A CONTENT BOOSTED HYBRID RECOMMENDATION SYSTEM

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

BY

SEVAL ÇAPRAZ

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

JANUARY 2016

Approval of the thesis:

A CONTENT BOOSTED HYBRID RECOMMENDATION SYSTEM

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

Prof. Dr. Gülbin Dural Ünver
Dean, Graduate School of **Natural and Applied Sciences** _____

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

Asst. Prof. Dr. Selim Temizer
Supervisor, **Computer Engineering, METU** _____

Examining Committee Members:

Prof. Dr. Ferda Nur Alpaslan
Computer Engineering Department, METU _____

Asst. Prof. Dr. Selim Temizer
Computer Engineering Department, METU _____

Prof. Dr. Nihan Kesim Çiçekli
Computer Engineering Department, METU _____

Assoc. Prof. Dr. Pınar Karagöz
Computer Engineering Department, METU _____

Prof. Dr. Erdoğan Doğdu
Computer Engineering Department, TOBB ETÜ _____

Date: 26.01.2016

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: SEVAL ÇAPRAZ

Signature :

ABSTRACT

A CONTENT BOOSTED HYBRID RECOMMENDATION SYSTEM

Çapraz, Seval
M.S., Department of Computer Engineering
Supervisor : Asst. Prof. Dr. Selim Temizer

January 2016, 102 pages

Nowadays, most of e-commerce and social media sites use recommendation systems to help users find more relevant products easily. The key feature of recommendation is personalization which means different products are being offered for different users according to each user s interests. In literature, there are a lot of algorithms and tools which implement recommendation systems. The most common techniques for recommendation systems include Collaborative Filtering (CF) and Content-Based Filtering (CBF). To increase efficiency and accuracy, these methods can be combined in a hybrid recommendation system. Apache Mahout is one of the tools which focuses primarily on algorithms in the areas of CF, clustering and classification. In this study, we used Apache Mahout for blending item-based and user-based methods of CF with switching approach. The Pearson Correlation Similarity and Nearest N-User Algorithm is used in user-based CF, while Tanimoto Coefficient Similarity and Generic Boolean Preference is used in item-based CF. Moreover, we added genre-based average ratings as content-based filtering so that the final recommendation list becomes more

relevant to user. The proposed hybrid algorithm is tested on MovieLens dataset and validated with k-fold cross validation. This new hybrid recommendation system that is used to find patterns in data and develop a model for the purpose of making accurate and efficient recommender systems is proposed and detailed in this thesis study.

Keywords: Recommendation Systems, Hybrid Model, Collaborative Filtering, Content Based Filtering

ÖZ

İÇERİK ARTTIRIMLI HİBRİT BİR ÖNERİ SİSTEMİ

Çapraz, Seval
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Yrd. Doç. Dr. Selim Temizer

Ocak 2016 , 102 sayfa

Günümüzde e-ticaret ve sosyal medya sitelerinin çoğu tüketicilerin daha kolay ilgili ürün bulmalarına yardımcı olmak için öneri sistemleri kullanmaktadır. Öneri sisteminin anahtar özelliği kişiselleştirilmiş olmasıdır, yani kullanıcının ilgisine göre farklı kullanıcılar için farklı ürünlerin önerilmesidir. Literatürde öneri sistemlerini gerçekleştiren birçok algoritma ve araç vardır. Öneri sistemleri için en yaygın teknikler İşbirlikçi Filtreleme(CF) ve İçerik-Tabanlı Filtrelemedir(CBF). Etkinliğini ve doğruluğunu artırmak için, bu yöntemler bir hibrit öneri sistemi içerisinde birleştirilebilir. Apache Mahout, genel olarak CF, kümeleme ve sınıflanfırma algoritmalarına odaklanan bir araçtır. Bu çalışmada CF yöntemlerinden parça-bazlı ve kullanıcı-bazlı yöntemlerin makas değiştirme yaklaşımı ile harmanlanması için Apache Mahout kullanılmıştır. Parça-bazlı CF için Tanimoto Katsayısı Benzerlik ve Genel Boole Tercih algoritmaları kullanılırken, kullanıcı-bazlı CF için Pearson Korelasyon Benzerlik ve En Yakın N-Kullanıcı algoritması kullanılmıştır. Bundan başka, içerik-tabanlı filtreleme için tür-tabanlı ortalama

değerlendirme eklenmiştir, böylece sonuç öneri listesi kullanıcı ile daha alakalı olmuştur. Önerilen hibrit algoritma MovieLens veri kümesi ile test edilmiş ve k-kat çapraz geçerleme yöntemi ile onaylanmıştır. Öneri sistemlerini daha doğru ve etkili yapabilmek amacıyla veriler içerisinde desenler bulmak ve bir model geliştirmek için kullanılan bu yeni hibrit öneri sistemi bu tez çalışması içerisinde önerilmekte ve detaylandırılmaktadır.

Anahtar Kelimeler: Öneri Sistemleri, Hibrit model, İşbirlikçi Filtreleme, İçerik Tabanlı Filtreleme

To my family and people who are reading this page

ACKNOWLEDGMENTS

I would like to thank my supervisor Asst. Prof. Dr. Selim Temizer for his constant support, guidance and friendship. It was a great honor to work with him for this interesting topic and our cooperation influenced my academical and world view highly. He motivated and influenced me highly in scientific context.

There are a lot of people that were with me during finishing my master's degree. I am very grateful to all people I know in TÜBİTAK BİLGEM YTE which has great software engineers. They are very hardworking and kind. I love working with them. I am very lucky to have them all. I would also like to thank my mother who is always listening and leading me from the start of my life. This work is also supported by TÜBİTAK-BİDEB National Graduate Scholarship Programme for MSc (2210-C, 2013-2).

Lastly, sincerest thanks to my husband and each of my family members for supporting and believing in me all the way through my academic life.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

ALS	Alternating Least Squares
CBF	Content Based Filtering
CD	Compact Disc
CF	Collaborative Filtering
CSV	Comma Separated Value
CUDA	Compute Unified Device Architecture
DRMAA	Distributed Resource Management Application API
DRMS	Distributed Resource Management System
F	Female
FPGA	Field-Programmable Gate Array
GBF	Generic Boolean Preference
HDFS	Hadoop Distributed File System
IB	Item Based
ID	Identification, Identifier
IMDB	Internet Movie Database
IR	Information Retrieval
KNN	K Nearest Neighbor
LDA	Latent Drichlet Allocation
M	Male
MAE	Mean Absolute Error
METU	Middle East Technical University
NN	Nearest N-User
PCS	Pearson Correlation Similarity
PLSA	Probabilistic Latent Semantic Analysis
RMSE	Root Mean Squared Error
SVD	Singular Value Decomposition
TCS	Tanimoto Coefficient Similarity
TF-IDF	Term Frequency – Inverse Document Frequency

UB	User Based
1M	One Million

CHAPTER 1

INTRODUCTION

Nowadays, internet is available everywhere due to the advances in technology. There are cell phones, computers and smart watches that have internet connection capabilities and they are really cheap to buy. People are using these tools and provide more and more information through internet. Therefore there is a huge information overload and it is growing day by day. Information is obtained by not only personal computers and mobile phones, but also cash registers which people use when they are shopping. Cash registers are recording every user transaction. As a result, corporate databases store excessive amount of data like point-of-sale transactions and credit card purchases. For instance, Wal-Mart operates a chain of discount department stores and every day it collects 20 million point-of-sale transactions. In addition to this, research centers have huge databases for scientific purposes. Astronomical observatories store images of galaxies and they record every second of the universe. As a result, databases have grown from gigabytes to terabytes. To emphasize its significance, if we assume that each book requires 1 megabyte, then a terabyte is equivalent to about 1 million books. As a result, large data sets have become available everywhere. What will the information holders do with this huge data set? Huge data set can be beneficial if it is monitored and managed appropriately. In this sense, data mining is needed.

Data mining is a technique to discover patterns in data. It also provides finding associations, changes, anomalies and statistically significant structures in data [8]. It is easier to apply data mining techniques now than in the past. While

the data is growing, scientists and engineers propose more improvements in technology such as available and affordable computing power. Therefore data mining has become popular in many applications, because raw data by itself does not provide much information. They have to process, organize, structure or present the data in a given context in order to make it useful and gain more insights.

Several companies use data mining for many reasons. To name a few examples, firstly, CapitalOne bank uses data mining for forecasting whether a loan applicant will default on the loan. The prediction is based on given information about user's demographics, credit history, type of loan, etc. Secondly, Facebook uses data mining methods for prediction of activeness of users within 3 months time period. Thirdly, British law enforcement and intelligence agencies use data mining to predict the future behavior of people so that they can take precautions against crimes or security threats. We can give more examples from the e-commerce sector. E-commerce sites use recommendation systems to offer effective suggestions. Netflix (the largest DVD-by-mail rental company) and Amazon.com use data mining to provide recommendations to their customers. So they are recommending new products which customers might also be interested in besides the other products in their website. Recommendation is done also with personalization. So people are going to see recommended products which they are interested in. Moreover the results may be useful for prediction in order to guess future buying behavior. Recommendation systems are popular not only in e-commerce sector, but also in digital media sector. The websites which offer music, movie or photography use recommendation systems. There are many reasons to apply recommendation systems. An internet radio can choose the next song to play using data mining algorithms, or a movie website can suggest a movie which users may like. There is too much to watch, listen to or read. So people need to filter, make choices and select only the content or information that is relevant or interesting for them personally. Therefore data mining helps them to eliminate unrelated content. This thesis investigates these data mining techniques for recommendation and proposes a hybrid method.

In early 1990s, the first automated systems were developed that could assist

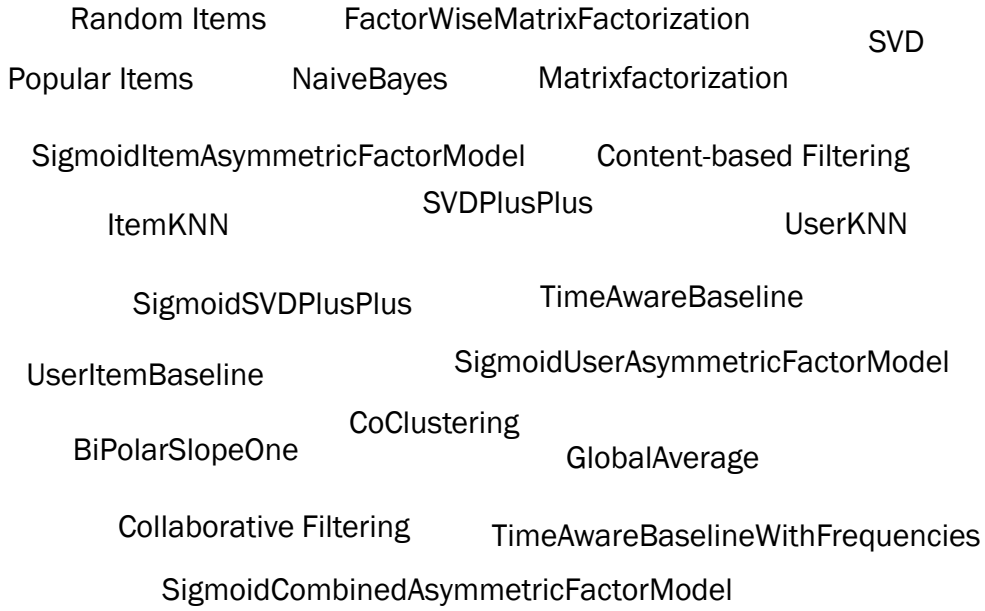

Figure 1.1: A word cloud of major data mining algorithms

users in filtering interesting content. The concept of collaborative filtering was born. It uses the behavior of the community as a guide for individual users. This idea was soon followed by other approaches on tackling the information overload problem and over the last 20 years hundreds of new recommendation algorithms or strategies were developed and researched. To create a recommendation system, we need to use data mining techniques. So the tasks are really obvious: first we need to find patterns in data. Second, develop a model for purpose. Lastly we need to make accurate and efficient recommendation.

If we want to recommend content to users, what algorithm do we use? There are a lot of data mining algorithms (Figure 1.1). How can we choose from so many and be sure to have made a good choice? There are many types of recommendation systems: search-based recommendations, category-based recommendations, collaborative filtering, clustering, association rules, information filtering, classifiers.

In the last few years there has even been a rise in software libraries such as

- Mymedialite

- Lenskit
- Mahout

They offer implementations of recommendation algorithms out of the box. **Mymedialite** is a lightweight, multi-purpose library of recommender system algorithms such as GlobalAverage, UserAverage, ItemAverage, SlopeOne, UserItemBaseline, ItemKNNPearson, FactorWiseMatrixFactorization, MatrixFactorization, BiasedMatrixFactorization, SVDPlusPlus. **Lenskit** provides flexible, measurable implementation of Item-based CF, User-based CF, Matrix factorization and Slope-One algorithms. **Mahout** also provides many implementation of algorithms such as Item-based CF, User-based CF, Matrix Factorization with ALS, Weighted Matrix Factorization, SVD PlusPlus etc.

Implementing an advice algorithm is now too easy like clicking a button, but again the problem now lies on choosing ideal advice algorithm for a given recommendation scenario.

Recent studies indicated that associating more than one data mining algorithm may give better results. For this reason, combining collaborative filtering (CF) and content based filtering (CBF) approach has been used to create a hybrid recommender system in this thesis. There are several reasons why these two algorithms are selected. First of all, they are not difficult to implement and they give favorable results as it is articulated in later chapters.

Collaborative filtering (CF) techniques make a comparison between customers depending on their previous acquisitions to make recommendations for similar customers. It is also called social filtering. To implement a CF algorithm, these steps should be followed: Firstly, find customers who are similar regarding tastes, preferences and past behaviors. To find similar customers, nearest neighbors algorithm can be applied. Secondly, accumulate weighted choices of these neighbors. Thirdly, make suggestion based on accumulated weighted choices.

To illustrate the concept, an example is given in Figure 1.2. We can recommend books to User 3. This user is very close to User 2. He has bought all the books User 2 has bought. Therefore Book 5 is highly recommended. User 4 is

	Book 1	Book 2	Book 3	Book 4	Book 5	Book 6
User 1	X			X		
User 2		X	X		X	
User 3		X	X			
User 4		X				X
User 5	X				X	

Figure 1.2: An Example of Collaborative Filtering

somewhat close to User 3. Book 6 is recommended to a lower extent. User 1 and User 5 are not similar to User 3 at all. Their weights are going to be zero. As a result, Book 5 and Book 6 are recommended to this user based on collaborative filtering.

There are certain advantages of CF. First of all, it is extremely powerful and efficient. Furthermore it finds very relevant recommendations. If the database is so big, it means there are a lot of past behaviors. Therefore CF provides better recommendations. The bigger the database, the more the past behaviors, the better the recommendations. There are also disadvantages of CF. It is difficult to implement, and it is somewhat resource and time-consuming. This algorithm is fiercely dependent on the past behavior of users. What about a new item that has never been purchased? It cannot be recommended because it does not have any history. What about a new customer who has never bought anything? He or she cannot be compared to other customers, therefore no items can be recommended to him or her.

The other algorithm which we used in the hybrid algorithm is content based filtering (CBF). This method considers the features of items. The features can be considered as attributes or characteristics of the item. For example, the attributes may be the type, stars, year etc. for a film. It can also be textual content like title, description, table of contents, etc. To implement the CBF, the average scores for each feature is calculated by using original data. The average scores are weights for suggestions. Each weight can be added to all recommendations if the recommended item has the feature. Suppose that, the average score of a user for movies which star Leonardo Dicaprio is 4. If the suggested movie also stars Leonardo Dicaprio, we can add 4 to the predicted

rating.

The second common usage of CBF is in information retrieval. CBF technique is mostly used for finding similar text documents in order to estimate the distance between two textual documents. CBF scans and parses a textual document and counts word occurrences. Several words or tokens which are too common in daily language are not accounted for CBF. These words are stop words. For instance, the, a, for... etc. Moreover, CBF includes the words that do not appear enough in documents. It transforms each document into a normed TFIDF (Term Frequency / Inverted Document Frequency) vector. The distance between any pair of vectors is calculated mathematically.

There are advantages of CBF. There is no need for past purchase history. It is not extremely difficult to implement. On the other hand, there are certain disadvantages, as well. It offers static recommendations. When the content is not very informative, it is not efficient. Information filtering is more suited to recommend technical books than novels or movies. Generally collaborative systems report a better performance than content-based approaches, but their success relies on the presence of a sufficient number of user ratings. So both algorithms have problems if we use them alone. We are going to combine these algorithms as a hybrid recommender system. Of course, this method is done in history many times. But we contributed by adding some innovative processes in it.

In literature, there are many researches about e-learning and e-commerce which propose hybrid methods but their general approach is not in total alignment with ours. When more than one algorithm is used, it is called hybrid. So the algorithms can be different for different hybrid methods. In our proposed method, we are going to use MovieLens 1M dataset. The recommendation is going to be made by comparing the watching habits and given scores of similar users. It is collaborative filtering (CF). Our approach also boosts this by offering movies that share characteristics with other movies that a user has rated highly. It is content based filtering (CBF). After the implementation of method, to improve the performance of content-based prototypes, it is necessary to use some

other ways of modeling the movies and re-estimating the similarities among them with a new similarity function.

Several studies analytically compare the performance of the hybrid with the pure collaborative and content-based methods and prove that the hybrid methods can provide more correct advices than pure approaches. These methods can also be used to accomplish some of the common problems in recommender systems such as cold start and the sparsity problem. If no user preference information is existing to form any basis for suggestions, it is called cold start problem [32]. The sparsity problem happens when existing data are insufficient for determining similar users (neighbors) and it is a main issue that limits the quality of advices and the applicability of collaborative filtering in general [28].

In this thesis, we used collaborative filtering and content based filtering method to recommend items to users by using different datasets like MovieLens dataset and METU Student Elective Course dataset. Firstly, we created movie recommendation system on MovieLens dataset by using Apache Mahout. Very big datasets can be run on Mahout with cloud computing platform. Instead of developing our system from scratch, we prefer to use Apache Mahout. It has the clustering algorithms we are looking for as well as the recommendation algorithms. The results are compared with user-based CF, item-based CF and hybrid method. We evaluate the accuracy of the system by using MAE (Mean Absolute Error). MAE just evaluates the accuracy based on rating values. It is comparing predicted rating and actual rating. So MAE is the average of the absolute errors between the real rating and the prediction. Besides, we used Precision and Recall to evaluate our results. We don't have any test dataset besides MovieLens 1M, so we segregated the test data randomly from original dataset. We implemented k-fold cross validation. Secondly, we used METU Student Elective Course dataset in order to see the accuracy of the algorithms in a practical real-world scenario. By using this dataset, we tried to implement student course recommendation system and validate it especially with students in real world. Calculating accuracy of recommendations was our first aim on this survey. For both of these goals, we used Apache Mahout library.

As a result, to implement the hybrid method, we performed experiments with MovieLens dataset and METU elective course dataset. The implementation is done by using Apache Mahout library which is an innovative and fabulous tool for recommender systems. The results are compared with validation metrics. The rest of the thesis is organized as follows. The background studies in the literature about CF, CBF and other recommender systems which are proposed in recent years are given in Chapter 2. The detailed descriptions and algorithms of recommendation systems are given in Chapter 3. The datasets which we used are explained in Chapter 4. To implement Recommendation Systems, we used Apache Mahout library. It is mentioned on Chapter 5. We explained our proposed system in detail by describing all parameters, datasets and innovations added to the baseline approach in Chapter 6. Lastly, we presented our experimental results and discussed the results which are validated with evaluation metrics as well as explaining the limitations of the study in Chapter 7. In Chapter 8, we summarized the work done and explained contributions of the study and possible improvement points of the proposed system.

CHAPTER 2

RELATED WORK

Data mining allows us to get recommendations or forecasts for future from very huge data. It is a process of getting meaningful information from meaningless raw data. The forecasts which data mining provides are crucial for recommender systems. The exact problem in this field is to get better and more accurate forecasts. Because of the better forecasts, not only the companies can sell more products, but also the customers can be satisfied because they are going to reach the desired products easily. The most common methods in this field are Collaborative Filtering, Content Based Filtering and hybrid approaches.

The first system that implemented the collaborative filtering method was the Tapestry project at Xerox PARC [15] in 1992. The project coined the collaborative filtering term. One of the other early systems is a music recommender named Ringo ([41], [35]) which is proposed in 1994. The other one is a system for rating USENET [30] in the same year. GroupLens is one of the first collaborative filtering recommendation system, which recommends movies [30]. Other examples are Amazon.com that recommends books, and the Jester system that recommends jokes.

If the relatively old studies on recommender systems are investigated, it is discernible that scientists mostly worked on text based domains to implement content-based filtering, collaborative and knowledge-based filtering. Sometimes, effective suggestions are produced on movie ([22]), music ([9], [18]) and web site domains. Besides, in recent years, efforts were made on e-government ([34]), e-learning and e-commerce domains ([40] [36], [1], [10]). Recent studies show that

filtering methods can give more effective results when they are used together. In this thesis, we aimed firstly the minimization of shortages of hybrid approach found in literature ([38], [12], [25], [23], [13]), and secondly the utilization of such an approach for another domain. The main purposes of this thesis are to investigate the new application areas of data mining and to calculate the effectiveness of our hybrid approach.

