

Hybrid Recommender System towards User Satisfaction

By

Raza Ul Haq

A thesis submitted to the
Faculty of Graduate and Postdoctoral studies
In partial fulfillment of the requirements for the degree of

Masters in Computer Science

Ottawa-Carleton Institute for Computer Science
School of Information Technology and Engineering
University of Ottawa
April, 2013

© Raza Ul Haq, Ottawa, Canada, 2013

Abstract

An individual's ability to locate the information they desire grows more slowly than the rate at which new information becomes available. Customers are constantly confronted with situations in which they have many options to choose from and need assistance exploring or narrowing down the possibilities. Recommender systems are one tool to help bridge this gap. There are various mechanisms being employed to create recommender systems, but the most common systems fall into two main classes: content-based and collaborative filtering systems. Content-based recommender systems match the textual information of a particular product with the textual information representing the interests of a customer. Collaborative filtering systems use patterns in customer ratings to make recommendations. Both types of recommender systems require significant data resources in the form of a customer's ratings and product features; hence they are not able to generate high quality recommendations. Hybrid mechanisms have been used by researchers to improve the performance of recommender systems where one can integrate more than one mechanism to overcome the drawbacks of an individual system.

The hybrid approach proposed in this thesis is the integration of content and context-based with collaborative filtering, since these are the most successful and widely used mechanisms. This proposed approach will look into the integration of content and context data with rating data using a different mechanism that mainly focuses on boosting a customer's trust in the recommender system. Researchers have been trying to improve system performance using hybrid approaches, but research is lacking on providing justifications for recommended products. Hence, the proposed approach will mainly focus on providing justifications for recommended products as this plays a crucial role in obtaining the satisfaction and trust of customers. A product's features and a customer's context attributes are used to provide justifications. In addition to this, the presentation mechanism needs to be very effective as it has been observed that customers trust more in a system when there are explanations on how the recommended products have been computed and presented. Finally, this proposed recommender system will allow the customer to interact with it in various ways to provide feedback on the recommendations and justifications. Overall, this integration will be very useful in achieving a stronger correlation between the customers and products. Experimental

results clearly showed that the majority of the participants prefer to have recommendations with their justifications and they received valuable recommendations on which they could trust.

Acknowledgements

I would like to thank my thesis supervisor, Dr. Thomas Tran, for his constant guidance, motivation, and support throughout my thesis writing. I would also like to extend my gratitude to all my friends who helped me succeed especially Kazi Masudul Alam. Finally, I would also like to acknowledge constant encouragement and appreciation from my family members throughout the whole program.

Table of Contents

Abstract	ii
Acknowledgments	iv
Table of Contents	v
List of Figures	vii
List of Tables	viii
List of Acronyms	ix
Chapter 1. Introduction	1
1.1. <i>Information Overload</i>	1
1.2. <i>Recommender Systems</i>	2
1.3. <i>Challenges and Problems</i>	5
1.4. <i>Motivation</i>	7
1.5. <i>Objectives and Contributions</i>	8
Chapter 2. Background and Related Work	10
2.1. <i>Input Data</i>	10
2.2. <i>Recommendation Techniques</i>	10
2.2.1 <i>Content-Based Filtering</i>	11
2.2.2 <i>Collaborative Filtering</i>	12
2.2.3 <i>Context-Based Filtering</i>	18
2.2.4 <i>Demographic Filtering</i>	18
2.2.5 <i>Knowledge-Based Filtering</i>	19
2.2.6 <i>Hybrid Filtering</i>	19
2.3. <i>Generation of Recommendation or Prediction</i>	22
2.4. <i>Related Work</i>	23
2.5. <i>Trust on Recommender Systems</i>	33
2.6. <i>Explanation Interfaces of Recommender Systems</i>	36
Chapter 3. The Proposed Model	39
3.1. <i>Hybrid Approach</i>	39
3.2. <i>Design of Proposed Model</i>	40
3.2.1 <i>New Customer Profile</i>	42
3.2.2 <i>Collaborative Filtering</i>	42

3.2.3	<i>Explanation of Recommendations</i>	46
3.2.3.1	<i>Customer's Feature and Preference Profiles</i>	47
3.2.3.2	<i>Justification Style</i>	51
3.2.3.3	<i>Explanation Interface</i>	51
Chapter 4.	System Implementation	55
4.1.	<i>System Architecture</i>	55
4.1.1	<i>Technologies and Tools</i>	56
4.1.2	<i>Database Design</i>	57
4.1.3	<i>System Performance</i>	59
4.1.4	<i>System Configuration</i>	60
4.2.	<i>System Implementation</i>	60
4.2.1	<i>Web Pages</i>	61
4.2.2	<i>JAVA Package</i>	62
4.2.3	<i>Screenshots of the Application</i>	65
4.2.4	<i>Data Flow Diagrams</i>	72
Chapter 5.	Experimental Evaluation.....	75
5.1.	<i>Dataset</i>	75
5.2.	<i>Evaluation Metrics</i>	75
5.3.	<i>Experimental Procedure</i>	79
5.3.1	<i>Evaluation Measures</i>	79
5.3.2	<i>Participants and Material</i>	81
5.3.3	<i>Experimental Platform</i>	82
5.4.	<i>Experiment Results</i>	82
5.5.	<i>Limitations of Evaluatoin Approach</i>	87
5.6.	<i>Discussion</i>	87
5.6.1	<i>Comparison with Related Work</i>	87
Chapter 6.	Conclusions and Future Research Directions.....	90
6.1.	<i>Conclusions</i>	90
6.2.	<i>Future Research Directions</i>	91
References		93
Appendices		96
Appendix A –	<i>Web Pages</i>	96
Appendix B –	<i>Java Package</i>	104

List of Figures

Figure 1: Amazon shopping cart recommendations

Figure 2: Design of the Proposed Model

Figure 3: Customer Product Matrix

Figure 4: Product Feature Matrix

Figure 5: User-Item Matrix

Figure 6: Customer Profile Matrix

Figure 7: Design of the Proposed Explanation Interface

Figure 8: Three Tier Architecture

Figure 9: Entity Relationship Model

Figure 10: Structure of Pages and Class Packages

Figure 11: Home Page of Prototype Application

Figure 12: New Customer Registration

Figure 13: Rating Form

Figure 14: Explanation Interface of Recommender System

Figure 15: Recommendation Process of Old Customer

Figure 16: Recommendation Process of New Customer

Figure 17: Evaluation Measure's Results

Figure 18: Surveys Results

List of Tables

Table 1: Aims for explanation

Table 2: Survey Questions

Table 3: Survey's Results [1]

Table 3: Survey's Results [2]

Table 4: Comparison Results

List of Acronyms

Acronym	Definition
IR	Information Retrieval
TF-IDF	Term Frequency- Inverse Document Frequency
CF	Collaborative Filtering
CB	Content-Based
IIA	Interactive Interface Agent
EPSS	Electronic Performance Support System
ERP	Enterprise Resource Planning
HRMS	Human Resources Managements System
DMS	Document Management system
LMS	Learning Management system
MAE	Mean Absolute Error
GUI	Graphical User Interface
IMDB	Internet Movie Database
JSP	Java Server Pages
JDBC	Java Database Connectivity
DDL	Data Definition Language
DML	Data Manipulation Language
ROC	Receiver Operating Characteristic

Chapter 1: Introduction

The rapid growth of the internet has resulted in a huge amount of online content including various types of products that can be purchased and sold on the Internet. According to some studies, there are more than 10 billion uniquely pages on the web. In addition to this, six terabytes of new content is added to the web every month [1]. While the web grows rapidly, the information it contains is also updated constantly. However, this information is not always well organized. Web resources are published and distributed in various ways, and there is no specific mechanism to find the existing web resources [2].

1.1. Information Overload

When searching for information about a specific topic on the internet, there is a large amount of information available that is very difficult to filter through. For example most search engines on the internet give hundreds to thousands of results on every query, while only a few of those results are really relevant to the desired search. As the web is rapidly increasing and changing, the web user community is also becoming very diverse. Users have different backgrounds and preferences. As a result, users have a hard time finding the right information at the right time, and it has been shown that too many choices make users unable to handle the amount of information and also lead to anxiety generated by worrying about whether something important is being missed. This crucial problem is often referred to as information overload. Information overload refers to the state of having too much information to make a decision about a topic [2, 3].

Generally, there are three approaches that are being used to find items: the retrieval approach, the filtering approach and the browsing approach. The retrieval approach is also called information retrieval, where the user makes a request to the information retrieval system with a query and then the information is provided using an information retrieval method. Information systems that use the filtering approach are designed to sort through large volumes of dynamically generated information. They provide useful resources to the users that are likely to satisfy his or her desired information [4]. The goal of an information filtering system is to boost the user's ability to identify useful information sources. This can

be accomplished by automatically selecting which sources of information to show. Furthermore, it has been shown that user satisfaction can be enhanced in interactive applications by using the techniques which exploit the strengths of both humans and machines [4]. Browsing is a process in which query formulation is tightly integrated with results presentation. In browsing, the user does not know exactly what he is looking for; hence, he explores all the available information and tries to find something according to his desired purpose. On the web, pages of information are linked together. Users browse through this huge collection of information by following these links. With browsing, they implicitly formulate what they are looking for by clicking on these links [4].

1.2. Recommender Systems

E-commerce companies have started offering a large quantity of various products and services to their customers. Hence, it becomes very difficult for customers to choose a product from amongst the overwhelming sets of available options. Most of the customers complain that the large amount of information that a customer has to browse to find the desired product makes the process harder and time consuming. Therefore, e-commerce companies have developed tools to help customers search for the most suitable product that meets their needs. The most successful and widely used tools in this area have been the recommender systems.

Recommender systems can make recommendations based on the top sellers of a specific website, the demographic information of consumers, or based on the previous purchase history of the customers. The output of a recommender system includes suggesting products to a consumer, providing personalized product information and providing community opinion and critiques [5]. Information filtering systems are also often used on e-commerce sites to help customers find specific products in which they are interested. These systems have some features in common with recommender systems, in that both systems provide lists of suggestions for a customer. Recommender systems gather various types of information about a customer's tastes, suggestions, preferences, and with a recommendation method, give recommendations that help customers find the most suitable products for them. Mainly, recommender systems are used to recommend various products in online stores. For

example, when a customer wants to buy a book “Creating a Web Site: The Missing Manual” through a website such as Amazon.com, the website can use a recommender system to provide recommendations like “Customers who bought this book also bought “JavaScript: The Missing Manual”. The figure below shows recommendations that offer customers product suggestions based on the products in their shopping cart [6].



Figure 1: Amazon shopping cart recommendations [6]

Most websites also show customer reviews and ratings for a particular product. A recommender system collects customer ratings and creates a profile of the customer. Customers can give explicit ratings or they may be inferred implicitly from the customer’s actions. For example, in [7], the authors designed an interface to help customers distinguish interesting web pages on a particular topic from the uninteresting ones. The customers can click on a ‘thumbs up’ symbol when visiting a web site they like and a ‘thumbs down’ symbol when visiting a web site they do not like. Once the feedback has been provided from the customer, a recommender system can make better recommendations for the customer [7].

A recommender system does not always result in a list of recommendations, i.e, a list of only products that the customer will find interesting; it can also provide other means of guiding a customer to those products that are of interest to him; e.g, highlighting interesting products, etc. The main concept here is ‘individualized’ or ‘personalized’: every customer gets recommendations or is guided to interesting or useful products in a way that best suits that customer based on his/her interest profile, his/her preferences and the product’s features [8]. Generally, recommendations can be provided to the customers in a pull-based and push-based manner. In pull-based recommenders, customers explicitly make a request for

recommendations since they control when recommendations are displayed. Recommender systems inform the customer that recommendations are available, but do not display recommendations until he or she wants. This request may appear in different ways, such as a request to evaluate a specific product, a request to find a gift, or a request for recommendations in a category. Most applications used pull delivery, because recommendation computation is expensive and could slow down the interactivity of a website. Currently, it is considered a choice for each application [5]. In push-based recommenders, recommendations are sent to customers without their specific request (although subscription may be required in some systems). It has the benefit of reaching out to a customer when the customer is not currently interacting with the seller. In e-commerce applications, e-mail is the most commonly used push technology for delivering recommendations. Sending recommendations, and perhaps promotional offers, invites the customer to return to the seller. Indeed, today's technology allows customers to click on a link in the e-mail message and to be taken directly to the recommended products on-line [5].

Recommender systems can also help customers in navigating through a complex product space. These are called conversational recommender systems. In conversational recommender systems, customers interact more closely with a recommender by providing feedback (critiques) on a recommended product. They work well for recommending products that customers rarely need to buy and where the customer needs to choose one product; for example a house and car [9].

Non-personalized recommender systems recommend products to customers based on what other customers have mentioned about the products. These recommendations are completely independent of the customer, so each customer receives the same recommendations [10]. They are considered automatic, because they need only minor customer effort to generate a recommendation. Non-personalized recommender systems are common in physical stores, since they can be set up on a display that is viewed without being personalized for a customer. For example, the average customer ratings displayed by Amazon.com and Moviefinder.com are non-personalized recommendations. Recommenders systems can also provide personalize recommendations to the customer because they are based on real input

from the customer. Whenever the customer buys or reviews a new product, a new recommendation list is created for that particular customer [9, 10]. They are also useful for retail companies to help their customers in finding the desired product. However, sometimes recommendations generated by these systems are not satisfactory. Therefore, in last few years, various extensions to recommender systems have been proposed that enhance recommendation capabilities and make recommender systems applicable to an even broader range of applications.

1.3. Challenges and problems

The rapid growth in the amount of available information and the numbers of web users produces the following crucial challenges for recommender systems.

Sparsity / Quality of recommendation

Collaborative filtering recommender systems produce high quality recommendations compared to other recommender systems. They collect ratings of products from many customers and generate recommendations based on those ratings to a given customer. In collaborative filtering, the process of comparing two customers with the goal of computing their similarity involves comparing the ratings they provided to products. In order to compare the ratings, it is important that the two customers rated at least some products in common. However in the case of movies or books, the number of products is very large while the number of products rated by every single customer is in general small. This means that it is very unlikely two random customers have rated any products in common and hence they are not comparable. Therefore the probability of finding a set of customers with significantly similar ratings is usually low. This creates an inability to find successful neighbors and finally produces weak recommendations [11, 12].

Scalability

Collaborative filtering algorithms are able to search tens of thousands of neighbors in real-time, but the demands of modern E-commerce systems are to search tens of millions of potential neighbors. Hence, there is a need to improve the scalability of the collaborative algorithms [13]. Scalability means how quickly a recommender system can generate recommendations. Recommender systems mostly have a huge dataset, therefore, the

algorithm needs to maintain its speed and work properly. Response time is very crucial factor to evaluate the performance of recommender systems. Thus, it is very important to implement efficient algorithms, which are capable of handling this situation very efficiently [11, 14].

New product / customer problem

A Recommender system can only work efficiently after a large collection of ratings has been obtained. When a new customer starts with no profile, a training period is required to create a profile before the recommender system can make recommendations. As recommendations follow from a comparison between the target user and other users based only on the accumulation of ratings, a user with few ratings becomes difficult to categorize [15]. This is called the “start-up” or “ramp-up” problem. Hence, in order to generate quality recommendations, a recommender system needs a customer’s preference profile. [11, 14, 16].

Gray Sheep problem

The gray sheep problem refers to individuals with opinions that are “unusual”, meaning that they do not agree or disagree consistently with any type or group of customers. They have low correlation coefficients with other customers, as they partially agree or disagree with other customers. Hence they may not receive accurate recommendations and they may negatively affect the recommendations of the rest of community [17].

Justifications

E-commerce is a field that influences people’s activities. It allows customers to write reviews on products, which other customers read and trust thereby influencing them to purchase a certain product. Due to the lack of face-to-face interaction with customers in online shopping, a customer's actions undertake a higher degree of uncertainty and risk compared with traditional shopping. Therefore, customers need recommendations in which they can trust. In order to obtain customer trust, a recommender system needs to provide justifications for recommended products. Trust creates a long-term relationship between a customer and an organization. Additionally, it has been shown that a customer’s trust is positively associated with a customer’s intentions to purchase a product and return to the website [18-20].

From the above-mentioned challenges, this proposed approach will mainly focus on providing valuable recommendations and knowledgeable explanations as justifications for recommended products. Furthermore, this approach will create feature and preference profiles to generate recommendations for the existing and new customers. Finally, this proposed recommender system will adopt a structured presentation mechanism that will display the recommendations with justifications for the customer.

1.4. Motivation

A crucial problem in e-commerce is the vast amount of information and its accessibility. Many web customers complain that the information about a specific product that a specific customer needs to find makes the process very complex and time consuming. The time and cost which is required for searching for a specific product can be greater than obtaining the product without e-commerce. Therefore, researchers have been focusing on browsing for the required product in the least time and for the best price. Recommender systems have been utilized in e-commerce sites to recommend products to their customers. These products can be recommended based on the demographic information of customers or based on the past purchasing history of customers [10]. However, the recommendations generated by these systems do not always meet the customer's requirements. Consequently, researchers have been working to improve the predictions or suggestions generated by these systems.

Various filtering mechanisms are being used to produce recommendations for customers. Each of these mechanisms however, has some limitations. Therefore, researchers have started to integrate more than one technique to provide accurate recommendations for customers. It has been shown by researchers that hybrid techniques can perform better than any individual technique. This motivated me to work on hybrid techniques in order to generate good quality recommendations so that customers may feel confident when making online decisions. In order to make online decisions it is very important to generate recommendations with useful justifications that will give the reasoning behind these recommendations to customers. Furthermore, the presentation of these recommendations is important to minimize the customer's decision making time.

1.5. Objectives and Contributions

Recommender systems have become popular on the web because the objective of such systems is to customize the information that they gather according to the customer's tastes and preferences. Furthermore, personal recommender systems help customers with specific products by predicting the customer's interest in that product [15]. Hybrid mechanisms have been a popular research area and have shown better results than using any individual technique. However, hybrid mechanisms are lacking in providing useful justifications for their recommended products.

