison of CBCF with PCB and PCF

	MAE					ROC-4				
Approach	u1	u2	u3	u4	u5	u1	u2	u3	u4	u5
PCB	0,95	0,927	0,919	0,918	0,93	0,6093	0,6073	0,602	0,6093	0,605
PCF	0,897	0,871	0,866	0,864	0,878	0,6083	0,6021	0,5996	0,5979	0,5949
CBCF	0,864	0,848	0,834	0,845	0,848	0,6138	0,6121	0,6044	0,6094	0,6055

On the MAE metric, CBCF performs 9.06% better than PCB and 3.3% better than PCF. On the ROC-4 metric, CBCF performs 0.4% better than PCB and 1.4% better than PCF. To conclude CBCF outperforms PCB and PCF on both MAE and ROC-4 metrics(See Table 4.12). Although the improvements are significant, CBCF can

perform better with the optimized values of user-specified parameters.

Table 4.12: Summary of comparison results of CBCF with PCB and PCF

	MAE	ROC-4
CBCF vs. PCB	%9,6	%0,4
CBCF vs. PCF	%3,3	%1,4

4.4 Online Evaluation

4.4.1 Methodology

In our online experiments we use Mean Absolute Error (MAE) and Receiver Operating Characteristics (ROC) as statistical and decision-support accuracy metrics, respectively. We calculate elapsed time by dividing the number of clock ticks to the clock ticks per second. Since we construct the clusters offline we are not interested with the time elapsed while constructing the clusters.

For each experiment, firstly we give brief background information. Secondly, we give the test results of experiments. Thirdly, we interpret the results of each experiment. We perform our experiments for four distinct training sets of MovieLens (i.e. $u4$, $u5$, ua, ub). Increase on the MAE metric(i.e. decrease in prediction accuracy) is regarded as a negative change. Similarly, the increase on the ROC-4 metric(i.e. increase in classification accuracy) is regarded as a positive change. For online user evaluation instead of employing user-specified vectors, we use a portion of training matrix. Although, different ratios can be specified, we use 10% of training matrices for all experiments given below.

In following sections, we analyze experiments conducted by different parameters. Unless explicitly stated, we fixed number of clusters (NC) to 10, degree of fuzziness (m) to 2, sensitivity threshold (ϵ) to 0,01 and number of active clusters (NAC) to 1. Default values and types of these parameters are given in Table 4.13.

Table 4.13: Default values and types of parameters for offline evaluation

Parameter	Default Value	Type
number of clusters (NC)	10	integer
degree of fuzziness (m)	2	{1.25, 1.5, 2, 3}
sensitivity threshold (ϵ)	0.01	decimal
number of active clusters (NAC)	1	integer

Before evaluating accuracy and efficiency of CBCF$_{dfc}$ (i.e. CBCF with dynamic fuzzy clustering) and CBCF$_{onl}$ (i.e. online CBCF without clustering), we evaluate the effects of some parameters to the results. In Section 4.4.2 we evaluate the effect of NS. In Section 4.4.3 we evaluate the effect of NAC.

4.4.2 Effect of Neighborhood Size to the Accuracy and Efficiency of CBCF

In this experiment, we tested accuracy and efficiency of CBCF$_{dfc}$ and CBCF$_{onl}$ with $NS = \{10, 20, \ldots, 100\}$. The results of our experiments are summarized in Table 4.14 and Table 4.15.

Table 4.14: Effect of NS to the accuracy of CBCF$_{dfc}$ and CBCF$_{onl}$

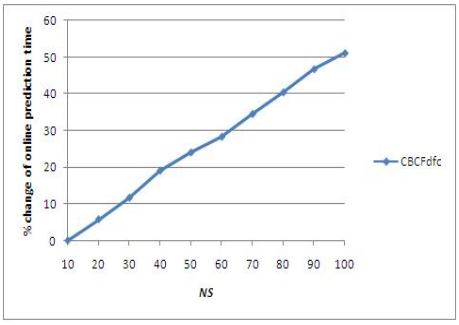
		MAE					ROC-4				
Approach	NS	u4	u5	ua	ub	% change of MAE from NS=10	u4	u5	ua	ub	% change of ROC-4 from NS=10
CBCF$_{dfc}$	10	0,820	0,827	0,831	0,846	-	0,6183	0,6153	0,6197	0,6258	-
CBCF$_{dfc}$	20	0,821	0,829	0,832	0,847	-0,09	0,6184	0,6154	0,6201	0,6259	0,024
CBCF$_{dfc}$	30	0,822	0,829	0,832	0,847	-0,11	0,6183	0,6155	0,6198	0,6260	0,018
CBCF$_{dfc}$	40	0,822	0,829	0,832	0,847	-0,14	0,6181	0,6151	0,6198	0,6262	0,002
CBCF$_{dfc}$	50	0,822	0,829	0,832	0,847	-0,15	0,6183	0,6150	0,6197	0,6261	-0,002
CBCF$_{dfc}$	60	0,822	0,829	0,832	0,847	-0,16	0,6184	0,6151	0,6200	0,6259	0,011
CBCF$_{dfc}$	70	0,822	0,829	0,832	0,847	-0,16	0,6183	0,6149	0,6198	0,6260	-0,007
CBCF$_{dfc}$	80	0,822	0,829	0,832	0,847	-0,16	0,6184	0,6152	0,6197	0,6260	0,008
CBCF$_{dfc}$	90	0,822	0,829	0,832	0,847	-0,16	0,6183	0,6151	0,6196	0,6259	-0,011
CBCF$_{dfc}$	100	0,822	0,829	0,832	0,847	-0,17	0,6184	0,6151	0,6197	0,6259	-0,003
CBCF$_{onl}$	10	0,822	0,823	0,829	0,847	-	0,6137	0,6130	0,6181	0,6228	-
CBCF$_{onl}$	20	0,822	0,822	0,829	0,847	0,04	0,6135	0,6131	0,6179	0,6230	0,013
CBCF$_{onl}$	30	0,821	0,822	0,829	0,846	0,05	0,6136	0,6127	0,6181	0,6231	-0,008
CBCF$_{onl}$	40	0,821	0,822	0,829	0,846	0,07	0,6137	0,6130	0,6178	0,6234	0,029
CBCF$_{onl}$	50	0,821	0,822	0,829	0,846	0,07	0,6137	0,6131	0,6180	0,6235	0,036
CBCF$_{onl}$	60	0,821	0,822	0,829	0,846	0,08	0,6137	0,6130	0,6180	0,6236	0,039
CBCF$_{onl}$	70	0,821	0,822	0,829	0,846	0,08	0,6137	0,6130	0,6180	0,6236	0,042
CBCF$_{onl}$	80	0,821	0,822	0,829	0,846	0,09	0,6137	0,6131	0,6182	0,6237	0,050
CBCF$_{onl}$	90	0,821	0,822	0,829	0,846	0,09	0,6137	0,6131	0,6182	0,6238	0,056
CBCF$_{onl}$	100	0,821	0,822	0,829	0,846	0,08	0,6137	0,6130	0,6182	0,6238	0,043