Data mining is the main part of recommender systems. In old recommender systems, data mining and data processing are utilized together and various methods are developed. In this thesis, data mining methods are studied together to reach at the hybrid approach. Similar studies have been performed since 1997 and have much more interest in recent years. Recommender systems are important not just for individuals but also for companies and governments. Future prediction is benefited in a lot of areas such as finance, shopping, internet, music, cinema, and e-government. In this thesis, one for such domain is selected applying our effective hybrid filtering approach.

Recommender systems are especially developed for e-commerce, e-government and e-learning ([39], [40], [19]) in recent studies. Moreover, a lot of recommender systems for music, cinema, book and entertainment domain (TV program recommendation [44], Flickr group recommendation [43]) are developed. In development of such systems, data mining techniques and algorithms are effectively used. Some of the most effective filtering examples are content-based, collaborative, user-based and data-based. Hybrid recommender system is the combined usage of these filtering methods. In 2013, AR (association rule) and SVD (singular value decomposition) are utilized together as a hybrid solution for the recruitment of the partner filtering recommender system.

In recent years, hybrid recommender systems have gained significant importance. For example, Netflix company organized a contest in 2006. The contest began on October 2, 2006 and continued through at least October 2, 2011. They wanted from contestants to create better movie recommendation by using their dataset. Netflix is already using a world-class movie recommendation system: CinematchSM. Its job is to predict whether someone will enjoy a movie based

on how much they liked or disliked other movies. However the contestants find better recommendations than CinematchSM. The contest by Netflix resulted in a big jump in the research of the recommender systems, more than 40,000 teams were trying to create a good algorithm. In the end, BellKor's Pragmatic Chaos [2][29][20] won the contest because their approach regards *time* unlike other studies. One of the other striking findings belong to Krishnan [21] who compared the recommendation results of machine against humans. According to the study of Krishnan, the machine won in most of the comparisons because machines can handle more data than humans can.

Hybrid approaches are developed for recommender systems used for e-commerce systems and TV programs. There are also some studies for social networks. One of them is about group recommendation system for Flickr. In another study, friend recommendation is performed using LinkedIn profiles. Music recommendation is investigated via auto-tagging and hybrid-matching. Some algorithms are developed for the recommendation used in search engines. Some studies made use of similarity trees such as fuzzy-tree. For example, recommender systems are developed for telecommunication products using this algorithm. Genetic algorithm and Bayes categorization are also utilized in some studies. These studies are beneficial to develop algorithms with scalable, accurate results.

One of the article in literature (Zhao and Shang *et al.* [45]) is about using Hadoop to get better performance and shorter response time in Collaborative-Filtering(CF) algorithm. The similarity measure method is The Pearson correlation coefficient. They run the algorithm on 9 computers which have Hadoop clusters. They do not consider the accuracy, but the running time. They claim that CF in Hadoop platform provides good performance. I come up with an idea to use hybrid recommendation system in Hadoop so it will provide not only better performance, but also better accuracy. This idea is already investigated by a few researchers in recent years. We can still use a different hybrid recommendation on Hadoop platform. Another algorithm which is described in an article (Wang and Zheng *et al.* [42]) is a hybrid recommendation algorithm which is run on Hadoop platform. Firstly, user's possible ratings are predicted by using content-based filtering algorithm. They used a fill sparse matrix in

content-based filtering algorithm, so it can fill the missing data. Then they used MapReduce to calculate the final score to get the list of recommended items. They used eight ordinary PC machines to build Hadoop clusters. Experimental results are given in the article. They claim that this method improves the accuracy of recommendation. Dooms *et al.* [11] focuses on responsiveness and scalability of User-specific online hybrid recommender systems. They have an online system called MovieBrain. They tried to get better results with user control on the recommendations.

In this thesis, some of the studies performed earlier are re-visited. Some of the proven algorithms are preferred to be utilized together. Firstly, Robin Burke's research studies [6] and [7] helped a lot in our work. We decided to use switching hybridization strategy by the help of these papers. Secondly, Manos Papagelis's studies [28] and [27] are so beneficial that we decided to use category boosted method for content-based filtering. Thirdly, Badrul Sarwar's study [31] led us to decide which evaluation metric we were going to use. Therefore many studies in literature helped us to implement and design a hybrid recommendation system.

The mentioned studies above could be found in resources section of the thesis. In this thesis, one of the promising sector "Movie" is selected as a domain to apply algorithms. The other data which we used is the registration records of junior and senior students in Middle East Technical University. We used the given scores of students to implement an elective-course recommendation system.

CHAPTER 3

RECOMMENDATION SYSTEMS

3.1 Recommendation Systems

Recommendation systems aim to improve accuracy of suggestions which users might interest. Recommender systems are based on cognitive science, information retrieval, approximation theory. It emerged as an independent area in mid 1990s with the focus on structures of ratings. In recommendation systems, the main job is to find unseen and unrated items for a user in order to choose the correct items with the highest estimation values. The systems try to estimate ratings by using domain knowledge, similarity algorithms and machine learning approaches. So a recommender system is responsible for predicting the rating or preference that a user would give to an item. When a user creates his or her profile, the system have to get the user preferences in order to provide her interesting recommendations.

Both consumers and sellers can benefit from recommendation systems. Due to the recommendation systems, users can reach the interesting and related items easily. Recommendation systems can be applied to a variety of applications. For example, some of the recommended items are movies, songs, news, books, research articles, social tags, search queries and products in general. In addition to this, there exist recommendation systems for jokes, restaurants, financial services, life insurance, friends (like Facebook or dating websites), and followers (like Twitter). You can see history of recommendation systems and the examples in Chapter 9.

	Movie 1	Movie 2	Movie 3	Movie 4
User 1	4	2	5	-
User 2	-	1	4	-
User 3	-	-	3	5

Figure 3.1: User x Item Matrix

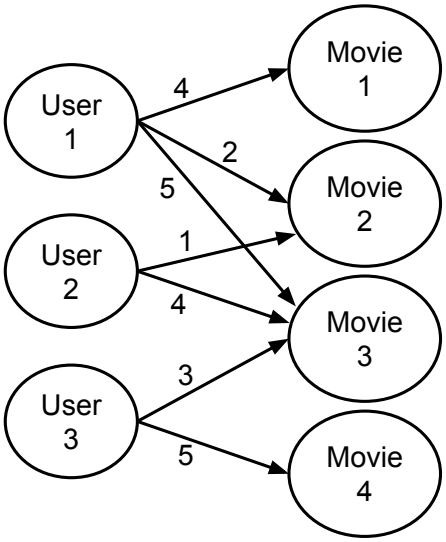

Figure 3.2: User x Item Graph

In most of the recommendation systems, the dataset includes three main elements: *user*, *item* and *rating*. The data can be represented by a matrix. In the matrix, rows point out users, columns point out items and matrix entries are the ratings. These three elements are enough for collaborative filtering so that we can find the users who are similar each other. If a user loves comedy movies and gives higher ratings for comedy movies, it is easy to find another user who also loves comedy movies, because ratings are available. If we want to increase the accuracy and quality of recommendations, we can consider the other parameters like features of items. For example, an item may have various features like price, year, time, location etc. Similarly, a user may have various features like gender, age, country, address etc. Besides the matrix representation, graph visualization of *user x item* matrix is used in some of the algorithms. In graph illustration, users and items are represented with nodes and the weighted edges which are the ratings of users for the items. The examples of matrix and graph representations are given in Figure 3.1 and Figure 3.2 [26].

Three basic approaches are used in literature for recommendation systems. The first of them is collaborative filtering (CF) which uses user similarity for item recommendation. The second one is content based filtering (CBF) which uses item similarity for item recommendation. The last one is hybrid approach which basically combines previous two approaches. The following sections present the recommendation methods. Firstly Collaborative Filtering is discussed, then Content-Based Filtering is given and lastly hybrid approach is mentioned.

3.2 Recommendation Systems with Collaborative Filtering

By using user similarity, we can decide which item to recommend in Collaborative filtering (CF) approach. This approach provides to find similar users in taste. We can recommend items rated by these alike users (neighbors). The underlying opinion of the CF approach is that if two people have same opinion on an argument, they are more likely to have same opinion on a different argument. For example, a CF recommendation system for television tastes could make predictions about which television show a user may like. This approach is based on given past ratings of that user. The algorithm uses *likes* and *dislikes* of users to predict the future behavior. People often get the best recommendation from other people who have similar tastes with themselves. The motivation for CF lies on that idea. CF methods search techniques to associate people on a topic and making suggestion on this topic.

k-Nearest Neighbors(KNN) method is the most widely used CF approach. In this way, firstly, k most similar users are identified. Then, the items are selected if it is marked by these users. The items should not be rated by the user, but should be rated by the neighbors. The rating values for all items are calculated. For this computation, weighted or notweighted average of the neighbors' ratings are calculated. Finally, the system decides on the items which have the highest scores. Besides KNN, there exist other methods used for collaborative filtering (CF):

- Clustering methods [5],

- Bayesian networks [5],
- Artificial Neural Networks [3],
- SVD [3],
- Probabilistic Latent Semantic Analysis (PLSA) [37],
- Latent Drichlet Allocation (LDA) [24].

CF is generally applied on very large data sets. The data sets can be varied in different sectors. For instance, CF can be used in mineral exploration to sense and monitor data. CF can also be used on financial data, such as financial service institutions that integrate many financial sources. Some CF application focus on user data, such as electronic commerce and web applications. This thesis focuses on CF for user data, although some of the methods may apply CF in other major applications as well.

Typically, the workflow of a CF system is:

A user gives ratings for items (e.g. books, movies, songs... etc.) of the system. The system compares this user's ratings with other users' ratings. As a result, the system finds the people with most similar tastes. They are called neighbors. Then the items are selected for recommendation. The recommended items should be the ones which have rated highly by the neighbors, but not yet being rated by this user. This process need some time to calculate prediction, therefore it cannot be run in real time. There should be also enough ratings of the user to run CF algorithm. If there are a lot of past entries of a user, the system can generate more accurate results.

CF systems can be run on two forms: user-based and item-based. Both of them have two steps: (1) find similar users, (2) find recommended item based on similar users. In user-based manner, the algorithm looks for users who share the same rating patterns with the active user. Then it calculates prediction for the active user based on the ratings from those like-minded users found in step 1. This falls under the category of **User-Based CF**. Alternatively, **Item-Based CF** offers the related products. These alike products are determined

by identifying similar users. If some of the users bought both x and y, a user who bought only x can get recommendation of y. So this algorithm proceeds in an item-centric manner. First it builds an item-item matrix determining relationships between pairs of items. Then it infers the tastes of the current user by examining the matrix and matching that user's data.

To sum up, the past ratings of similar users are used in CF in order to get recommended items. CF methods build a matrix of the user preferences for the items. In this matrix, each row represents a user profile, whereas the columns are items. The value Ru_i, i_j is the rating of the user u_i for the item i_j. Figure 3.1 depicts the matrix of user–item ratings. CF can be one of two types:

(1) User-based CF:

1. Find the neighbors whose ratings are similar to the ratings of the selected user U.
2. Calculate a prediction for the selected user U by using the ratings from those like-minded users found in Step 1.

(2) Item-based CF:

1. Build an item-item matrix determining relationships between pairs of items.
2. Calculate a prediction for the selected user U by using this matrix and data on the current user.

The details of both types of CF is given below.

3.2.1 User–based neighborhood

The predicted rating value of item i for the active user u is $P_{u,i}$. It can be calculated as the average of the ratings' values of the users similar to u. The predicted rating score of item i for user u is given in Equation 3.1 [16], where $\bar{R}_u$ is the average rating of user u, and $R_{u,i}$ is the rating of the user u for the

item i.

$$P_{u,i} = \bar{R}_u + \frac{\sum_{v \in Neighbors(u)}^{k} sim(u,v)(R_{v,i} - \bar{R}_v)}{\sum_{v \in Neighbors(u)}^{k} sim(u,v)}. \tag{3.1}$$

This method is also known as user–based CF. However, to predict $P_{u,i}$, the set of users who are similar (e.g. like-minded people) to u ($v \in Neighbors(u)$), the similarity between them ($sim(u,v)$), and the size of this set (k) are needed to be known by the algorithm. This is similar solving the user–profile matching problem. Pearson correlation, cosine similarity, and clustering based on stereotypes are the most common methods to find the neighbors of u.

3.2.2 Item–based neighborhood

Item–based approach benefits from the similarity among the items. This method examines the set of items that have been rated by a user, and calculates the similarity among the target item in order to decide whether recommendation is worthy to the user or not. As a first step in this approach, the similarity between i and j is obtained. In the calculation of this similarity, one could benefit from Cosine Similarity, Pearson Correlation, Adjusted Cosine, or computing the conditional probability, $P(j|i)$. The definition of the cosine similarity is given in Equation 3.2 [16], where U represents the set of users who rated i and j and $R_{u,i}$ symbolizes the rating of user u on item i:

$$sim(i,j) = cos(\vec{i},\vec{j}) = \frac{\vec{i}.\vec{j}}{\|i\| * \|j\|} = \frac{\sum_{u \in U} R_{u,i} R_{u,j}}{\sqrt{\sum_{u \in U} R_{u,i}^2} \sqrt{\sum_{u \in U} R_{u,j}^2}}. \tag{3.2}$$

On the other hand, for the item–based similarity, the differences in rating scale between different users are not taken into account by the Cosine Similarity. It is possible to make the use of user average rating from each co–rated pair, and cope with the limitation of Cosine Similarity with the Adjusted Cosine Similarity, given in Equation 3.3. In Equation 3.3, $\bar{R}_u$ represents the average rating of the

u–th user:

$$sim(i,j) = \frac{\sum_{u \in U}(R_{u,i} - \bar{R}_u)(R_{u,j} - \bar{R}_u)}{\sqrt{\sum_{u \in U}(R_{u,i} - \bar{R}_u)^2}\sqrt{\sum_{u \in U}(R_{u,j} - \bar{R}_u)^2}}. \tag{3.3}$$

Pearson r correlation commonly used for correlation–based similarity. The correlation between two variables shows the degree to which the variables are related. The correlation similarity is defined in Equation 3.4 [16], where $\bar{R}_i$ represents the average rating of the i–th item:

$$sim(i,j) = \frac{Cov(i,j)}{\sigma_i \sigma_j} = \frac{\sum_{u \in U}(R_{u,i} - \bar{R}_i)(R_{u,j} - \bar{R}_j)}{\sqrt{\sum_{u \in U}(R_{u,i} - \bar{R}_i)^2}\sqrt{\sum_{u \in U}(R_{u,j} - \bar{R}_j)^2}}. \tag{3.4}$$

Equation 3.5 [16] defines similarity using conditional probability, $P(j|i)$:

$$sim(i,j) = P(j|i) \simeq \frac{f(i \cap j)}{f(i)}. \tag{3.5}$$

where $f(X)$ represents the number of customers by whom the item set X has been purchased. The only asymmetric metric is this one, so $sim(i,j) \neq sim(j,i)$. After the computation of the similarity among the items, the next step is the prediction of a value for the active item, i, to the target user, u. Usually, how the user rates the similar items of i is captured for this task. The predicted value for item i to user u is given in Equation 3.6 [16], where $S^k(i;u)$ represents the set of k neighbors of item i, that is rated by the user u. The weighted sum of the user's ratings, $\forall j \in S^k(i;u)$ is required for the predicted value.

$$P_{u,i} = \frac{\sum_{j \in S^k(i;u)} sim(i,j) R_{u,j}}{\sum_{j \in S^k(i;u)} sim(i,j)}. \tag{3.6}$$

The Slope One algorithm can be given as an example of item-based CF. It was introduced in 2005. It can be seen as the simplest form of item-based algorithm which is based on ratings.

The original item-based recommendation is totally based on user-item ranking (e.g., a user rated a movie with 3 stars, or a user "likes" a video). When we compute the similarity between items, we are not supposed to know anything other than all users' history of ratings. So the similarity between items is computed based on the ratings instead of the meta data of item content.

Here is an example of Collaborative Filtering. Suppose we have only access to some rating data like Table 3.1.

Table3.1: Example of Collaborative Filtering data

user 1 likes: movie, cooking
user 2 likes: movie, biking, hiking
user 3 likes: biking, cooking
user 4 likes: hiking

Suppose that we want to make recommendations for user 4. First we create an inverted index for items, we will get results as Table 3.2.

Table3.2: Example of Inverted Index For Items

movie: user 1, user 2
cooking: user 1, user 3
biking: user 2, user 3
hiking: user 2, user 4

Since this is a binary rating (like or not), we can use a similarity measure like Jaccard Similarity (Formula 3.7) to compute item similarity.

$$JaccardSimilarity(movie, cooking) = \frac{|user1|}{|user1, 2, 3|} = \frac{1}{3}. \tag{3.7}$$

In the enumerator, user1 is the only user who has both movie and cooking. In the Figure 3.2, the union of movie and cooking has 3 distinct users (user 1, user 2, user 3). In Jaccard Similarity 3.7, $|.|$ here denote the size of the set. So we know the similarity between movie and cooking is $1/3$ in our case. We can just do the same thing for all possible item pairs (i, j).

After similarity computation is done for all pairs, we can continue to find recommendation for a specific user. When we look at the similarity score of $similarity(hiking, x)$ where x is any other tags we might have. If we need to

make a recommendation for user 3, we can aggregate the similarity score from each items in its list. For example,

$$score(movie) = Similarity(biking, movie) + Similarity(cooking, movie)$$

$$score(hiking) = Similarity(biking, hiking) + Similarity(cooking, hiking)$$

There exist some challenges in CF approach. Firstly, if there are not enough ratings of a user (e.g new users), the system can not recommend accurately because finding similar users becomes difficult. This problem is called cold start. **Cold start** problem may occur when the system cannot draw any inferences for users or items about which it has not yet gathered sufficient information. Secondly, similar to new user, the system may not response well when a new item is added into the system. If a new item added to the system, it have to wait to be recommended until it is rated by a few users. Third, some problems can be occurred because of data sparsity during finding neighbors of the user. If number of items is bigger than the number of users in a system, the recommendations can be more different for two neighbors. In the same way, if a user rated items which were not rated by others, it is hard for that user to have accurate recommendations since there is not many neighbors.

3.3 Recommendation Systems with Content Based Filtering

Bu using features of items, Content Based Filtering (CBF) approach can be used for generating recommendations. If we know the content of either user or item, we can calculate content-based similarity. The most of the systems have user-profile and item-profile. For example, a movie can have stars, directors, genres etc. For user profile, we can do the same thing based on the user's likes some movie stars, directors or genres etc. Then the similarity of user and item can be computed using similarity functions (e.g. Cosine Similarity).

With content-based filtering, the recommended movies are going to be more relevant to the user. For instance, if a user rated mostly comedy movies with higher values, the system would recommend comedy movies more than other

types of movies. Features to define items help to calculate similarity between items. For example, in a movie recommendation system, information of genre, year, actors and directors can be beneficial for CBF. CBF approaches are divided into two parts: heuristics based CBF approaches and machine learning based CBF approaches. Heuristic based methods include weights of features, TF-IDF (Term Frequency-Inverse Document Frequency) values etc. Machine learning based methods are composed of Bayesian classifiers, decision trees, clustering etc.

There are a few certain limitations in CBF. Firstly, content-based approaches require features of items. The items do not have features usually. Gathering the features and concluding the recommender job require a significant knowledge engineering effort. Secondly, distinguishing the items is impossible when two items are represented with the same features somehow. Thirdly, the recommended items can become repetitive, because only similarity of items is considered. Although the recommended items should be different on disparate subjects, CBF causes a problem of getting always same recommendation. Lastly, new users have not rated enough number of items, so they may suffer from not getting accurate recommendation. This problem is known as **cold start problem** [14] in the literature. When a new user or new item is introduced to the system, the system knows nothing about them and therefore, it can not recommend anything.

One of the popular examples of a content-based recommender system is Pandora Radio which plays music based on the songs that users listen in the past. There exist other examples of content-based recommendation systems, such as movie recommendation or book recommendation.

The early CBF approaches focused on the text domain. They usually used techniques from Information Retrieval (IR) like extracting meaningful information from the text. On the other hand, the recent studies focus on CBF solutions that cope with more complex domains, such as movie, music, TV program, book. Multimedia domain is so improved in recent years, and there is a lot of available data sets which include item features more than in the past. The scientists im-

proved feature extraction and machine learning algorithms by using these data sets.

CBF methods focus on an objective distance among the items, while CF methods focus on subjective factor. The distance between two items can be computed by a similarity function. Most of the distance metrics deal with numeric attributes, or single feature vectors. Some common distances, given two feature vectors x and y, are: Euclidean (Equation 3.8), Manhattan (Equation 3.9), Chebychev (Equation 3.10), cosine distance for vectors (see previously defined Equation 3.2), and Mahalanobis distance (Equation 3.11) (All formulas are taken from [16]).

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}. \tag{3.8}$$

$$d(x,y) = \sum_{i=1}^{n}|x_i - y_i|. \tag{3.9}$$

$$d(x,y) = max_{i=1..n}|x_i - y_i|. \tag{3.10}$$

$$d(x,y) = \sqrt{(x-y)^T S^{-1}(x-y)}. \tag{3.11}$$

Some of the distance metrics accept that the attributes are orthogonal such as Euclidean (Equation 3.8), Manhattan (Equation 3.9), Chebychev (Equation 3.10) distances. The Mahalanobis distance uses the covariance matrix S. As a result, Mahalanobis is more vigorous to the dependencies among features. A delta function can be used when the attributes are nominal (not numeric). One of simple delta functions is like: $\delta(a,b) = 0 \Leftrightarrow a = b$, and $\delta(a,b) = 1$ otherwise. Then, a distance metric among nominal attributes can be defined as Equation 3.12 [16] (where ω is a reduction factor):

$$d(x,y) = \omega \sum_{i=1}^{n}\delta(x_i, y_i). \tag{3.12}$$

If we want to deal with both numeric and nominal features, then the final distance metric can combine two equations. One of them is Equation 3.12 for

nominal attributes. For numeric attributes we can use one of these equations: 3.8, 3.9, 3.10, 3.11.