- My thesis will mainly focus on providing valuable recommendations and various forms of justifications for recommended products as this plays a crucial role in obtaining the satisfaction and trust of customers. My goal is to produce useful explanations that make it faster for customers to decide which recommended product is best or suitable for them.
- An attractive and organized explanation interface that will reduce the customer's decision making time and provide more confidence was also created. It has been noticed by various researchers that the presentation mechanism to display the recommendations attracts the customer more effectively and boosts the trust on the recommender system [21-23].
- The proposed recommender system will allow the customer to interact with it in order to provide feedback on the recommendations. This feedback can be provided by giving a rating on recommended products and also by modifying the previous ratings. In addition to this, customers can make a suggestion on the justifications that are based on product's features in order to revise the recommendations.
- A prototype application that implements the proposed approach was also developed. Hence, movies are being used as an example of products in this thesis. This prototype is a web based application that recommends movies to customers and provides justifications for its recommendations. This application has a categorized

explanation interface that mainly boosts the trust of customers and helps them make a decision. This prototype implementation currently works only for movies; however it can be easily modified for a wide range of products such as books, cameras, etc.

Chapter 2: Background and Related Work

Usually, each recommender system consists of three steps. These are gathering input data, computing the recommendations using different recommendation techniques, and giving results to the customer. The following terms that will be used throughout this thesis.

Active Customer – The customer for whom the system would generate recommendations.

Target Product – The product for which the system would predict a rating value.

2.1. Input Data

Each recommendation technique requires some input upon which to base its recommendations. Usually this input is provided by the customer. However, it is possible that the input may also be provided by the business. Various ways exist to gather input data and some of the common ways are listed below [5, 10].

- Customer information – This includes personal information such as name, gender, personal interests and preferences.
- Customer history – Recommender systems can store the preferences of their customers by observing their transaction histories or browsing activities.
- Product ratings – This is mostly used in collaborative-filtering recommender systems whereby customers are requested to rate some products to indicate their preferences.

2.2. Recommendation Techniques

Every recommender system employs a technique to find the required products for their current customer (the active user). There are various techniques that have been applied to the basic problem of making accurate and efficient recommender systems. Recommendation techniques often work in the following challenging environments [6]:

- A retailer has large amounts of data, tens of millions of customers and millions of distinct catalog products.
- New customers typically have extremely limited information, based on only a few purchases or product ratings.

- Older customers have a greater amount of information, based on thousands of purchases and ratings.
- Customer data is volatile: Each interaction provides valuable customer data, and the algorithm should work accordingly.

There are various techniques that have been implemented to generate useful recommendations for customers. These are mainly categorized into various types: content-based filtering, collaborative filtering, context-based filtering, demographic filtering, knowledge-based filtering and hybrid filtering. These are described in detail below.

2.2.1. Content-Based Filtering

Content-based recommender systems match the textual information of a particular product with the textual information representing the interests of a customer. For example, a movie recommender that employs a content-based approach might use the metadata of the movie, e.g, title, genre, etc., or reviews by other customers to make recommendations [12]. It recommends a product to a customer based upon a description of the product and the profile of a customer's interests. It is mainly based on the contents of the products and transaction history of the products that a customer has purchased before [16]. Furthermore, it also uses attribute-based filtering that is purely based on the syntactic properties of products and a customer's interests.

Information retrieval (IR) and machine learning mechanisms have been used in content-based recommender systems. One of the most widely used IR mechanisms is the assignment of weights to keywords. Term frequency-inverse document frequency (TF-IDF) is used to assign weights to keywords [21]. This integrates two measures, the term frequency and the inverse document frequency. The vector representation of a document 'd' contains the TF-IDF value for each term in the document 'd'. Cosine similarity can be used to calculate the similarity between two or more documents. The term frequency (TF) assigns a weight to a term based on how many times the term 't' is available in a document 'd'. The inverse document frequency (IDF) provides the importance of the least occurring terms. It shows that the IDF of a rare term is high and the IDF of a frequent term is low.

Other mechanisms used for content-based recommendations include probabilistic models, such as Bayesian classifiers, and machine learning mechanisms like artificial neural networks. These mechanisms give predictions by learning the underlying model with statistical analysis and machine learning mechanisms. The full details of the content-based mechanisms are beyond the scope of this thesis; the interested reader is referred to Manning et al., for more information [24].

The following are the limitations of this technique [16]:

- In order to have a good profile for a customer's interests, they have to rate a sufficient number of products before a recommender system can understand the user's preferences and give recommendations to the customer.
- Since this technique is purely based on content, if two different products have similar contents then the recommendation system is not able to differentiate between them.
- The TF-IDF approach is not suitable with synonym words, for example the words "automobile" and "car" are not considered the same, even though they have the same meaning.

2.2.2. Collaborative Filtering

Collaborative recommender systems are mainly based on identification of other customers with similar tastes. Generally, it collects ratings of various products from many individuals and then gives recommendations based on those ratings to a given customer [16]. The typical input of collaborative recommender systems is represented as a matrix of ratings, in which the customers are represented as rows, the products are represented as columns and the values in the cells represent the rating given by a customer about the product. Each entry in the matrix denotes the opinion that a customer has about the product as a numeric value. These ratings can either be the explicit ratings or implicit ratings deduced from the customer's reviews, purchase history or browsing history. Many entries in the matrix will be empty, as a customer is not aware of all the products in the system and hence might not have rated all the products or simply might have chosen not to rate a product.

This assumes that like minded people have almost similar preferences. For example, if Tom and Ron have liked many of the same books and Tom liked “Creating a Web Site: The Missing Manual”, which Ron has not read yet, then the system can recommend this same book to Ron. Researchers have categorized a number of collaborative filtering algorithms into two types, memory-based (user-based) and model-based (item-based).

Memory-based algorithms check every other customer and whether they are the active customer’s neighbor, which is determined by the personal correlation coefficient. Once a neighborhood of customers is found, various algorithms are used to combine the preferences of neighbors to generate a prediction or top N-recommendations for an active customer. The prediction of the product, which the active customer has not rated, is calculated as an aggregation of all ratings of a customer’s neighbor for the same product. These techniques are known as customer-based collaborative filtering [11, 21]. However, scalability is the major issue of user-based collaborative filtering: It is very hard to calculate a user’s similarity when the number of users gets too large. It works well only with a small dataset. Model-based algorithms generate product recommendations by creating a model of a customer’s ratings. It uses the ratings to learn a model, which is then used to create the ratings for those products which are not rated by active customers. This model can be created by various machine learning techniques such as clustering and rule-based approaches [25]. A clustering model considers collaborative filtering that works by clustering similar customers in the same class/cluster and estimating the probability that a particular customer is in a particular class, and then calculates the conditional probability of ratings.

Item-based is another collaborative filtering algorithm that is based on item/product relations and not on customer relations. It builds the model of item/ product similarities and consists of following three main steps [14]:

- First, all the products rated by an active customer are retrieved.

- Second, the item-based approach looks into the retrieved products from step 1 and determines how similar they are to target product ‘i’ and then selects the ‘K’ most similar products. In addition, their corresponding similarities are also determined.
- Finally, after finding the most similar items, the prediction is then determined by taking a weighted average of the active users rating on these similar products ‘K’.

Neighborhood Formation

Neighborhood formation is crucial to find the similarities between customers in the customer-product matrix. It is implemented in two steps: initially, the similarities between all the customers in a customer-product matrix are calculated with the help of the proximity metrics. The second step is the actual neighborhood generation for the active customer, where the similarities of customers are processed in order to select those customers from whom the neighborhood of the active customer will consist [11, 18, 23].

Similarity of Customers / Products

Pearson correlation similarity or cosine / vector similarity metrics have been used to find the similarities between customers in the literature. The most widely used similarity measurement is Pearson Correlation Coefficient. This can be calculated using the following equation [26].

$$w_{ab} = \frac{\sum_j (r_{aj} - \bar{r}_a)(r_{bj} - \bar{r}_b)}{\sqrt{\sum_j (r_{aj} - \bar{r}_a)^2 \sum_j (r_{bj} - \bar{r}_b)^2}} \quad (1)$$

From above equation, r_{aj} indicates the rating value of customer ‘a’ on item ‘j’; \bar{r}_a indicates the mean rating value of user a. The calculation is made on the products that are rated by both customers ‘a’ and ‘b’ [21].

Pearson Correlation Coefficient has also been used to calculate the similarities between the products. In order to calculate the similarities between two products, the customers who

have rated both products first need to be isolated and then a similarity computation technique is applied.

In the case of the Pearson correlation similarity, the difference is that we are not using the ratings that two customers have given for a common product, but instead the ratings of two products ‘k’ and ‘l’, whose similarity we want to calculate, have been given by a common customer ‘u’[11]. This can be calculated using the following equation [27].

$$sim(k, l) = \frac{\sum_{u=1}^m (R_{u,k} - \bar{R}_k)(R_{u,l} - \bar{R}_l)}{\sqrt{\sum_{u=1}^m (R_{u,k} - \bar{R}_k)^2} \sqrt{\sum_{u=1}^m (R_{u,l} - \bar{R}_l)^2}} \quad (2)$$

From the above equation, $sim(k, l)$ denotes the similarity between the products ‘k’ and ‘l’; ‘m’ denotes the total number of customers, which rated both the products ‘k’ and ‘l’; \bar{R}_k, \bar{R}_l are the average ratings of the products ‘k’ and ‘l’, respectively; $R_{u,k}, R_{u,l}$ denotes the rating of customer ‘u’ on the product ‘k’ and ‘l’ respectively [27].

Cosine similarity also has been used to calculate the similarity of customers. In the cosine-based approach, each customer's ratings are modeled as a vector and the similarity between two customers is measured by the cosine of the angle between the two vectors. The similarity between the active customer ‘a’ and another customer ‘u’ is measured using their rating vectors ‘a’ and ‘u’, respectively [21]:

$$sim(\vec{a}, \vec{u}) = \frac{\vec{a} \cdot \vec{u}}{|\vec{a}| |\vec{u}|} \quad (3)$$

From the above equation, $\vec{a} \cdot \vec{u}$ denotes the dot product between the two vectors and $|\vec{a}| |\vec{u}|$ is the product between the two vectors' Euclidean lengths. The Euclidean length of a vector ‘a’ with ‘m’ components is given by [21]:

$$|\vec{a}| = \sqrt{\sum_{i=1}^m \vec{a}_i^2} \quad (4)$$

This, however, has one shortcoming. The difference in the rating scale between different customers will show in quite a different similarity. For instance, if Tom only rates a score of 4 on the best movie, never rates 5 on any movie, and rates 1 on the bad movie and, in contrast, Ron always rates according to the standard level (score of 5 on the best movie, and 2 on the bad movie). The adjusted cosine similarity has been used to consider this drawback and is specifically used for item-based collaborative filtering. This can be calculated with the following equation [16].

$$sim(k, l) = \frac{\sum_{u=1}^m (R_{u,k} - \bar{R}_u)(R_{u,l} - \bar{R}_u)}{\sqrt{\sum_{u=1}^m (R_{u,k} - \bar{R}_u)^2} \sqrt{\sum_{u=1}^m (R_{u,l} - \bar{R}_u)^2}} \quad (5)$$

From the above equation, $sim(k, l)$ denotes the similarity between the product 'k' and 'l'; 'm' denotes the total number of customers, which rated both the products 'k' and 'l'; \bar{R}_u is the average rating of the customer 'u'; $R_{u,k}, R_{u,l}$ denote the rating of customer 'u' on the product 'k' and 'l', respectively.

Once we have calculated the similarities between customers in the customer's-product matrix, the next step is to generate a prediction for that product [11]. The most common way to achieve this is through a weighted sum. The predicted rating is the weighted sum of the ratings given by other customers for that product, where the weights are the similarity coefficient of the active customer with the other customers. It clearly indicates that the rating expressed by a very similar customer has a larger influence on the rating predicted for the active customer [18]. It can be calculated using the following equation [11, 18].

$$p_{a,i} = \bar{r}_a + \frac{\sum_{u=1}^k w_{a,u} (r_{u,i} - \bar{r}_u)}{\sum_{u=1}^k w_{a,u}} \quad (6)$$

From the above equation, $p_{a,i}$ indicates the predicted rating that the active customer 'a' would possibly give for product 'i', r_u is the average of the rating provided by customer 'u', $w_{a,u}$ is the customer similarity weight of 'a' and 'u' as computed in first step and 'k' is the number of customers whose ratings of product 'i' are considered in the weighted sum [18].

Neighborhood Generation

After neighborhood formation, it is mandatory to distinguish a single customer, called the active customer. The active customer is for whom we would like to make the predictions. This can be accomplished using various approaches [11]:

Center based scheme

The center-based scheme creates a neighborhood for the active customer by selecting the row of the similarity matrix, which corresponds to the active customer [11].

Aggregate Neighborhood Scheme

The aggregate neighborhood scheme creates a neighborhood of customers, not by finding the customers who are closest to the active customer, but by collecting the customers who are closest to the centroid of the current neighborhood. It forms a neighborhood by first selecting the user who is the closest to the active customer. Those two customers will now form the current neighborhood, and the selection of the next neighbor will be based on them. This is an effective method in the case of a very sparse user-item matrix [12].

Collaborative filtering also has the following limitations:

- It can only work efficiently after a large collection of ratings. This is called the start-up or ramp-up problem. For example, if there is a new product, the recommender system would not be able to recommend it until it is rated by a number of customers, and in the same way, in order to get good recommendations, a new customer has to rate many products to create a strong profile.
- The process of comparing two customers with the goal of computing their similarity involves comparing the ratings they provided to products. In order to compare the ratings, it is important that the two customers have rated at least some products in common. It is common that in online stores even the most active customers have purchased or rated a very limited percentage of products, when compared to the available total. This leads to sparse user-item

matrices, the inability to locate successful neighbors and, finally, the generation of weak recommendations. Hence, in the case of movies or books, the number of products is very large, while the number of products rated by every single customer is in general small. This means that it is very unlikely that two random customers have rated any products in common, and hence they are not comparable [18].

- If a product is purchased often by many customers, then it will always be recommended to every user. This is called as a banana problem.
- It does not work properly when a customer's interests change [16].

2.2.3. Context-Based Filtering

Traditional recommender systems consider only the customer and product dimensions in the recommendation process. It may not be enough to consider only customers and products—it is also important to incorporate the contextual information of the customer's decision scenario into the recommendation process. For example, in the case of personalizing content on a Web site, it is important to determine what content needs to be delivered to a customer and when [28]. Context aware recommender systems predict customer's tastes and preferences by incorporating available contextual information into the recommendation process. The contextual information can be gathered explicitly from the customer, and also implicitly from the application data or environment. For example, one can obtain explicit data regarding a change in the location of the customer detected by a cellular company, and also implicitly obtained data from the timestamp of a transaction [28, 29]. Context-aware recommender systems have been used for generating recommendations for music and movies, since contextual information is very important for these two products recommendations. For a music recommender system, customers want music according to context such as an event or occasion and to an emotional state, rather than a singer or an artist [30].

2.2.4. Demographic Filtering

Demographic-based recommender systems classify customers or products based on their personal attributes and produce recommendations based on demographic classifications. Demographic data refers to information such as the age, gender, address and the education of

users. This data is normally gathered explicitly from the customer [11, 14]. Demographic characteristics of customers play a crucial role in identifying categories of customers who like a specific type of products or have similar tastes. This has been used to recommend books based on personal information and also for marketing research to suggest a range of products and services [15]. In some situations, collaborative and content-based algorithms are not able to predict ratings or recommend products to customers, specifically in the case of new customers or products [31]. Demographic techniques form “people-to-people” correlations like the collaborative technique, however, use different data. The benefit of a demographic approach is that it may not require a history of customer ratings of the type needed by collaborative and content-based techniques [15].

2.2.5. Knowledge-based Filtering

Knowledge-based recommender systems recommend the product to a customer by using the knowledge of the product domain. It collects the customer’s preferences on a specific product, and uses its knowledge to find the products according to the customer’s preferences. It does not need a rating database since its recommendations do not depend on a customer’s ratings. It can easily manage the recommendations if a customer’s interest changes, since its recommendations are independent of the user’s preferences [32]. However, in order to make useful recommendations for a customer, a system has to fully understand the product domain. It should have knowledge of all important features of the product and be able to get these features where this information is stored. The crucial task in this mechanism is the process needed for extracting the desired information and building the needed models, such as knowledge acquisition and representation [32, 33].

2.2.6. Hybrid Filtering

Collaborative, content and knowledge-based filtering techniques are widely used to build the recommendation systems for e-commerce websites. Collaborative systems are very successful due to their simplicity and better results. However, such systems have an important problem known as cold-start or ramp-up. This means that when a new customer comes, it cannot provide useful recommendations, because of the lack of required data for its recommendation computation process. In order to overcome this problem, a useful technique

is hybridizing different methods of recommendation. My research is also on these hybrid techniques. Researchers have been working on these hybrid techniques to boost the performance of recommender systems in different manners.

Hybrid filtering methods integrate more than one technique in order to avoid the limitations of individual filtering technique. Generally, they integrate collaborative and content-based methods or combine collaborative and knowledge-based methods, in order to overcome the cold-start or ramp-up problem and performance issues. The hybrid methods are detailed below [15].

Weighted hybrid:

A weighted hybrid recommender is one in which the score of a recommended item is computed from the results of all the available recommendation techniques present in the system. This integrates the scores from each technique using a linear formula. Therefore, the various techniques must be able to produce their recommendation score, which can be linearly combinable. It is very useful that all of the system's capabilities are brought to bear on the recommendation process and it is easy to perform post-hoc credit assignment and adjust the hybrid system accordingly [15].

Switching hybrid:

This uses different criteria to switch between recommendation techniques. The DailyLearner system uses a content/collaborative hybrid in which a content-based recommendation method is used first. If the content-based system cannot make a recommendation with sufficient confidence, then a collaborative recommendation is attempted. However, this creates complexity into the recommendation process since the switching criteria must be determined.

Mixed hybrid:

It is based on the merging of multiple-ranked lists into one. Each technique in this hybrid should be able to create a recommendation list with ranks, and the core algorithm merges them into a single ranked list. For example, it uses content-based techniques based on the content information of TV shows, and collaborative information about the preferences of

other users. Recommendations from the two techniques are combined together in the final recommendations. This mixed hybrid avoids the “new item” start-up problem: the content-based component can be relied on to recommend new shows on the basis of their descriptions even if they have not been rated by anyone [15].

Feature combination hybrid:

It treats collaborative information as additional feature data and uses content-based techniques over this augmented data set. It allows the system to consider collaborative data without relying on it exclusively, so it reduces the sensitivity of the system to the number of users who have rated an item.

Cascade hybrid:

It involves a staged process. In this technique, one recommendation technique is used to produce a list of recommended items, and then a second technique is used to refine the recommendation among the previous created list. The restaurant recommender EntreeC, is a cascaded knowledge-based and collaborative recommender. It uses its knowledge of restaurants to make recommendations based on the user’s stated interests [15].