As it can be seen from Table 4.14 the accuracy of CBCF$_{dfc}$ remains nearly constant

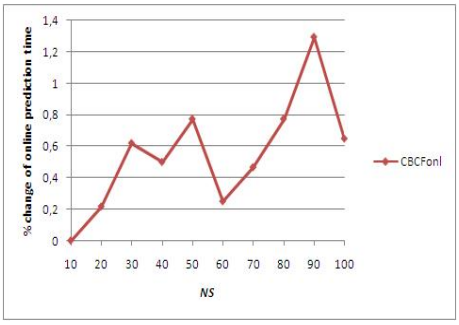
Table 4.15: Effect of NS to the efficiency of CBCF$_{dfc}$ and CBCF$_{onl}$

		online prediction time (s)				
Approach	NS	u4	u5	ua	ub	% change of time (s) from NS=10
CBCF$_{dfc}$	10	2,32	2,33	2,32	2,35	-
CBCF$_{dfc}$	20	2,47	2,45	2,47	2,47	-5,80
CBCF$_{dfc}$	30	2,61	2,61	2,59	2,60	-11,70
CBCF$_{dfc}$	40	2,84	2,72	2,76	2,78	-19,10
CBCF$_{dfc}$	50	2,88	2,96	2,85	2,87	-24,04
CBCF$_{dfc}$	60	3,02	2,98	2,98	2,98	-28,33
CBCF$_{dfc}$	70	3,17	3,16	3,10	3,11	-34,56
CBCF$_{dfc}$	80	3,37	3,25	3,23	3,24	-40,46
CBCF$_{dfc}$	90	3,47	3,49	3,36	3,36	-46,79
CBCF$_{dfc}$	100	3,47	3,53	3,54	3,55	-51,18
CBCF$_{onl}$	10	8,10	8,19	8,10	8,17	-
CBCF$_{onl}$	20	8,09	8,21	8,08	8,17	-0,22
CBCF$_{onl}$	30	8,22	8,18	8,14	8,16	-0,62
CBCF$_{onl}$	40	8,22	8,12	8,18	8,12	-0,50
CBCF$_{onl}$	50	8,18	8,21	8,24	8,17	-0,77
CBCF$_{onl}$	60	8,17	8,16	8,23	8,13	-0,25
CBCF$_{onl}$	70	8,18	8,14	8,16	8,18	-0,47
CBCF$_{onl}$	80	8,22	8,17	8,19	8,20	-0,77
CBCF$_{onl}$	90	8,32	8,28	8,20	8,19	-1,29
CBCF$_{onl}$	100	8,28	8,18	8,17	8,21	-0,65

on both metrics. It is an expected result, because our motivation for clustering was to be able create cluster centroids which represent users properly. Another factor for constancy in the accuracy is using fuzzy clustering instead of hard clustering. A proper representing set of users can be created with small number of neighbors by using fuzzy clustering techniques [84].



(a) percentage change of online prediction time from NS=10 to $NS = \{20, 30, \ldots, 100\}$ for CBCF$_{dfc}$



(b) percentage change of online prediction time from NS=10 to $NS = \{20, 30, \ldots, 100\}$ for CBCF$_{onl}$

Figure 4.6: Effect of NS to the efficiency of CBCF$_{dfc}$ and CBCF$_{onl}$

Change of efficiency with increasing values of NS is shown in Figure 4.6. Figure 4.6(a) and Figure 4.6(b) show the percentage change of efficiency in terms of online prediction time from NS=10 to $NS = \{20, 30, \ldots, 100\}$ for CBCF$_{dfc}$ and CBCF$_{onl}$, respectively.

As can be seen from Table 4.15 and Figure 4.6, online prediction time of CBCF$_{dfc}$ increases linearly where online prediction time of CBCF$_{onl}$ remains almost constant. Therefore, as we stated earlier(See Section 3.5) online prediction time of CBCF$_{dfc}$ is directly proportional with NS because of the similarity computation with NS users for each online insertion. And online prediction time of CBCF$_{onl}$ is not related with NS because of the similarity computation with NU users for each online insertion. As a result, setting NS to a small number is plausible.

4.4.3 Effect of Neighbor Clusters to the Accuracy and Efficiency of CBCF$_{dfc}$ and CBCF$_{onl}$

In this experiment, we tested the prediction and efficiency of CBCF$_{dfc}$ with $NAC = \{1, 2, \ldots, 10\}$. The results of our experiments are summarized in Table 4.16 and Table 4.17. Change of the accuracy and efficiency with increasing values of NAC is shown in Figure 4.7.