Here is a concrete example of CBF. For instance, our user-profile (using binary encoding, 0 means *not-like*, 1 means *like*) is like in Table 3.3, which contains user's preference over 5 movie stars and 5 movie genres:

Table3.3: Example of User Profile

Users	Movie stars 0-4	Movie Genres
user1:	0 0 0 1 1	1 1 1 0 0
user2:	1 1 0 0 0	0 0 0 1 1
user3:	0 0 0 1 1	1 1 1 1 0

Our movie-profile is like in Table 3.4.

Table3.4: Example of Movie Profile

Movies	Movie stars 0 - 4	Movie Genres
movie1:	0 0 0 0 1	1 1 0 0 0
movie2:	1 1 1 0 0	0 0 1 0 1
movie3:	0 0 1 0 1	1 0 1 0 1

To calculate how good a movie is for a user, we use Cosine Similarity (Formula 3.13):

$$CosSimilarity(user, movie) = \frac{DotProduct(user, movie)}{||user|| \cdot ||movie||}. \tag{3.13}$$

$CosSimilarity(user1, movie1) = \frac{0\cdot0+0\cdot0+0\cdot0+1\cdot0+1\cdot1+1\cdot1+1\cdot1+1\cdot0+0\cdot0+0\cdot0}{\sqrt{5}\cdot\sqrt{3}}$
$= \frac{3}{(\sqrt{5}\cdot\sqrt{3})} = 0.77460$

Similarly:

$CosSimilarity(user2, movie2) = \frac{3}{(\sqrt{4}\cdot\sqrt{5})} = 0.67082$

$CosSimilarity(user3, movie3) = \frac{3}{(\sqrt{6}\cdot\sqrt{5})} = 0.54772$

We used content-based filtering after hybrid approach between User-based CF and Item-Based CF. The user's ratings are aggregated based on movie's genre. So each genre has average rating for a user. We add this average genre rating

value to the last predicted rating of each movie based on its genre. For instance, a user gave three ratings for three movies:

- movie1 (Action, Adventure) 3.0
- movie2 (Comedy, Romance) 4.0
- movie3 (Comedy, Adventure) 5.0

So the average value of each genre will be like:

- Action 3.0/1 = 3
- Adventure (3.0+5.0)/2 = 4
- Comedy (4.0+5.0)/2 = 4.5
- Romance 4.0/1 = 4

The recommended movies and their predicted ratings for this user are given:

- movie4 (Comedy, Romance) 4.0
- movie5 (Adventure) 4.0
- movie6 (Thriller) 4.0

Movie4, movie5 and movie6 have same predicted ratings after Collaborative Filtering method. We need to add genre average values so that user's ratings for each genre are going to determine the final recommendation list. After genre-based filtering, the ratings are aggregated with genre's average values.

- movie4 (Comedy, Romance) 4.0 + (4.5 + 4) = 12.5
- movie5 (Adventure) 4.0 + 4 = 8
- movie6 (Thriller) 4.0

In normalization, we are reducing the highest score to 5. Therefore we are multiplying all scores with (5/maxScore). In this example, we multiplied all scores with (5/12.5)=0.4. After normalization:

- movie4 (Comedy, Romance) 5.0
- movie5 (Adventure) 3.2
- movie6 (Thriller) 1.6

As you can see, the movie4 is more suitable to the taste of User1 than movie5 and movie6. It seems that movie6 is not related to the user. The genre-based filtering is a kind of content-based filtering. We can do the same for directors, actors and actresses. However the MovieLens dataset does not have much information about movies.

3.4 Recommendation Systems with Hybrid Approach

Hybrid methods basically combines collaborative filtering (CF) and content based filtering (CBF) approaches to give recommendations. For Hybrid Filtering, we need to use same dataset for both CF and CBF algorithms, so the dataset should be suitable for both of them. Two algorithms have problems when they are used alone. Hybrid approaches not only tries to solve the problems but also tries to increase the accuracy of prediction.

The common problems of recommender systems are explained below.

- **Early rating:** CF method has problem of early rating. This means that first user in the system is going to rate items without receiving any recommendation. So we should decide which items to recommend without looking at the past ratings of this user. CF approach cannot provide recommendations for new users since there are no user ratings. Same problem is valid for new items since there are no user ratings on the item to forecast.
- **Data sparsity:** When there is not much information in dataset, the systems can not calculate correct similarity. Sometimes the system can not

associate the user with other users, so it can not find any recommended item. Besides this, if two users rated same items, both are going to see same recommendation. Therefore it is a hard problem to compute similarity.

- **Cold start for user:** If a new user participates the system, he or she is going to see no recommendation, because there is no given rating for items. If user does not have sufficient number of ratings, he or she can suffer from unrelated recommendation. This may occur when similar users to this user cannot be found. This problem is called Cold Start.
- **Cold start for item:** If an item is introduced to the system, it can not be recommended until it is rated by anybody. Also if an item does not have enough ratings, it can suffer from not being recommended. This problem is called Cold Start too.
- **Attacks:** If there exist attacks to the recommendation system, the system should recognize it and try to avoid it. For instance, some user can copy the other user's profile and can get same recommendations. The system should separate the users who are attacker and the users who are very similar.

CBF approaches solve some of the defects of the CF like early–rater problem. When a new item is added, the similarity can be computed by looking the other items. In CBF, we do not need the ratings of users on an item to recommend it, so any item can be recommended without being rated by users. This is impossible in CF methods. We can say that CBF solves the cold start for item problem, however other problems still exist in pure algorithms. Therefore the proposed solutions in literature try to solve these problems by creating hybrid approaches.

We can say that a recommender system should provide accuracy, coverage, novelty, diversity, stability, resistance to attacks. We can summarize the requirements as follows [26]:

- **Accuracy:** Providing better recommendations and prediction.

- **Coverage:** Capacity of estimating rating of an item.
- **Novelty:** Degree of diversity between recommended and known items.
- **Diversity:** Degree of diversity among recommended items.
- **Stability:** Not firmly changing recommendations in a short period of time.
- **Resistance to attacks:** Not being damaged by attacks

We researched about hybrid recommender systems in this thesis. The most of the studies in literature are mostly about making the algorithm faster by using more clusters on Hadoop which is cloud platform for distributed computing. They focus on scalability and responsiveness of recommender systems. Hadoop and MapReduce can make the algorithm faster and easier. Some of scientists used 8 computers so they had 8 clusters of Hadoop. The recommendation algorithm can response in minutes with 8 clusters. The studies compared the response time of Hadoop with using more and more clusters. However in this thesis, we focused on the accuracy of hybrid approach than the scalability and response time.

In order to do hybrid filtering, CF and CBF algorithms can be implemented individually and displayed separately. In a second option, both scores of CF and CBF algorithms can be multiplied the ranking scores in order to merge them into a single recommendation set. In literature, there exist seven hybridization techniques that are explained briefly below [7]:

- **Weighted:** The score of both recommendation components are combined numerically.
- **Switching:** The system selects one of the recommendation components and applies the selected one.
- **Mixed:** Recommendations from different recommenders are presented in a combined list.
- **Feature Combination:** Features of different knowledge sources are combined together and given to a single recommendation algorithm.

- **Feature Augmentation:** A set of features can be calculated by one technique, and then they can be the input to the next technique.
- **Cascade:** Recommenders are given strict priority, with the lower priority ones breaking ties in the scoring of the higher ones.
- **Meta-level:** A model can be produced by one technique, which is then the input used by the next technique.

First of all, we used weighted approach. Secondly we used switching approach in this thesis. We agreed on using switching method at the end. Our hybrid solution works like the following:

First, the result of user-based algorithm is called s1 while the result of item-based is called s2. There are 100 movies in each result list so that the most of the movies have both s1 and s2 predicted rating values.

If a movie is not in list of 100 results of user-based, then it's s1 value equals to zero. In the same way, if a movie does not exist in 100 results of item-based, then s2 value equals to zero. Consequently, in weighted method, the s1 and s2 scores are going to be multiplied with weights and are aggregated in a single value. In switching method if a movie has both s1 and s2 results, the system is going to select the one which is higher precision.

In weighted method, we calculated the total weighted score of all movies in result sets. The total score is calculated as (w1*s1)+(w2*s2). In the formula, w1 and w2 are weights in decimal number. The first 100 movies of the total result are regarded.

After trying the different numbers for w1 and w2, we can look at precision and recall. In simple terms, high precision means that an algorithm returned substantially more relevant items than irrelevant items, while high recall means that an algorithm returned most of the relevant results.

In switching method, we chose recommended items by looking the precision and recall of the randomly chosen 1000 users in MovieLens 1M data (There are 6040 users and 3952 different movies in total).

We run the algorithms separately with weighted and switching hybridization techniques. We tried the pure collaborative filtering methods (user-based CF and item-based CF) alone and get the precision values. After that we chose the hybridization technique. Switching method gave better results than the weighted method. Therefore we focus on it in later studies. In switching hybridization, CF method with the highest precision value is selected. Then, at the last step, we applied content-based filtering. The implementation details of our hybrid approach is given in Chapter 6. The pseudo code of our final hybrid approach which includes content-based filtering and switching hybridization between user-based CF and item-based CF is given in Algorithm 1, Algorithm 2, Algorithm 3, Algorithm 4, Algorithm 5.

After getting results, we needed to evaluate the results. The recommendation systems need evaluation metrics to calculate the error rate. There are statistical methods to measure error rate:

- Root Mean Squared Error (RMSE)
- Mean Absolute Error (MAE)

In statistics, the mean absolute error (MAE) is a quantity used to measure how close predictions are to the eventual outcomes. The root-mean-square deviation (RMSD) or root-mean-square error (RMSE) is a frequently used measure of the differences between values (sample and population values) predicted by a model or an estimator and the values actually observed. In this thesis, we calculated MAE and compared it other methods with k-fold cross validation technique. The details of validation metrics are given in Chapter 7.

Algorithm 1 Pseudo Code of Hybrid Recommendation System

```
1: procedure HYBRID–RECOMMENDER
2:     movieData ← movies.csv
3:     for Fold k=1 to 5 do
4:         model ← trainingdata_k.csv
5:         testModel ← testdata_k.csv
6:         for each user u in testModel do
7:             result_UB ← UserBasedCF(model, u.ID)
8:             CalcValidationMetrics(result_UB, testModel)
9:             result_IB ← ItemBasedCF(model, u.ID)
10:            Normalize(result_IB)
11:            CalcValidationMetrics(result_IB, testModel)
12:            result_UB−IB ← merge(result_UB, result_IB)
13:            for each movie m in result_UB−IB do
14:                if precision_IB > precision_UB then
15:                    resultCf_HYBRID ← result_IB
16:                else
17:                    resultCf_HYBRID ← result_UB
18:                end if
19:            end for
20:            Sort(resultCf_HYBRID)
21:            result_HYBRID ← ContentBasedFiltering(resultCf_HYBRID)
22:            Normalize(result_HYBRID)
23:            Sort(result_HYBRID)
24:            CalcValidationMetrics(result_HYBRID, testModel)
25:            precision_k ← add precision_u to end of file.
26:            recall_k ← add recall_u to end of file.
27:            fmeasure_k ← add fmeasure_u to end of file.
28:            mae_k ← add mae_u to end of file.
29:        end for
30:    end for
31: end procedure
```

Algorithm 2 Pseudo Code of User Based CF

1: **procedure** USER–BASED CF
2: $userSimilarity \leftarrow$ PearsonCorrelationSimilarity($model$)
3: $neighborhood \leftarrow$ NearestNUserNeighborhood(numberOfNeighbors, $userSimilarity$, $model$)
4: $recommender \leftarrow$ GenericUserBasedRecommender($model$, $neighborhood$, $userSimilarity$)
5: $cachingRecommender \leftarrow$ CachingRecommender($recommender$)
6: $recommendations \leftarrow cachingRecommender$.recommend($userId$, numberOfRecommendations)
7: **return** $recommendations$
8: **end procedure**

Algorithm 3 Pseudo Code of Item Based CF

1: **procedure** ITEM–BASED CF
2: $itemSimilarity \leftarrow$ TanimotoCoefficientSimilarity($model$)
3: $recommender \leftarrow$ GenericBooleanPrefItemBasedRecommender($model$, $itemSimilarity$)
4: $cachingRecommender \leftarrow$ CachingRecommender($recommender$)
5: $recommendations \leftarrow cachingRecommender$.recommend($userId$, numberOfRecommendations);
6: **return** $recommendations$
7: **end procedure**

Algorithm 4 Pseudo Code of Content Based Filtering

```
1: procedure CONTENT–BASED FILTERING
2:     for each genre g in model do
3:         Avg_g ← Calculate average rating of user
4:     end for
5:     for each movie m in resultCfHybrid do
6:         for each genre g which m has do
7:             rating_m ← rating_m + Avg_g
8:         end for
9:     end forreturn resultCfHybrid
10: end procedure
```

Algorithm 5 Pseudo Code of Validation Metrics

```
1: procedure VALIDATION–METRICS
2:     precision ← number of found in test / number of recommendation size
3:     recall ← number of found in test / number of test size
4:     F − measure ← (2 * p * r) / (p + r)
5:     MAE ← (Total differences between predicted and actual rating) / number of common items in result and test data.
6: end procedure
```

CHAPTER 4

DATASETS

In this chapter, the datasets which are used for experiments to implement hybrid recommendation system are discussed and given in detail.

4.1 MovieLens Dataset

The first dataset we chose is **MovieLens**. This dataset contains 1,000,209 anonymous ratings of 3,952 movies made by 6,040 MovieLens users who joined MovieLens in 2000. The correctness of data is not guaranteed but this data is very suitable for movie recommendation systems. Research usage of MovieLens Dataset is allowed under some conditions.

MovieLens data can be used in many research related to information filtering, collaborative filtering, and recommender systems. This dataset helped a lot to improve collaborative filtering and content-based methods in history. It is such an old dataset because it includes movies which are released in 2000. The dataset summary is given in Figure 4.1. All ratings are contained in the file "ratings.dat" and are in the following format:

UserID::MovieID::Rating::Timestamp

- UserIDs range between 1 and 6040
- MovieIDs range between 1 and 3952
- Ratings are made on a 5-star scale (whole-star ratings only)

```
     user          movie          rating          time
Min.   :   1   Min.   :   1   Min.   :1.000   Min.   :9.567e+08
1st Qu.:1506   1st Qu.:1030   1st Qu.:3.000   1st Qu.:9.653e+08
Median :3070   Median :1835   Median :4.000   Median :9.730e+08
Mean   :3025   Mean   :1866   Mean   :3.582   Mean   :9.722e+08
3rd Qu.:4476   3rd Qu.:2770   3rd Qu.:4.000   3rd Qu.:9.752e+08
Max.   :6040   Max.   :3952   Max.   :5.000   Max.   :1.046e+09
```

Figure 4.1: Summary of MovieLens 1M data

- Timestamp is represented in seconds since the epoch as returned by time(2)
- Each user has at least 20 ratings

User information which in the file "users.dat" is in the following format:

UserID::Gender::Age::Occupation::Zip-code

All demographic information is provided voluntarily by the users and is not checked for accuracy. Only users who have provided some demographic information are included in this data set.

- Gender is denoted by a "M" for male and "F" for female
- Age is chosen from the following ranges:

  1: "Under 18" 18: "18-24" 25: "25-34"

  35: "35-44" 45: "45-49" 50: "50-55"

  56: "56+"
- Occupation is chosen from the following choices:

  0: "other" or not specified 1: "academic/educator"

  2: "artist" 3: "clerical/admin"

  4: "college/grad student" 5: "customer service"

  6: "doctor/health care" 7: "executive/managerial"

  8: "farmer" 9: "homemaker"

  10: "K-12 student" 11: "lawyer"

  12: "programmer" 13: "retired"

  14: "sales/marketing" 15: "scientist"

  16: "self-employed" 17: "technician/engineer"

  18: "tradesman/craftsman" 19: "unemployed"

  20: "writer"

Movie information is in the file "movies.dat" and is in the following format:

MovieID::Title::Genres

- Titles are identical to titles provided by the IMDB (including year of release)
- Genres are pipe-separated and are selected from the following genres:

Action	Adventure	Animation
Children's	Comedy	Crime
Documentary	Drama	Fantasy
Film-Noir	Horror	Musical
Mystery	Romance	Sci-Fi
Thriller	War	Western

- Some MovieIDs do not correspond to a movie due to accidental duplicate entries and/or test entries
- Movies are mostly entered by hand, so errors and inconsistencies may exist

This dataset is separated into two parts like Training data and Test data. One part includes some of ratings of from randomly chosen 1000 users. This part is used as test data. The other part includes all ratings except the ones which are taken for test data. So this part is used as training data. The ratings for the test data are chosen randomly from the users. So, both users and ratings are randomly chosen for test set.

4.2 METU Student Elective Course Dataset

The third dataset which we use in this thesis is **METU Student Elective Course dataset**. This dataset is not open for everyone and it can be used with permission of Computer Engineering Department of Middle East Technical University. In this dataset, there are 1056 ratings of 300 students which they have given at the beginning of 3 semesters for 20 courses. We can think of it as a matrix. There are students as rows and the elective courses as columns in

a matrix. The intersection of rows and columns shows scores. A student can give at most 100 score in total if we sum up all scores which he or she has given for one semester. For example, if a student can give 90 points for an elective course, then he or she can give at most 10 points for another course. He or she cannot give more than 10 points anymore. This method is done for placing or distributing the students among all elective lessons by their scores. To sum up, students are selected for the elective courses by the scores which they gave.

The dataset summary is given in Figure 4.2. A snippet of semester1.csv is given in Table 4.1.

Table4.1: Unnormalized and normalized data example for METU Elective Course Data Set

Unnormalized Dataset	**Normalized Dataset**
student,lecture,score,semester	student,lecture,score,semester
1,476,1,1	1,476,1.01,1
1,498,99,1	1,498,100.0,1
2,498,99,3	2,498,100.0,3
3,469,99,2	3,469,100.0,2
4,424,1,1	4,424,1.42,1
4,443,70,1	4,443,100.0,1
4,465,4,1	4,465,5.71,1
4,495,25,1	4,495,35.71,1
4,352,50,3	4,352,100.0,3
4,498,50,3	4,498,100.0,3

This dataset has real world content and it consists of 3 semesters. The students' identity is not given in dataset because their names replaced with anonymous ids like 1, 2, 3, etc. However the lecture names are given correctly like 476, 498, etc. We used this dataset to implement course recommendation system with our hybrid method. The validation of recommended items were analyzed with a survey among students in METU Computer Engineering department.

There were some problems in dataset, so we solved them with normalization before processing. First of all, the system allows students to give scores dependent on how many lectures they are allowed to take for one semester. Some of students are allowed to take just two elective courses for a semester while some

of them are not. So students can be selected for the lecture of either the higher score or the other. When the students are restricted to select two courses, they are dividing the 100 points into two. On the other hand, students who are restricted to select three courses are dividing the 100 points into three. This causes a problem of finding similar scores among students because they are dividing scores based on their number of elective course limit. When you look at the example snippet of dataset, S1 and S4 gave whole 100 points for one lecture. S2 separated the scores into three and S3 separated the scores into two. So while S2 gave 35 points for Course 5, S3 gave 95 points for the same course. However both S2 and S3 wanted to enroll Course 5 which is their most desired course for both of them. So 35 and 95 scores should point out the similarity for S2 and S3. Therefore we needed normalization.

Secondly, The system always enables students to enter score for one more lesson based on their balance. For example, if a student is allowed to enroll one course, the online system allows to enter scores for two courses. As a result, this student can be elected for one of these courses. However some of students gave 100 points for just one lecture so they do not distribute the scores. In the example above, S1 and S4 gave 100 points for one lecture. This cause problem of finding recommendations. They gave entire score to a course and the system automatically elected them because they are the ones who gave higher scores. Lets say another student gave 99 points for this lecture, this means this student want to enroll this course as much as them. Thus we subtracted 1 from 100, so we assume the students gave 99 points for the course. This is one of the parts of normalization.

After the normalization the biggest score of a user became 100 and other scores are multiplied with (100 / biggestScore) to normalize. So if a user wants to enroll a course, we can identify them by their scores. Normalization and exporting data is done by Java programming and the normalization code is given at Appendix A.

The courses have genre types. There are 9 genre types. Genres are pipe-separated and are selected from the following genres:

```
    student          lecture          score            semester
Min.   :  1.0   Min.   :316.0   Min.   :  1.010   Min.   :1.0
1st Qu.: 75.0   1st Qu.:443.0   1st Qu.:  1.485   1st Qu.:1.0
Median :137.0   Median :462.0   Median : 83.330   Median :2.0
Mean   :138.5   Mean   :449.8   Mean   : 56.946   Mean   :1.9
3rd Qu.:202.0   3rd Qu.:478.0   3rd Qu.:100.000   3rd Qu.:3.0
Max.   :300.0   Max.   :498.0   Max.   :100.000   Max.   :3.0
```

Figure 4.2: Summary of Metu Elective Course data

Computational	Algorithm	Database
Theory	Automata	Programming
Software	Computer Systems	Vision

CHAPTER 5

RECOMMENDATION SYSTEM WITH APACHE MAHOUT

In this chapter, the implementation details of hybrid recommendation system with Apache Mahout are discussed and given in detail.