Feature augmentation hybrid:

It is similar to the feature-combination hybrid, but different in that the contributor generates new features. One technique is used to produce a rating of an item and that information is then incorporated into the processing of the next recommendation technique. For example, the Libra system makes content-based recommendations of books based on data found at Amazon.com, using a naive Bayes text classifier [15]. Furthermore, it is more flexible and adds a smaller dimension than the feature-combination method. It is also attractive because it offers a way to improve the performance of a core system.

Meta Level:

It offers a way in which two recommendation techniques can be combined using the model generated by one as the input for another. The benefit of the meta-level method, especially for the content/collaborative hybrid is that the learned model is a compressed form of user’s

interest, and a collaborative mechanism that follows can operate on this information easier than on raw rating data [15].

2.3. Generation of Recommendation or Prediction

The output of a recommender system varies with product type, quantity, and display of the information provided to the customer. The most common type of output is to provide a single suggestion. The word “this” is usually used for the recommendation of a single product. With a recommendation of a single item, the seller has more chances that the customer will pay more attention to it. However, a drawback is that it places all of the risk in a single recommendation which can be rejected by the customer. Typically, recommender systems provide a set of suggestions in the form of a list for a customer in a particular context. Some web developers prefer to leave the list unordered, to avoid giving the impression that a particular recommendation is the best one [5].

Prediction Generation

The main goal of a collaborative filtering algorithm is to find a set of products for an active customer that he might like, or find a prediction value for one of the empty slots in his rating’s vector. Generally, the output of the collaborative filtering algorithm will either be a prediction value or a top-N recommendation list. The prediction value is a numerical value in the same scale as the other ratings by the active customer for a product that he has not rated before. The predicted rating indicates how much the customer may like the product [11]. These predicted ratings can help customers understand the strength of a recommendation. Predicted ratings can be displayed in the context of individual recommendations, lists of recommendations, or in the context of general item information [5].

Top-N Recommendation Generation

The top-N recommendation list is a list of products that the active customer will like the most. These products should not appear in the list of products already rated by the active customer [12].

Most-Frequent Item Recommendation

Most-frequent item recommendation searches into the neighborhood of the active customer and performs a frequency count of the products that each neighbor customer has purchased or rated. After considering all the neighbor customers and the total counts for their rated products have been calculated, the system will exclude products already rated by the active customer. It will then sort the remaining products according to their frequency counts and give the recommendation of the most frequent products for the active customer [12].

2.4. Related Work

Researchers have been working to boost the performance of recommender systems using hybrid mechanisms. Over the past years, they have found promising results with the integration of more than one technique. Various hybrid approaches such as multi-clustering, incomplete preference relation, unified approach and mobile hybrid recommender systems have been shown the good results. Below is summarized related work on hybrid recommender systems relevant to this thesis.

In [32], a hybrid architecture integrates collaborative and knowledge-based filtering techniques. This architecture has the following components:

- The interactive interface agent (IIA)
- The knowledge-based engine
- The knowledge base of the product domain
- The collaborative filtering engine
- The database of user's ratings for items
- The product database

The interactive interface agent works as a controller in this architecture. In other words, it works as an intermediary between the customer and the two recommender systems. It selects the appropriate system for the recommendation process. As stated earlier collaborative filtering is not efficient for a specific customer until a large number of customers, whose profiles are known, have rated enough products. In this architecture, the interactive interface

agent has values for these two variables. When a recommendation process runs, it compares the values of these two variables with their thresholds. If either of these variable values is less than its threshold, then the interactive interface agent selects the knowledge-based recommender system. Otherwise the collaborative filtering recommender system will be used [32].

With the selection of a knowledge-based system, first it helps the customer sign in to the system. If this is not a new customer then it forwards the user's name and password to the collaborative filtering engine, which will open the customer's profile for modification. If this is a new customer, then the IIA helps the customer sign in to the system and suggests to the user to rate some products. This rated information will help create the customer's profile in the collaborative filtering engine. After obtaining the customer's preferences, the IIA transfers it to a knowledge-based engine. The knowledge-based recommender system generates recommendations after consulting the product domain and the customer's preferences. In the end, IIA receives these recommendations and gives them to the customer. This information can be presented in a graphical user interface.

In [34], a hybrid architecture integrates collaborative and knowledge-based filtering techniques. This model is based on incomplete preference relations in a knowledge-based system. It is very useful to get rid of the cold start problem. This model consists of three phases.

- Acquiring the user preference information
- Building the user profile
- Recommendation

Acquiring the user preference information: It gathers the user's preferences in two phases:

Setting favorite examples: the user will provide two or three favorite examples and with the help of this, user can provide an incomplete preference relation as a form of one row.

Filling preference relation: This incomplete preference relation can be completed using algorithms. These algorithms try to get a preference relation with a consistency of maximum degree.

Building the user profile: The system can create the user profile with the complete preference relation and description of the items from the product database.

Recommendation: Finally, the user profile is used to search for the product according to the user's preferences.

REJA: A Restaurant recommender system

This system has been implemented for restaurants for the province Jean, in Spain. The system provides recommendations to its users about existing restaurants. It uses the collaborative approach for the restaurant database and registered customers. In this system, a customer has to be registered and provide ratings of known restaurants. This system also uses the knowledge-based approach for new customers or when the collaborative approach does not work. Knowledge-based systems use the incomplete preference relation to get the minimum amount of information about the customer and the knowledge that the system has about the restaurants [34].

In [27], the authors suggested a technique that introduces the contents of products into the product-based collaborative filtering system to improve the performance of a prediction algorithm. It is called the product-based clustering hybrid approach. In this approach, they first applied the clustering algorithm to group the products. The main purpose was to group the products into various sets and provide content-based information to determine similarities. Each product has its own attributes, such as the movie product, which may have actor, actress, director, etc. Thus, they grouped the items-based on those attributes. They implemented the K-means clustering algorithm to group the products with some adjustments.

After the implementation of the clustering algorithm, they determined the sub-similarity of the group-rating matrix, and then calculated the sub-similarity of a product-rating matrix. Finally, they determined the total similarity that is the linear combination of the above two. They implemented the Pearson correlation measures to determine the similarity that measures the degree to which a linear relationship exists between two variables. They also implemented the adjusted cosine similarity, in order to address the problem between different rating scales given by the customers. The difference in rating scales between different

customers will give different similarities. Finally, they performed a weighted average of deviations from the neighbor's mean to give a prediction for a product. This can work well in the cold start problem. In order to address this problem, they proposed two methods: one was to use the average rating of all ratings on the new product's nearest neighbors, which is inferred by the group rating matrix; the second is using a weighted sum method for prediction.

In [35], this hybrid recommender system integrates collaborative and content-based approaches. Firstly, the content-based filtering algorithm is applied to find customers, who share similar interests. Secondly, a collaborative algorithm is applied to make predictions. It integrates the product information and product ratings to calculate the product - product similarity, called product-based clustering method. It also integrates a customer's information and a customer's ratings to calculate the customer - customer similarity, called customer-based clustering method.

Hybrid filtering approach

The hybrid filtering approach is classified into two parts: an off-line and an online module. The off-line module is a batch processing unit that runs periodically. It further consists of two modules: the training and the clustering module. In the training module, it creates new rating data and stores them in a database. This new rating data is determined using a collaborative prediction to fulfill the sparse customer-rating matrix. In the clustering module, it performs clustering again on the new rating matrix, in order to find a group of like-minded customers. This group is small in size compared to the original set, thus making the technique scalable.

The online module further consists of three modules; registration, rating and recommendation module. The registration module gathers a customer's information such as sex, age, occupation, education, address, postal code, etc. This information is gathered by the customer in a web form at the time of registration. This module also gathers a product's information that is recorded by an administrator. On the other hand, the rating module gathers a customer's ratings of products in a rating database. The customer's ratings range starts from zero to five stars. Zero stars denote extreme dislike for a product, whereas five

stars denote high praise. The recommendation module searches for similarities between an active customer and a group of like-minded customers that it obtained from the offline component. They used the preferences of like-minded customers to recommend Top-N recommended products to the current active customers by using the Pearson's correlation coefficient algorithm.

In [12], the authors proposed a content-based predictor to enhance existing user data, and then generated personalized suggestions through collaborative filtering. They implemented a bag-of-words naive Bayesian text classifier extended to handle a vector of bags of words; where each bag-of-words corresponds to a movie-feature such as actor and director. Furthermore, they used the classifier to learn a user's profile from a set of rated movies. The learned profile is then used to predict the rating of unrated movies. User-based collaborative filtering is used to create the neighborhood with the Pearson correlation algorithm.

In this approach, the authors created a pseudo user-ratings vector for every user 'u' in the database. The pseudo user-ratings vector 'vu' consists of the item ratings provided by the user 'u', where available, and those predicted by the content-based predictor. The pseudo user-ratings vectors of all the users provide the dense pseudo rating's matrix 'V'. Finally, they performed collaborative filtering using this dense matrix.

Weighted Harmonic Mean

The accuracy of a pseudo user-ratings vector depends on the number of products that have been rated. If a customer rates many products, the content-based predictions will be a lot better, and additionally the pseudo user-ratings vector will be accurate. On the other hand, if a customer rates only a few products, then the pseudo user-ratings vector will not be as accurate. It has been observed that the inaccuracies in the pseudo user-ratings vector sometimes produce wrong correlations between the active customer and their neighbors. Therefore, in order to make a strong correlation, the authors used a weighted harmonic mean factor in this approach [13].

$$\begin{aligned}
hm_{i,j} &= \frac{2m_i m_j}{m_i + m_j} \\
m_i &= \begin{cases} \frac{n_i}{50} : \text{if } n_i < 50 \\ 1 : \text{otherwise} \end{cases}
\end{aligned} \tag{7}$$

In the above equation, ‘ n_i ’ refers to the number of products that customer ‘ i ’ has rated. The harmonic mean tends the weight towards the lower of the two correlated values – ‘ m_i ’ and ‘ m_j ’. Thus, the correlation between a customer’s rating profiles, with at least 50 customers rated products each, will receive the highest weight regardless of the actual number of movies each customer rated. On the other hand, even if one of the customer’s rating profile is based on less than 50 customer-rated products, the correlation will be devalued appropriately [12].

In [36], the authors proposed moreTourism, a hybrid recommender system that provides valuable information about tourist resources depending on the customer’s profile, location, schedule and the amount of time available for visiting their favorite places. Mobile technology has evolved to offer valuable services for computation and connectivity. For this reason, the authors suggested a platform that helps customers make decisions based on their location, timetable, context and mobility needs. This proposed system consists of two components – the smart phone and a server. A smart phone is a device that connects the user with the system; the user can perform all the actions at any time. A server provides various functionalities to the user such as presentation, recommendation, personalization, socialization and advertising.

This proposed approach integrates collaborative filtering with content-based approaches. Furthermore, these approaches are improved by social recommendations, taking into account the tags provided by customers. These tags are used to create the customer’s tag cloud with all the tags provided by the customer, weighted by the ratings and the attraction tag cloud with all the tags that the customers have provided. In addition to that, a folksonomy is created as an undirected graph that reflects the relationships among tags.

The social content-based approach computes the recommendations by comparing the customer tag cloud with the attraction tag cloud considering not only the coincident tags in both clouds, but also the relationships between tags reflected in the folksonomy. The social collaborative filtering approach creates for each attraction a new tag cloud from the tag clouds of those customers who liked it. In this manner, they achieve the target customer tag cloud, which is shaped to the customers who may like this attraction. Then, the tag cloud of the customer who wants to receive recommendations is compared to the target customer tag cloud taking into account the relationships between tags by the folksonomy. This proposed moreTourism service is currently being developed for Android Dev Phone 1 terminal with SDK 2.1 and it has been successfully tested. This mobile application proved very useful for the customer for rating tourist resources and achieving recommendations.

In [14], the authors proposed a unique cascading hybrid recommendation approach by integrating the customer's rating, a product's feature, and demographic information about the products. This approach builds product models based on a customer's rating, a product's feature, and demographic information about the products.

Proposed Algorithm

This proposed algorithm consists of the following three steps: first, the similarity between products using rating data, demographic data, and feature data is computed and stored. Furthermore, adjusted cosine similarity between two products is used for measuring the similarity over rating data, and vector similarity between two products is used for measuring the similarity using demographic and feature vectors.

Boosted similarity, BoostedSim, is determined by a function, f_{max} that combines RISim, DISim, FISim, RDSim, DDSim, and FDSim over a set of products in the training set. Let RISim, DISim, and FISim represent the rating, demographic, and feature similarity between the products respectively. Furthermore, let RDSim, DDSim, and FDSim represent the rating similarity, demographic similarity, and feature similarity among candidate products found after applying the rating correlation among all products. Formally, it can be specified as follows:

$$f_{max} = \operatorname{argmax}_{f \in F} u(m_i, n_j) : \forall m_i \in M^T, \forall n_j \in N^T. \quad (8)$$

This describes how to choose the function which boosts the utility of all customers M^T over set of products N^T in the training set. This function uses the following equation for making prediction, and tries to boost the utility.

$$P_{m_a, n_t} = \frac{\sum_{i=1}^k \left(\text{Boosted}_{sim}(n_t, c_i) \times r_{m_a, c_i} \right)}{\sum_{i=1}^k \left(|\text{Boosted}_{sim}(n_t, c_i)| \right)}. \quad (9)$$

In [37], the authors proposed a content-based collaborative hybrid recommender which determines similarities between the customers-based on their content-based interest profiles rather than comparing their rating patterns. Recommender systems are being used to suggest relevant information according to the customer's preferences, thus EPSSs (Electronic Performance Support System) could take benefit of the recommendation mechanisms that have the effect of guiding customers in a large space of possible options. The JUMP project intends to integrate an EPSS with a hybrid recommender system.

They proposed a mechanism to determine a group of customers, who have similar interest profiles, by determining similarity values without requiring overlapping ratings. Customer interest profiles are made with machine learning techniques by analyzing both textual metadata about the product's description and the corresponding customer ratings. This textual metadata contains natural language processing mechanisms, which depend on the linguistic knowledge stored in the WordNet lexical database. Furthermore, the "captured" knowledge is stored in semantic customer profiles. This approach overcomes the limitations of the similarity measures based on co-rated products, because customers might be considered similar not only if they like or dislike the same products. Finally, collaborative recommendations are provided by a nearest-neighbor algorithm, which predicts scores for the products to be recommended.

The Architecture of the JUMP EPSS

The objective of the JUMP project was to create an infrastructure that provides intelligent communication among different systems. The key idea of this project was to use a recommender system as the central engine that will provide the best information against the various requests issued by the customers in a specific working context.

The EPSS has a central recommender system that communicates with legacy systems, such as Enterprise Resource Planning (ERP), Human Resources Management system (HRMS), Document Management system (DMS), and Learning Management system (LMS). These systems manage structured and unstructured information, and the communication among them that is based on a common ontology describing and connecting the various knowledge bases. This common ontology provides a semantic representation of information that can be used in a personalized way. Furthermore, in this approach a customer's profile is divided by implementing a bisecting k-means clustering algorithm for calculating neighborhood. Particularly, the clustering algorithm is applied to the set of positive and negative profiles representing positive and negative customer interests. The neighborhood for the active customer is the union of all the customers contained in the cluster of positive and negative profiles the active user belongs to. Clusters obtained by positive parts indicate groups of similar customers, because they share the same interests, while clusters obtained by negative parts indicate groups of similar customers, because they share common dislikes.

The authors validated the hypothesis that the knowledge contained in customer profiles is useful to boost the performance of recommendations within an EPSS. Mean Absolute Error (MAE) is used to measure the quality of recommendations, it calculates the average absolute deviation between a predicted rating and the customer's true rating. A lower MAE value indicates more accuracy in the recommender system. Experimental results indicated a decrease in MAE values that are achieved by using clusters of semantic-based profiles compared to those obtained using the other techniques.

In [38], the authors proposed a hybrid approach that integrates content-based, collaborative and demographic filtering techniques to improve the prediction accuracy, to allow better coverage, and to overcome the cold start problem. The demographic characteristics of customers (e.g, gender, race, age, employment status, occupation, etc.) are used to overcome the cold start problem by categorizing the customers into various categories using a nearest neighbor technique. They used the KNN algorithm to find the nearest neighbors. Each category contains customers sharing similar demographic characteristics. For a new customer, products are recommended using only the cluster to which this customer belongs. In the same way, the combination of demographic characteristics and content-based approaches allow to solve the problem of new products that are added in the system. The authors evaluated their hybrid approach and compared it with various algorithms. Their results indicated that their approach performs well for all customers as well as for the cold start problem compared to the other methods.

Researchers have been working to boost the performance of recommender systems using hybrid mechanisms. My proposed approach also incorporates the integration of content-based with collaborative filtering techniques. As discussed in the related work, this integration has shown better recommendation results. However, research is lacking on providing useful justifications of recommended products. This proposed approach will mainly focus on providing justifications for recommended products as this plays a crucial role in obtaining the satisfaction and trust of customers. Trust creates a long term relationship between a customer and an organization. It has been shown that a customer's trust is positively associated with a customer's intentions to purchase a product and return to the website. Furthermore, this proposed approach will also consider having an attractive and organized explanation interface that will reduce the customer's decision making time and provide more confidence on it. It has been noticed by various researchers that the presentation mechanism to display the recommendations attracts the customer more effectively and boosts the trust on the recommender system [21-23]. Finally, this proposed recommender system will allow the customer to interact with it in order to provide feedback on the recommendations. Once the feedback has been provided, recommender system can revise the recommendation process to provide better recommendations in future.

2.5. Trust in Recommender Systems

Trust is a widely-used term in computer science, such as trust management, trusted computing and reputation system. In recommender systems, trust is the customer's belief that the recommender system (trustee) has the ability to create or recommend useful products. There are two classes of belief: the trustee's ability to create useful products and the trustee's ability to recommend useful products. On the web, two types of judgments can be used to estimate the trust for generating recommendations: a trust statement and a customer's ratings. A trust statement can be either binary or scaled. Intuitively, the scaled one can provide more information about the strengths of belief [26].

E-commerce is a field that influences people's activities. It allows customers to write reviews on products, which other customers read and trust thereby influencing them to purchase a certain product. Due to the lack of face-to-face interaction with customers in online shopping, customers' actions undertake a higher degree of uncertainty and risk compared to traditional shopping. Hence, researchers have been researching these trust related issues in the e-commerce area, and various trust models have been validated in different circumstances. It is crucial to study these issues in recommender system where the traditional salesperson, and subsequent relationship, is replaced by a product-recommender agent.