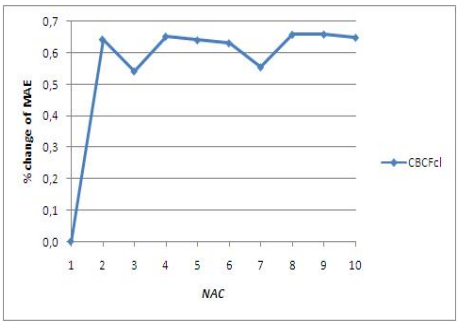
Table 4.16: Effect of NAC to the accuracy of CBCF$_{dfc}$

		MAE					ROC-4				
Approach	NAC	u4	u5	ua	ub	% change of MAE from NAC=1	u4	u5	ua	ub	% change of ROC-4 from NAC=1
CBCF$_{dfc}$	1	0,822	0,829	0,832	0,847	-	0,6183	0,6153	0,6198	0,6262	-
CBCF$_{dfc}$	2	0,818	0,819	0,827	0,843	0,64	0,6138	0,6115	0,6179	0,6225	-0,56
CBCF$_{dfc}$	3	0,821	0,820	0,827	0,843	0,54	0,6137	0,6123	0,6173	0,6233	-0,52
CBCF$_{dfc}$	4	0,818	0,819	0,827	0,843	0,65	0,6136	0,6120	0,6178	0,6230	-0,53
CBCF$_{dfc}$	5	0,818	0,820	0,827	0,843	0,64	0,6135	0,6121	0,6177	0,6230	-0,54
CBCF$_{dfc}$	6	0,818	0,819	0,827	0,844	0,63	0,6131	0,6124	0,6182	0,6232	-0,51
CBCF$_{dfc}$	7	0,825	0,819	0,827	0,843	0,55	0,6133	0,6128	0,6182	0,6237	-0,56
CBCF$_{dfc}$	8	0,818	0,819	0,827	0,843	0,66	0,6130	0,6120	0,6184	0,6227	-0,54
CBCF$_{dfc}$	9	0,818	0,819	0,827	0,843	0,66	0,6130	0,6117	0,6186	0,6229	-0,54
CBCF$_{dfc}$	10	0,818	0,819	0,827	0,843	0,65	0,6133	0,6121	0,6179	0,6233	-0,52

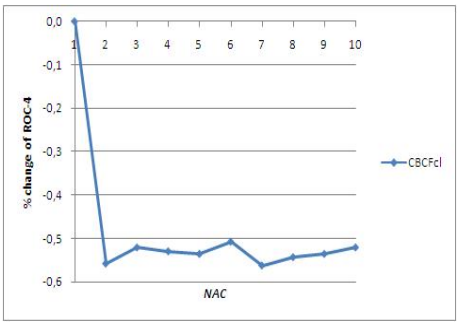
Table 4.17: Effect of NAC to efficiency of CBCF$_{dfc}$

		online prediction time (s)				
Approach	NAC	u4	u5	ua	ub	% change of time NAC=1
CBCF$_{dfc}$	1	2,63	2,63	2,63	2,62	-
CBCF$_{dfc}$	2	4,84	4,82	4,84	4,83	83,92
CBCF$_{dfc}$	3	7,23	7	7	6,98	168,41
CBCF$_{dfc}$	4	9,16	9,21	9,15	9,16	249,00
CBCF$_{dfc}$	5	11,38	11,34	11,39	11,28	331,87
CBCF$_{dfc}$	6	13,53	13,52	13,51	13,51	414,46
CBCF$_{dfc}$	7	15,65	16,15	15,6	15,67	500,09
CBCF$_{dfc}$	8	17,75	17,75	17,82	17,84	577,07
CBCF$_{dfc}$	9	20,58	20	20,61	20,59	678,12
CBCF$_{dfc}$	10	22,1	22,19	22,19	22,08	742,63

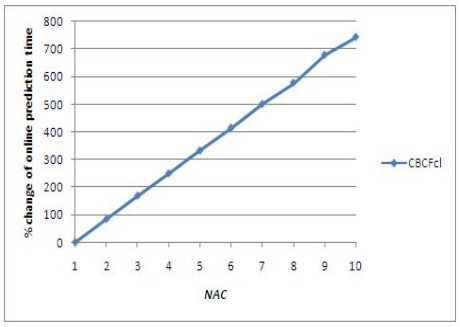
As it can be seen from Table 4.16, the statistical accuracy (i.e. MAE) of CBCF$_{dfc}$ increases first then remains nearly constant(See Figure 4.7(a)). Conversely, decision-support accuracy (i.e. ROC-4) of CBCF$_{dfc}$ decreases first then remains nearly constant (See Figure 4.7(b)). However, the changes on both metrics are insignificant (i.e. $<$0.1%) so that the optimal value for NAC should be chosen according to the efficiency of CBCF$_{dfc}$ with different values of NAC.



(a) percentage change of MAE from NAC=0 to $NAC = \{1, 2, \ldots, 10\}$ for CBCF$_{dfc}$



(b) percentage change of ROC-4 from NAC=0 to $NAC = \{1, 2, \ldots, 10\}$ for CBCF$_{dfc}$



(c) percentage change of online prediction time from NAC=0 to $NAC = \{1, 2, \ldots, 10\}$ for CBCF$_{dfc}$

Figure 4.7: Effect of NAC to the accuracy and efficiency of CBCF$_{dfc}$

Figure 4.7(c) shows the percentage change of efficiency in terms of online prediction time from NS=10 to $NS = \{20, 30, \ldots, 100\}$ for CBCF$_{dfc}$. As can be seen from Table 4.17 and Figure 4.7(c), online prediction time of CBCF$_{dfc}$ increases linearly with increase of NAC. As it is explained in Section 3.5, number of similarities computed in *top-NAC nearest* clusters approach is NACxNS causing linear increase of online prediction time, similar to the case in the above section. Consequently, for efficiency

in online prediction, setting NAC to a small number is plausible.