5.1 Recommendation System Implementation with Apache Mahout

Recommender systems provide reaching more products, knowledge and similar people easily for users. A recommender system can predict the taste of a user based on his or her past, item similarities and the associations with other people. The recommendation systems become popular over the past 20 years. In this period, a rich collection of tools that enable to implement recommendation systems are emerged. There are a lot of recommender tools in order to use and evaluate for the recommendation algorithms. One of the most significant of those tools is Apache Mahout.

Apache Mahout is one of the machine learning libraries. It is scalable so that it can handle very large data sets. Apache Mahout provides parallelizable running of algorithms with MapReduce paradigm on Hadoop distributed computing platform. People involved in the Apache Lucene project started the Mahout project to deal with machine learning algorithms for clustering and categorization. Apache Mahout is tried to get example recommendations. It is an official Apache project and freely distributed on internet thus available from any of the Apache mirrors.

Apache Mahout can deal with the following tasks:

1. **Recommendation systems:** Mahout administers wide range of CF algorithms. Recommendation Job workflow is given in Figure 5.1 which is taken from [17].

2. **Clustering:** Mahout can compute TF-IDF and can cluster text documents so that we can find related texts.

3. **Classification:** Mahout implements learning methods and can classify documents. It can find the correct category for unlabeled documents.

4. **Frequent item-set mining:** Mahout can take a group of items and detects which individual items usually appear together.

Mahout includes recommendation algorithms such as slope one, user based, item based and is incredibly easy to extend. It also has some pretty useful clustering algorithms which support dimension reduction features. This is useful for scientists in case their matrix is sparse (that is, a lot of tags that have very few usage stats). Also Mahout supports Lucene which has frequency–inverse document frequency (TF-IDF) features to cluster tags and documents. In addition to this, Mahout works in a collaboration with Solr. Both are Apache projects. Solr is an indexing tool built on Apache Lucene. Solr is highly reliable, scalable and fault tolerant, providing distributed indexing, replication and load-balanced querying, automated failover and recovery, centralized configuration and more.

In this thesis, Apache Mahout used for implementation of CF recommendation on MovieLens dataset and METU Elective Course Dataset. Recommender of Mahout processes a data set in a text file which each line includes user id, item id and rating value. The ids and rating value have to be separated with comma as described in Chapter 4. Therefore every line should have the format "userID,itemID,ratingValue". The rating value can be in different range in different applications. For a movie recommendation system, the rating values are varied from 0 to 5 as an integer number.

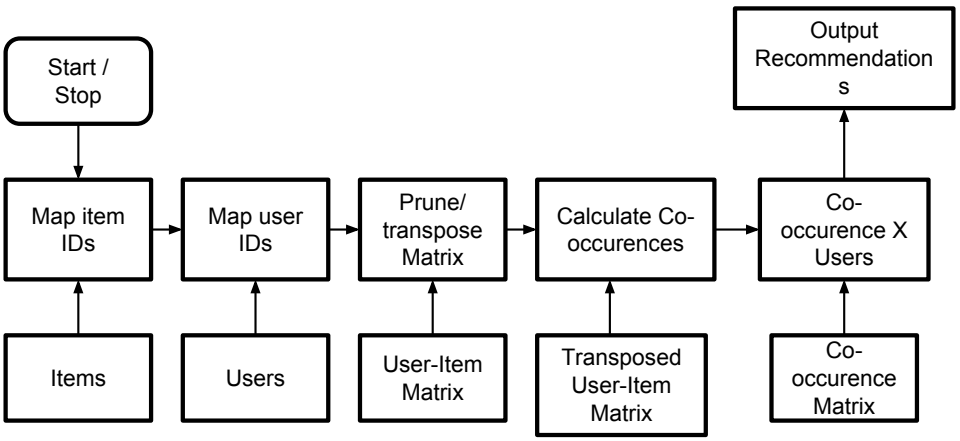

Figure 5.1: Recommender Job workflow of Apache Mahout

5.2 User Based Collaborative Filtering Implementation with Apache Mahout

In this thesis, firstly, we created a user-based recommender on Apache Mahout. The idea behind this approach is described in Chapter 3. To calculate the similarity of users, we have to compare their interactions. There are several methods for doing this. One popular method is to compute the correlation coefficient between their interactions. In Mahout, we used Pearson Correlation Similarity. The codes below runs Pearson Correlation Similarity (PCS) algorithm to implement user similarity. After that, we had to define which similar users we wanted to leverage for the recommender. This example program (Table 5.1) recommends 10 movies for user 1. Nearest N-User Neighborhood algorithm is used to find 3 similar users. The short version of our Java code which was used in this study is given at Appendix B.

Table5.1: Java Code example of User-based CF on Apache Mahout

```
1 UserSimilarity userSimilarity = new PearsonCorrelationSimilarity(
      model);
2 UserNeighborhood neighborhood = new NearestNUserNeighborhood(3,
      userSimilarity, model);
3 Recommender recommender = new GenericUserBasedRecommender(model,
      neighborhood, userSimilarity);
4 Recommender cachingRecommender = new CachingRecommender(recommender
      );
5 List<RecommendedItem> recommendations = cachingRecommender.
      recommend(1, 10);
```

PCS is a well known algorithm to find similarity of nodes. This metric measures how highly correlated are two variables. PCS is measured from -1 to +1 unlike the Euclidean Distance similarity score which is scaled from 0 to 1. A Pearson Correlation Coefficient of 1 means that the data objects are perfectly correlated while a score of -1 indicates that the data objects are not correlated at all. In essence, the Pearson Correlation score finds the ratio between the covariance and the standard deviation of both objects. In the mathematical form, the score can be described as formula 5.1 [33].

$$Pearson(x,y) = \frac{\sum xy - \frac{\sum x \sum y}{N}}{\sqrt{(\sum x^2 - \frac{(\sum x)^2}{N})(\sum y^2 - \frac{(\sum y)^2}{N})}}. \tag{5.1}$$

K-nearest neighbor (KNN) is an algorithm to find nearest items among the whole collection, so it can be used in collaborative filtering as well. The most important idea lies in a term "similarity". To recommend something to the user in question, you find people from his neighborhood that have similar profile.

5.3 Item Based Collaborative Filtering Implementation with Apache Mahout

There is a pretty standard Mahout item-based recommender for movies using MovieLens data. We used **Tanimoto Coefficient Similarity** and **Generic Boolean Preference Item Based Recommender** to calculate item-based recommendations. The item based filtering can be done on mostly boolean data sets. If input data set does not have a preference value, we can call it like boolean data set. For example, input data set would be like the following format in Table 5.2.

Table5.2: Boolean data set format

UserId1,ItemId1
UserId2,ItemId2

Here it can be based on some data where a user either likes an item or he or she does not. There is no preference value associated with this. For similarity algo-

rithm, we can either go in for Tanimoto Coefficient Similarity or Log Likelihood Similarity. For recommender, we need to use Generic Boolean Pref User Based Recommender or Generic Boolean Pref Item Based Recommender. Mahout provides a number of item-based implementations – a generic recommender, Boolean preferences recommender, Slope one, SVD and KNN. To implement an item based recommendation, we chose Tanimoto Coefficient Similarity and Generic Boolean Preference Item Based Recommender (Table 5.3).

Table5.3: Java Code example of Item-based CF on Apache Mahout

```
1 ItemSimilarity itemSimilarity = new TanimotoCoefficientSimilarity(
      model);
2 ItemBasedRecommender recommender = new
      GenericBooleanPrefItemBasedRecommender(model, itemSimilarity);
```

We are experimenting with injecting content-based knowledge into the recommender, so that we can most highly recommend movies that are not only similar in the normal collaborative filtering sense, but also similar in the sense that they share many common terms.

The movies content similarities can be computed by cosine similarity of TF-IDF vectors and they are precomputed using a Mahout batch and read from a dataset file. This is not possible for Movielens Dataset because features of movies are limited with their title, year and genre. We have just these three information on MovieLens dataset, so we used just item-based filtering. The distance measure which name is **Tanimoto Coefficient Similarity**. The Tanimoto coefficient between two points, a and b, with k dimensions is calculated as in the Equation 5.2. The Tanimoto similarity is only applicable for a binary variable, and for binary variables the Tanimoto coefficient ranges from 0 to +1 (where +1 is the highest similarity).

$$TanimotoCoefficient = \frac{\sum_{j=1}^{k}(a_j \times b_j)}{\sum_{j=1}^{k} a_j^2 + \sum_{j=1}^{k} b_j^2 - \sum_{j=1}^{k}(a_j \times b_j)}. \tag{5.2}$$

5.4 Hybrid Approach Implementation on Apache Mahout

We have a few options for hybridization (explained in 3.4). One of them is using weights. We can use weights in linearly merging two result set. Therefore, the new result set should cover the test data at most. The second option is switching approach. We can choose one of the algorithm which covers the test data at most.

"A switching hybrid builds in item-level sensitivity to the hybridization strategy: the system uses some criterion to switch between recommendation techniques. Tran and Cohen (1999) proposed a more straightforward switching hybrid. In their system, the agreement between a user's past ratings and the recommendations of each technique are used to select the technique to employ for the next recommendation." *Robin Burke, 2002* [6]

We are trying to get maximize not only the number of coincidences but also the ratings of test data. We have to consider the rating of the user for a specific movie genre in training data before we recommend it. So we can get the average of the ratings of the user for the all watched movie genres which exist on training dataset. So we added content based filtering at the end of the process.

Hybrid approach and content-based filtering are not supported by Apache Mahout. We created our hybrid approach and content-based filtering without using Apache Mahout library. The details are given in Chapter 6.

CHAPTER 6

EXPERIMENTS

In this chapter, the experiments to implement hybrid recommendation system are discussed and given in detail. For experiments, we used Apache Mahout library for collaborative filtering. The algorithms are mentioned in Chapter 3 and the details of Apache Mahout is given in Chapter 5. In this chapter, we gave only the details of experiments. We created and implemented hybrid recommendation system. Firstly, we did experiments with User-based CF. We used Pearson Correlation Similarity and Nearest N-User Algorithm for User-based CF. Secondly, we did experiments with Item-based CF. We used Tanimoto Coefficient Similarity and Generic Boolean Preference algorithms for Item-based CF. Thirdly, we blended item-based and user-based methods of CF with weighted and switching hybridization approaches. Lastly, we added genre-based average ratings as content-based filtering so that the final recommendation list became more relevant to user. We used 2 different datasets which are discussed in Chapter 4. The proposed hybrid algorithm is tested on MovieLens dataset and METU Elective Course dataset. The results are validated with k-fold cross validation which is given in Chapter 7.

6.1 Experiments on MovieLens Dataset

In this section, each CF and CBF methods are explored by performing experiments in practice on MovieLens Dataset.

6.1.1 Experiments with Collaborative Filtering on MovieLens Dataset

We tried out two types of CF: User-based CF and Item-based CF. Firstly, we did experiments with User-based CF which uses Pearson Correlation Similarity and Nearest N-User Algorithm. Secondly, we did experiments with Item-based CF which uses Tanimoto Coefficient Similarity and Generic Boolean Preference algorithms.

6.1.1.1 Experiments with User Based Filtering on MovieLens Dataset

First of all, we wanted to make recommendation for User 1 on MovieLens. we looked at his MovieLens profile and then at profiles of other users in MovieLens. We found 10 people with similar profiles and checked what they like. If 8 of 10 people with similar profiles like new film, most probably User 1 will like it too.

The movies which *User 1* watched and scored are given in Appendix C. According the movies which User 1 watched, we can group the movies in their genre so that we can find the average score of the user for each genre 6.1.

Table6.1: Average score of the user1 for each genre

Genre	**How many movies?**	**Average Score**
Drama	21	4.42
Children's	20	4.25
Animation	18	4.11
Musical	14	4.28
Comedy	14	4.14
Romance	6	3.66
Action	5	4.20
Adventure	5	4.00
Fantasy	3	4.00
Thriller	3	3.66
Sci-Fi	3	4.33
Crime	2	4.00
War	2	5.00

So we can say that this user likes drama and animation more than other type of movies. It is obvious that this user is going to get recommendations mostly on these topics: drama, children's, animation, musical and comedy. The similar

users should be selected from the ones who gave better points for the drama and animation movies. In first configuration, we found 3 neighbors of User 1 based on nearest neighbor algorithm. The recommended movies are limited to 10. The recommended movies for this user on Mahout with **Nearest N-User Neighborhood** is in the following Table 6.2.

Table6.2: The result of Nearest 3-User Neighborhood

1 :	item:2763 value:5.0	Thomas Crown Affair, The (1999) Action|Thriller
2 :	item:2571 value:5.0	Matrix, The (1999) Action|Sci-Fi|Thriller
3 :	item:110 value:4.5	Braveheart (1995) Action|Drama|War
4 :	item:480 value:4.5	Jurassic Park (1993) Action|Adventure|Sci-Fi
5 :	item:551 value:3.5	Nightmare Bef(1993) Children's|Comedy|Musical
6 :	item:2581 value:3.5	Never Been Kissed (1999) Comedy|Romance
7 :	item:2701 value:3.3	Wild Wild West (1999) Action|Sci-Fi|Western
8 :	item:2723 value:2.6	Mystery Men (1999) Action|Adventure|Comedy
9 :	item:2683 value:2.5	Austin Powers: The Spy Who (1999) Comedy
10 :	item:1513 value:2.5	Romy and Michele's High Sc (1997) Comedy

The similarity values of Pearson Correlation plus Nearest N-User Algorithm generate predicted ratings which range from 0 to 5 for MovieLens dataset. As you can see, the similar users to User 1 watched *Thomas Crown Affair* and *Matrix* and they gave higher scores for these movies, so the predicted ratings for these movies are 5. The similar users also like action and thriller movies. The

other recommended movies are in Drama, Children's and Comedy which are suitable for User 1. We can try the same algorithm with different configurations like more number of neighbors.

If we expand our neighbors from 3 to 10, then **Nearest N-User Neighborhood** recommendation gives the following result Table 6.3.

Table6.3: The result of Nearest 10-User Neighborhood

0 :	item:858 value:5.0	Godfather, The (1972) Action|Crime|Drama
1 :	item:2858 value:5.0	American Beauty (1999) Comedy|Drama
2 :	item:1617 value:5.0	L.A. Confi. (1997) Crime|Film-Noir|Mystery|Thriller
3 :	item:587 value:5.0	Ghost (1990) Comedy|Romance|Thriller
4 :	item:2763 value:5.0	Thomas Crown Affair, The (1999) Action|Thriller
5 :	item:3578 value:5.0	Gladiator (2000) Action|Drama
6 :	item:1517 value:5.0	Austin Powers: International Man (1997) Comedy
7 :	item:141 value:5.0	Birdcage, The (1996) Comedy
8 :	item:1221 value:5.0	Godfather: Part II, The (1974) Action|Crime|Drama
9 :	item:457 value:5.0	Fugitive, The (1993) Action|Thriller

It can be seen from the table above that the results are getting better and the similarity results are all 5. There is more Drama movies in the recommended movies now. The algorithm found 10 similar users to User 1 and naturally it found more recommendation. If we expand the neighbor size more and more, the accuracy of predicted movies is going to lower again until it reaches the

optimal size of neighbors. In literature, some of studies shows that 30 neighbors is enough for kNN, so 30 neighbors are acceptable most of the time [31]. In our hybrid study, we used 30 neighbors for MovieLens 1M dataset.

If we use **Threshold User Neighborhood** rather than Nearest N-User Neighborhood, we can get different results. The similarity function is the same as **Pearson Correlation**. When the threshold is 0.1, the result of **Threshold User Neighborhood** is like the following Table 6.4.

Table6.4: The result of Threshold User Neighborhood

0 :	item:3245 value:5.0	I Am Cuba (Soy Cuba/Ya Kuba) (1964) Drama
1 :	item:2503 value:5.0	Apple, The (Sib) (1998) Drama
2 :	item:1420 value:5.0	Message to Love: The Isle (1996) Documentary
3 :	item:559 value:5.0	Paris, France (1993) Comedy
4 :	item:1002 value:5.0	Ed's Next Move (1996) Comedy
5 :	item:649 value:5.0	Cold Fever (1994) Comedy|Drama
6 :	item:2197 value:5.0	Firelight (1997) Drama
7 :	item:669 value:4.8	Aparajito (1956) Drama
8 :	item:1872 value:4.7	Go Now (1995) Drama
9 :	item:2129 value:4.6	Saltmen of Tibet, The (1997) Documentary

The Threshold Neighborhood gave totally different results than other Neighborhood algorithm. The genres of the recommended items are similar like Drama and Comedy. To sum up, we used Collaborative Filtering to find and recommend

movies to User 1. However the recommendations are not good enough for this user because he loves drama and children's genre more than action and thriller. The recommended movies are chosen by looking other people who watched the same movies and gave similar points. We did not look at the types of movies, and this may change the recommendations a lot. Therefore we need content based filtering too.

6.1.1.2 Experiments with Item Based Filtering on MovieLens Dataset

The 10 results of item-based recommendation in Mahout on MovieLens 1M data set for User 1 are given in Table 6.5.

Table6.5: The 10 results of item-based recommendation in Mahout on MovieLens 1M data for User 1

0 :	item:364 value: 12.99	Lion King (1994) Animation|Children's|Musical
1 :	item:1196 value: 12.38	Star Wars: Episode V (1980) Action|Adventure| Drama|Sci-Fi|War
2 :	item:1265 value: 12.37	Groundhog Day (1993) Comedy|Romance
3 :	item:2174 value: 12.31	Beetlejuice (1988) Comedy|Fantasy
4 :	item:2081 value: 12.28	Little Mermaid, The (1989) Animation| Children's|Comedy|Musical|Romance
5 :	item:2716 value: 12.08	Ghostbusters (1984) Comedy|Horror
6 :	[item:1073 value: 12.04	Willy Wonka and the Choco (1971) Adventure| Children's|Comedy|Fantasy
7 :	item:1198 value: 12.03	Raiders of the Lost Ark (1981) Action|Adventure
8 :	item:318 value: 12.03	Shawshank Redemption, The (1994) Drama
9 :	item:1307	When Harry Met Sally... (1989) Comedy|Romance

Table 6.5 Item-based result (Continued)

	value: 11.99	

The result of Mahout with **Tanimoto Coefficient Similarity** gave some results with higher similarity values. The predicted values of ratings with Tanimoto and Generic Boolean Pref Item Based Recommender range from 0 to 13 for MovieLens dataset for User 1. If we look at the genres of the recommended movies, they are mostly from type of Comedy, Animation and Children's. So the recommendations are suitable with the User 1's profile. How we can combine these informations is given in the next section in hybrid approach.

6.1.2 Hybrid Approach Implementation on MovieLens Dataset

In previous sections, the experiments of content based and collaborative filtering algorithms are discussed. So, for a given pair of movies, we have:

- The user similarity (Pearson Correlation) 0 <= s1 <= 5
- The content similarity (Tanimoto) 0 <= s2 <= 13

In our initial run, we run the same algorithms to find 100 recommended movies. So the common recommended movies are investigated. The Nearest N-User Algorithm is run to find 50 nearest user in between 6040 users in MovieLens 1M dataset. There are 3952 different movies in dataset. Therefore 100 movies are selected from these criteria. As a result, there are 20 common recommended movies which are given in Table 6.6.

Table6.6: Common recommended movies and their scores from User-based (s1) and Item-based (s2)

Common Movies	**s1**	**s2**
Honey, I Shrunk the Kids (1989)	5.0	10.54
Witness (1985)	5.0	10.28
Few Good Men, A (1992)	5.0	10.64

Table 6.6 Common movies (Continued)

Common Movies	s1	s2
Good Will Hunting (1997)	5.0	11.38
Matrix (1999)	4.81	11.67
Aliens (1986)	4.75	10.91
Godfather (1972)	4.72	10.66
Casablanca (1942)	4.71	10.17
Forrest Gump (1994)	4.6	11.77
Fugitive (1993)	4.6	11.63
Braveheart (1995)	4.54	10.83
American Beauty (1999)	4.52	10.55
Lady and the Tramp (1955)	4.5	11.40
True Lies (1994)	4.5	10.57
Animal House (1978)	4.5	10.53
Hunt for Red October (1990)	4.5	10.85
League of Their Own (1992)	4.5	10.46
Shawshank Redemption (1994)	4.5	12.03
Rocky (1976)	4.5	10.29
Mask (1994)	4.5	10.71

To see the accuracy of results, the data is divided into two parts. In initial configuration, from user 1 to user 1000, the half of ratings are separated into training and test data. The rest of users stayed same, so we can calculate the ratings by using all 6040 users. After the separation, the results are compared with test results. There are more coincidences in item-based algorithm than user-based algorithm.

The results which is run on MovieLens 1M dataset can be seen at figure 6.1. In the figure, NN means Nearest N-User Algorithm which is used for user-based calculation and TCS means Tanimoto Coefficient Similarity which is used for item-based calculation. The Pearson Correlation Similarity and Nearest N-User (NN) Algorithm are executed to find 100 nearest users and recommend 100 movies.