In [26], the authors proposed a solution to deal with collaborative filtering issues by associating similarity measurement from customers' rating patterns with trust metric. In collaborative recommender systems, user similarity can be used as one of the sources for estimating recommender trust. User similarity is the measurement pertaining to how two customers' rating patterns are correlated. The Pearson correlation coefficient has been widely used for calculating the similarity between customers and products. Customer similarity can be used as one of the sources for estimating recommender trust. By combining customer ratings and author information, one can obtain the author's trust. It is also one of the sources for estimating recommender trust. According to their experimental results, they found that customer similarity and trust are strongly correlated. They claimed that customers who have

high trust but low similarity or vice versa should be seen as the unreliable advisors. The experimental results showed that a good prediction strategy comes from filtering the ratings from these unreliable customers [26].

Many researchers discovered that, in order to gain the trust of customers, the output of a recommender system should be justified [19, 20, 22]. This increases the customer's trust on the specific recommender system. Trust creates a long-term relationship between a customer and an organization. Additionally, it has been shown that a customer's trust is positively associated with a customer's intentions to purchase a product and return to the website. Providing explanations of recommendations justifies the selection of the product and allows the customer to determine how much confidence to put in the recommendation. Explanations help to educate the customer about the process of recommendations. It removes the black box from the recommender system and provides the following major benefits:

- **Justification:** It gives a customer reasoning behind recommendations in order to help for the customer make a decision faster.
- **User Involvement:** It plays an important role in the recommendation process because it gives the customer a chance to interact with it to provide feedback on the recommendations and justifications.
- **Acceptance:** This depends on the customer's satisfaction. Hence, providing reasoning behind the recommendations boosts the customer's decision process and overall acceptance of the system.

The table below describes seven possible aims of explanation facilities in recommender systems [19]:

Aim	Definition
Transparency	Describes how the system operates
Scrutability	Allows users to interact with the system
Trust	Boosts the user's confidence in the system
Effectiveness	Aids users in good decision making
Persuasiveness	Sway users to buy

Efficiency	Aids users in faster decision making
satisfaction	Gives users easiness with navigation

Table 1: Aims for explanation [19]

In [19], the authors showed that explanations are basically linked with the way of displaying recommendations, and with the degree of interactivity offered. Furthermore, they considered how to measure the ‘quality’ of explanations for each of the aims mentioned in the above table. For example, one way to evaluate how effective explanations are is to measure the ‘liking’ of the recommended product prior to and after using it. They also used a mechanism to elicit the key features from the product domain. The rationale behind studying product features is that simply saying that two products are similar does not obtain the attention of the customers to observe the commonality between products, while an explanation using feature-based information can be useful to help a user understand how two products are related. Furthermore, in the study, they performed an experiment to investigate whether the balance between more features in more detail has an impact on gaining the customer’s trust. They validated this with trust questionnaires to determine how much participants put their trust in recommender systems that are based on explanation. According to their findings, the results of this study were not definite, however some customers trust was impacted positively by more features, and others by more detail.

In [20], the authors described Amazon’s justification style: “Customers who bought product ‘X’ also bought products’ Y and Z’. This is called the “nearest neighbor” style of justification. In contrast, another way to recommend is: “Product ‘Y’ is recommended because you rated product ‘X’”. This is called the influence style of justification. Many researchers claim that the influence style is better than the nearest neighbor style, since it allows customers to accurately predict their true opinion of a product. Both of these styles cannot adequately justify their recommendations, since they are based only on data about customer’s ratings or navigational data, ignoring content data, which is extracted in the form of features that are obtained from the products. Content data contains a valuable source of information in addition to rating data. In addition to this, content data can be useful for providing the justifications against recommended products. In order to provide justifications,

content-based filtering is used to explore the features of a product. Hence, the integration of content with rating data helps to achieve a stronger correlation between customers and products, which provide more accurate recommendations with justifications.

2.6. Explanation Interfaces of Recommender Systems

The interfaces of a recommender system play a crucial role in obtaining the satisfaction of customers, since it boosts the customers' decision process by reducing their time and effort to find their desired products. Researchers have started to investigate the effective design factors that can accelerate and impact the promotion of a customers' trust, as well as their behavioral intentions [23]. Recommender systems provide information about the products they recommend. This includes product description, reviews written by other customers, average user ratings, and predicted personalized ratings for the active customer. Some of the recommender systems also provide a way for customers to rate a product when it is recommended.

The attractive interface of a recommender system is crucial for explaining the reasoning behind the recommended products, as it boosts the customer's decision process and provides trust, security and privacy as well. Hence, researchers have started to consider this as a crucial design factor. Only a few researchers have investigated the effect of interfaces on the use of recommendations, most research in recommender systems has focused on achieving the best results from recommendation algorithms. Their research has showed that customers trust more in such a system that has a good GUI interface to explain the reasoning of recommended products [2, 23].

Reasoning-based recommender systems use traditional strategies for displaying and explaining recommendations to the customers and commercial websites display the recommended products in a rank-ordered list and use a "why" component along with each product to explain the computational reasoning behind it [23]. Various measures have been used to evaluate the trust and interfaces of recommender systems. In [23], the authors measured variables that are related to their proposed competence-based trust model for recommender systems. This trust model consists of three main components: system-design

features, competence-inspired trust, and trust-induced behavioral intentions. System-design features are useful for the promotion of overall competence perceptions that include three measures: recommendation quality, transparency, and user-control. The overall competence consists of crucial measures: perceived ease of use, perceived usefulness decision confidence, cognitive effort, and satisfaction, which have been determined as the primary factors of persuading customers to accept and use a certain technology. Trusting intentions are behavioral attitudes expected from customers once their trust has been built. They determine whether customers will potentially reduce their decision-making effort in repeated visits upon establishing a certain trust level with the recommender system [23].

In [19], the authors compared various mechanisms of presenting explanations. They wanted to measure the qualitative feedback from various interfaces. They focused on various things, with a sixty seven participants. In this experiment, they compared graphical and textual equivalents. They found that a graphical interface gained the most attention of the participants; it did not have a textual counterpart. Furthermore, they received a great qualitative feedback, including the relevance of the content. For example, most of the participants felt that information about the recommendation confidence should be excluded altogether, or at least not be displayed as a justification for a recommended item. Rather, the participants felt that uncertain recommendations should simply be excluded.

This proposed approach adopts an attractive GUI-based explanation interface that mainly displays the recommendations for the customer. This approach has a categorized presentation mechanism that displays the same type of recommendation under one category, rather than displaying each product individually. Furthermore, it explains the reasoning behind the recommended products. This interface mainly consists of two sections. The first section displays the current user's rated products and the user's favorite products type. The second section displays the recommendations with justifications. A movie's contents, such as movie type, actors, director and movie release date, are being used to justify the reasoning behind the recommendations. Furthermore, the average rating of each recommended movie is being displayed to the customer that shows the popularity of that movie among other users. Finally, this interface allows the customer to interact with the recommender system to provide

feedback on the recommendations. Overall, the strategy of displaying and explaining justifications will be very effective, as it has been observed that customers trust more in a system when there are explanations on how the recommended products have been computed and displayed.

From the above-discussed recommendation techniques, I am going to use collaborative filtering, content-based filtering and context-based filtering in my proposed model. These three techniques will be used to generate recommendations and justifications for the customer. From collaborative filtering, item-based collaborative filtering will be used to generate item-based recommendations from the rating data. Furthermore, content and context-based filtering will be mainly used to generate useful justifications for recommendations. As this proposed approach mainly focuses on providing justifications, a product's features and the active customer's context dimensions will be used. Finally, an attractive presentation mechanism will be used to display all the information about recommendations.

Chapter 3: The Proposed Model

Hybrid techniques have been used by researchers to overcome the drawbacks of individual technique and mainly to improve the predictions of recommender systems. There are various ways to integrate more than one technique as discussed in the chapter 2. My proposed hybrid approach is the integration of content and context-based systems with collaborative recommender systems, since this is the most successful and widely-used mechanism. Research is lacking on providing justifications against recommended products. In this proposed approach, I will mainly focus on providing valuable recommendation and various types of justifications for recommended products, as this plays a crucial role in obtaining the satisfaction and trust of customers. In addition to this, the method of presenting recommendations and explaining justifications will be very effective as it has been observed that customers trust more in a system when there are explanations on how the recommended products have been computed and displayed [22]. Finally, this proposed recommender system will allow the customer to interact with it to provide feedback in various forms on the recommendations and justifications.

3.1. Hybrid approach: Content and context-based collaborative filtering

Initially, many recommender systems were very simple query based information retrieval systems, which are called content-based recommender systems. They recommend products to a customer-based upon a description of the product and a profile of the customer's interests. Therefore, reliable and automatic determining of the results of content analysis and customer's preferences are required for this system [25]. Furthermore, in order to have a sufficient set of features, the content must be in a form whose features can be extracted automatically. To overcome these above limitations, researchers have been working on the integration of content data with rating data in various ways. My proposed approach will look into the integration of content and context data with rating data using a different mechanism that will mainly focus on providing justifications for the recommended products. This integration will be very useful in achieving a stronger correlation between customers and products. Finally, this approach will have a categorized attractive explanation interface that will display the recommendations with justifications and also minimize the customer's decision-making time.

3.2. Design of the Proposed Model

This proposed hybrid recommender system consists of three major components as shown in the diagram below:

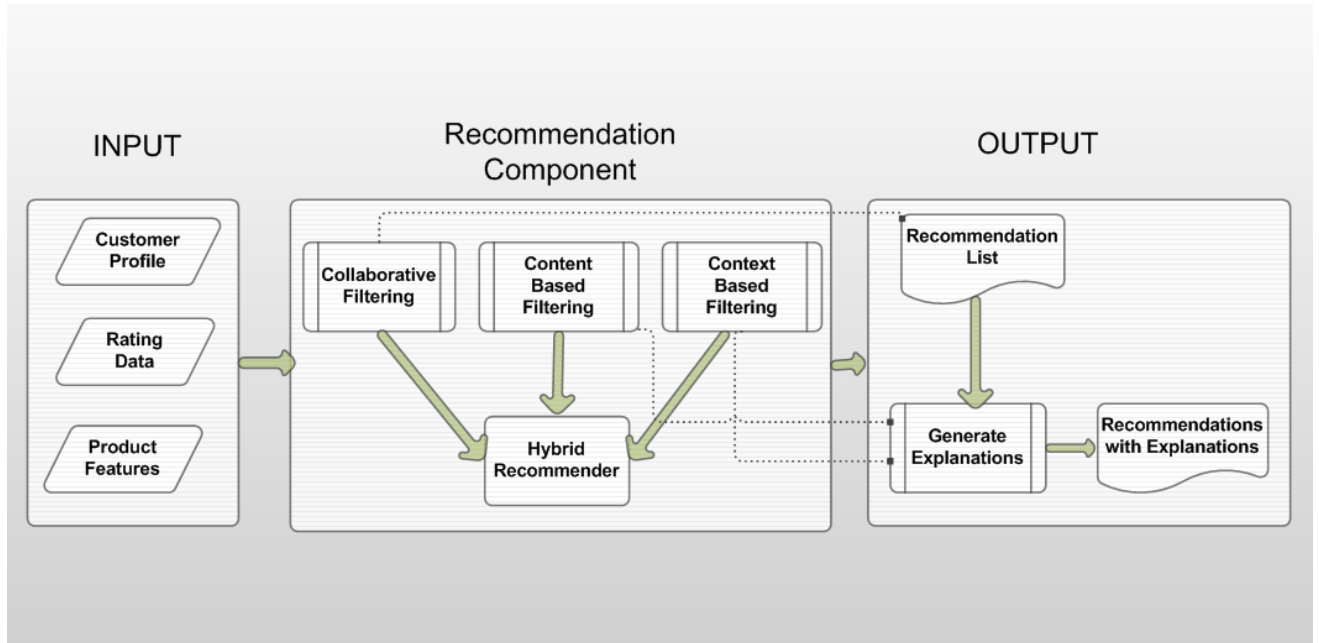


Figure 2: Design of the Proposed Model

Each recommendation technique requires some input upon which to base the recommendations. The input of this proposed system also contains a customer's interest profile, community or rating data and product information. Accurate customer information has a crucial role for integrating different recommendation techniques. I have classified a customer's profile into two types of information: customer description and customer behavior information. Customer description information includes basic information such as name, gender, geographical location, occupation, etc. That information is usually gathered in the process of customer registration. Customer behavior information can be gathered in various ways such as from previous purchasing history, navigation of pages, customer's ratings on products and feedback provided from the customer.

In this proposed approach, customer behavior information is gathered by giving ratings to the products as shown in the customer-product matrix below, where c1 to c5 represents the

customers and p1 to p5 represents the products. Customers used a rating scale of 1 to 5, where 1 indicates a less favorite and 5 indicates a top favorite product. The customers that have not given ratings to products are indicated by a value 0. Furthermore, a customer's rating information also incorporates contextual information that is linked to the application. This means that the customer's context is being considered when they provided a rating, as shown in the below rating function:

$$R: \text{Customers} * \text{Product} * \text{Context} \rightarrow \text{Rating} \quad (10)$$

This contextual information is gathered explicitly from the customer and also implicitly from the application. Context aware recommender systems predict a customer's preferences by incorporating contextual information into the recommendation process. Hence, it is very important to consider not only information about products and customers, but also contextual information such as the shopping date, time, location and who accompanies the main customer / buyer [28, 29].

	P1	P2	P3	P4	P5
C1	3	4	5	3	4
C2	4	3	4	5	0
C3	3	5	3	4	5
C4	0	3	4	5	0
C5	4	5	3	0	4

Figure 3: Customer Product Matrix

With the customer's behavior information, the system can get an idea of their rating behavior and an idea of their favorite and non-favorite products. In this approach, we assumed that each product has four features (f1, f2, f3, f4) as shown in the product feature matrix below, where 1 indicates that this feature is available in the product. With these features, the system will create a feature profile for each customer containing their favorite features. This feature

profile is created using a customer’s preference profile and the products feature profile. In order to create a feature profile, the system will select only those products that are highly rated by a customer. These steps or components are described in detail in the following sections.

	f1	f2	f3	f4
P1	1	1	1	1
P2	1	1	1	1
P3	1	1	1	1
P4	1	1	1	1
P5	1	1	1	1

Figure 4: Product Feature Matrix

3.2.1. New customer profile

For a new customer, description and behavior information will be gathered during the process of customer registration. For customer description information, new customers will find a link “Register here” on the main page of the system. This will be required mainly to enter login credentials for creating a new account. In the next step, this proposed approach will ask the new customer to rate some products. The system will offer a set of products and the customer will select the closest ones to their necessities, tastes, or preferences. In this step, the system will offer only a small subset of the products. This subset will include favorite products as well as those products that are not liked by many customers. In addition, these products will belong to all the available categories that will allow customers to find more appropriate products that are suitable to their tastes. The customer will provide rating information regarding which of these products are closer to their real expectations. With this, the system will have the feature and preference profile of a new customer.

3.2.2. Collaborative Filtering

This mainly consists of two types: user and item-based collaborative filtering. As described earlier, user-based collaborative filtering has crucial limitations such as sparsity, scalability,

performance and does not work efficiently with a large dataset [11]. Therefore, item-based collaborative filtering was used in the proposed approach. It is based on item relations instead of user relations, as in user collaborative filtering. Item-based collaborative filtering consists of representation, neighborhood formation and recommendation generation, as described below [11, 14]:

- Firstly, all the products rated by an active customer are retrieved.
- Secondly, the item-based approach looks into the retrieved products from the step 1 and determines how similar they are to target product 'i' and then selects the 'K' most similar products. In addition, their corresponding similarities are also determined.
- Finally, after finding the most similar items, the prediction is then determined by taking a weighted average of the active users rating on these similar products 'k'.

This process is illustrated using the following diagram. Here, the purpose is to find Alice's rating for item5. As the algorithm described above, the first step is to retrieve Alice's ratings of all the items. In the second step, a set of the most similar products (K) to item5, with their similarities, are selected. In the final step, we take Alice's ratings for these similar items to predict the rating for Item5 [39]. The process of item similarity computation and prediction generation is described in details in the following section.

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

Figure 5: User-Item Matrix [39]

Algorithm 1: Item-based Collaborative Filtering Algorithm

Input:

Active customer (User ID)

Community Ratings

Output:

S= Set of Item-based recommendations

P= Set of preferences of predicted items

Main Procedure: Calculate the recommendations for active customer

1- Create a MYSQL data source

2- Assign the data source properties (Parameters)

Set data source= root (user)

Set data source= raza (Password)

Set data source= localhost (Server Name)

Set data source= 3306 (Port)

Set data source= hybridrecommender (Database Name)

3- Create a JDBC data model

Assign the parameters to data model

Set data model= data source

Set data model= Database table 'ratings' with the fields ("User_id", "movie_id", "rating", "rating_date")

4- Calculate the item similarities using equation 11(Pearson Correlation) based on the data model

```

Set similarity = PearsonCorrelationSimilarity (model)
5- Build the item-based recommender based on the model and similarity
Set recommender = ItemBasedRecommender (model, similarity)
6- Run recommend function using equation 12 that takes two parameters of User ID and
    Maximum recommendations required
Set recomendations = recommender.recommend (user ID, maxRecomendationsReqd)
7- Display all the recommendations and add them in the list of recommendations (S)
For i= 1 to recomendations.size
    Display recomendations.get(i).getItemID - recommended item
    Display recomendations.get(i).getValue - Predicted preference value
    Add recommended item in the list
End For

```

Item Similarity Computation

The Pearson Correlation Similarity or the Cosine / Vector Similarity metrics have been used to find the similarity between customers and products in the literature. The most widely used similarity measurement is the Pearson Correlation Coefficient. In order to calculate the similarity between two products one must first isolate the customers who have rated both of these products and then apply a similarity computation technique. In the case of the Pearson Correlation Similarity, the difference is that we are not using the ratings that two customers have given for a common product, but the ratings that two products ‘i’ and ‘j’, whose similarity we want to calculate, have been given by a common customer ‘u’. This can be calculated using the following equation [11, 25, 27].

$$sim(i, j) = \frac{\sum_{u \in U} (R_{u,i} - \bar{R}_i)(R_{u,j} - \bar{R}_j)}{\sqrt{\sum_{u \in U} (R_{u,i} - \bar{R}_i)^2} \sqrt{\sum_{u \in U} (R_{u,j} - \bar{R}_j)^2}}. \quad (11)$$

Here $R_{u,i}$ indicates the rating of user u on product i, \bar{R}_i is the average rating of the i-th product [25]. Once we have calculated the similarities between all the products in the user-

product matrix, collaborative filtering isolates the most similar products based on the similarity measures. Once this is done, the next step is to generate a prediction for that product.