4.4.4 Comparing the Accuracy and Efficiency of and CBCF$_{dfc}$ with other approaches

For default values of our variables (See Table 4.13) we compare PCB, PCF and CBCF. The results of our experiments are summarized in Table 4.11. CBCF stands for offline CBCF which was already computed in Section 4.3.

Table 4.18: Comparison of CBCF$_{dfc}$ with CBCF and CBCF$_{onl}$

	MAE				ROC-4				online prediction time (s)			
Approach	u4	u5	ua	ub	u4	u5	ua	ub	u4	u5	ua	ub
CBCF	0,846	0,849	0,852	0,858	0,6238	0,6183	0,6160	0,6190	-	-	-	-
CBCF$_{onl}$	0,822	0,823	0,829	0,847	0,6137	0,6130	0,6181	0,6228	8,10	8,19	8,10	8,17
CBCF$_{dfc}$	0,820	0,827	0,831	0,846	0,6183	0,6153	0,6197	0,6258	2,32	2,33	2,32	2,35

As it is given in Table 4.19; on the MAE metric, CBCF$_{dfc}$ performs 2.40% better than CBCF and 0.14% better than CBCF$_{onl}$. On the ROC-4 metric, CBCF$_{dfc}$ performs 0.08% better than CBCF$_{onl}$ and 0.47% better than CBCF. Although CBCF$_{dfc}$ outperforms other approaches on both metrics the difference between them is quite marginal. Major gain of CBCF$_{dfc}$ is its efficiency compared to CBCF$_{onl}$. In terms of online prediction time CBCF$_{dfc}$ performs % 249 better than CBCF$_{onl}$. As a conclusion, using clustering on online prediction is considerably reasonable in respect to the accuracy and efficiency.

Table 4.19: Summary of comparison results of CBCF$_{dfc}$ with CBCF and CBCF$_{onl}$

	MAE	ROC-4	time (s)
CBCF$_{dfc}$ vs. CBCF	2,40%	0,08%	-
CBCF$_{dfc}$ vs. CBCF$_{onl}$	0,14%	0,47%	249%

4.5 Evaluation of Initialization of Fuzzy c-Means Clustering

As it is explained in 3.5, in addition to random initialization we developed a new method for fuzzy clustering which we called max-distance cluster initialization method. Differently from random initialization in max-distance initialization we iteratively se-

lect cluster centers according to their distance to initial cluster centers. We compare the accuracy and efficiency of these two methods in Section 4.5.1.

In addition to random selection of initial clusters in max-distance initialization method we developed a new method which is supposed to select more distinguishable initial users compared to random selection. Briefly, we obtain these users by taking the users having maximum standard deviation values. We compare accuracies of these two methods in Section 4.5.1.

for fuzzy clustering which we called max-distance cluster initialization method. Differently from random initialization in max-distance initialization we iteratively select cluster centers according to their distance to initial cluster centers. We compare the accuracy and efficiency of these two methods in Section 4.5.2.

4.5.1 Comparison of Cluster Initialization Methods

The results of our experiments are summarized in Table 4.20 and Table 4.20. In the tables given below, CBCF$_{dfc,rand}$ stands for CBCF$_{dfc}$ using random initialization method and CBCF$_{dfc,dist}$ stands for CBCF$_{dfc}$ using max-distance initialization method.

Table 4.20: Comparing accuracy of used cluster initialization methods

	MAE					ROC-4				
Approach	u4	u5	ua	ub	average	u4	u5	ua	ub	average
CBCF$_{dfc,rand}$	0,821	0,828	0,831	0,846	-	0,6180	0,6145	0,6191	0,6250	-
CBCF$_{dfc,dist}$	0,821	0,828	0,831	0,846	-	0,6186	0,6155	0,6198	0,6259	-
% change of MAE and ROC-4 from CBCF$_{dfc,rand}$	0,00	0,02	0,00	0,00	0,00	0,09	0,16	0,10	0,15	0,13

As it can be seen from Table 4.20, on the MAE metric, CBCF$_{dfc,rand}$ and CBCF$_{dfc,dist}$ perform almost equal. On the ROC-4 metric, CBCF$_{dfc,dist}$ performs 0.13% better than CBCF$_{dfc,rand}$. The difference between CBCF$_{dfc,rand}$ and CBCF$_{dfc,dist}$ is slightly marginal but still significant. The reason behind this difference in the accuracy is relatively balanced separation of cluster centers in CBCF$_{dfc,dist}$ according to the situation in CBCF$_{dfc,rand}$. As a result, with random selection of cluster centers we cannot guarantee to obtain ideal cluster centers so that we use max-distance initialization method in our approach.

Table 4.21: Comparing efficiency of used cluster initialization methods

	offline cluster construction time (s)					online prediction time (s)				
Approach	u4	u5	ua	ub	average	u4	u5	ua	ub	average
$CBCF_{dfc,rand}$	217	190	132	130	-	0,0285	0,0276	0,0281	0,0277	-
$CBCF_{dfc,dist}$	302	234	166	157	-	0,0282	0,0274	0,0273	0,0273	-
% change of MAE and ROC-4 from $CBCF_{dfc,rand}$	-40	-23	-26	-21	-28	1,1	0,7	3,0	1,5	1,6

Table 4.21 shows the comparison of efficiency between $CBCF_{dfc,rand}$ and $CBCF_{dfc,dist}$. According to offline cluster construction time, $CBCF_{dfc,dist}$ takes % 28 more time than $CBCF_{dfc,rand}$. However, since we are not interested with offline time, increase in cluster construction time does not pose a problem. On the other side, online efficiency of $CBCF_{dfc,dist}$ performs % 1,6 better than $CBCF_{dfc,rand}$. To conclude, there is a trade-off between online and offline times. Although, decline in the offline time is much more bigger than improvement in the online time as a percentage, due to our considerations mentioned in Section 3.5 we choose max-distance initialization method for cluster construction.

4.5.2 Comparison of Initial Cluster Selection Methods

The results of our experiments are summarized in Table 4.22.