MovieLens 1M Dataset Results								
					Test Data Found			
User	# of Training Data	# of Test Data	# of NN and TCS Common Result	# of Found Test Data In Common Result	# of Found In NN Result	Recall (%)	# of Found In TCS Result	Recall (%)
1	27	26	8	3	4	15.38	13	50.00
2	65	64	11	2	8	12.50	22	34.38
3	26	25	1	1	2	8.00	13	52.00
4	11	10	5	0	0	0.00	10	100.00
5	99	99	12	6	6	6.06	28	28.28
6	36	35	1	0	1	2.86	9	25.71
7	16	15	11	1	1	6.67	12	80.00
8	70	69	16	2	4	5.80	20	28.99
9	53	53	15	5	7	13.21	19	35.85
10	201	200	16	5	17	8.50	51	25.50
11	69	8	68	4	6	75.00	25	312.50
12	12	11	13	2	2	18.18	4	36.36
13	54	54	5	1	2	3.70	28	51.85
14	13	12	5	0	1	8.33	7	58.33
15	101	100	10	5	7	7.00	32	32.00
16	18	17	0	0	0	0.00	9	52.94
17	106	105	13	3	7	6.67	33	31.43
18	153	152	10	4	12	7.89	40	26.32
19	128	127	10	4	8	6.30	36	28.35
20	12	12	9	1	2	16.67	6	50.00

Figure 6.1: Comparing Algorithm Results with Test Data of MovieLens 1M

Tanimoto Coefficient Similarity(TCS) and Generic Boolean Preference(GBF) Item Based Recommendation is used to recommend 100 movies. So the coincidences are found in two hundred recommended movies. The recall of Item-based approach is better than the User-based approach.

We needed a new hybrid solution to get better results. After the hybridization, the recall and precision values are not changing a lot but the accuracy of predicted ratings are changing a lot in average. We have a few options for hybridization. First of it weighted hybrid solution. We tried it first, then switching hybrid solution is tried. The switching solution gives better results for movie domain than the weighted hybrid solution.

In the weighted hybrid solution, the score of different recommendation components are combined numerically. We tried this method to implement a weighted hybrid recommender system, but the results are not promising enough. The results are discussed in next chapter.

We used switching hybridization technique which is using one of the best algorithm after trying weighted one. As a result, we combine user-based and item-based algorithms with switching hybridization method. The results are

better than the weighted one. The switching technique works like selecting the best algorithm after first try. The evaluation metric is precision which is a ratio between number of test data found in recommendation list and size of the recommendation list. We used k-fold cross validation. We get the average of precision for each fold. For MovieLens dataset, k is 5.

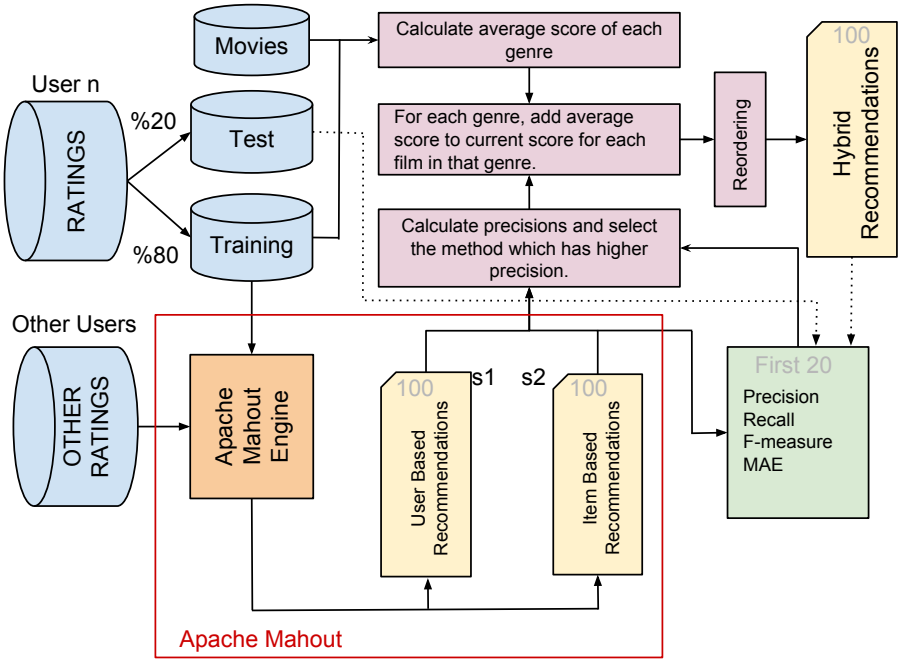

Figure 6.2: Hybrid Solution Overview

In the case where the content similarity is not null, we want to use its value to weight the user similarity, in order to give a boost to movies with similar contents. To achieve this, after the combining two method, we added the average ratings for genres to the movies. Therefore the result set is ordered again with content-based filtering which is now genre-based. The hybrid approach overview is given in Figure 6.2. The switching hybridization method is used between User-based CF and Item-based CF. After then the average genre scores are added to the predicted scores as weights. Then the scores are normalized to range between 0 and 5. The program which uses Apache Mahout is written in Java. The datasets are hold as comma separated values. The results are also recorded as comma separated values in different files.

Lets look at the results of User 1 again after creating training and test data by dividing the data into two parts. This user has 53 ratings in initial data (Appendix C). To explain the how the system works, we extracted 26 ratings of the User 1 into test data. So 27 ratings are left for training data. We calculated the NN Algorithm and GBF Item Based Recommendation with training dataset. There are 8 coincidences in the results of these two algorithms. Just the 3 of them are included in test data. The all results are given in Table 6.7.

Table6.7: The movie list which are found in both test data set and result

Movies Name	**Genre**	**Found In**	**Rating**
Apollo 13 (1995)	Drama	Common	5
Schindler's List (1993)	Drama War	Common	5
Mary Poppins (1964)	Children's Comedy Musical	TCS	5
Rain Man (1988)	Drama	TCS	5
Cinderella (1950)	Animation Comedy Musical	TCS	5
Wizard of Oz The (1939)	Adventure Children's Drama Musical	Common	4
Aladdin (1992)	Animation Children's Comedy Musical	TCS	4
Big (1988)	Comedy Fantasy	TCS	4
Ferris Bueller's Day Off (1986)	Comedy	TCS	4
Fargo (1996)	Crime Drama	TCS	4

Table 6.7 The movies found in both (Continued)

Movies Name	Genre	Found In	Rating
	Thriller		
Airplane! (1980)	Comedy	TCS	4
Bambi (1942)	Animation Children's	TCS	4
To Kill a Mockingbird (1962)	Drama	NN	4
Princess Bride The (1987)	Action Adventure Comedy Romance	TCS	3

There are 14 movies which are found in recommendations and test data. So the user is already watched and gave ratings for these movies. Let's call these movies as coincidences. We ordered the coincidences based on ratings of user 1. If two movies have same rating, then the common one should be on the top. The common means that the movie is recommended from both in user-based and in item-based method.

6.1.2.1 The Weighted Hybrid Approach

The item-based method alone gives better results than user-based method. Naturally, in weighted approach, the weight of item-based method should be bigger than the user-based one. To sum up, we chose the weights like below:

Weight for user-based method = 0.11;

Weight for item-based method = 0.91;

For a movie, the algorithm finds user-based predicted score(s1) and item-based predicted score(s2). Both scores ranges from 0 to 5 and they are decimal numbers. If the both scores exist for a movie, the scores are multiplying with weights.

Otherwise, the score stays as it is for hybrid. Pseudo code is given below:

- If s1 != 0 and s2 != 0, then assign (w1*s1 + w2*s2) as hybrid score.
- If s1 == 0 and s2 != 0, then assign s2 as hybrid score.
- If s1 != 0 and s2 == 0, then assign s1 as hybrid score.

After the multiplication, the whole recommendation result list is ordered based on new scores. The ordered list then enters a new process to add average scores of genres which is content-based filtering.

6.1.2.2 The Switching Hybrid Approach

The switching approach means we are going to choose the algorithm after trying both of it. First item-based method alone is applied on the dataset. The precision and recall values are calculated. Then the user-based method alone is applied on the same dataset. After that, we can choose the one which has better precision result.

For a movie, the algorithm finds user-based predicted score(s1) and item-based predicted score(s2). After normalization, both scores ranges from 0 to 5 and they are decimal numbers. If the one of scores exists for a movie, the precision, recall, f-measure and MAE are calculated. Then the hybrid approach can choose one of the algorithm based on one of these evaluation metrics. We chose precision metric to compare two algorithms. User-based precision (precision1) and item-based precision (precision2) are calculated. Pseudo code is given below:

- If precision1 > precision2, then assign s1 as hybrid score.
- If precision2 > precision1, then assign s2 as hybrid score.

After the switching, the whole recommendation result list is ordered based on new scores. The ordered list then enters a new process to add average scores of genres which is content-based filtering.

6.1.3 Experiments with Content Based Filtering on MovieLens Dataset

In content-based approach, average score of each actor, director, and genre are calculated. We have just genre information in MovieLens dataset, so the program calculates average score of genre. For each genre, it adds average score to current score for each film in that genre. Then it calculates average score (predicted rating) for each movie.

The implementation of hybrid recommendation system is done in Java programming language. It is using Apache Mahout mr 0.12.0 library (mahout-mr-0.12.0-SNAPSHOT.jar) to implement user-based and item based functions like Nearest N-Neighbor algorithm. However for Content-Based filtering, there is no Mahout Library exists, so we did it by writing pure Java code. Firstly, we calculated the average rating for each genre which a user watched and gave scores. Then, we added these scores to the final hybrid scores. After that, we normalized all scores to range between 0 and 5, so that we could compare the final predicted rating with test data. After the normalization, the final result was reordered from highest score to lowest score. The first 20 results were evaluated with evaluation metrics. The overview of category boosted content based filtering is given on Figure 6.3.

We validated our method with k-fold cross validation and calculated precision, recall, f-measure and MAE on each fold execution. The configuration of Nearest N-Neighbor algorithm is set to find 30 nearest neighbors, because using 30 neighbors are optimal and recommended in literature [31]. The configuration of the Java program is set like below. The validation is given in next chapter.

For each user:

How many neighbors found?: 30

How many recommended movies found?: 100 for user-based, 100 for item-based

How many recommended movies in result are regarded for validation?: First 20 movies

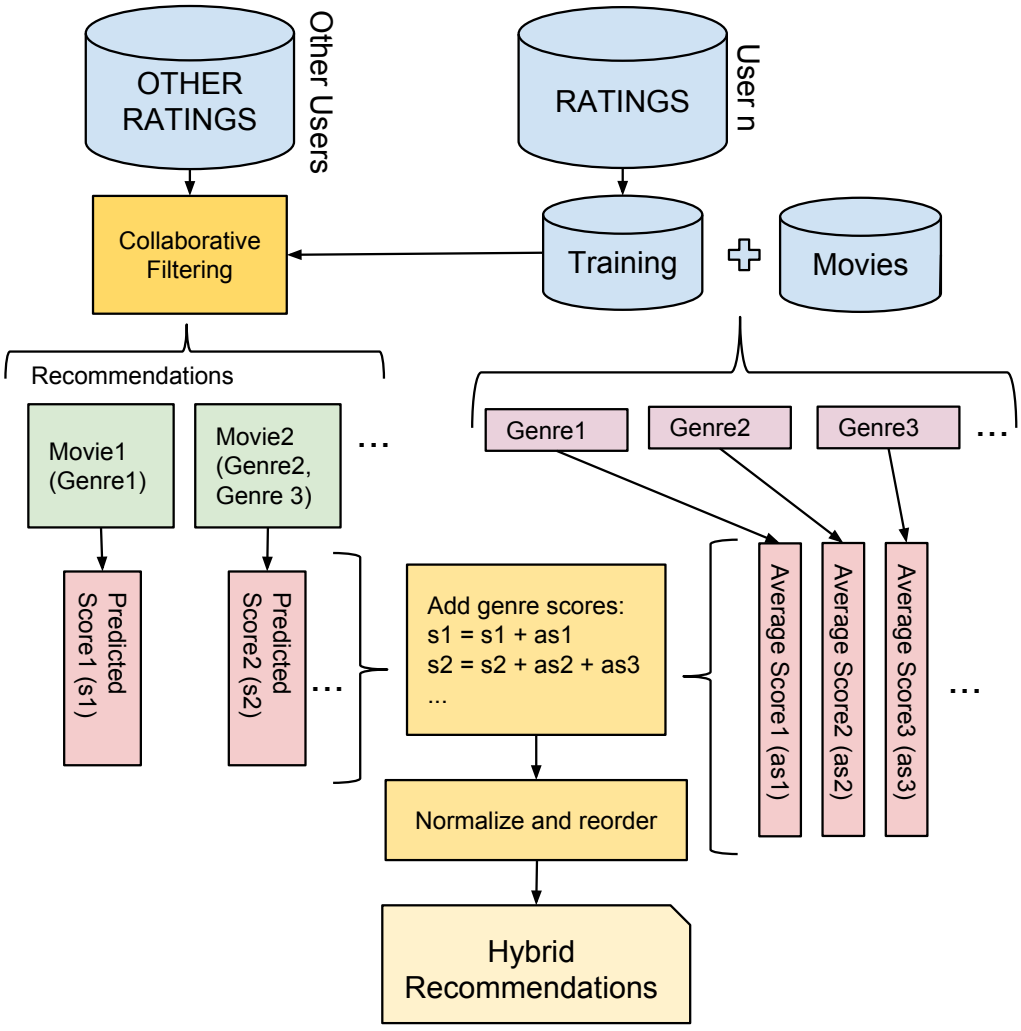

Figure 6.3: Category Boosted Content Based Filtering Overview

6.2 Experiments on METU Student Elective Course Dataset

6.2.1 Hybrid Approach Implementation on METU Student Elective Course Dataset

We used switching hybridization technique plus category-boosted CBF on METU Student Elective Course Dataset. In previous sections, the CF and CBF methods are discussed one by one on MovieLens dataset. The final solution is run on METU Student Elective Course dataset with minor changes. The 3-fold cross validation is done. Precision, Recall, F-Measure and MAE are calculated after each fold. The results which is run on METU Elective Course dataset can be seen in next chapter.

For each user:

How many neighbors found?: 10

How many recommended courses found?: 15 for user-based, 15 for item-based

How many recommended courses in result are regarded for validation?: First 5 courses

CHAPTER 7

RESULTS AND VALIDATION

In this chapter, the results are evaluated by MAE (Mean Absolute Error), Precision, Recall and F-Measure metrics. The results of hybrid approach on Metu Elective Course Dataset is also evaluated by a user study.

7.1 Validation Metrics

In order to validate the results of Hybrid approach, k-fold cross validation is used. Cross-validation (or rotation estimation), is a model validation technique for assessing how the results of a statistical analysis will generalize to an independent data set. When the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice, cross-validation technique can be used.

In each round of cross-validation, the data set is partitioned into two subsets. One subset is used as training set to performing the analysis. The other subset is used as testing set to validate the analysis. Multiple rounds should be run on different subsets so that the final validation result is the average of all results.

In k-fold cross-validation, k equal sized subsets are generated from the original data set randomly. Among k subsets, one of the single subset is selected as test data, and the remaining subsets are used as training data. The cross-validation process is then repeated k times (the folds), with each of the k subsets used exactly once as the test data. A single prediction is produced from k results from the all folds by getting the average of them. In literature, 10-fold cross-

validation is commonly used. We used **5-fold cross-validation** for MovieLens dataset 7.1 and **3-fold cross-validation** for METU Student Elective Course dataset since each user does not have enough ratings for accurate estimation.

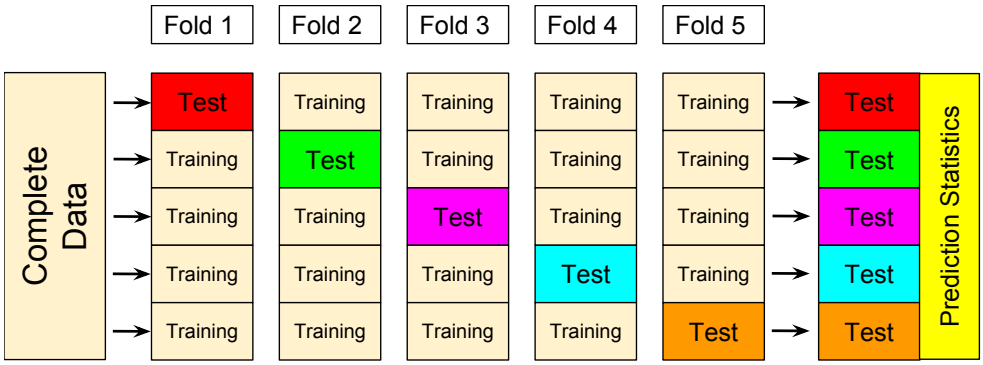

Figure 7.1: K-fold cross validation overview

Precision is ratio of number of recommended items which is in the test subset and the recommendation size 7.1. Recall is ratio of number of recommended items which is in the test subset and the test data size 7.2. F-measure is one of the combined measure of precision and recall 7.3. Generally F1 is used and it is one of the common F-measure. The F1 score is the 2*((precision*recall)/(precision+recall)). The F1 score conveys the balance between the precision and the recall. We used F1 method for F-measure.

$$precision = \frac{|relevantMovies| \cap |retrievedMovies|}{|retrievedMovies|}. \tag{7.1}$$

$$recall = \frac{|relevantMovies| \cap |retrievedMovies|}{|relevantMovies|}. \tag{7.2}$$

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}. \tag{7.3}$$

We believe the hybrid method gives better recommendations but the Precision and Recall measures are lower than pure item-based CF. The precision may not be the indicator for best algorithm because it looks the number of items which are found both in result list and test subset. There are two reasons why the precision and recall values are not an indicator for recommendation systems:

(1) First of all, when we offer 100 recommendations, it covers the test subset better. If there are 100 movies in the test subset, the recall can be $100/100 = 1$ at most. However offering 100 recommendation does not good for users. For example, Amazon shows just a few recommended books like 3 or 5 books. If we generate 5 recommendation with algorithm, it will just cover at most 5 movies in the test subset. In this case, our recall value is going to be $5/100$ at most, and it is very low. So the recall is really dependent on number of recommendation in result. Moreover, if the number of recommended items is changing frequently, we can not get correct precision. For example, the test subset size can be dependent on the user's rating size. Therefore, both precision and recall do not reflect the correct evaluation.

(2) If something is searched in a search engine (e.g. Google), related results are very important so that the recall should be very high. Recall value is considerable criterion for information retrieval. On the other hand, any user may like the items which are not related to their past information and which are not in the test subset. Irrelevant items can be recommended in recommendation systems and users can like them. For example, users who bought x, also bought y. Therefore the sellers are putting x and y side by side at the same floor so that users can reach the related products easily. We cannot trust recall value in this case because the test subset may not include the items which should be recommended for a user. For instance, the user-based method gives lower recall values. If we do a user-study on users, we can see that the recommended items are mostly liked by users. Because user-based methods are not regarding similar items, but similar user's behaviors.

If we do not use recall values, we cannot use F-measure too. However many studies in literature used both precision and recall, so we calculated all of them. The most reliable technique is asking the users if the recommendation is good or bad. User-study is not applicable for MovieLens dataset. The users are not known and the dataset includes old ratings from 2000. For this reason, we calculated the precision and recall values.

The Precision, Recall and F-Measure, the MAE (Mean Absolute Error) are

calculated. The formula of MAE is given in Equation 7.4. The MAE is an average of the absolute errors $|e_i| = |f_i - y_i|$, where f_i indicates the prediction and y_i indicates the true value. Relative frequencies may be included in the formula as weight factors. The MAE is a common measure of forecast error in time series analysis.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|f_i - y_i| = \frac{1}{n}\sum_{i=1}^{n}|e_i|. \tag{7.4}$$

7.2 Validation of Hybrid Approach on MovieLens 1M Dataset

The MovieLens 1M dataset is partitioned into 5 equal sized subsamples. If a user have 100 ratings, there are random 20 ratings in first test subset and remaining 80 ratings in first training subset. In each fold, the test subset is changed with other random 20 ratings. The precision, recall, f-measure and MAE values are calculated for user-based, item-based and hybrid method in each phase.