Weighted Sum

The prediction of a user on a target product is computed after we have a similarity score of all the other products to the target product. The most common way to achieve this is through a weighted sum. The predicted rating is the weighted sum of the ratings given by the other customers to that product, where the weights are the similarity coefficient of the active customer with the other customers. This clearly indicates that the rating expressed by a very similar customer has a larger influence on the rating predicted for the active customer. This can be calculated using the following equation [25].

$$P_{u,i} = \frac{\sum_{Allsimilaritems,N}(S_{i,N} * R_{u,N})}{\sum_{allsimilaritems,N}(|s_{i,N}|)} \quad (12)$$

Where $s_{i,N}$ is the similarity between product i and the other product in set N . $R_{u,N}$ is the rating of user u on the products in the set N . N is the set of the products which are rated by user u . The predicted value will have the same scale as the rated products such as 1-5. However, in order to determine the accuracy of a recommender system, the proposed algorithm sets the threshold value to determine whether this product is relevant or irrelevant for the active customer. This threshold value is based on the predicted ratings that are achieved after running the item-based collaborative filtering algorithm.

3.2.3. Explanation of Recommendations

Once the item-based recommendations are determined, the next step of this proposed approach is to generate final recommendations with justifications. It is very important for a recommender system to explain the reasoning behind the recommended products. When customers are selecting the right product to purchase, the ability to convince them to buy that

product should be an important goal of any recommender system. Furthermore, an attractive presentation mechanism is needed to display the recommendations with justifications that will take customer's less time to find their desired products. Therefore, these key points are taken into consideration in my proposed approach.

3.2.3.1. Customer's Feature and Preference Profiles

In this proposed approach, a customer's feature and preference profiles are being created for providing final recommendations with justifications. The product features (f1, f2, f3, f4, etc.) mainly describes the characteristics of the product. With the customer behavior information, the system will create a feature profile for each customer that will have their favorite features. This feature profile is created using a customer's rating information and the products feature profile. In this feature profile, the system will select only those products that are very highly-liked by a customer. In addition to this, from the customer behavior information, the system will also create a preference profile that will have their favorite context attributes. In order to generate more personalized recommendations and justifications, it is very important to consider contextual dimensions that are associated to the application. Hence accurate prediction of customer preferences depends on the degree to which relevant contextual information is considered in the recommendation process [28, 29]. This clearly indicates that the integration of content and context with rating data helps to achieve a stronger correlation between customers and products, which provides more accurate recommendations. In addition to this, the consideration of content and context can provide high quality justifications for recommendations.

A database view has been created, which stores the customer's feature profile with their favorite features and customer's preference profile with their context attributes, as shown in the customer profile matrix below. In this approach, we assumed that each customer has at least four favorite features (f1, f2, f3, f4, etc.) and two context attributes (cn1, cn2, etc.) that are indicated by value 1 in the matrix. A database view is similar to the database table which also consists of a row and column, so you can get and modify data on it in the same way as a table. Database management systems stores database views as SQL SELECT statements with

various joins. When the tables that are the data source of a database view changes, the view reflects that changes as well [40].

	f1	f2	f3	f4	cn1	cn2
C1	1	1	1	1	1	1
C2	1	1	1	1	1	1
C3	1	1	1	1	1	1
C4	1	1	1	1	1	1
C5	1	1	1	1	1	1

Figure 6: Customer Profile Matrix

Once these profiles have been created, the system will use these profiles to filter the final recommendations from the initial item-based recommendations. Content and context post-filtering mechanism are being used in this proposed approach that filter out those recommendations that are irrelevant to the active customer’s content and context. Contextual post-filtering is a derivation of recommendations via contextual preference elicitation and estimation mechanism. It has been widely used for tourist and mobile recommender systems, where current context plays a key role in generating more personalized recommendations. [28, 29]. It ignores context information (C) initially from the input data and the unknown ratings are predicted using traditional recommender systems. Finally, recommendations are generated for each customer according to a customer’s favorite content and context [29].

This proposed approach employs the post filtering mechanism, where item-based collaborative filtering is applied to the customer’s ratings that generate the initial recommendations. Once these initial recommendations are generated, these will be filtered out according to content and context. The process of computing the content and context-based recommendations with justifications consists of comparing each recommended product’s features and context attributes with the customer’s feature and preference profiles, as described in the proposed algorithm 2. Hence, it is very crucial for the customer to rate the products very carefully, since recommendations are fully dependent on the profiles.

Algorithm 2: Proposed Algorithm for Creating Recommendations with Justifications

Input:

S= Set of initial recommendations (item-based recommendations)

F= Set of product's features (f1, f2, f3, f4, etc.)

CF= Active customer's feature profile

CP= Active customer's preference profile with context dimensions

CP-S= Customer's preference profiles of other customers who have rated to recommended products

Main Procedure: It consists of two steps:

- Comparison of recommended product's features with the feature's profile of the active customer
- Comparison of active customer's preference profile with other customers who have rated recommended products

For each item in S {s1, s2, s3, etc.}

1- Compare features of s1 with the feature profile of the active customer (CF)

If s1's product type (f1) IN CF then

- i- Include s1 in the final list of recommendations (N)
- ii- Include feature f1 (product type) as a justification
- iii- Compare other features of s1 with the feature profile of the active customer (CF)

If s1 (f2, f3, f4) IN CF then

 Include these features as a justification

Else

 Ignore these features

End If

2- Compare active customer's preference profile (CP) with other customer's preference profiles (CP-S) who have rated to recommended products (S)

If features of CP-S in CP then

 Include these features as a justification

```

    Else
        Ignore these features
    End If
Else
    Reject s1
End If
End For
Output:
N= Final list of recommendations
Display content-based justifications
Display context-based justifications

```

In order to implement this proposed approach, movies are being used as an example product in this thesis. Hence, a movie's features such as movie type (f1), actors (f2), director (f3) and movie release date (f4) are being used for providing content-based justifications. When a customer rates any specific movie, the system stores the features of those movies that are highly rated by the customer. This rating data also includes a customer's context attributes such as the time of week, as well as their companion, which means that the rating given to a movie by a customer depends on how the movie has been watched and at what time of the week. These context attributes are being used to generate personalized recommendations with justifications.

The process of computing the justifications is shown in the proposed algorithm 2. If it finds a similarity in these profiles then these attributes will be used to provide the justifications of recommended movies. The above proposed algorithm compares a movie's features such as movie type, actor, director and movie release date of each movie from the initial set of item-based recommendations with the active customer's feature profile. In addition to this, it compares a customer's context dimensions from the preference profile such as the time of week and their companion, with other customer's profiles who have rated the recommended movies. First, the movie type of a recommended movie should belong to the types of rated movies. This clearly indicates that this recommender system would only recommend the

same type of products (movies) that the active customer has shown interest to. Second, the other content and context attributes will be matched to the rated movies. Finally, for the matched product types, a product's features and customer's context attributes will be used to generate final recommendations with justifications. Hence, it is very crucial for the customer to rate the products very carefully since recommendations are fully dependent on the profiles.

3.2.3.2. Justification Style

Various justification styles such as nearest neighbor and influence styles have been used. However, they cannot adequately justify their recommendations, since they are based only on data about a customer's ratings or navigational data and ignore content and context data. Content and context data is extracted in the form of features that are obtained from the products and the contextual information that is associated to the customer. It contains a valuable source of information that can be used to provide justifications. The justification style used in this approach combines the "keyword based" with the influence styles that will include justifications from a product's features and the active customer's context dimensions. It has the following form: "Product X is recommended, since it contains the product's features and context dimensions which are included in products Z and W that the active customer has already rated". For example, "the movie 'The Godfather' is recommended since it contains the actor 'Al Pacino' who also acted in the movie 'Glengarry Glen Ross'. Furthermore, this movie has been watched on a weekday with a friend". This clearly indicates that the active customer's behaviors with context dimensions are being used to generate more personalized recommendations that will boost the customer's trust on recommender system.

3.2.3.3. Explanation Interface

This proposed approach adopts a GUI-based explanation interface that displays all the information regarding recommendations and their justifications to the customer. This approach has a categorized presentation mechanism that displays the same type of recommendations under one category rather than displaying the same type of product individually. It clearly eliminates duplicate information and saves time when making a decision. This proposed interface mainly consists of two sections as shown in the diagram

below. The first section displays the current customer's rated products and the customer's most favorite products type. The second section displays the recommendations with their justifications.

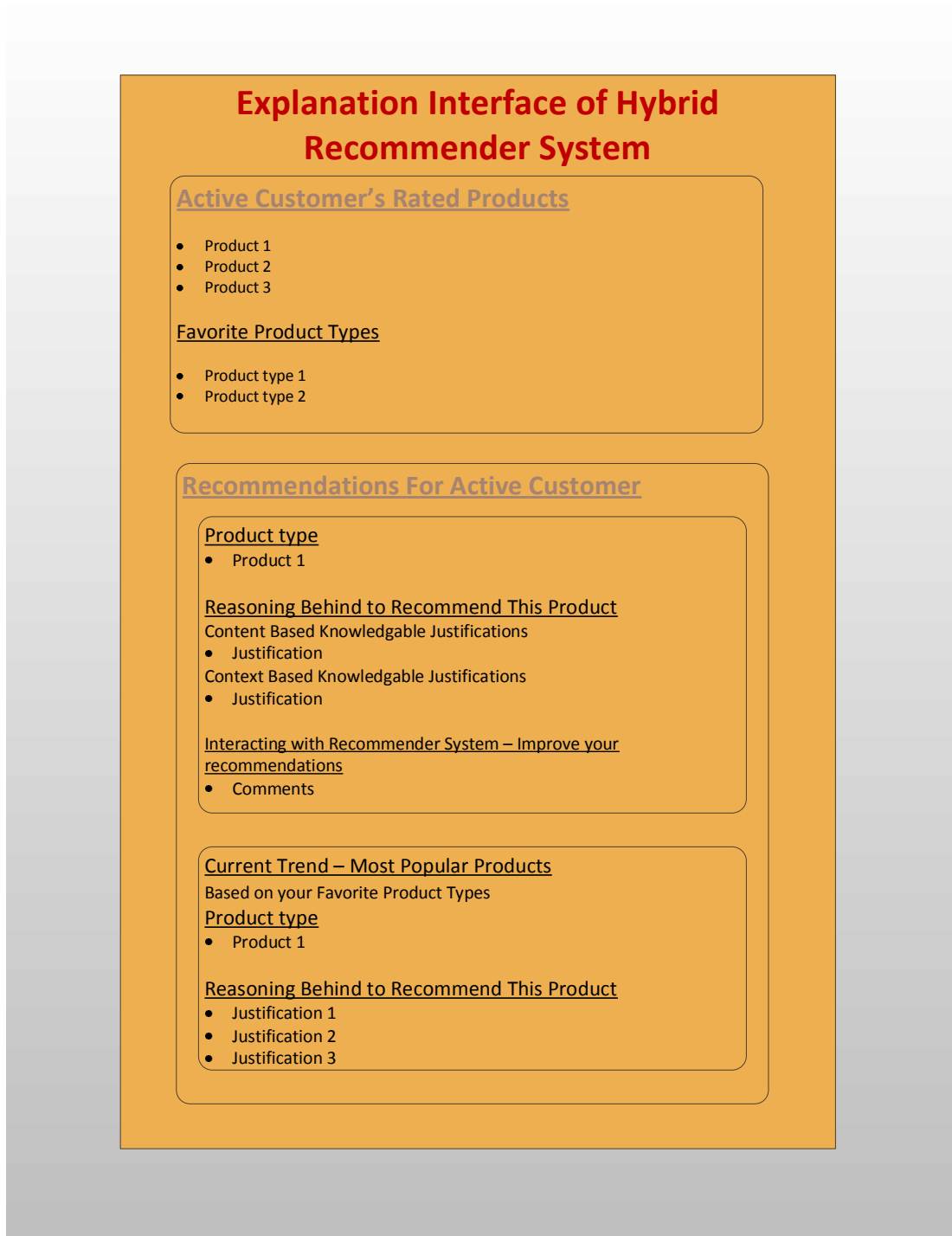


Figure 7: Design of the Proposed Explanation Interface

As movies are being used as an example product in this thesis, hence this explanation interface displays the movies recommendations for the active customer. The first section displays the active customer's rated movies and the most favorite movie types. In the second section, the recommendations will be presented in order for the active customer. The recommendations from the most favorite movie types will be displayed first and then other movies will be displayed. The movie type is being used to categorize the recommendations in different sections or blocks. This means that if there are two recommendations with the same movie type, then it will be displayed under the same category (movie type). Furthermore, justifications will always be displayed with each recommendation that clearly explains the reasoning behind this recommendation. In addition to this, the average rating of each recommended movie is being displayed to the customer that shows the popularity of that movie among other users. Finally, the current most-popular movies that belong to the active customer's favorite movie types will be recommended. This will indicate the current trend of other customers that have the same preferences with the active customer. It has been noticed by researchers that the organized view of recommended product lists was more accepted by customers to accelerate the process of product comparison and choice making than the traditional "why" based list view [23]. Overall, the strategy of displaying and explaining justifications will be very effective, as it has been observed that customers trust more in a system when there are explanations on how the recommended products have been computed and displayed.

Interacting with Recommender System

Once the recommendations are presented to the customer, the recommender system should allow the customer to interact with it. It is very crucial for the performance of recommender systems to allow the customer to provide feedback on it, since this feedback would provide a chance for the recommender system to improve the prediction. It has been noticed that if customers are allowed to make alteration to the recommended products then they can find their desired products quicker [22]. Hence, this proposed explanation interface allows the customer to provide feedback on recommendations. This feedback can be provided by modifying the previous ratings and also providing direct suggestions in the form of 'like' or 'dislike'. In addition, customers can make a suggestion on a product's features that were

used to provide justifications. This process can improve the recommendations as described below.

As movies are being used as an example product in this thesis, hence movie's features are being used to provide justifications. Once justifications are provided, customer's can provide feedback on the movies features (f1, f2, f3, f4, etc.) in the various following forms such as:

- I want further recommendations on Action movies (f1)
- I do not want further recommendations on movies with actor such as 'Al Pacino' (f2).
- I do not want further recommendations on movies directed by such as 'George Lucas' (f3).

Once the feedback is provided, the system can update the active customer's feature and preference profiles based on this feedback. Now the process of computing the justifications should be based on the updated customer's profile. Hence, the system will be able to generate better recommendations for the customer.

Chapter 4: System Implementation

This chapter presents the implementation of a recommender system, which is built on the basis of the proposed hybrid approach discussed in Chapter 3.

4.1. System Architecture

In order to evaluate my proposed approach, I developed a web based application to experiment the applicability and performance of the designed model. This web application is designed on a three-tier model and consists of the three main components shown in the figure below [41]:

- A client PC (Browser)
- Application or Web Server
- Database Server

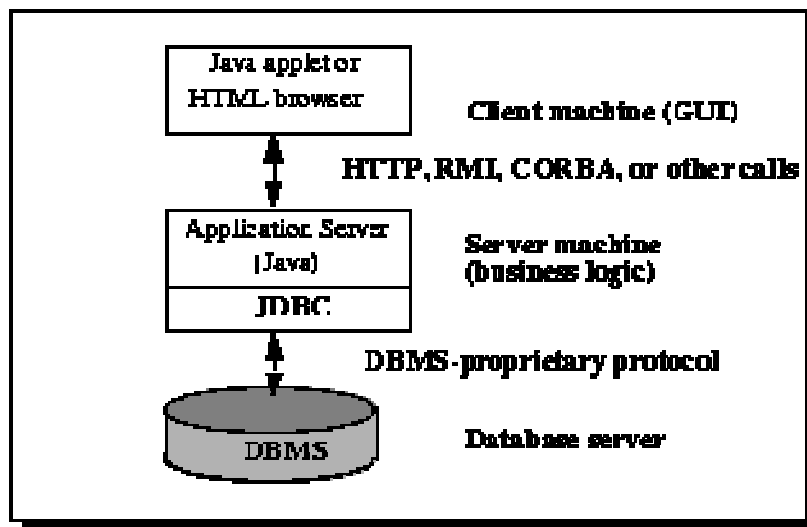


Figure 8: Three Tier Architecture [41]

The three-tier architecture consists of presentation and application logic in the client, application and business logic in a middle tier application server, and data managed by database servers in the third tier. The application is typically controlled by user interface in the client with substantial application processing taking place in the middle tier application server. In the three-tier model, commands or requests are sent to a "middle tier" of services,

which then sends the commands to the data source. The data source processes the commands and sends the results back to the middle tier, which then sends them to the user. JDBC API is being used more and more in the middle tier of three-tier architecture. There are some features that make JDBC a good server technology, such as its support for connection pooling and distributed transactions [41].

4.1.1. Technologies and Tools

This proposed hybrid recommender system is developed using the following open-source tools and technologies with Java platform in the Windows XP environment.

- Netbeans IDE: It is an open source integrated development environment for developers to develop various types of desktop and web-based applications.
- Java Server Pages (JSP): a server-side Java technology was used to create web pages. JSP allows software developers to create dynamically-generated web pages, with HTML, CSS and Java Script, in response to a web client request to a server.
- Java Database Connectivity (JDBC) is used to communicate with the database. The JDBC API is a Java API that can access any kind of tabular data, especially data stored in a relational database management system. Mainly, it establishes a connection with the data source. A data source can be a DBMS, a legacy file system, or some other source of data with a corresponding JDBC driver. Typically, a JDBC application connects to a target data source using one of two classes: 'Driver Manager' and 'Data Source'.
- MYSQL database engine version 5.1 was used for storing and manipulating with the MovieLens dataset.
- MySQL Command Line Client and MySQL Query Browser: These are front-end tools that were used for DDL (data definition language) and DML (data manipulation language) statements.
- Tomcat web server: This web application is hosted using Tomcat web server. It is an open source software implementation of the Java Servlet and Java Server Pages technologies.
- Notepad++: This is a very useful source code editor that was used to write and edit the Java code and the MovieLens text files.