Table 4.22: Comparing accuracy of used initial cluster selection methods

	MAE					ROC-4				
Approach	u4	u5	ua	ub	average	u4	u5	ua	ub	average
$CBCF_{dfc,dist-rand}$	0,822	0,829	0,832	0,847	-	0,6182	0,6148	0,6197	0,6257	-
$CBCF_{dfc,dist-stdev}$	0,822	0,829	0,832	0,847	-	0,6186	0,6153	0,6199	0,6263	-
% change of MAE and ROC-4 from $CBCF_{dfc,dist-rand}$	0,01	0,01	0,00	0,00	0,01	0,06	0,09	0,03	0,10	0,07

As it can be seen from Table 4.22, on the MAE metric, $CBCF_{dfc,dist-rand}$ and $CBCF_{dfc,dist-stdev}$ perform almost equal. On the ROC-4 metric, $CBCF_{dfc,dist-stdev}$ performs 0.07% better than $CBCF_{cl,dist-rand}$. The difference between $CBCF_{dfc,dist-rand}$ and $CBCF_{dfc,dist-stdev}$ is slightly marginal but still significant.

The reason behind this difference is similar to the reason explained in the above section. As it is expected, choosing the users having bigger standard deviation values yield more distinguishable initial clusters so that increases the accuracy. To conclude,

$\text{CBCF}_{dfc,dist-stdev}$ is more appropriate for initial cluster selection.

4.6 User Evaluation

4.6.1 Methodology

The main goal of user tests is to gather usefulness data of our interactive movie recommender. We have conducted the tests in January 2010 with 35 people all between 25-35 years. The tests have been conducted online each taking 15 minutes in average. Each test user rated movies on his own computer which is connected to the server. After entering rating information, Movie-Rec displays a recommendation list to the screen. Thereafter, the test user gives feedback about relativeness of movies in the recommendation list. Recommendation list is composed of 15 movies, each having six choices as it is explained in Section 3.6.

The most frequently encountered problem occurring during the tests was, that the most of the movies are not known by users. Genre information and imdb link are provided for each movie to solve this problem.

4.6.2 Test Results

Evaluation results of 35 users are given in Table 4.23.

Table 4.23: User evaluation results

	"already watched"			"not watched yet"			general attitude		
	like	dislike	neutral	may like	may not like	no idea	positive	negative	neutral
user1	2	1	0	5	2	5	47%	20%	33%
user2	3	0	0	6	1	5	60%	7%	33%
user3	0	0	0	9	2	4	60%	13%	27%
user4	0	0	1	10	2	2	67%	13%	20%
user5	4	1	1	6	1	2	67%	13%	20%
user6	5	0	0	7	1	2	80%	7%	13%
user7	1	0	0	9	2	3	67%	13%	20%
user8	2	1	0	10	1	1	80%	13%	7%
user9	3	2	0	7	1	2	67%	20%	13%
user10	3	0	1	8	1	2	73%	7%	20%
user11	4	0	0	5	0	6	60%	0%	40%
user12	2	1	0	9	0	3	73%	7%	20%
user13	3	1	0	7	0	4	67%	7%	27%
user14	3	0	1	5	2	4	53%	13%	33%
user15	2	0	0	9	0	4	73%	0%	27%
user16	1	0	0	12	0	2	87%	0%	13%
user17	1	0	1	7	2	4	53%	13%	33%
user18	4	1	0	7	2	1	73%	20%	7%
user19	3	1	2	8	1	0	73%	13%	13%
user20	4	0	0	7	3	1	73%	20%	7%
user21	2	0	0	9	2	2	73%	13%	13%
user22	2	0	1	7	1	4	60%	7%	33%
user23	1	1	0	7	2	4	53%	20%	27%
user24	3	0	1	8	0	3	73%	0%	27%
user25	4	2	0	6	0	3	67%	13%	20%
user26	1	0	0	9	1	4	67%	7%	27%
user27	1	1	0	10	2	1	73%	20%	7%
user28	0	0	0	10	0	5	67%	0%	33%
user29	1	1	0	8	1	4	60%	13%	27%
user30	0	0	0	11	3	1	73%	20%	7%
user31	2	1	1	9	2	0	73%	20%	7%
user32	3	1	0	7	1	3	67%	13%	20%
user33	1	0	0	9	3	2	67%	20%	13%
user34	4	0	0	8	0	3	80%	0%	20%
user35	1	0	0	8	3	3	60%	20%	20%

Table 4.24 summarizes the evaluation results. As it can be seen from Table 4.24 68% of recommended movies are evaluated as positive, 12% are evaluated as negative and 21% are evaluated as neutral by test users. The standart deviation of positive, negative and neutral feedbacks are 9%, 5% and 7%, respectively.

Table 4.24: Summary of user evaluation results

	mean	st. dev.
positive attitude	68%	9%
negative attitude	12%	5%
neutral attitude	21%	7%

According to user evaluation results, we can conclude that Movie-Rec recommends movies to the users with an acceptable accuracy.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

In this thesis, we propose a novel approach for movie recommendation, which resolves some limitations by combining collaborative filtering and content-based filtering techniques. Our hybrid approach is a unified model, which employs some user-specified parameters in order to get more accurate and efficient recommendations. We use well-known MovieLens rating data and the IMDB movie information for experiments. Our recommendation algorithm is based on Content-Boosted Collaborative Filtering (CBCF) algorithm which is proposed by Melville et al. in [52]. In our work, we extend and improve their proposed system by adding some new techniques and components.

Firstly, we enhance the naive Bayesian classifier (NBC) proposed in [52] in order to get a better classification of movies. In addition to maximum-likelihood that Melville et al. used, we use a different decision rule named average-likelihood. By means of test results, we see that NBC using average-likelihood outperforms NBC using maximum-likelihood on both accuracy metrics. We also use *additive smoothing* with a user-specified parameter (AV) to overcome anomalies caused by existence of zeros in the multiplication of the likelihood estimates. Test results indicate the optimal value of AV as a decimal number close 1. We compare accuracy of CBCF using average-likelihood with pure content-based filtering (PCB) and pure collaborative filtering (PCF). According to test results, CBCF outperforms PCB and PCF significantly on both metrics.