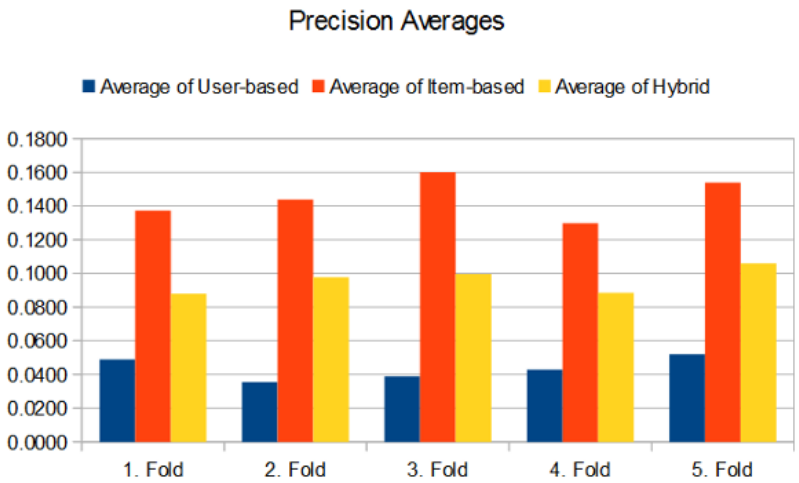

Figure 7.2: Precision Averages of Weighted Hybridization

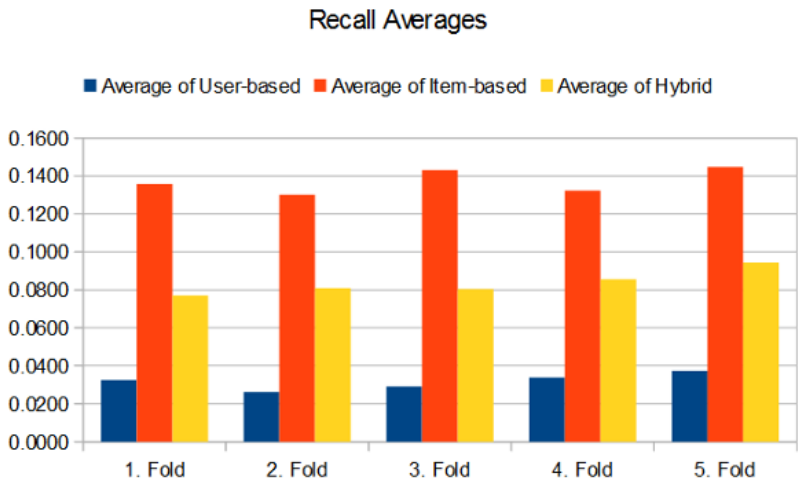

Figure 7.3: Recall Averages of Weighted Hybridization

For the weighted approach, the averages of precision, recall and F1 in each fold are given at Figure 7.2, Figure 7.3 and Figure 7.4. The averages of all folds are given in Table 7.1.

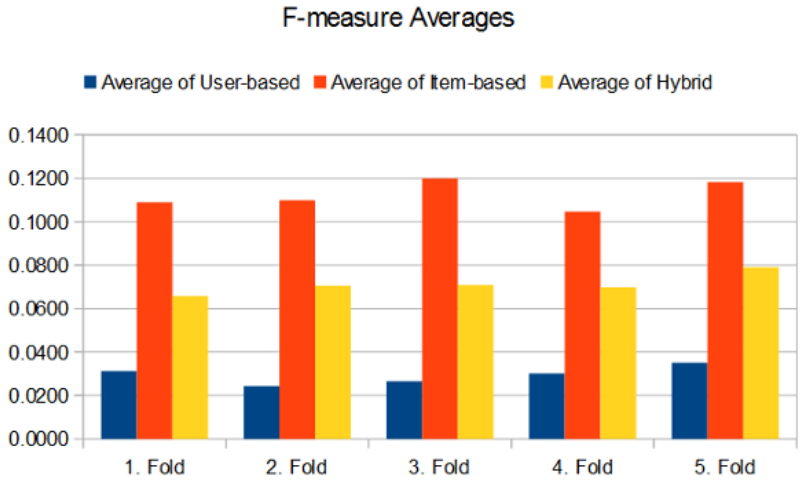

Figure 7.4: F-measure Averages of Weighted Hybridization

Table7.1: Average Values of Precision, Recall and F-measure of Weighted Hybrid Method on MovieLens data set after 5-Fold Cross Validation

Method	**Avg. Precision**	**Avg. Recall**	**Avg. F-measure**
User-based CF	0.043	0.031	0.029
Item-based CF	**0.145**	**0.137**	**0.112**
Hybrid Method	0.096	0.083	0.071

In weighted hybridization approach, the hybrid method is always in between

user-based and item-based method. The item-based method always covers the test subset better than other methods. Therefore, we needed to change our hybridization method.

In second configuration, we run the algorithm with **switching hybridization approach** without content based filtering (CBF). The results are slightly better than the weighted method. We select the algorithm with the highest precision among user-based CF and item-based CF. The program selects the algorithm for each user differently. If precision of user-based is greater for user 1, we used user-based CF or vice versa. For the switching approach, the averages of precision, recall and F1 in each fold are given at Figure 7.5, Figure 7.6 and Figure 7.7.

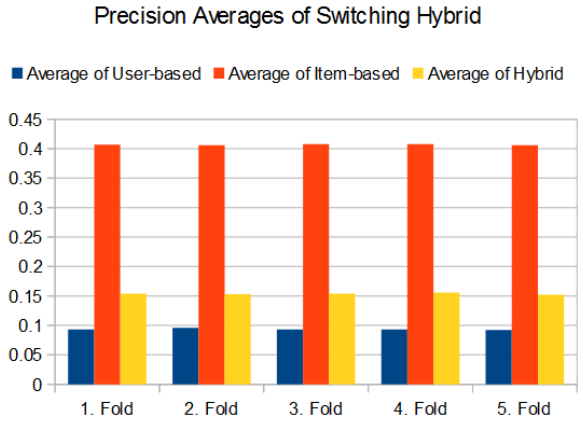

Figure 7.5: Precision Averages of Switching Hybridization without Category-Boosted CBF

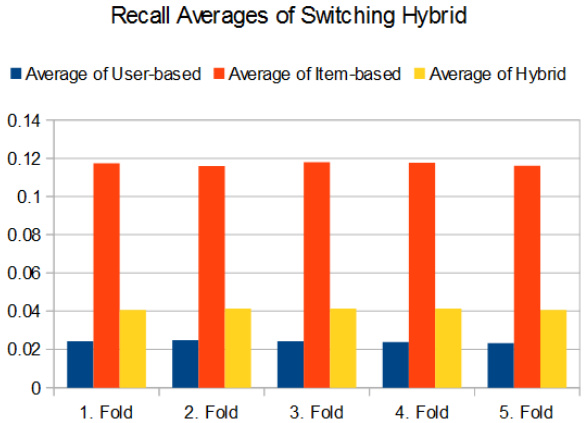

Figure 7.6: Recall Averages of Switching Hybridization without Category-Boosted CBF

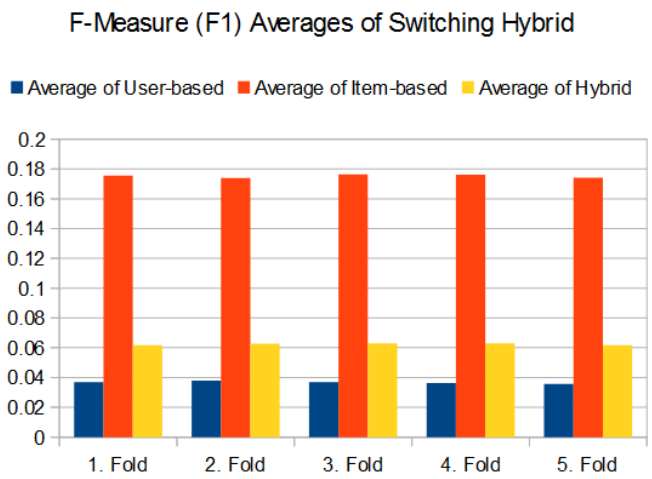

Figure 7.7: F-measure Averages of Switching Hybridization without Category-Boosted CBF

In third configuration, we run the algorithm with switching hybridization approach and plus category-boosted CBF. The results are slightly better than the only weighted and only switching method. For the switching approach with category-boosted CBF, the averages of precision, recall and F1 in each fold are given at Figure 7.8, Figure 7.9 and Figure 7.10.

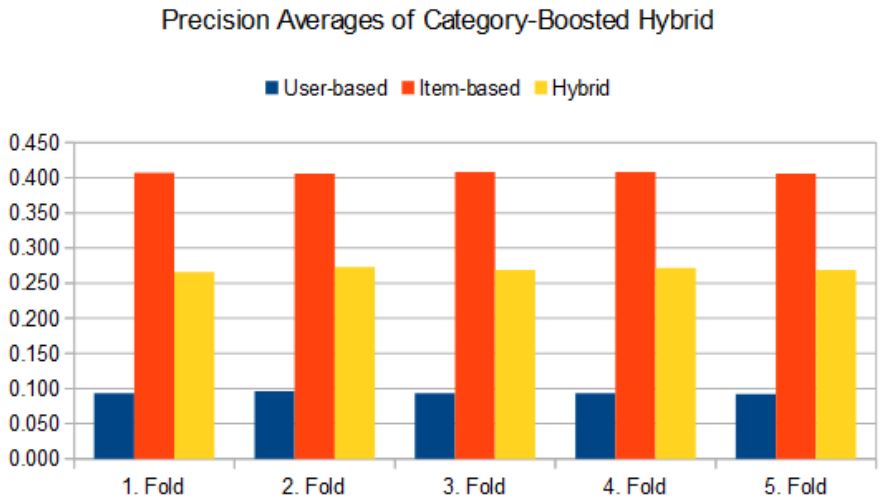

Figure 7.8: Precision Averages of Switching Hybridization with Category-Boosted CBF

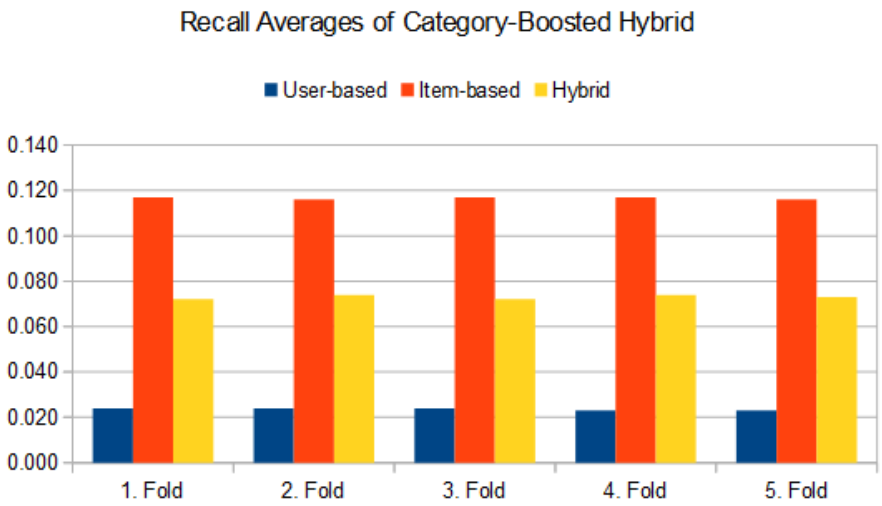

Figure 7.9: Recall Averages of Switching Hybridization with Category-Boosted CBF

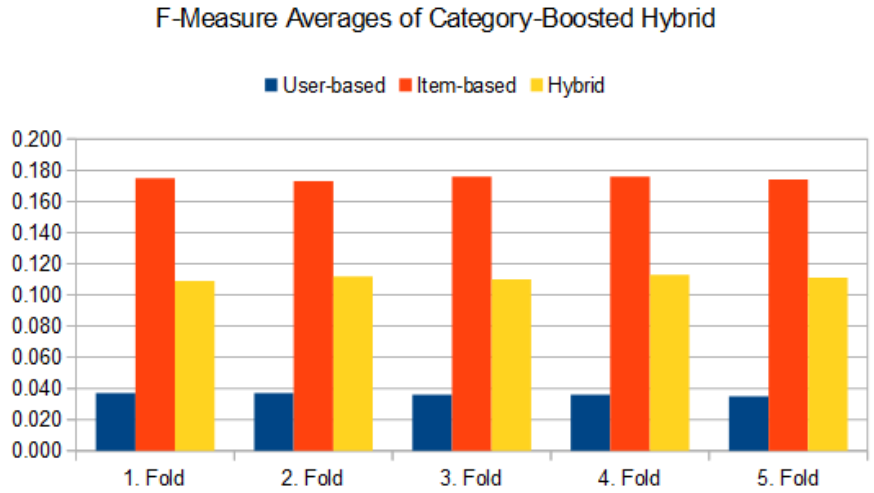

Figure 7.10: F-measure Averages of Switching Hybridization with Category-Boosted CBF

The highest 100 precisions of switching hybrid method are compared with other methods (user-based and item-based). We did it 5 times because of 5-fold cross-validation. The results of first fold are given in Figure 7.11 and Figure 7.12. In addition to this, recall of first fold can be seen on Figure 7.13 and Figure 7.14 and F-measure can be seen on Figure 7.15 and Figure 7.16. The averages of all folds are given in Table 7.2 for without category-boosted and in Table 7.3 for with category-boosted.

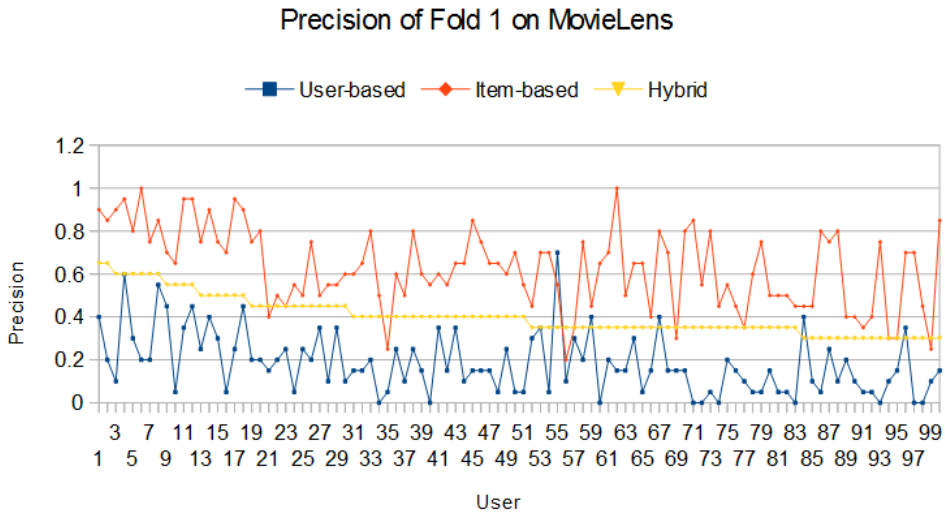

Figure 7.11: Precision of switching hybrid method without category-boosted CBF on MovieLens 1M Dataset - 1. Fold

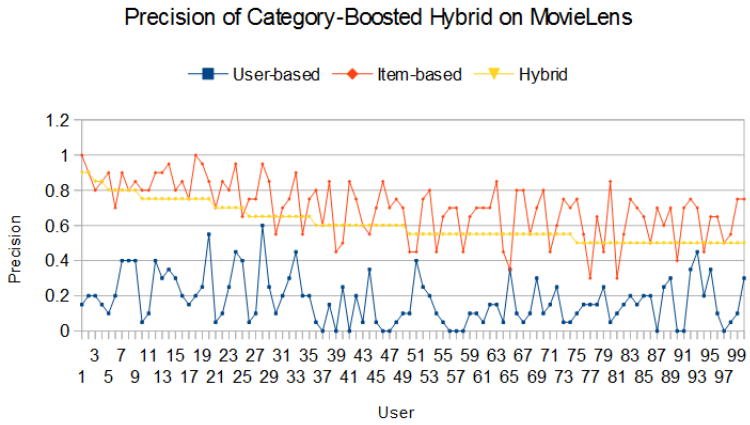

Figure 7.12: Precision of switching hybrid method with category-boosted CBF on MovieLens 1M Dataset - 1. Fold

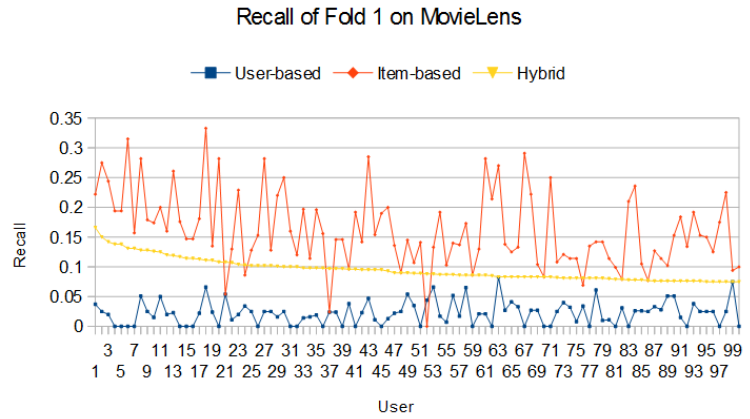

Figure 7.13: Recall of switching hybrid method without category-boosted CBF on MovieLens 1M Dataset - 1. Fold

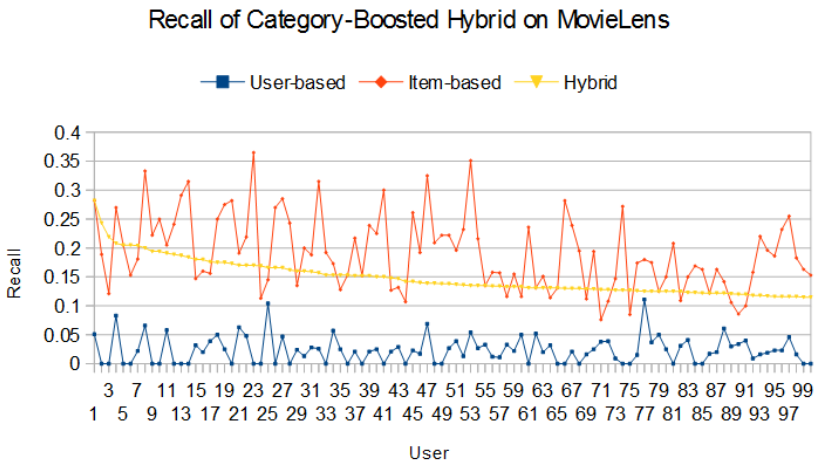

Figure 7.14: Recall of switching hybrid method with category-boosted CBF on MovieLens 1M Dataset - 1. Fold

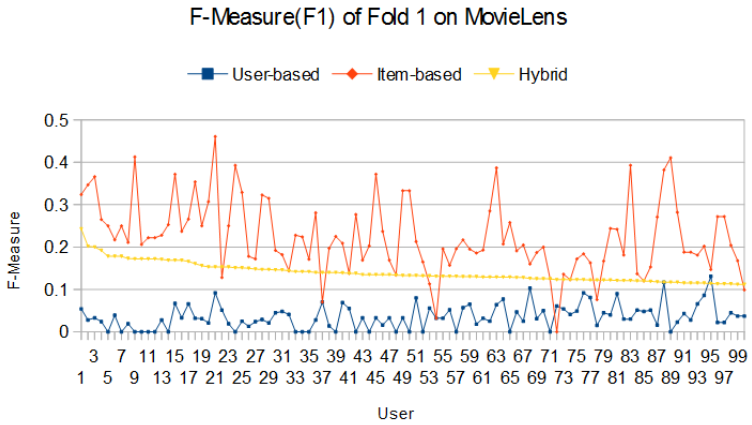

Figure 7.15: F-Measure(F1) of switching hybrid method without category-boosted CBF on MovieLens 1M Dataset - 1. Fold

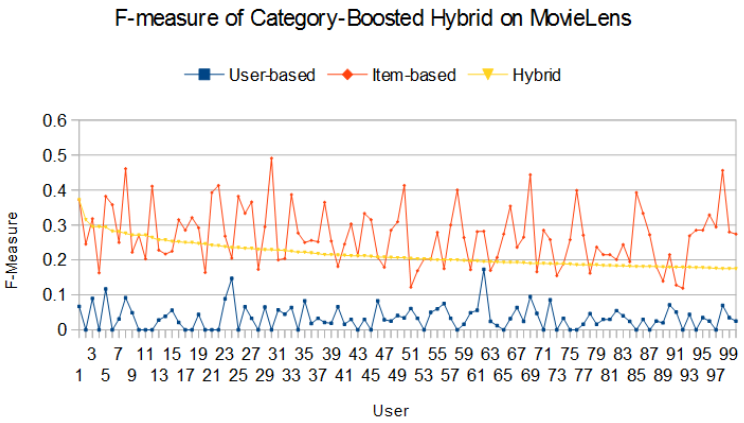

Figure 7.16: F-Measure(F1) of switching hybrid method with category-boosted CBF on MovieLens 1M Dataset - 1. Fold

Table7.2: Average Values of Precision, Recall and F-measure of Switching Hybrid Method without category-boosted CBF on MovieLens data set after 5-Fold Cross Validation

Method	**Avg. Precision**	**Avg. Recall**	**Avg. F-measure**
User-based CF	0.093	0.024	0.036
Item-based CF	**0.407**	**0.117**	**0.175**
Hybrid Method	0.153	0.041	0.062

Table7.3: Average Values of Precision, Recall and F-measure of Switching Hybrid Method with category-boosted CBF on MovieLens data set after 5-Fold Cross Validation

Method	**Avg. Pre.**	**Avg. Re.**	**Avg. F-mea.**	**Avg. MAE**
User-based CF	0.093	0.024	0.036	2.876
Item-based CF	**0.407**	**0.117**	**0.175**	1.041
Hybrid Method	0.270	0.073	0.111	**0.941**

For switching technique, 879 users are randomly chosen for test. Their ratings are separated 5 parts randomly. So 5-cross validation is used. Among 879 users, how many of them resulted with higher precision values after hybrid method is calculated. The result is not better than the item-based. Using hybrid method diminishes the precision of item-based. Using category boosted method also increases the precision a little bit. All results which calculated after each fold are given in Table 7.4 to compare results of switching hybrid method with category-boosted CBF.