MS-Visio 2010: Is a Microsoft tool to create or design various types of diagrams. This was used to create the entity relationship diagrams for my database schema.

4.1.2. Database Design

In order to implement this proposed approach, movies are being used as an example product in this thesis. Hence, the MovieLens dataset was used for evaluating the proposed approach. In order to implement the proposed approach, the MovieLens dataset was modified accordingly. A 'password' attribute was added in the customers table, in order to login to the application. The MovieLens dataset does not have content and context data, hence the dataset was modified accordingly as explained below in the algorithm 3. The data regarding a movie's features such as actors and directors was extracted from the international movie database (IMDB). IMDB is an online database for movies and television shows. It has all the information that is associated with movies such as actors, directors and movie types. The information for those movies that were in the MovieLens dataset was collected [42]. In addition to this, a customer's context dimensions (time_of_week and companion) were added to generate more personalized recommendations with justifications. The context data for time of the week and companion dimensions indicates that the rating given to a movie by a customer depends on how the movie has been watched and at what time of the week. Algorithm 3 describes the process to add a customer's content and context data. For implementation, matching content and context data was added in the dataset. Hence, the proposed algorithm 2 finds the matching data in the active customer's profile and the other customer's profile who rated the recommended movies.

Algorithm 3: Proposed Algorithm for Updating the MovieLens Dataset

Input:

S= Set of rated movies

Main Procedure: Addition of content and context data

For each movie in S{s1, s2, s3, etc.}

 s1(actor) = IMDB database

 s1(director) = IMDB database

 If s1(movie_type) = 'Action' OR s1(movie_type) = 'Comedy' Then

 time_of_week = 'Weekday'

 companion= 'Family or Friends'

```

End IF

If s1(movie_type) = 'Romance' Then
    Time_of_week = 'Weekend'
    companion = 'Significant Other'
End IF

End For
Output:

```

M= Set of rated movies with movie's features and customer's context data

For database implementation, a MYSQL database was created with the name of 'HybridRecommender'. Once the database was created, various tables were created as shown in the entity relationship diagram below. In order to load the dataset, MYSQL load utility was used that reads rows from a text file into a table at a very high speed. The file name must be given as a literal string. All MovieLens text files were placed in the root (C:\) directory and the following command was used to load data into the table.

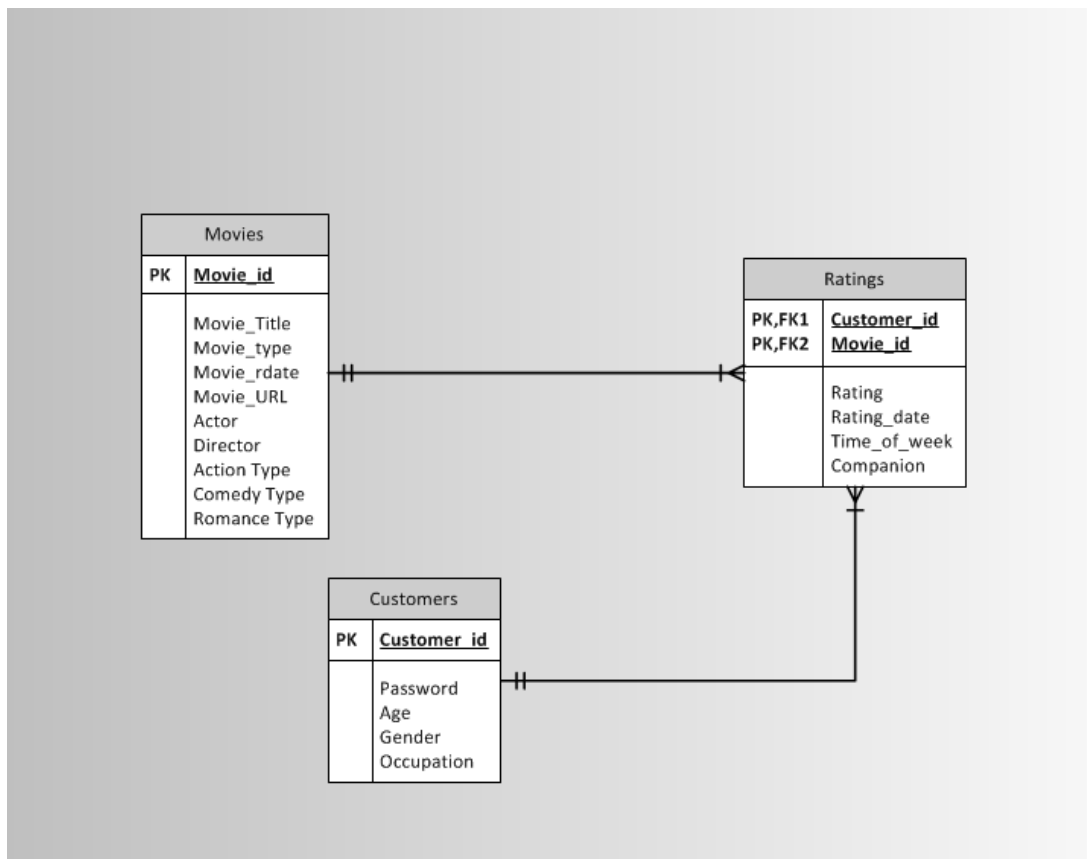


Figure 9: Entity Relationship Model

MYSQL Load Statement

```
LOAD DATA LOCAL INFILE '/users.txt'  
INTO TABLE customers  
FIELDS TERMINATED BY '|'  
LINES TERMINATED BY '\n'  
(User_id,age,gender,occupation,zip_code);
```

The database view 'Customers_profile_V' was mainly created to retrieve a customer's favorite features and context attributes. The database view is dynamic because it is not related to the physical schema. The database view is stored as view definition as SELECT statements. The database view always shows updated data since it always retrieves the data from the database tables [40] . The database objects are:

- Customers: It has customer_id, password, age, gender, occupation and zip code attributes.
- Products or Movies: It has movie_id, movie_title, movie_rdate, movie_url, actor, director and various movie_types (action, comedy, romance) attributes.
- Ratings: It has customer_id, movie_id, rating, rating_date, time_of_week and companion attributes.
- Customers_Profile_V: It has customer_id, movie_id, actor and director, movie type, time_of_week and companion attributes.

4.1.3. System Performance

This approach is being tested with the training and test data from the MovieLens dataset. It has 100,000 ratings from 943 users on 1682 movies [14, 25]. As this is a large dataset, it is very important to consider the response time of the algorithm. Therefore, database indexes have been implemented on the database tables. Generally, indexes are something extra that you can enable on your MySQL tables to boost the system performance. The index entries work like pointers to the table rows, allowing the query to quickly find which rows match a condition in the WHERE clause, and retrieve the other column's values for those rows [43, 44].

4.1.4. System Configuration

As this proposed approach was tested using Windows XP operating system, a new parameter needed to be entered in the windows registry. Since, MovieLens is a large dataset, the JDBC interface needs enough capacity to open many connections from the TCP ports. This parameter controls the maximum port number that is used when a program requests any available user port from the system. Without this entry from the JDBC, the system gives the following exception:

Communications link failure: The last packet sent successfully to the server was 0 milliseconds ago. The driver has not received any packets from the server [45].

4.2. System Implementation

This web based application is developed in JAVA using NetBeans IDE and movies are being used as an example product in this thesis. Hence, this implementation has been done for movies (product). The figure below is taken from NetBeans IDE, and shows the structure of the entire project. It mainly contains the web pages and classes of the packages as described below in detail.

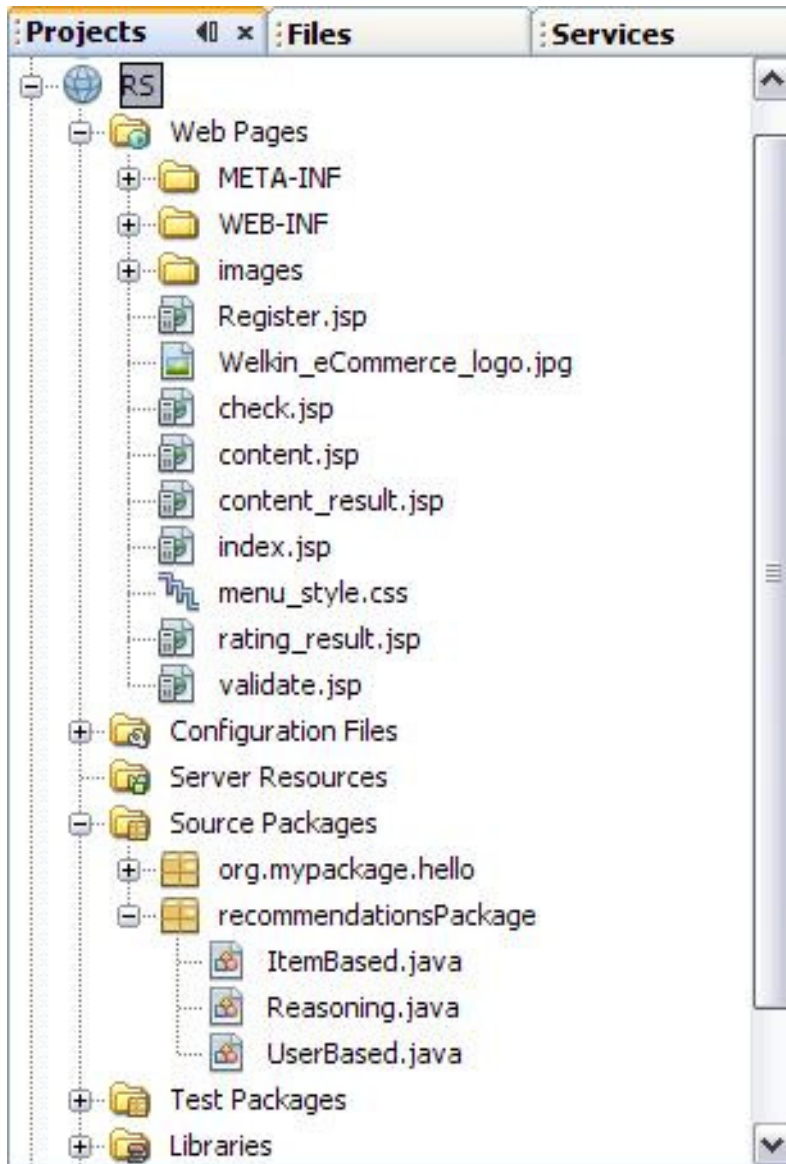


Figure 10: Structure of Pages and Class Packages

4.2.1. Web Pages

The following are the JSP web pages that were developed for the implementation of this proposed recommender system. Here is the brief description of each web page and the code is described in the Appendix section.

- Index: This is the main page of the application. This page briefly describes the proposed hybrid recommender system and highlights the main contributions of this system. It contains

the login form for existing customers to login to the application to obtain recommendations, as well as a web link for new customer to the registration page.

- Check: This page validates the login credentials of the active customer from the database. If this validation is successful then it displays the recommendations on the explanation interface of this proposed recommender system. This page mainly contains the logic to create an explanation interface that mainly displays the recommendations and their justifications.
- Register: This page is created for the new customers to get registered in the system. It requires basic information to create an account in the system.
- Validate: This page validates and stores the new customer's information into the database. Furthermore, in order to create a profile of the new customer it displays some movies for which the new customer has to provide ratings to show his /her preferences or tastes about the available products.
- Rating-result: This page validates the entered rating and displays the result whether the rating has been successfully entered into the system or not. It is very important for the active customer to rate with the same rating scale that was originally used by the system. This system uses 1 to 5 to rate the products, where 1 indicates the least favorite and 5 indicates the most favorite.
- Content: This page has a search form for the active customer, to allow them to search for a specific available product. It has the functionality to search through the entire database about the required product.
- Content-result: This page validates and displays the search results for the active customer.

4.2.2. JAVA Package

The web based application has a Java package that has various classes in it. This package contains all the core business logic that is required to develop the proposed hybrid recommender system. The code of the following classes is described in the Appendix section.

- Item-based recommender
- Explaining reasoning

Item-Based Recommender

As described in the last section, an item-based recommender system is used in the proposed approach. 'Mahout Library' is used to build the item-based recommender. Mahout provides a scalable machine learning library that includes core algorithms for clustering, classification and batch-based collaborative filtering [46]. A JAVA class with the name of 'Itembased' was created, in a package, which has the functionality to recommend products to the customers. It mainly consists of the following steps.

- First, create a database data source with the parameters of schema name, password, server name, database name and port. This data source is used to create a MYSQL database data model. This data model is based on the customer's rating data of the movies. It is created with the following 'MySQLJDBCDataModel' function of the item-based recommender class. This function takes six parameters. It consists of the data source's name, rating (table name), user_id, movie_id, rating and rating date.

```
MySQLJDBCDataModel model = new  
MySQLJDBCDataModel(dataSource,"ratings", "User_id", "movie_id", "rating",  
"rating_date");
```

- Second, the 'Pearson Correlation' coefficient is used (equation 7) to compute the item similarities between the items that are loaded in the data model. It returns the degree of similarity of two items, based on the preferences (ratings) that users have expressed for the items. It returns values in the range -1.0 to 1.0, with 1.0 representing perfect similarity.

- Finally, the item-based recommender is built on the previously-computed data model and similarities. Once the item-based recommender is built, then the recommender function is executed which takes two parameters (User ID and Maximum recommendations). This following line of code indicates that three items will be recommended to the customer1.

```
List<RecommendedItem> recomendations = recommender.recommend(1,3);
```


Explaining the Reasoning

This class is created to explain the reasoning behind the recommended products. It has all the required functionality to provide justifications to the customer in order to obtain their trust on the recommender system. As I described in chapter 3, a customer's feature profile is created based on the favorite features of the customer. Once the customer's feature profile has been created, the system can recommend the closest products to the customer. Mainly, this feature profile is being used to provide justifications against recommendations. Hence, it is very crucial for the customer to rate the products very carefully since recommendations are fully dependent on their profile. The process of computing the justifications consists of comparing each recommended product's features (f1, f2, f3, f4, etc.) with the customer's feature profile and active customer's context dimensions with the other customers who rated recommended products. This process is described in the proposed algorithm of creating justifications mentioned in chapter 3. This class has the following two functions:

- ReasoningMoviesType (user): This function takes the active customer as a parameter and generates a recommendation of the same (product) movie type that belongs to the active customer's favorite movie types. It makes sure that the movie type of the recommended movie belongs to the types of rated movies. This clearly indicates that this recommender system would only recommend the same type of products (movies) that the active customer has shown interest to.
- Reasoning (user): This function takes the active customer as a parameter and generates the reasoning behind the recommendations. It considers all other movie features such as actors, directors and movie release date and customer's context dimensions such as time_of_week companion that are compared with the active customer's feature and preference profiles. If it finds similarity in these profiles then these attributes or features will be used to explain the reasoning or provide the justification of recommended movies. The justification style combines the "keyword based" with the influence styles. It has the following form: "Product 'X' is recommended, since it contains features which are included in products 'Z and W' that you have already rated or liked." Furthermore, the average rating of the

recommended movies will be shown to the customer as a justification. The average rating indicates the popularity of these specific movies from all the customers.

4.2.3. Screenshots of the Application

The following are various screenshots of the application:

Main Page

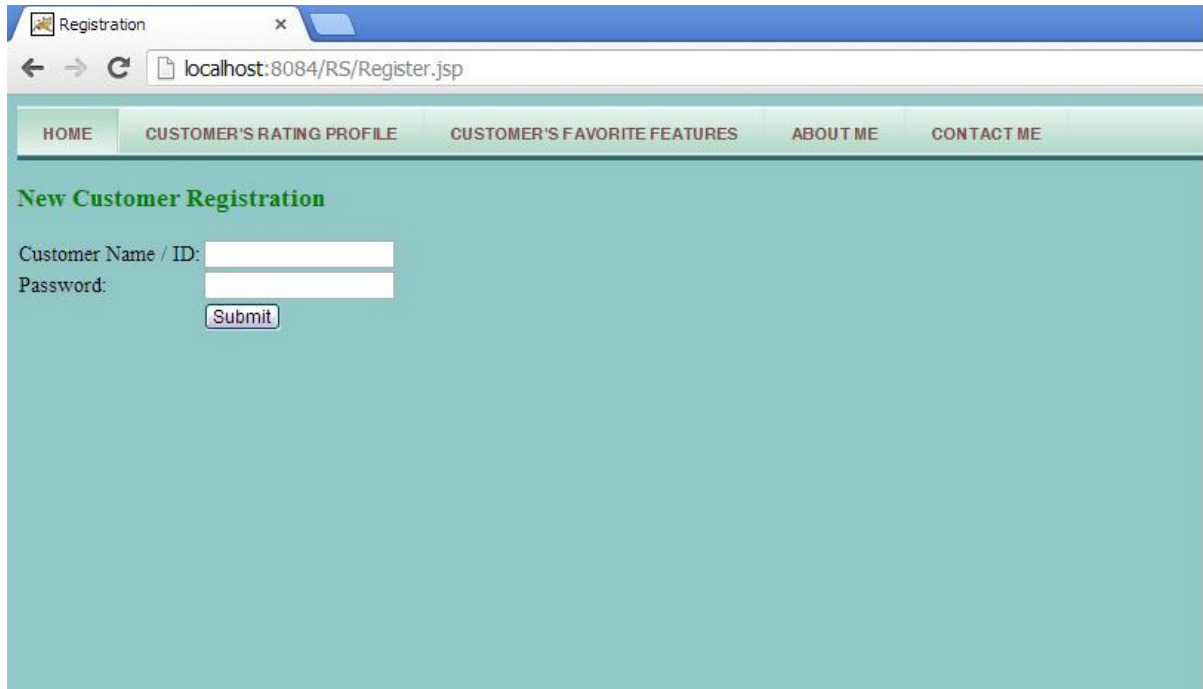
This is the main page of the application that highlights the features of this proposed hybrid recommender system. It mainly has a login form for an existing customer, as well as a navigation link for the new customer registration.



Figure 11: Home Page of Prototype Application

New Customer Registration

This page contains the customer registration form that would be used for new customers to get registered in the system.



The screenshot displays a web browser window with a single tab titled 'Registration'. The address bar shows the URL 'localhost:8084/RS/Register.jsp'. Below the address bar is a navigation menu with five items: 'HOME', 'CUSTOMER'S RATING PROFILE', 'CUSTOMER'S FAVORITE FEATURES', 'ABOUT ME', and 'CONTACT ME'. The main content area has a teal background and is titled 'New Customer Registration' in green text. The form contains two input fields: 'Customer Name / ID:' and 'Password:'. A 'Submit' button is positioned below the password field.