Secondly, in order to be able to make online recommendations efficiently we use a fuzzy

clustering model (i.e. fuzzy c-means clustering) that combines model and memory-based techniques. Although, fuzziness increases offline initialization time of clusters, it does not pose a problem because we are interested only in online recommendation time. We compare CBCF$_{dfc}$ (CBCF with dynamic fuzzy clustering) with CBCF$_{onl}$ (online CBCF without clustering) and CBCF (offline CBCF) in terms of accuracy and efficiency metrics. Results are surprising; in addition to dramatic decline from O(n) to O(1) in online recommendation time, CBCF$_{dfc}$ performs even better than CBCF$_{onl}$ on accuracy metrics. To sum up, accuracy of CBCF$_{dfc}$ is comparable with the accuracy of the CBCF$_{onl}$ and CBCF, on the other side scalability of CBCF$_{dfc}$ is considerably better than the scalability of CBCF$_{onl}$.

Thirdly, we propose a new cluster initialization method in order to increase accuracy and efficiency. In addition to random selection of clusters (CBCF$_{dfc,rand}$) we iteratively select cluster centers according to their distance to initial cluster centers (CBCF$_{dfc,dist}$). CBCF$_{dfc,dist}$'s offline cluster construction time is more than CBCF$_{dfc,dist}$'s cluster construction time. However, since we are not interested with offline time, it does not pose a problem. In all experiments, CBCF$_{dfc,dist}$ performed better than CBCF$_{dfc,rand}$ in accuracy and online efficiency. Also, in addition to random selection of initial clusters (CBCF$_{dfc,dist-rand}$), we select more distinguishable initial users by picking the users having maximum standard deviation values (CBCF$_{dfc,dist-stdev}$). Test results were meaningful, CBCF$_{dfc,dist-stdev}$ marginally better than CBCF$_{dfc,dist-stdev}$.

Fourthly, we evaluate the effect of user-specified parameters those significantly effect the accuracy and efficiency of the recommendations. These parameters are adjustment value (AV) in NBC, neighborhood size (NS) and similarity threshold (ST) in collaborative filtering engine, weight of content-based ratings (CW) in combination part of CBCF and number of clusters (NAC) in clustering part. AV is used for *additive smoothing* in NBC and according to test results choosing AV a relatively sizeable value increases accuracy as it is explained above. NS and ST are used to limit the number of neighbors of a user. Test results show that, effect of these parameters to the accuracy are similar. As the number of neighbors increases we see that accuracy increases until some point then remains almost constant. On the other side, increase of NS negatively effects efficiency of CBCF$_{dfc}$. The reason is clear; online recommendation time of CBCF$_{dfc}$ is linearly dependent with calculation time of neighborhood

selection in CF. So for NS and ST there is a trade-off between accuracy and efficiency. As the fourth parameter, CW, determines weight of content-based and collaborative characteristics of CBCF. If CW is a relatively big number than CBCF gets closer to PCB, otherwise CBCF gets closer to PCF. By means of various experiments we see that there is an optimal interval for CW on both accuracy metrics, so it is reasonable to keep CW in this optimal interval. As the last parameter, we employ NAC which corresponds number of active clusters used for the computation of final predictions in CBCF$_{dfc}$. Increase of NAC marginally improves accuracy, however negatively effects efficiency. As a result keeping NAC small is resonable.

Fifthly, we evaluate opinions of users about recommended movies. For user evaluation we use feedbacks obtained by the user interface. Feedbacks are grouped under three categories; positive, negative and neutral. Approximately 70% of users users agreed with recommended movies to them and approximately 10% disagreed. Received feedbacks are quite satisfactory in comparison with other studies in recommender systems literature.

In conclusion, our system CBCF$_{dfc}$ can be evaluated as an accurate and efficient online recommender for movie domain.

5.2 Future Work

- In CBCF$_{dfc}$ we only perform user-based similarity. This means while selecting neighbors only the similarity between users according to ratings they give are taken into consideration. In addition to user-based similarity, item-based similarity can be used. Item-based (i.e. movie-based) similarity measurement can improve the accuracy of the system.

- In our approach we do not implement reclustering of the user space. In dynamic databases it is a necessity to recluster database so that cluster centroids can properly represent users as the number of user insertions into the database increases.

- In addition to user-based clustering, item-based (movie-based) clustering also can be performed. Item-based clustering applied with user-based clustering can

decrease online recommendation time for each user.

- The user interface can be improved in order to provide a better evaluation framework.

REFERENCES

[1] Fuzzy c-means clustering. `http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/cmeans.html`, 2006.

[2] G. Adomavicius and Y. Kwon. New recommendation techniques for multicriteria rating systems. *IEEE Intelligent Systems*, 22(3):48–55, 2007.

[3] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[4] A. Ansari, S. Essegaier, and R. Kohli. Internet recommendations systems. *Journal of Marketing Research*, pages 363–375, 2000.

[5] C. Avery and R. Zeckhauser. Recommender systems for evaluating computer messages. *Communications of the ACM*, 40(3):88–89, 1997.

[6] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999.

[7] M. Balabanovic and Y. Shoham. Hybrid recommender systems: Survey and experiments. *Communications of the ACM*, 40(3):66–72, 1997.

[8] C. Basu, H. Hirsh, and W. Cohen. Recommendation as classification: Using social and content-based information in recommendation. Technical Report WS-98-08, Cornell University, 1998.

[9] P. Baudisch and L. Brueckner. Tv scout: Lowering the entry barrier to personalized tv program recommendation. In *Proceedings of the 2nd International Conference on Adaptive Hypermedia and Adaptive Web Based Systems*, 2002.