Table7.4: Number of the highest precision values for each method when using switching hybrid method with category-boosted CBF on MovieLens

	UB	IB	Hybrid	UB(%)	IB(%)	Hybrid(%)
1. fold	9	692	168	1.02	78.73	19.11
2. fold	11	677	182	1.25	77.02	20.71
3. fold	10	688	172	1.14	78.27	19.57
4. fold	13	692	169	1.48	78.73	19.23
5. fold	11	678	182	1.25	77.13	20.71

We calculated the MAE for switching method with category boosted CBF, it can bee seen in Figure 7.17 while the averages of calculated MAE for first fold are given in Figure 7.18. The figures shows that the calculated MAE of hybrid

method is better than the item-based method. It is also sometimes better than the user-based method too.

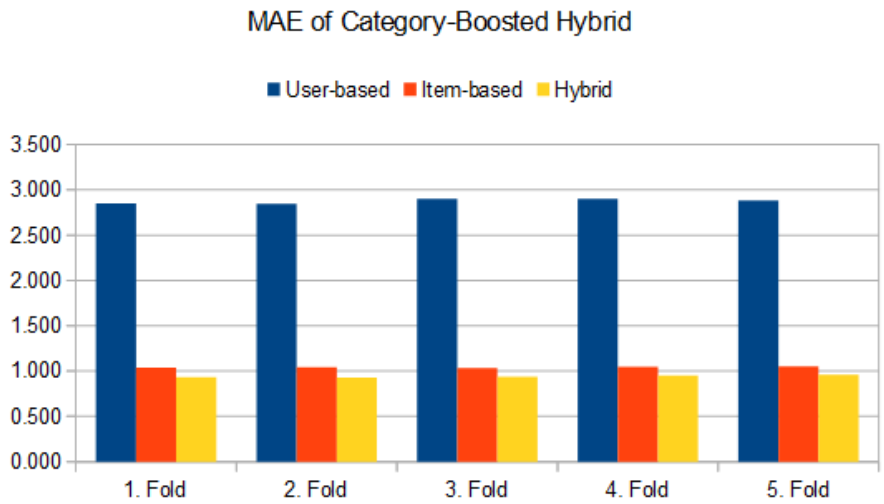

Figure 7.17: MAE of each fold on MovieLens 1M Dataset when using switching hybrid method with category-boosted CBF

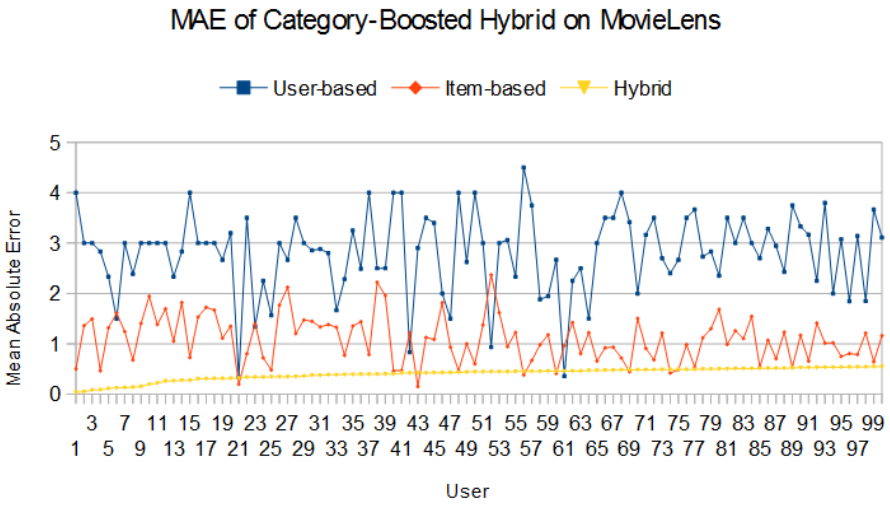

Figure 7.18: MAE of first fold on MovieLens 1M Dataset when using switching hybrid method with category-boosted CBF

7.3 Runtime Performance of Algorithm on MovieLens 1M

The more data in a recommendation system, the better the results. Mahout provides performance on big data. For best results, we used following command-line flags to our JVM:

-server: Enables the server VM, which is generally appropriate for long running, computation intensive applications.

-Xmx4096m -Xms2048m: Make the heap as big as possible. Giving a lot of memory for caching helps performance of Mahout.

-da -dsa: Disable all assertions.

-XX:NewRatio=9: Increase heap allocated to old objects, which is most of them in this framework.

The program was run on Eclipse on Java7se and is compressed in a jar file. The jar file is a Java Archive file format which enables developers to bundle multiple files into a single archive file. This jar file is run on different computers which have different performances.

The first computer have these features: 4 cores (Intel Core i5, 2.60 GHz), 8 GB RAM and Windows 8 64-bit. The algorithm took 3 hours 40 minutes 44 seconds to run on console for 5-fold cross validation on MovieLens 1M dataset. For each fold, it took 45 minutes to finish. To calculate the precision, recall, f-measure and MAE, the algorithm is run on 8 hours 30 minutes in same computer. Each fold is calculated in 1 hour 42 minutes.

The same program with same dataset is also run on Slurm machine with 20 nodes. Each node has 1 core and 3 GB RAM. The Slurm was installed on Ubuntu. We changed the system to run on distributed system with 5 threads. Each thread was run on different fold in 5-fold cross validation. The program was resulted in 74 minutes to implement algorithm and calculate the evaluation metrics. We can say that, it took 1 hour 14 minutes to finish for 5 folds. It means each fold can be calculated in 14 minutes. Slurm made the recommendation system 7.2 times faster.

7.4 Validation of Hybrid Approach on METU Student Elective Course Dataset

The same algorithm is run on METU Elective Course Dataset and the evaluation metrics are calculated. The configuration parameters are given below:

- Number of neighbors : 10
- Number of recommendations : 15
- Number of recommendations which is used in validation : First 5 courses

The time to run whole algorithm took less than 1 minute because the dataset includes nearly 1055 ratings and it is smaller than the MovieLens dataset which includes over 1 million ratings. Therefore we do not need to run it on Slurm since there is no performance issue on METU Elective Course Dataset. The results are evaluated with precision (Figure 7.21), recall (Figure 7.22), f-measure (Figure 7.23) and MAE (Figure 7.27 and Figure 7.28). Precision averages are given in Figure 7.24, recall averages are given in Figure 7.25, and f-measure averages are given in Figure 7.26). Average values of precision, recall and f-measure of switching hybrid method with category-boosted CBF on METU Elective Course dataset after 3-fold cross validation is given in Table 7.5. Number of the highest precision values for each method when using switching hybrid method with category-boosted CBF on METU Elective Course dataset is given in Table 7.6. The results show that the pure item-based collaborative filtering gives best result. Therefore we needed to do a user-study.

We prepared a user survey and asked students who are currently a student in METU in 3rd or 4th grade. The dataset includes their actual ratings, so we asked them if they likes the recommendations or not. There were 10 question which 8 of them are our recommendations. 2 of them are not our recommendations but the pure algorithms generated them. We were expecting a dislike for these 2 questions. The results of user-study showed that our hybrid approach is successful on predicting the elective lectures. The results can be seen on Figure 7.19 and Figure 7.20.

65 students participated in our survey. There are 300 students in the dataset. This means that more than 10 percent of whole students participated to the user survey. The most of the students liked the 5 of 8 recommendations which are produced by our hybrid algorithm. The most of the students disliked all 2 recommendations which are not produced by our hybrid algorithm and this is an expected result. So the 7 out of 10 recommendations are verified by students. Our hybrid algorithm is successful in 70 percent probability. The results can be seen on following figures. The user-survey results proved that our algorithm is better than the pure algorithms.

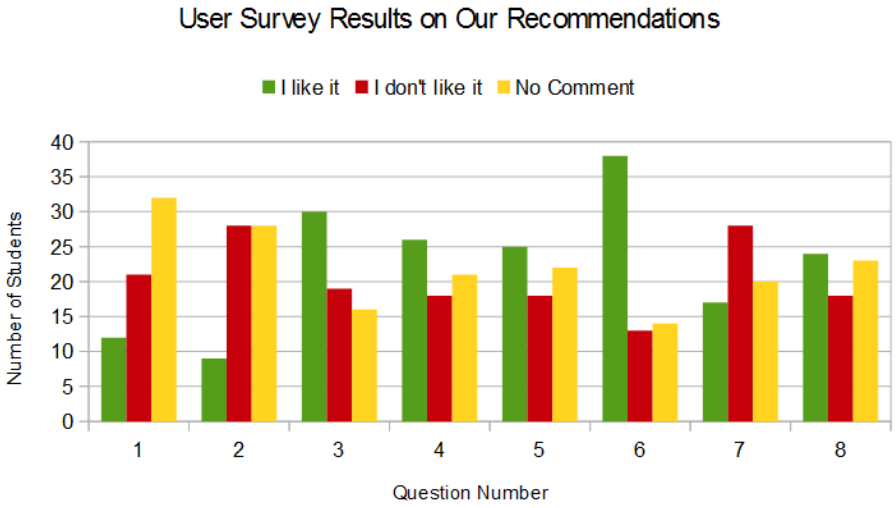

Figure 7.19: User Survey Results on Our Recommendations

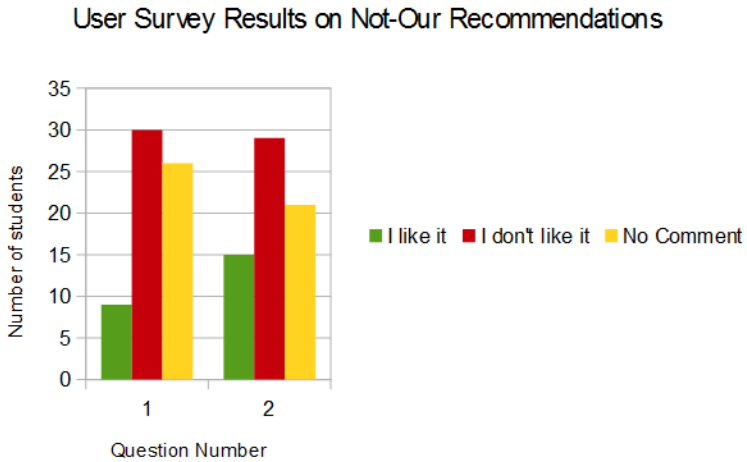

Figure 7.20: User Survey Results on Not-Our Recommendations

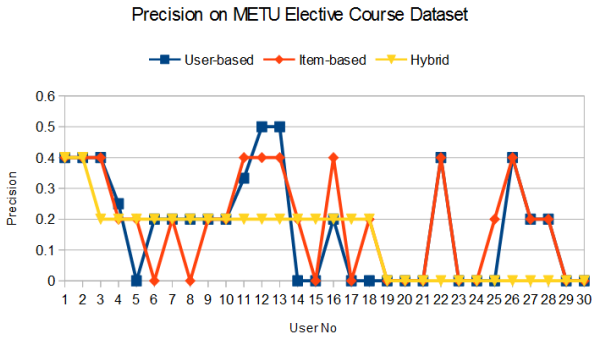

Figure 7.21: Precision of switching hybrid method with category-boosted CBF on METU Elective Course Dataset - 1. Fold

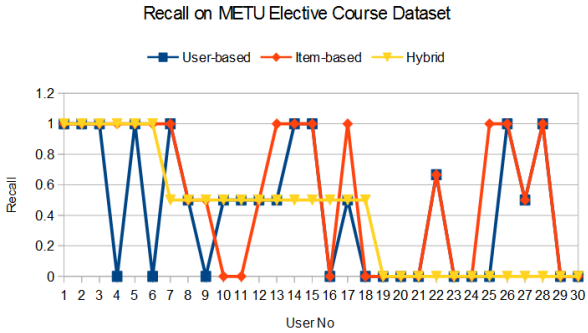

Figure 7.22: Recall of switching hybrid method with category-boosted CBF on METU Elective Course Dataset - 1. Fold

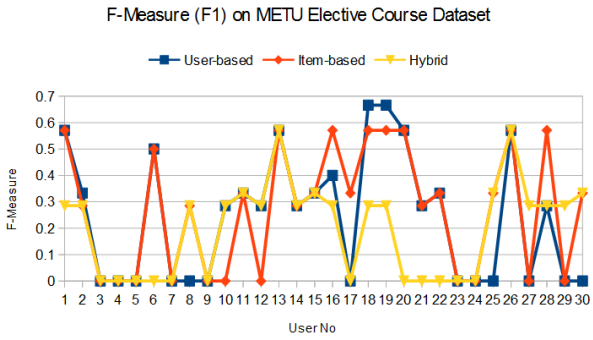

Figure 7.23: F-Measure(F1) of switching hybrid method without category-boosted CBF on METU Elective Course Dataset - 1. Fold

Table7.5: Average Values of Precision, Recall and F-measure of Switching Hybrid Method with category-boosted CBF on METU Elective Course dataset after 3-Fold Cross Validation

Method	**Avg. Pre.**	**Avg. Re.**	**Avg. F-mea.**	**Avg. MAE**
User-based CF	0.157	0.468	0.231	25.653
Item-based CF	**0.176**	**0.585**	**0.266**	**18.269**
Hybrid Method	0.121	0.394	0.182	27.351

Table7.6: Number of the highest precision values for each method when using switching hybrid method with category-boosted CBF on METU Elective Course dataset

	UB	IB	Hybrid	UB(%)	IB(%)	Hybrid(%)
1. fold	3	3	12	10	10	40
2. fold	2	4	11	6.6	13.3	36.6
3. fold	4	4	11	13.3	13.3	36.6

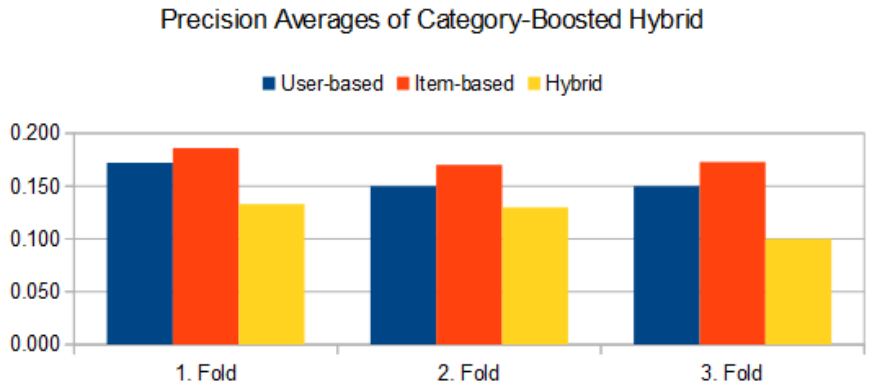

Figure 7.24: Precision Averages of each fold on METU Student Elective Course Dataset when using switching hybrid method with category-boosted CBF

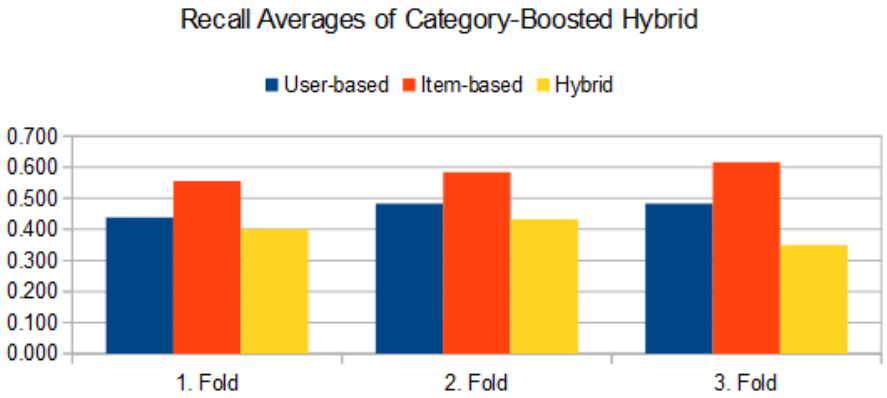

Figure 7.25: Recall Averages of each fold on METU Student Elective Course Dataset when using switching hybrid method with category-boosted CBF

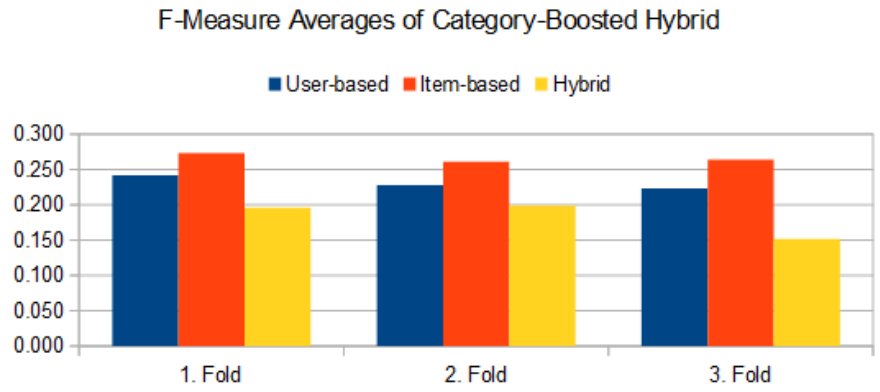

Figure 7.26: F-measure Averages of each fold on METU Student Elective Course Dataset when using switching hybrid method with category-boosted CBF

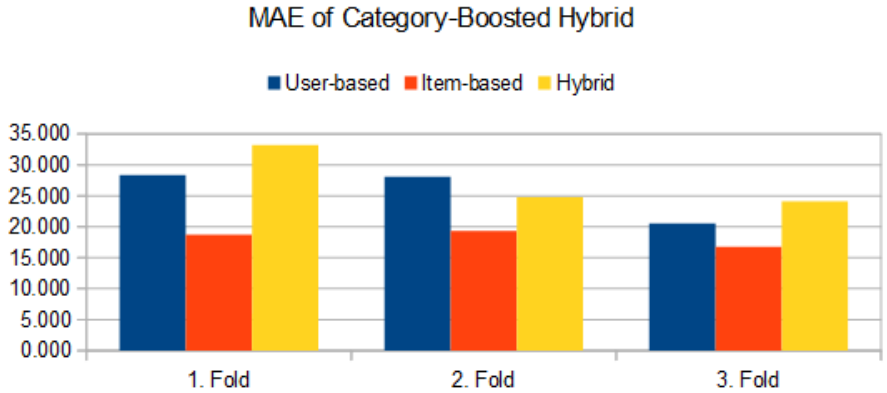

Figure 7.27: MAE of each fold on METU Student Elective Course Dataset when using switching hybrid method with category-boosted CBF

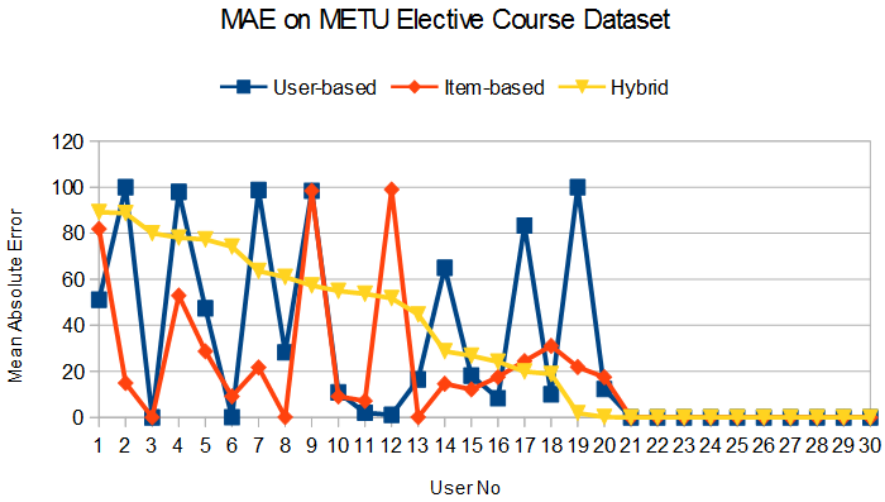

Figure 7.28: MAE of first fold on METU Student Elective Course Dataset when using switching hybrid method with category-boosted CBF

CHAPTER 8

CONCLUSION

Nowadays, There is an information overload in the internet and people hardly find the things which fit their taste. People require to filter the information, so recommendation systems help them in this way. Recommendation systems become popular due to their benefits for both companies and users. The companies want to offer to the user specific information to increase the purchases while users want to reach the relevant items easily. There are three prevailed approaches in literature for better recommendation: content based filtering (CBF), collaborative filtering (CF) and hybrid approaches. The most of the recommendation applications use the algorithms of these three approaches. The performance of the hybrid approach is compared with the pure collaborative and content-based methods in several studies. As a result, the studies demonstrate that pure approaches have problems and are not enough for a better recommendation system and the hybrid method is needed. The CBF and CF methods can be used in a hybrid approach to overcome some of the common problems in recommender systems such as cold start and the sparsity problem. We improved a hybrid method and performed experiments with bigger and more varied data set. After we done on experiments, we compared results with validation metrics such as precision, recall, f-measure and mean absolute error (MAE). We used k-fold cross validation technique for this study. After the experiments are done, we also did a survey on users to whom the dataset belongs.

The results show that most of the time, using only Item-Based CF gives better precision, recall and f-measure. However Item-Based CF generates predicted

values with higher error rate and it does not good enough for prediction of user ratings. On the other hand, User-Based CF gives worse prediction, recall and f-measure, but it predicts the closest ratings. For these reasons, we needed to implement a new hybrid method by using Item-Based and User-Based CF. We tried weighted and switching hybridization techniques and we chose "switching approach" at the end. Switching method is more suitable for combining Item-based and User-based CF and generates better results. Our algorithm selects the correct method based on the highest precision after first try. Also we added category boosted filtering which is one kind of CBF. Using CBF increases the prediction's accuracy. The results after experiments are evaluated with evaluation metrics and a user study. As a result, we verified that our method gives better MAE values and better results for users. Our method is also easy to implement and more suitable for recommendation systems, therefore it can be used in real life.