Figure 12: New Customer Registration

Rating Movies

Once the customer is registered the next step is to rate the movies.

HOME CUSTOMER'S RATING PROFILE CUSTOMER'S FAVORITE FEATURES ABOUT ME CONTACT ME

Registration Successful!

Please Rate Movies

Rating Scale: 1 to 5

1: Less Favorite

5: Most Favorite

Delicatessen (1991)

Groundhog Day (1993)

Star Wars (1977)

3 Ninjas: High Noon At Mega Mountain (1998)

Conspiracy Theory (1997)

Free Willy 3: The Rescue (1997)

Figure 13: Rating Form

Explanation Interface

This interface display mainly consists of two sections. The first section displays the active customer's rated movies and the favorite movie types. The second section displays the recommendations with justifications.

HOME CUSTOMER'S RATING PROFILE CUSTOMER'S FAVORITE FEATURES ABOUT ME CONTACT ME

Login Successful

Your rated movies

Angels and Insects (1995)
Desperado (1995)
Three Colors: White (1994)
Glengarry Glen Ross (1992)
Delicatessen (1991)
Groundhog Day (1993)
Hunt for Red October, The (1990)

Your Most Favorite Movie Type
Romance movies

Figure 14 (A): Explanation Interface of the Recommender System

Recommended Movies For You
Movie type: Romance Movies

1: Godfather, The (1972)



Reasoning Behind To Recommend This Movie

Popularity Among Other Users

Average Rating of this Movie is: 4/5

Content Based Knowledgeable Justifications

- Recommending you this movie because you like romnace movies
- Recommending you this movie because you like the actor(s) Marlon Brando
- Recommending you this movie because you like the director(s) Francis Ford Coppola

Context Based Knowledgeable Justifications

- Recommending you this movie because you have watched the movie on a weekend
- Recommending you this movie because you have watched romnace movies with a Significant other

Interacting With the Recommender System - Improve Your Recommendations

Give a Rating :

- I Like this Movie
- I want further recommendations on Romance movies

Figure 14 (B): Explanation Interface of Recommender System

Movie type: Action Movies

3: Star Wars (1977)



Reasoning Behind To Recommend This Movie

Popularity Among Other Users

- Average Rating of this Movie is: 4/5

Content Based Knowledgeable Justifications

- Recommending you this movie because you like action movies
- Recommending you this movie because you like the actor(s) Harrison Ford
- Recommending you this movie because you like the director(s) George Lucas

Context Based Knowledgeable Justifications

- Recommending you this movie because you have watched the movie on a weekday
- Recommending you this movie because you have watched action movies with a Friend

Interacting With the Recommender System - Improve Your Recommendations


Give a Rating :

- I Like this Movie
- I want further recommendations on action movies
- I do not want further recommendations on movies with Harrison Ford


Figure 14 (C): Explanation Interface of Recommender System

Top Watched Romance Movies
Your Most Favorite Movie type: Romance Movies

Six Degrees of Separation (1993)



Fallen (1998)



Reasoning Behind To Recommend These Movies

Popularity Among Other Users

- **Average Rating of these Movies is 4/5**

Content Based Knowledgeable Justifications

- **Recommending you these movies because you like romance movies**

Context Based Knowledgeable Justifications

- **Recommending you these movies because you have watched the movie on a weekend**
- **Recommending you these movies because you have watched romance movies with your significant other**

Figure 14 (D): Explanation Interface of Recommender System

Interacting with Recommender System

This explanation interface also allows the customer to provide feedback on the recommendations. The feedback can be provided by giving ratings on the recommendations and also by providing feedback on the products features justifications.

Movie type: Action Movies

3: Star Wars (1977)



Reasoning Behind To Recommend This Movie

Popularity Among Other Users

- Average Rating of this Movie is: 4/5

Content Based Knowledgeable Justifications

- Recommending you this movie because you like action movies
- Recommending you this movie because you like the actor(s) Harrison Ford
- Recommending you this movie because you like the director(s) George Lucas

Context Based Knowledgeable Justifications

- Recommending you this movie because you have watched the movie on a weekday
- Recommending you this movie because you have watched action movies with a Friend

Interacting With the Recommender System - Improve Your Recommendations

Give a Rating :

- I Like this Movie
- I want further recommendations on action movies
- I do not want further recommendations on movies with Harrison Ford

Figure 14 (E): Explanation Interface of Recommender System

4.2.4. Data Flow Diagrams

The following are high level data flow diagrams that were created to describe the process of obtaining recommendations from this proposed hybrid recommender systems.

Existing Customer

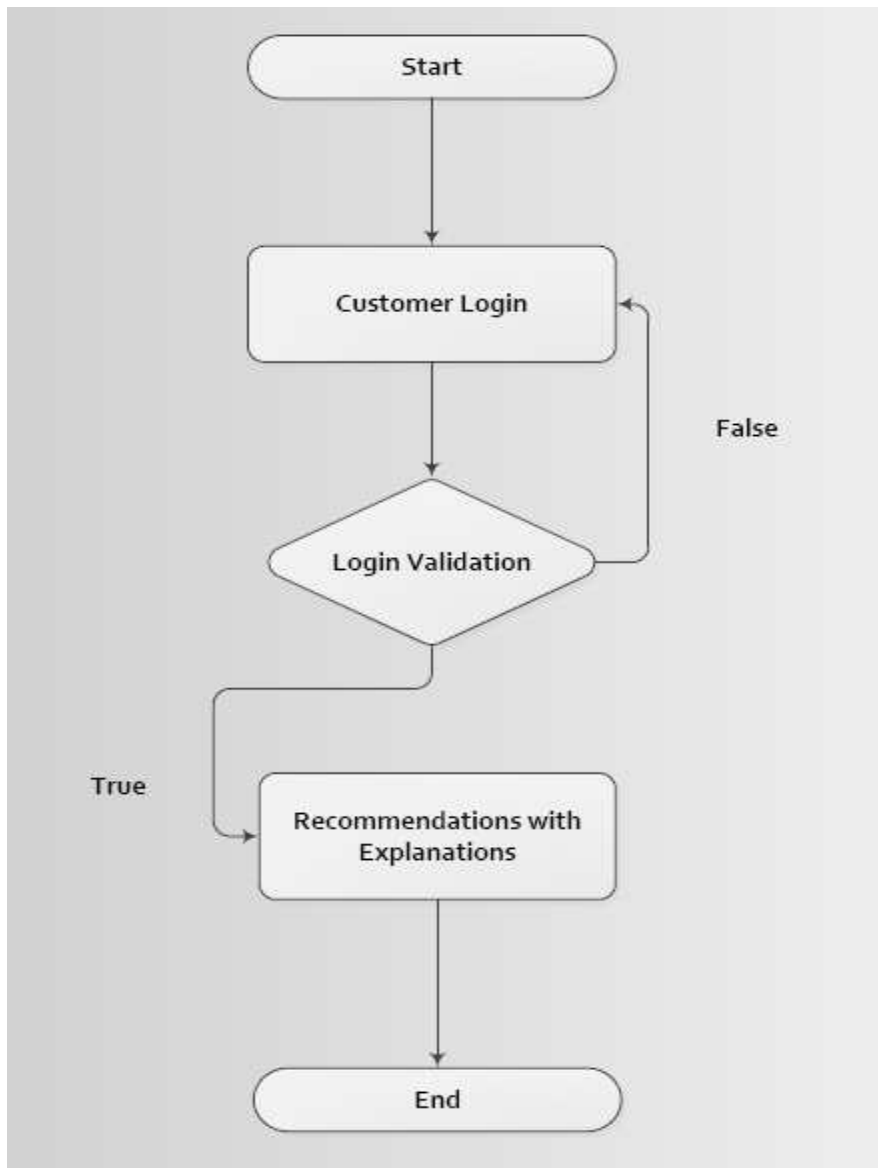


Figure 15: Recommendation Process of Old Customer

New Customer

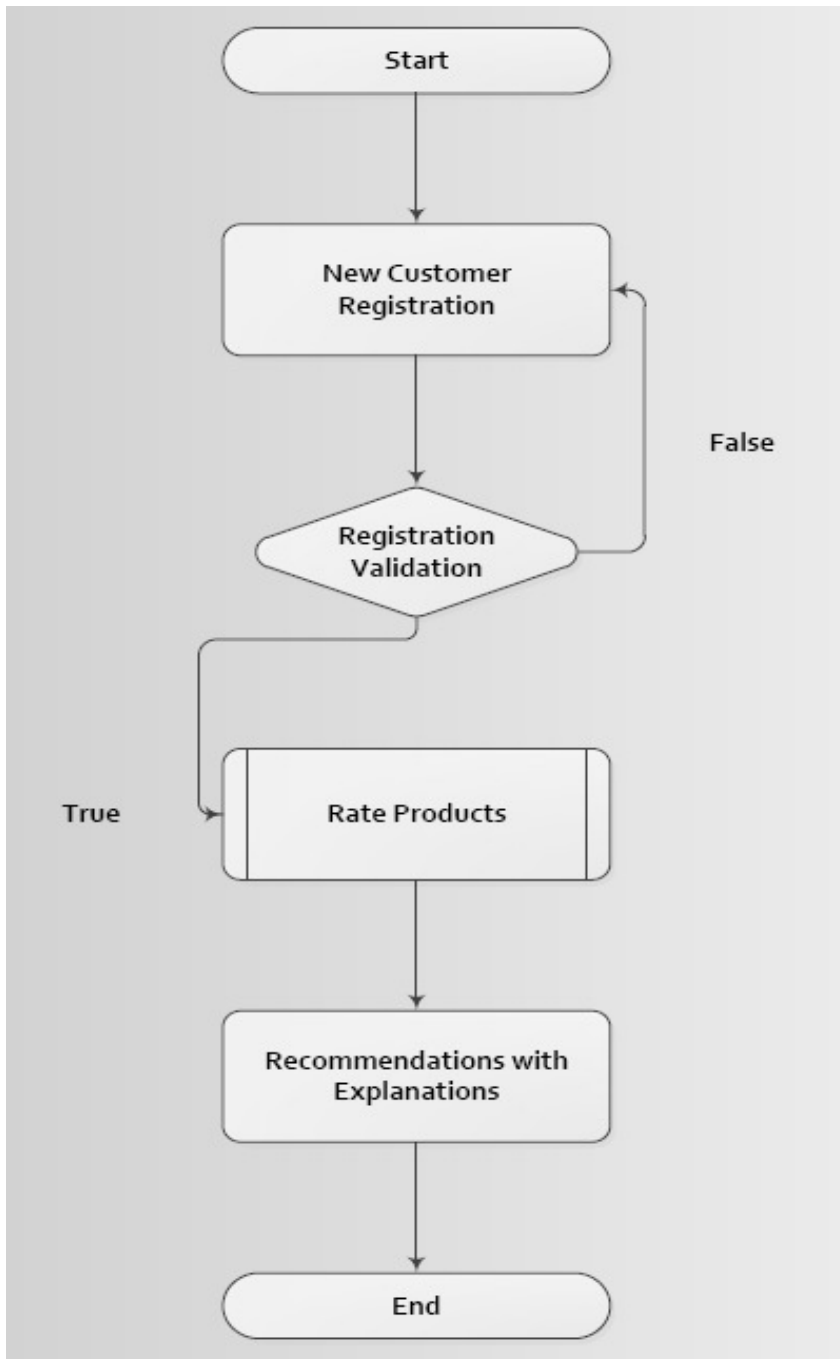


Figure 16: Recommendation Process of New Customer

Chapter 5: Experimental Evaluation

This chapter will describe the experimental methodology and the metrics that will be used to evaluate the performance of the algorithms. Furthermore, the results of the proposed approach will be detailed.

5.1. Dataset

The MovieLens dataset was used for evaluating the proposed approach. MovieLens is a web-based research recommender system that was started in 1997. Each week, hundreds of users visit MovieLens to rate and receive recommendation for movies. This recommender system has been used in many research projects. It contains 943 users, 1682 movies, and 100 000 ratings on an integer scale of 1 (bad) to 5 (excellent) [14, 25]. In this dataset all movies are not rated by all customers, but each movie must be rated by at least one of the users. It also contains 19 movie genres and a movie can belong to more than one genre. A binary value (0 or 1) is used to indicate whether a movie belongs to a specific genre. The dataset is divided into disjoint training (80%) and test (20%) sets. The training and test data was used to evaluate the evaluation measures [31]. The training dataset was used to evaluate the evaluation measures and the test dataset was used to compare the results with the predicted results.

5.2. Evaluation Metrics

There are various metrics that have been used to evaluate the recommendation algorithms in the literature. Systems that generate predictions should be considered separately from systems whose output is a top-N recommendation, hence distinct evaluation schemes or metrics should be considered in each case. Researchers have classified these measures in the following categories [11, 12, 14, 25]:

- Metrics evaluating Statistical Accuracy
- Metrics evaluating Decision Support Accuracy
- Metrics evaluating top-N Recommendation Quality
- Metrics evaluating Performance

In order to evaluate this proposed approach, the metrics that evaluates statistical accuracy of recommendations and top-N recommendations will be used. Furthermore, a survey will be conducted to evaluate the overall performance of a justification-based recommender system's interface. The evaluating measures are described as follows:

Metrics evaluating Statistical Accuracy

Statistical accuracy metrics evaluate the accuracy of a recommender system or predictor by comparing predicted values with customer provided values. Mean Absolute Error (MAE) is a widely used metric for this. It measures the average absolute deviation between a recommender system's predicted rating and actual rating given by the user [11, 12, 14, 25]. The MAE is measured only for those products, for which the active customer u_i has expressed his opinion. It can be calculated with the following equation, where ar_{ij} and r_{ij} indicate the actual ratings and predicted ratings respectively, and n_i are the total rated products by the active customer. Lower MAE values indicate good performance of a recommender system algorithm [14].

$$MAE_i = \frac{\sum_{j=1}^{n_i} |ar_{ij} - r_{ij}|}{n_i} \quad (13)$$

Metrics evaluating Decision Support Accuracy

Decision support accuracy metrics measure how well a recommender system or predictor helps customers choose their desired products. Receiver operating characteristic (ROC) sensitivity has been used for this objective [14, 18]. It assumes the prediction process is a binary operation and products are predicted as either good or bad. A predictor can be treated as a filter, where predicting a high rating for a product is equivalent to accepting the product, and predicting a low rating is equivalent to rejecting the product. In [12], the authors define that the ROC sensitivity is given by the area under the ROC curve— a curve that plots sensitivity versus specificity for a predictor. Sensitivity is defined as the probability that a good product is accepted by the filter; and specificity is defined as the probability that a bad product is rejected by the filter. They consider a product good if the customer gave it a rating

of 4 or above, otherwise it is considered bad. They referred to this ROC sensitivity with a threshold of 4 as ROC-4

Metrics evaluating top-N Recommendation Quality

These metrics are used to evaluate the quality of top-N recommendations. The main focus in evaluating such a system is to find out the value of that list or to find out whether the customer would be interested in rating or purchasing some or all the items included in that top-N list. Precision and recall are the two most widely-used measures for this purpose [11, 47, 48]. Basically, these measures are used to evaluate search strategies. These measures assume that there is a set of records in the database which are either relevant or irrelevant to the desired query and the actual retrieval set may not perfectly match the set of relevant records.

The precision of an algorithm is the percentage of suggested products that were actually desirable products or the ratio of the number of relevant records retrieved to the total number of irrelevant and relevant records retrieved. A high precision is best [11, 47, 48]. Recall is the percentage of relevant products returned by the algorithm or the ratio of the number of relevant records retrieved to the total number of relevant records in the database. A high recall rate is best [11, 47-49]. In other words, precision is the proportion of recommended movies that are actually good and recall is the proportion of all good movies recommended. They can also be described using the following equations [50]:

$$Precision = \frac{tp}{tp + fp} = \frac{|good\ movies\ recommended|}{|all\ recommendations|} \quad (14)$$

$$Recall = \frac{tp}{tp + fn} = \frac{|good\ movies\ recommended|}{|all\ good\ movies|} \quad (15)$$

Metrics evaluating Performance

There are various evaluation techniques that are considered for the performance of the system. Response time is a widely-used performance metric, which has been used for various purposes and in different domains. In the case of recommender systems, it defines the time that has elapsed between a customer's stated request and the system's response to that

request. A recommender system must be able to efficiently guide a customer through a product-space and, in general, short recommendation sessions are to be preferred. For this evaluation, researchers measure the length of a session in terms of recommendation cycles, i.e, the number of products viewed by users before they accepted the system's recommendation. User's perceived satisfaction about a recommendation algorithm is obtained by conducting a survey in the form of a questionnaire [11, 12]. Another way of evaluating a filtering algorithm is through storage requirement. It is very common to expect online stores, such as Amazon and eBay, to provide services that reach millions of customers and have an even greater number of products. Hence, it is crucial to evaluate how these systems manipulate the space provided to them. Storage requirements are usually analyzed in two ways: by checking their main memory requirement, which represents the on-line space usage of the system, and by checking their secondary storage requirement, which refers to the off-line space usage of the system [11].

Coverage measures the percentage of items for which a recommender system is capable of making predictions [14]. In some cases, a recommender system will not be able to generate a prediction for specific products due to the sparsity in the data of the initial customer product matrix or other restrictions, which are set during the recommender system's execution. Such cases will lead to low coverage values [11]. A low coverage value indicates that the recommender system will not be able to help the customer with many of the products he has not rated and a high coverage value indicates that the recommender system will be able to generate useful recommendations of products for the customers. Assuming that n_i are the products for which customer u_i has given a rating, and np_i is the number of those products for which the recommender system was able to generate a prediction, where clearly np_i less than or equal to n_i , then coverage can be calculated using the following equation [11]:

$$Coverage = \frac{\sum_{i=1}^m np_i}{\sum_{i=1}^m n_i} \quad (16)$$

5.3. Experimental Procedure

The proposed hybrid recommender system generates recommendations in the form of top-N recommendations. Algorithm 1 generates item-based recommendations and algorithm 2 generates final recommendations with justifications to the customers. After algorithm 2, the number of recommendations can decrease from the initial item-based recommendations. The following measures are applied to the final recommendations. The training and test data from MovieLens dataset is used to evaluate these measures. The predicted values from training data were compared with the actual values in the test data.