[10] N. J. Belkin and W. B. Croft. Information filtering and information retrieval: Two sides of the same coin? *Communications of the ACM*, 35(12):29–38, 1992.

[11] D. Billsus and M. Pazzani. Learning collaborative filters. In M. Kaufman, editor, *Proceedings of ICML*, 1998.

[12] D. Billsus and M. Pazzani. User modeling for adaptive news access. *User Modeling and User-Adapted Interaction*, 10(2-3):147–180, 2000.

[13] G. Bordogna and G. Pasi. Hierarchical-hyperspherical divisive fuzzy c-means (h2d-fcm) clustering for information retrieval. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, 2009.

[14] J. Breese, D. Heckerman, and C. Kadie. Empirical analysis of prediction algorithms for collaborative filtering. In *14th Conference on Uncertainly in Artificial Intelligence*, 1998.

[15] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, 1998.

[16] R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.

[17] C. Burkey and B. Krulwich. The infofinder agent - learning user interests through heuristic phrase extraction. *IEEE Expert: Intelligent Systems and Their Applications*, 12(5):22–27, 1997.

[18] J. Canny. Collaborative filtering with privacy via factor analysis. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information retrieval*, 2002.

[19] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin. Combining content-based and collaborative filters in an online newspaper. In *Workshop on Recommender Systems: Algorithms and Evaluation*, 1999.

[20] W. W. Cohen, R. E. Schapire, and Y. Singer. Learning to order things. *Journal of Articial Intelligence Research*, 10:243–270, 1999.

[21] M. Condliff, D. Lewis, D. Madigan, and C. Posse. Bayesian mixed-effects models for recommender systems. In *ACM SIGIR Workshop on Recommender Systems: Algorithms and Evaluation*, August 1999.

[22] B. J. Dahlen, J. A. Konstan, J. L. Herlocker, N. Good, A. Borchers, and J. Riedl. Jump-starting movielens: User benefits of starting a collaborative filtering system with "dead data". Technical Report TR-98-017, University of Minnesota, 1998.

[23] P. Domingos and M. Pazzani. On the optimality of the simple bayesian classifier under zero-one. *Machine Learning*, 29:103–130, 1997.

[24] P. Domingos and M. Richardson. Mining the network value of customers. In *Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining*, 2003.

[25] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley and Sons, 2001.

[26] S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of the Seventh International Conference on Information and Knowledge Management*, 1998.

[27] J. C. Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3:32–57, 1973.

[28] D. Fisher, K. Hildrum, J. Hong, M. Newman, M. Thomas, and R. Vuduc. Swami: A framework for collaborative filtering algorithm development and evaluation. *Research and Development in Information Retrieval*, pages 366–368, 2000.

[29] P. A. Flach and N. Lachice. Naive bayesian classification of structured data. *Machine Learning*, 29:1–37, 2004.

[30] P. W. Foltz. Using latent semantic indexing for information filtering. In R. B. Allen, editor, *Proceedings of the Conference on Office Information Systems*, 1990.

[31] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[32] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. A constant time collaborative filtering algorithm. *Information Retrieval Journal*, 4(2):133–151, 2001.

[33] N. Good, J. B. Schafer, J. A. Konstan, B. M. Borchers, A. Sarwar, J. L. Herlocker, and J. Riedl. Combining collaborative filtering with personal agents for better recommendations. In J. Hendler and D. Subramanian, editors, *Sixteenth National Conference on Artificial Intelligence*, 1999.

[34] J. A. Hanley and B. J. McNeil. The meaning and use of the area under a receiver operating characteristic (roc). *Radiology*, 143:29–36, 1982.

[35] J. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, 1999.

[36] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, 2004.

[37] W. Hill, L. Stead, M. Rosenstein, and G. Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of CHI*, 1995.

[38] T. Hoffman. Collaborative filtering via gaussian probabilistic latent semantic analysis. In *Proceedings of the 26th Annual International ACM SIGIR Conference*, 2003.

[39] A. Jennings and H. Higuchi. A user model neural network for a personal news service. *User Modeling and User-Adapted Interaction*, 3:1–25, 1993.

[40] B. Jeong, J. Leeb, and H. Chob. Improving memory-based collaborative filtering via similarity updating and prediction modulation. *Information Sciences*, 2009, 2009.

[41] P. Kazienko and P. Kolodziejski. Personalized integration of recommendation methods for e-commerce. *Artificial Intelligence Magazine*, 3(3):12–26, 2006.

[42] J. Kim, W. D. Oard, and K. Romanik. User modeling for information filtering based on implicit feedback. In *ISKO*, 2001.

[43] K. Kim and H. Ahn. A recommender system using ga k-means clustering in an online shopping market. *Expert Systems with Applications*, 34(2):1200–1209, 2008.

[44] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. Grouplens: Applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):77–87, 1997.

[45] Y. Koren. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009.

[46] M. Krulwich. Lifestyle finder: Intelligent user profiling using large-scale demographic data. *Artificial Intelligence Magazine*, 18(2):37–45, 1997.

[47] K. Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the 12th International Conference on Machine Learning*, 1995.

[48] Q. Li and B. M. Kim. Clustering approach for hybrid recommender system. In *Proceedings of IEEE/WIC International Conference*, 2003.

[49] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. In *IEEE Internet Computing*, January 2003.

[50] P. Maes, R. H. Guttman, and A. G. Moukas. Agents that buy and sell. *Communications of the ACM*, 42(3):81, 1999.

[51] A. McCallum, R. Rosenfeld, Mitchell T., and K. Nigam. Improving text classification by shrinkage in a hierarchy of classes. In *Proceedings of the International Conference on Machine Learning*, 1998.

[52] P. Melville, R. J. Mooney, and R. Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, 2002.

[53] B. Mojtaba and M. Reza. A new hybrid recommender system using dynamic fuzzy clustering. In *3rd International Conference on Information and Knowledge Technology*, January 2008.