We used MovieLens dataset in our first experiment. The program which runs our algorithm recommends 20 movies for each user at the end of the experiments. So the number of recommendation is fixed. In future work, we can recommend different number of recommendations to each user rather than recommending constant 20 recommendation. The number of movies can be bound to determined threshold. For example, recommending the movies which have predicted ratings greater than the threshold 4.0. Also we can limit the maximum number of recommended movies like 20 movies. If there are more than 20 movies which have predicted rating greater than 4.0, we can take the first 20 movies and omit the rest of them. This method enables the special number of recommended items for each user. We can calculate the precision by finding the common items which exists in recommendations and test subset. Changing the recommendation sizes based on users may give better results.

We also used METU Elective Course Dataset which includes actual ratings of junior and senior students for elective courses in a university. We implemented the hybrid algorithm on this dataset and took results. The recommended courses are asked to the students. The students are responded for 10 recommendations. The user survey showed that the accuracy of hybrid recommendations are 70

percent.

In future work, this study can run on distributed computing platform. Apache Mahout works alongside Hadoop which is distributed computing platform, so we can scale it out easily. We are using MapReduce which supports Hadoop and MapReduce. We did not use Hadoop and Mapreduce for this study, however they can be used in the future work to implement movie recommendation system with scalable format.

REFERENCES

[1] A. Albadvi and M. Shahbazi. A hybrid recommendation technique based on product category attributes. *Expert Systems with Applications*, 36(9):11480–11488, 2009. (Cited in Page 9)

[2] R. M. Bell, M. Jahrer, and A. Töscher. The BigChaos Solution to the Netflix Prize 2008. *Netflix Prize, Report*, pages 1–17, 2008. (Cited in Page 11)

[3] D. Billsus and M. J. Pazzani. Learning Collaborative Information Filters. In *Icml '98*, pages 46–54, 1998. (Cited in Page 16)

[4] J. Bobadilla, F. Ortega, a. Hernando, and a. Gutiérrez. Recommender systems survey. *Knowledge-Based Systems*, 46:109–132, 2013.

[5] J. Breese, D. Heckermanx, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Uai*, 1998. (Cited in Pages 15, 16)

[6] R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and UserAdapted Interaction*, 12(4):331–370, 2002. (Cited in Pages 12, 46)

[7] R. Burke. Hybrid web recommender systems. *The adaptive web*, pages 377–408, 2007. (Cited in Pages 12, 28)

[8] A. Dhond, A. Gupta, and S. Vadhavkar. Data mining techniques for optimizing inventories for electronic commerce. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '00*, pages 480–486, 2000. (Cited in Page 1)

[9] J. Donaldson. A hybrid social-acoustic recommendation system for popular music. In *Proceedings of the 2007 ACM conference on Recommender systems - RecSys '07*, page 187, 2007. (Cited in Page 9)

[10] F. Dong, J. Luo, X. Zhu, Y. Wang, and J. Shen. A Personalized Hybrid Recommendation System Oriented to E-Commerce Mass Data in the Cloud. In *2013 IEEE International Conference on Systems, Man, and Cybernetics*, pages 1020–1025, 2013. (Cited in Page 9)

[11] S. Dooms, T. De Pessemier, and L. Martens. Offline optimization for user-specific hybrid recommender systems. *Multimedia Tools and Applications*, pages 1–24, 2013. (Cited in Page 12)

[12] S. J. E. Lee. A Study on the Hybrid Recommendation System. *International Journal of Research in Engineering and Technology*, 2, 2013. (Cited in Page 10)

[13] M. H. Esfahani and F. K. Alhan. New hybrid recommendation system based On C-Means clustering method. In *The 5th Conference on Information and Knowledge Technology*, pages 145–149, 2013. (Cited in Page 10)

[14] D. Fang, K. R. Varshney, J. Wang, K. N. Ramamurthy, A. Mojsilovic, and J. H. Bauer. Quantifying and Recommending Expertise When New Skills Emerge. In *2013 IEEE 13th International Conference on Data Mining Workshops*, pages 672–679, 2013. (Cited in Page 22)

[15] 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. (Cited in Page 9)

[16] Ò. C. Herrada. *Music Recommendation and Discovery in the Long Tail.* PhD thesis, Universitat Pompeu Fabra, 2009. (Cited in Pages 17, 18, 19, 23)

[17] G. Ingersoll. Apache Mahout: Scalable machine learning for everyone. *IBM Developer Works*, pages 1–19, 2011. (Cited in Page 42)

[18] M. Kaminskas, F. Ricci, and M. Schedl. Location-aware music recommendation using auto-tagging and hybrid matching. In *Proceedings of the 7th ACM conference on Recommender systems - RecSys '13*, pages 17–24, 2013. (Cited in Page 9)

[19] A. Klašnja-Milićević, B. Vesin, M. Ivanović, and Z. Budimac. E-Learning personalization based on hybrid recommendation strategy and learning style identification. *Computers & Education*, 56(3):885–899, 2011. (Cited in Page 10)

[20] Y. Koren. The BellKor Solution to the Netflix Grand Prize. *Baseline*, (August):1–10, 2009. (Cited in Page 11)

[21] V. Krishnan, P. K. Narayanashetty, M. Nathan, R. T. Davies, and J. a. Konstan. Who predicts better? Results from an online study comparing humans and an online recommender system. In *Proceedings of the 2008 ACM conference on Recommender systems - RecSys '08*, volume October, page 211, 2008. (Cited in Page 11)

[22] G. Lekakos and P. Caravelas. A hybrid approach for movie recommendation. *Multimedia Tools and Applications*, 36(1-2):55–70, 2008. (Cited in Page 9)

[23] H. Liu, M. Amin, B. Yan, and A. Bhasin. Generating supplemental content information using virtual profiles. In *Proceedings of the 7th ACM conference on Recommender systems - RecSys '13*, pages 295–302, New York, New York, USA, 2013. ACM Press. (Cited in Page 10)

[24] B. M. Marlin. Modeling User Rating Profiles For Collaborative Filtering. *Advances in Neural Information Processing Systems*, 16:627–634, 2003. (Cited in Page 16)

[25] F. Mourão, L. Rocha, J. A. Konstan, and W. Meira. Exploiting non-content preference attributes through hybrid recommendation method. In *Proceedings of the 7th ACM conference on Recommender systems - RecSys '13*, pages 177–184, 2013. (Cited in Page 10)

[26] M. G. Ozsoy and F. Polat. Trust based recommendation systems. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining - ASONAM '13*, pages 1267–1274, New York, New York, USA, 2013. ACM Press. (Cited in Pages 14, 27)

[27] M. Papagelis and D. Plexousakis. Qualitative analysis of user-based and item-based prediction algorithms for recommendation agents. *Engineering Applications of Artificial Intelligence*, 18:781–789, 2005. (Cited in Page 12)

[28] M. Papagelis, D. Plexousakis, and T. Kutsuras. Alleviating the sparsity problem of collaborative filtering using trust inferences. In *Proceedings of the Third international conference on Trust Management*, pages 224–239, 2005. (Cited in Pages 7, 12)

[29] M. Piotte and M. Chabbert. The Pragmatic Theory solution to the Netflix Grand Prize. *Working Paper*, (August):1–92, 2009. (Cited in Page 11)

[30] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, J. G. Riedl, and An. Open Architecture for Collaborative Filtering of Netnews. In *in Proceedings of CSCW'94 (Chapel Hill NC, October*, pages 175–186, 1994. (Cited in Page 9)

[31] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, volume 1, pages 285–295, 2001. (Cited in Pages 12, 51, 60)

[32] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '02*, volume 46, page 253, 2002. (Cited in Page 7)

[33] T. Segaran. *Programming Collective Intelligence: Building Smart Web 2.0 Applications*. 2007. (Cited in Page 44)

[34] Q. Shambour and J. Lu. A framework of hybrid recommendation system for government-to-business personalized e-services. In *ITNG2010 - 7th International Conference on Information Technology: New Generations*, pages 592–597, 2010. (Cited in Page 9)

[35] U. Shardanand and P. Maes. Social information filtering: algorithms for automating 'word of mouth'. *Proceedings of the ACM Conference on Human Factors in Computing Systems*, 1:210–217, 1995. (Cited in Page 9)

[36] Y.-Y. Shih and D.-R. Liu. Hybrid Recommendation Approaches: Collaborative Filtering via Valuable Content Information. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, volume 00, pages 1–7, 2005. (Cited in Page 9)

[37] L. Si and R. Jin. Flexible Mixture Model for Collaborative Filtering. *Machine Learninginternational Workshop*, 20(2):704, 2003. (Cited in Page 16)

[38] J. Sobecki, E. Babiak, and M. Słanina. Application of Hybrid Recommendation in Web-Based Cooking Assistant. *Knowledge-Based Intelligent Information and Engineering Systems: Lecture Notes in Computer Science*, 4253:797–804, 2006. (Cited in Page 10)

[39] T. Tang and G. McCalla. Smart Recommendation for an evolving E-learning system: Architecture and experiment. *Workshop on Technologies for Electronic Documents for Supporting Learning, International Conference on Artificial Intelligence in Education (AIED 2003)*, 2003. (Cited in Page 10)

[40] T. Tang and G. McCalla. Beyond Learners' Interest: Personalized Paper Recommendation Based on Their Pedagogical Features for an E-learning System. *PRICAI 2004 Trends in Artificial Intelligence*, 3157:301–310, 2004. (Cited in Pages 9, 10)

[41] S. Upendra. Social information filtering for music recommendation. *SM Thesis, Program in Media Arts and Sciences, Massachusetts Institute of Technology*, 1994. (Cited in Page 9)

[42] C. Wang. The research of recommendation system based on Hadoop cloud platform. In *2014 9th International Conference on Computer Science & Education*, number Iccse, pages 193–196, 2014. (Cited in Page 11)

[43] Y. Wang, L. Chen, and J. Wu. Content-boosted maximum margin matrix factorization for Flickr group recommendation. In *Proceedings of the 2103 workshop on Data-driven user behavioral modelling and mining from social media - DUBMOD '13*, pages 13–16, 2013. (Cited in Page 10)

[44] Y. Zhang, W. Chen, and Z. Yin. Collaborative filtering with social regularization for TV program recommendation. *Knowledge-Based Systems*, 54:310–317, 2013. (Cited in Page 10)

[45] Z.-D. Zhao and M.-S. Shang. User-Based Collaborative-Filtering Recommendation Algorithms on Hadoop. In *2010 Third International Conference on Knowledge Discovery and Data Mining*, pages 478–481, 2010. (Cited in Page 11)

APPENDIX A

JAVA CODE FOR NORMALIZATION FOR METU ELECTIVE COURSE DATASET

```
public static String[] normalize(String[] list){
  String[] normalizedList = new String[list.length];

  int max = 0;

  //find max value
  for(int i=0; i<list.length ; i++){
    if(list[i].length() == 0){
      continue;
    }
    if(Integer.parseInt(list[i])>max){
      max = Integer.parseInt(list[i]);
    }
  }
  double normalization_coefficient = 100.0 / max;

  //multiply all values with normalization coefficient
  for(int i=0; i<list.length ; i++){
    if(list[i].length() == 0){
      normalizedList[i] = "";
      continue;
    } else if(list[i].equals("0")){
      normalizedList[i] = "0";
      continue;
    }
```

```
26      double result = ((int)((double)((Integer.parseInt(list[i]) *
            normalization_coefficient)) * 100.0)) / 100.0;
27      normalizedList[i] = result + "";
28    }
29    return normalizedList;
30  }
```

APPENDIX B

MAHOUT JAVA CODE FOR MOVIE RECOMMENDATION ON MOVIELENS

```
1  package mahoutOnMovieLens;
2
3  import java.io.BufferedReader;
4  import java.io.File;
5  import java.io.FileNotFoundException;
6  import java.io.FileReader;
7  import java.io.IOException;
8  import java.util.HashMap;
9  import java.util.Iterator;
10 import java.util.List;
11 import java.util.Map;
12
13 import org.apache.mahout.cf.taste.common.TasteException;
14 import org.apache.mahout.cf.taste.impl.model.file.FileDataModel;
15 import org.apache.mahout.cf.taste.impl.neighborhood.
       NearestNUserNeighborhood;
16 import org.apache.mahout.cf.taste.impl.neighborhood.
       ThresholdUserNeighborhood;
17 import org.apache.mahout.cf.taste.impl.recommender.
       CachingRecommender;
18 import org.apache.mahout.cf.taste.impl.recommender.
       GenericUserBasedRecommender;
19 import org.apache.mahout.cf.taste.model.DataModel;
20 import org.apache.mahout.cf.taste.neighborhood.UserNeighborhood;
21 import org.apache.mahout.cf.taste.recommender.RecommendedItem;
```

```
22 import org.apache.mahout.cf.taste.recommender.Recommender;
23 import org.apache.mahout.cf.taste.recommender.UserBasedRecommender;
24 import org.apache.mahout.cf.taste.similarity.UserSimilarity;
25 import org.apache.mahout.cf.taste.impl.similarity.
       PearsonCorrelationSimilarity;
26 /**
27  * This program recommends movies to user 1 in MovieLens 1M Dataset
28  * by doing Collaborative Filtering
29  * @author Seval Capraz
30  *
31  */
32 public class Main {
33
34   static Map<Long, String> mapA;
35   static Map<Long, String> mapOfUser;
36
37   public static void main(String[] args) throws
         FileNotFoundException,
38       TasteException {
39
40     mapA = new HashMap<Long, String>();
41
42     Main.loadMovieList("moviesForMahout.dat");
43     printUserMovieNames();
44
45     DataModel model;
46     try {
47       model = new FileDataModel(new File("ratingsForMahout.dat"));
48
49       System.out.println("\n##### 
             NearestNUserNeighborhoodRecommendation #####");
50       NearestNUserNeighborhoodRecommendation(model);
51
52       System.out.println("\n###### ThresholdUserNeighborhood ######")
             ;
53       ThresholdUserNeighborhood(model);
54
```

```
55          System.out.println("\nThe␣program␣is␣run␣successfully.");
56
57      } catch (IOException e) {
58          e.printStackTrace();
59      }
60
61  }
62
63  public static void NearestNUserNeighborhoodRecommendation(
        DataModel model)
64          throws TasteException {
65      UserSimilarity userSimilarity = new
            PearsonCorrelationSimilarity(model);
66      UserNeighborhood neighborhood = new NearestNUserNeighborhood
            (10,
67              userSimilarity, model);
68      Recommender recommender = new GenericUserBasedRecommender(model
            ,
69              neighborhood, userSimilarity);
70      Recommender cachingRecommender = new CachingRecommender(
            recommender);
71
72      List<RecommendedItem> recommendations = cachingRecommender.
            recommend(1,
73              10);
74      int num = 0;
75      for (RecommendedItem recommendedItem : recommendations) {
76          System.out.println(num + "␣:␣" + recommendedItem);
77          System.out.println("␣␣␣␣␣"
78                  + Main.findMovieName(recommendedItem.getItemID()));
79          num++;
80      }
81  }
82
83  public static void ThresholdUserNeighborhood(DataModel model)
84          throws TasteException {
```

```
85      UserSimilarity userSimilarity = new
            PearsonCorrelationSimilarity(model);
86      UserNeighborhood neighborhood = new ThresholdUserNeighborhood
            (0.1,
87           userSimilarity, model);
88      UserBasedRecommender recommender = new
            GenericUserBasedRecommender(
89           model, neighborhood, userSimilarity);
90
91      CachingRecommender cachingRecommender = new CachingRecommender(
92           recommender);
93
94      List<RecommendedItem> recommendations = cachingRecommender.
            recommend(1,
95           10);
96      int num = 0;
97      for (RecommendedItem recommendedItem : recommendations) {
98        System.out.println(num + "␣:␣" + recommendedItem);
99        System.out.println("␣␣␣␣␣"
100           + Main.findMovieName(recommendedItem.getItemID()));
101       num++;
102     }
103   }
104
105   public static String findMovieName(Long itemId) {
106     return mapA.get(itemId);
107   }
108
109   public static void loadMovieList(String fileName) {
110     BufferedReader in;
111     String nextline;
112
113     try {
114       in = new BufferedReader(new FileReader(fileName));
115       nextline = in.readLine(); // Read first line
116       String newL[];
117       while (nextline != null) {
```

```
118           newL = nextline.split("::");
119           mapA.put(Long.parseLong(newL[0]), newL[1] + "␣" + newL[2]);
120           nextline = in.readLine(); // Read another line
121         }
122
123         in.close(); // And close on EOF
124
125       } catch (IOException e) {
126         System.out.println("Exception" + e);
127         System.exit(0);
128       }
129     }
130
131     public static void loadMovieOfUserList(String fileName) {
132       BufferedReader in;
133       String nextline;
134
135       try {
136         in = new BufferedReader(new FileReader(fileName));
137         nextline = in.readLine(); // Read first line
138         String newL[];
139         for (int i = 1; i < 54; i++) {
140           newL = nextline.split(",");
141           mapOfUser.put(Long.parseLong(newL[1]), newL[2]);
142           nextline = in.readLine(); // Read another line
143         }
144
145         in.close(); // And close on EOF
146
147       } catch (IOException e) {
148         System.out.println("Exception" + e);
149         System.exit(0);
150       }
151     }
152
153     public static void printUserMovieNames() {
154       mapOfUser = new HashMap<Long, String>();
```

```
155     Main.loadMovieOfUserList("ratingsForMahout.dat");
156
157     Iterator it = mapOfUser.entrySet().iterator();
158     int num = 1;
159     while (it.hasNext()) {
160       Map.Entry pair = (Map.Entry) it.next();
161       System.out.println(num + ")␣"
162           + Main.findMovieName((Long) pair.getKey()) + "␣&␣"
163           + pair.getValue() + "␣\\\\␣");
164       it.remove();
165       num++;
166     }
167   }
168 }
```

APPENDIX C

MOVIE LIST OF USER 1 AND HIS SCORES

1)	Toy Story (1995) Animation|Children's|Comedy	5
2)	E.T. the Extra-Terrestrial (1982) Children's|Drama|Fantasy|Sci-Fi	4
3)	Mulan (1998) Animation|Children's	4
4)	Pleasantville (1998) Comedy	3
5)	Bambi (1942) Animation|Children's	4
6)	Star Wars: Episode IV - A New Hope (1977) Action|Adventure|Fantasy|Sci-Fi	4
7)	Gigi (1958) Musical	4
8)	Dead Poets Society (1989) Drama	4
9)	Saving Private Ryan (1998) Action|Drama|War	5
10)	Apollo 13 (1995) Drama	5
11)	Erin Brockovich (2000) Drama	4
12)	Meet Joe Black (1998) Romance	3

13) Wizard of Oz, The (1939) 4
Adventure|Children's|Drama|Musical
14) Schindler's List (1993) 5
Drama|War
15) My Fair Lady (1964) 3
Musical|Romance
16) Girl, Interrupted (1999) 4
Drama
17) Back to the Future (1985) 5
Comedy|Sci-Fi
18) Bug's Life, A (1998) 5
Animation|Children's|Comedy
19) Pocahontas (1995) 5
Animation|Children's|Musical|Romance
20) Secret Garden, The (1993) 4
Children's|Drama
21) James and the Giant Peach (1996) 3
Animation|Children's|Musical
22) Hunchback of Notre Dame, The (1996) 4
Animation|Children's|Musical
23) Run Lola Run (Lola rennt) (1998) 4
Action|Crime|Romance
24) Fargo (1996) 4
Crime|Drama|Thriller
25) Close Shave, A (1995) 3
Animation|Comedy|Thriller
26) Sound of Music, The (1965) 5
Musical
27) Ferris Bueller's Day Off (1986) 4
Comedy
28) Cinderella (1950) 5
Animation|Children's|Musical

29) Tarzan (1999) 3
Animation|Children's
30) Mary Poppins (1964) 5
Children's|Comedy|Musical
31) Dumbo (1941) 5
Animation|Children's|Musical
32) Christmas Story, A (1983) 5
Comedy|Drama
33) The Last Days of Disco (1998) 5
Drama
34) Big (1988) 4
Comedy|Fantasy
35) Miracle on 34th Street (1947) 4
Drama
36) Airplane! (1980) 4
Comedy
37) Rain Man (1988) 5
Drama
38) Ben-Hur (1959) 5
Action|Adventure|Drama
39) Titanic (1997) 4
Drama|Romance
40) Driving Miss Daisy (1989) 4
Drama
41) Ponette (1996) 4
Drama
42) Antz (1998) 4
Animation|Children's
43) One Flew Over the Cuckoo's Nest (1975) 5
Drama
44) Aladdin (1992) 4
Animation|Children's|Comedy|Musical

45) Princess Bride, The (1987) 3
Action|Adventure|Comedy|Romance

46) Toy Story 2 (1999) 4
Animation|Children's|Comedy

47) Beauty and the Beast (1991) 5
Animation|Children's|Musical

48) Snow White and the Seven Dwarfs (1937) 4
Animation|Children's|Musical

49) Hercules (1997) 4
Adventure|Animation|Children's|Comedy|Musical

50) Sixth Sense, The (1999) 4
Thriller

51) To Kill a Mockingbird (1962) 4
Drama

52) Wallace and Gromit: The Best of Aar. (1996) 3
Animation

53) Awakenings (1990) 5
Drama