[54] R. J. Mooney and L. Roy. Content-based book recommending using learning for text categorization. In *Workshop on Recommender Systems: Algorithms and Evaluation*, 1999.

[55] D. M. Nichols. Implicit rating and filtering. In *Proceedings of the Fifth DELOS Workshop on Filtering and Collaborative Filtering*, 1997.

[56] K. Nigam, A. McCallum, S. Thrun, and Mitchell T. Learning to classify text from labeled and unlabeled documents. In *Proceedings of the 15th International Conference on Artificial Intelligence*, 1998.

[57] D. Nikovski and V. Kulev. Induction of compact decision trees for personalized recommendation. In *Proceedings of the 2006 ACM symposium on Applied computing*, 2006.

[58] D. L. Nkweteyim. *A Collaborative Filtering Approach to Predict Web Pages of Interest from Navigation of Past Users within an Academic Website*. PhD thesis, University of Pittsburgh, 2005.

[59] M. Papagelis. Crawling the algorithmic foundations of recommender technologies. Master's thesis, University of Crete, 2005.

[60] D. Pavlov and D. Pennock. A maximum entropy approach to collaborative filtering in dynamic, sparse, high-dimensional domains. In *Proceedings of the 16th Annual Conference on Neural Information Processing Systems*, 2003.

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

[62] M. Pazzani and D. Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine Learning*, 27:313–331, 1997.

[63] D. M. Pennock, E. Horvitz, S. Lawrence, and C. L. Giles. Collaborative filtering by personality diagnosis: A hybrid memory and model-based approach. In *Proceedings of the Sixteenth Annual Conference on Uncertainty in Artificial Intelligence*, 2000.

[64] R. Popescul, L. H. Ungar, D. M. Pennock, and Lawrence S. Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, 2001.

[65] A. M. Rashid, I. Albert, D. Cosley, S. K. Lam, S. McNee, J. A. Konstan, and J. Riedl. Getting to know you: Learning new user preferences in recommender systems. In *Proceedings of the 2002 Conference on Intelligent User Interfaces*, 2002.

[66] J. A. Recio-Garcia, G. Jimenez-Diaz, A. A. Sanchez-Ruiz, and B. Diaz-Agudo. Personality aware recommendations to groups. In *Proceedings of the third ACM conference on Recommender systems*, 2009.

[67] P. Resnick, N. Iakovou, M. Sushak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 Computer Supported Cooperative Work Conference*, 1994.

[68] G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. Technical Report TR87-881, Cornell University, 1987.

[69] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Analysis of recommendation algorithms for e-commerce. In *ACM Conference on Electronic Commerce*, October 2000.

[70] J. B. Schafer, J. A. Konstan, and J. Riedl. Meta-recommendation systems:user-controlled integration of diverse recommendations. In *Proceedings of the 11th International Conference on Information and Knowledge Management*, November 2002.

[71] S. Schechter, M. Krishnan, and M. Smith. Using path profiles to predict http requests. In *7th International World Wide Web Conference*, 1998.

[72] L. Schmidt-Thieme. Compound classification models for recommender systems. In *Proceedings of the Fifth IEEE International Conference on Data Mining*, 2005.

[73] U. Shardanand and P. Maes. Social information filtering: Algorithms for automating word of mouth. In *Proceedings of the Conference on Human Factors in Computing Systems*, 1995.

[74] B. Sheth and P. Maes. Evolving agents for personalized information filtering. In *Proceedings of the 9th IEEE Conference on Artificial Intelligence for Applications*, 1993.

[75] M. Shigeki, I. Naomi, and H. Kazuo. Study on default voting for collaborative filtering. *Joho Shori Gakkai Kenkyu Hokoku*, 2001(20):183–188, 2001.

[76] I. Soboroff and C. Nicholas. Combining content and collaboration in text filtering. In *IJCAI Workshop: Machine Learning for Information Filtering*, August 1999.

[77] X. Su and T. M. Khoshgoftaar. Advances in artificial intelligence. *Journal of Machine Learning Research*, 2009:1–20, 2009.

[78] B. S. Suryavanshi, N. Shiri, and S. P. Mudur. An efficient technique for mining usage profiles using relational fuzzy subtractive clustering. In *Proceedings of the IEEE International Workshop Challenges in Web Information Retrieval and Integration*, April 2005.

[79] Grouplens Research Team. Movielens data sets. `http://www.grouplens.org/node/73`, March 2006.

[80] L. Terveen, W. Hill, B. Amento, D. McDonald, and J. Creter. Phoaks: A system for sharing recommendations. *Communications of the ACM*, 40(3):59–62, 1997.

[81] L. H. Ungar and D. P. Foster. Clustering methods for collaborative filtering. In *Recommender Systems*, 1998.

[82] Wikipedia. Bayes' theorem. `http://en.wikipedia.org/wiki/Bayes'_theorem`, December 2009.

[83] Wikipedia. Bayesian network. `http://en.wikipedia.org/wiki/Bayesian_network`, December 2009.

[84] Wikipedia. Cluster analysis. `http://en.wikipedia.org/wiki/Cluster_analysis`, December 2009.

[85] Wikipedia. Latent semantic analysis. `http://en.wikipedia.org/wiki/Latent_semantic_analysis`, December 2009.

[86] B. Yapriady and A. L. Uitdenbogerd. *Combining Demographic Data with Collaborative Filtering for Automatic Music Recommendation.* Springer Berlin / Heidelberg, 2005.

[87] T. Zhang and V. S. Iyengar. Recommender systems using linear classifiers. *Journal of Machine Learning Research*, 2:313–334, 2002.

[88] Y. Zhang and J. Jianxin. An associative classification-based recommendation system for personalization in b2c e-commerce applications. *Expert Systems with Applications*, 33(2):357–367, 2007.

[89] L. Zhu, F. L. Chung, and S. Wang. Generalized fuzzy c-means clustering algorithm with improved fuzzy partitions. *Systems Management and Cybernetics*, 39(3):578–591, 2009.