5.3.1. Evaluation Measures

Precision and Recall

As described previously, precision and recall are the most important measures for evaluating top-N recommendations. Algorithm 4 describes the calculation process for precision and recall measures. The threshold value was required to select the highest predicted value recommendations. This value is based on the predicted preference value that is determined by algorithm 1. This value is set to greater than or equal to 3.5; hence it selects those recommended products that meet this condition.

Mean Absolute Error (MAE)

It measures the average absolute deviation between a recommender system's predicted ratings and the actual rating given by the user [11, 12, 14, 25]. It can be calculated with equation 14 as described in algorithm 4.

Algorithm 4: Proposed Algorithm for Calculation of Evaluation Measures

Input:

A= List of actual ratings from the test dataset

N= List of recommendations

Output:

Precision and Recall

MAE

Main Procedure: Calculate the evaluation measures for active customer

-
- 1- Set V = relevance threshold value
 - 2- Set relevantItem = 0(zero)
 - 3- Set irRelevantItem = 0(zero)
 - 4- Set allRelevantItems = 0(zero)
 - 5- Select highest recommendations from the actual ratings (test dataset)
For $i = 1$ to recommendations.size (A)
 If actual rating (i) $\geq V$ then
 Set allRelevantItems = allRelevantItems + 1
 End If
End For
 - 6- Select relevant and irrelevant recommendations from the final recommendations(N)
For $i = 1$ to recommendations.size (N)
 Set $p =$ recommendations.get(i).getValue (Predicted preference value)
 If $p \geq V$ then
 If $p(i) =$ actual rating (i)
 Set relevantItem = relevantItem + 1
 Else
 Set irRelevantItem = irRelevantItem + 1
 End If
 End If
End For
 - 7- Calculate Precision using equation 14
 Set Precision = $\text{relevantItem} * 100 / \text{relevantItem} + \text{irRelevantItem}$
 - 8- Calculate Recall using equation 15
 Set Recall = $\text{relevantItem} * 100 / \text{allRelevantItems}$
 - 9- Calculate MAE using equation 13
 Set MAE = $\text{Sum}(\text{value } p(i) - \text{actual rating (i)}) / \text{Count of actual ratings from list A.}$
-

5.3.2. Survey Participants and Materials

A survey was conducted to evaluate the proposed hybrid recommender system's explanation interface. The objective of this survey was to measure the overall performance of a justification-based recommender system. The participants volunteered to take part in this survey. Fifty participants were selected that have been using e-commerce web sites for their shopping needs. The majority were university students from different disciplines. Most of them were Master's students from computer science, electrical engineering and other engineering disciplines. Some of them were professional people who have online shopping experience.

SurveyMonkey was used to conduct the survey [51]. This is a very useful web-based tool that is used to design a survey, collect responses and analyze the results [51]. The proposed hybrid recommender system's interface was presented to the participants with various survey questions, as listed in the table 2. Furthermore, other related justification based recommender systems [19, 20] were also presented to the participants for comparison with the proposed system. This survey was done online by sending the participants the URL and also in real-time by using the system. Researchers have been using various types of measurements to evaluate the overall quality of recommender systems. Many of them claimed that statistical accuracy metrics (MAE) and evaluating top-N recommendations (precision and recall) can only partially evaluate the recommender system. It has been noticed that customer satisfaction and trust on recommender systems are the most crucial measurements to evaluate the overall performance of recommender system [18, 22]. Hence, these survey questions are specifically designed to consider these measurements. The following set of questions were prepared and asked to each individual in this survey [23]. The participants were required to give their answers to all of the following questions in the form of YES or NO, in addition to comments on each question.

Evaluation Purpose	Questions
Recommendation quality	Does this hybrid recommender system give you very good movie recommendations?
Perceived ease of use	Do you find this interface easy to use?

Perceived usefulness	Is this explanation interface (with content and context-based justifications) competent to help you effectively in finding movies (products) that you really like? Do you find this interface useful in comparison to the other given interfaces in order to improve your shopping performance?
Decision confidence	Are you confident that the recommended movie is really the best choice for you?
Intention to purchase	If you had to search for a movie (product) online in the future and an interface like this was available, would you be very likely to use it?

Table 2: Survey Questions

5.3.3. Experimental Platform

As mentioned in the chapter 4 (4.2) this application is implemented and tested in Java programming language. Furthermore, this application is tested on Windows XP based machine with Intel Core Duo CPU processor having a speed of 2.26GHz and 1.98GB of RAM.

5.4. Experimental Results

Since this recommender system has an explanation-based interface, recommendations are justified to the customers. For all used evaluation measures such as precision, recall, and Mean Absolute Error (MAE), this proposed recommender system is evaluated with various users on the training and the test dataset was used to compare the results. This proposed recommender system is able to generate the recommendations for existing and new customers because this dataset does not have sparsity. It was tested with the various random customers and also for the new customer as well. However, the system needs a good feature and preference profiles for the existing and new customer to generate valuable recommendations. The obtained values indicate that this proposed algorithm is able to generate maximum valuable recommendations for the active customer, as shown by precision and recall values in the figure below. These values can lie between 0 and 1 and the maximum value is best. However, these precision and recall values depend on the threshold

value that was set in the calculation of these measures. The threshold value was set to maximum in order to generate valuable recommendations. Finally, MAE (Mean Absolute Error) values clearly indicate that this proposed recommender system is able to generate accurate predictions as shown in the figure below. Lower MAE values indicate that this proposed recommender system predicts more accurate customer's ratings.

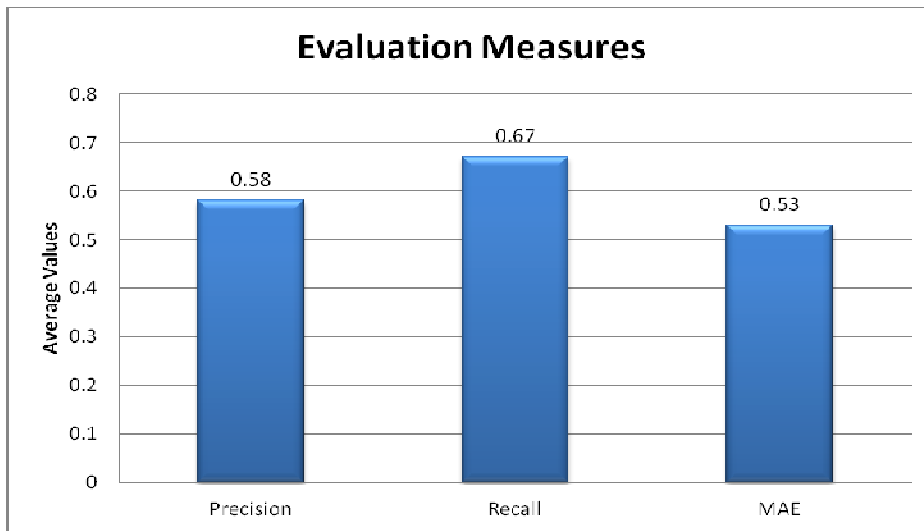


Figure 17: Evaluation Measure's Results

The tables below are based on the participant's answers of the survey questions. Approximately, 30% of the participants completed the survey using the system in real time. The rest were sent various snapshots of the proposed system and other related systems. Both groups of participants had almost the same experiences, except that the real time participants understood the system better because they went through all the recommendation process. It was clear that the majority of both types of participants preferred to have recommendations with their justifications. Their answers clearly described that they received valuable recommendations on which they can trust. Especially, both content and context-based justifications were very helpful to provide strong reasoning behind these recommendations. Furthermore, they found the explanation-based interface easy to use. Some of their comments described that it is easy to use, as long as the interface requires minimal information. In addition to this, the majority of the participants really liked the method of displaying the justifications with a recommended product where the interface does not

require any further input from the customer. However, some of them mentioned that in-depth justifications for the recommendation should show up only once the recommended product is clicked, or perhaps if the mouse hovers over it.

Number	Questions	Yes	No
1	Does this hybrid recommender system give you very good movie recommendations?	93.3%	6.7%
2	Do you find this interface easy to use?	100%	0 %
3	Is this explanation interface (with content and context-based justifications) competent to help you effectively in finding movies (products) that you really like?	93.3%	6.7%
4	Do you find this interface useful in comparison to the other given interfaces in order to improve your shopping performance?	93.3%	6.7%
5	Are you confident that the recommended movie is really the best choice for you?	100%	0%
6	If you had to search for a movie (product) online in the future and an interface like this was available, would you be very likely to use it?	100%	0%

N (number of participants who used the system in real time) =15

Table 3: Survey's Results[1]

Number	Questions	Yes	No
1	Does this hybrid recommender system give you very good movie recommendations?	91.4%	8.6%
2	Do you find this interface easy to use?	97.2%	2.8%
3	Is this explanation interface (with content and context-based justifications) competent to help you effectively in finding movies (products) that you really like?	91.4%	8.6%
4	Do you find this interface useful in comparison to the other given interfaces in order to improve your shopping performance?	94.3%	5.7%
5	Are you confident that the recommended movie is really the	91.4%	8.6%

	best choice for you?		
6	If you had to search for a movie (product) online in the future and an interface like this was available, would you be very likely to use it?	97.2%	2.8%

N (number of participants who viewed images of the system) =35

Table 3: Survey's Results[2]

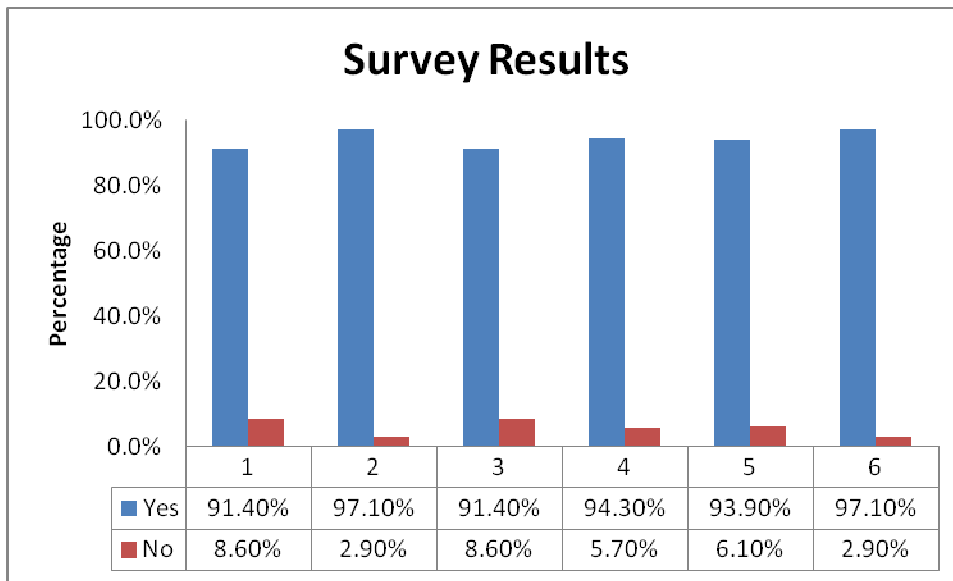


Figure 18: Survey Results of Both Groups Combined)

In addition to this, the participants liked the style of the explanation interface that displays the recommendations by category rather than displaying them individually with some of the same information. Their comments described that it saves time when making a decision and attracts the customer more effectively. Judgments on whether the product is the best will be clearer once the product is received, but that they are certainly more confident about the quality of their purchase than would be if there was no recommendation software.

In comparison with the other related proposed recommender systems [12, 19, 20], this system works different in various ways. First this system uses item-based collaborative filtering rather than user-based collaborative filtering. Secondly, this system uses more product features and a customer's context attributes (dimensions) to provide high-quality

knowledgeable justifications for recommendations. In addition, it uses a categorized presentation mechanism to display recommendations with justifications. Table 4 below shows a comparison between the proposed explanation interface and that by Tinarev [19] which is content-based explanation interface of recommender systems. Finally, this system allows the customer to provide feedback on the recommendations and justifications. This feedback can be provided by giving a rating on the recommended movies, as well as by modifying the previous ratings. In addition, customers can make suggestions on the justifications that are based on a product’s features in order to revise the recommendations in the following ways.

- I want further recommendations on Action movies
- I do not want further recommendations on movies with the actor Al Pacino.
- I do not want further recommendations on movies directed by George Lucas.

Questions Addressed in the Paper [19]	Qualitative Results	My Proposed Approach Results
Number of Participants	67	50
How data was gathered	Through focus group survey	Online survey
Compared a graphical with textual interface for recommender systems	Most of the users preferred graphical interface	Graphical interface
Recommendation quality (Effectiveness)	Users found that the recommendations are useful with the content-based justifications. However, they suggested that uncertain recommendations should simply be omitted.	This system generates accurate recommendations that contain content and context-based justifications (therefore the recommendations are more personalized and less uncertain)
Design of the recommender system interface	Users found that the interface was not very efficient and suggested more details and possible improvements to the interfaces.	This proposed interface was easy to use (97.1% of users) and more efficient because it displays recommendations categorically and eliminates duplicates

Table 4: Comparison Results

5.5. Limitations of Evaluation Approach

This approach is the integration of content and context data with rating data to provide valuable recommendations with justifications. The MovieLens dataset does not contain content and context data; therefore, it was modified accordingly for evaluation purposes as explained in chapter 4. We had to manually add the required data to the MovieLens database since it was required for evaluation purposes. In our literature review, to the best of our knowledge, no other context aware recommender system is evaluated with the MovieLens dataset. Hence, we could not compare our final recommendations quantitatively with other related recommender systems. An extrapolation for future work would be to apply our system to multiple other data sets and compare their performance both quantitatively and qualitatively.

Since this proposed system presents justifications along with final recommendations, we had to measure their quality using human evaluators. For this purpose, a survey was conducted as justifications could not be measured quantitatively. The survey was conducted to evaluate the overall performance of the proposed recommender system. This survey was completed online by sending the participants the URL and also in real-time by using the system. Approximately, 30% of the participants completed the survey using the system in real time. The rest were sent various snapshots of the proposed system and other related systems. It was done this way as it was not feasible for all of the participants to interact with the system in real-time.

5.6. Discussion

This section describes how this proposed approach is different from other related approaches and how it provides accurate recommendations with the justifications for existing and new customers.

5.6.1. Comparison with Related Work

This proposed hybrid recommender is related to the work of the authors who integrated collaborative, content-based and context-based filtering approaches. The proposed approach was inspired by various authors from [12, 19, 20], who also proposed hybrid recommender

systems. According to these works, the integration of a product's content with rating data has been shown to give good recommendations and also provided justifications against its recommendations. It has been noticed by many researchers that a lot of work is required to provide effective explanations that would be useful to boost the customer's trust on a recommender system. In addition to this, a very attractive explanation interface is required to display the recommendations along with the justifications, thereby reducing the customer's decision making time and effort to interact with the recommender system.

Hence, this proposed approach mainly focuses on the above-mentioned two major challenges. This system is mainly different in various factors. Firstly, item-based collaborative filtering rather than user-based collaborative filtering was used, since user-based collaborative filtering has some major limitations. Second, an attractive categorized explanation interface was created to display the same type of recommendations in a category format rather than duplicating the same information. Product features and active customer's context dimensions are used to create useful explanations for recommendations. Therefore, customers would get only those recommendations for which they can get justifications. Finally, this proposed recommender system allows the customer to interact with it to provide feedback on the recommendations and also on the justifications. Once the feedback is there the recommender system can revise the recommendations for the active customer accordingly.

For the new customer, this proposed system created preference and feature profiles. For creating a good profile, a new customer has to rate at least ten products. Once these profiles have been created, the system can generate good recommendations and justifications for the new customer. A customer's feature and preference profiles play a key role in this proposed approach as this matches the contents of profiles with the contents of recommended products as described in the proposed algorithm². In the results, the system will recommend only those products that have even one of the features in the current customer's profiles. Hence, this proposed approach only recommends those products for which it can provide justifications.

With the comparison of [19, 20], Netflix and AMAZON, this proposed system uses more product features and active customer's context dimensions in order to provide quality recommendations and knowledgeable justifications for recommendations. Furthermore, a categorized presentation mechanism is utilized to display recommendations with justifications. Finally, this system allows the customer to provide feedback on the recommendations and also on justifications. This feedback can be provided by giving a rating on the recommended movies and also by making a suggestion on the justifications that are based on a product's features as follows.

- I want further recommendations on action movies
- I do not want further recommendations on movies with the actor Al Pacino.
- I do not want further recommendations on movies directed by George Lucas.

Our survey results clearly show that the number of features for a product and customer's context plays a crucial role in obtaining the satisfaction and trust of the customers and additionally a categorized presentation mechanism is really liked by the customers as it reduces decision making time.

Chapter 6: Conclusion and Future Research Directions

6.1. Conclusion

Recommender systems have made significant progress over the last years since hybrid recommendation methods have been proposed and implemented. My proposed hybrid approach mainly focuses on providing justifications for the recommendations. This proposed hybrid approach is the integration of content and context data with rating data, since it boosts the prediction performance of a recommender system and also provides accurate justifications for recommendations. A product's features and active customer's context dimensions have been used to create a customer's feature and preference profiles that mainly play a crucial role in providing justifications. Hence, this proposed approach only recommends those products for which it can provide justifications. Experimental results indicate that the majority of participants preferred to have recommendations with their justifications.

Furthermore, this proposed system adopts a categorized explanation interface that displays the recommendations in a group or a category rather than duplicating the same information which reduces the time for decision making of customers. Furthermore, this system will allow the customer to interact with it in order to provide feedback on the recommendations and justifications. The results have clearly shown that interact with the customer more effectively and boosts the customer's satisfaction on the recommender system. Interacting with the recommender system allows the customer to achieve their desired product quicker. This proposed approach is implemented by a prototype web-based application in the JAVA platform. This prototype is implemented for movies; however it can be easily implemented for other products. However, despite all of these advances, the current generation of recommender systems still requires further improvements to make recommendation methods more effective in a broader range of applications. Specifically, there is lot of work needed in the area of providing effective explanations that will increase the customer's trust on the recommender systems and also boosts the business of the organization.

6.2. Future Research Directions

As this proposed approach focuses on providing explanations against recommendations, there are different areas of explanations in which there is room for improvement.

- Explanations have shown very good results in reasoning-based recommender systems and expert systems. It removes the black box from the recommender system, and provides transparency. It has been noticed that customers like and feel more confident in recommendations perceived as transparent. However, a lot of work is required to investigate various mechanisms in order to achieve system transparency. Especially, crucial challenges are to achieve more meaningful explanations from various computational models that would be useful to boost the trust in recommender systems.
- A product's features have been used by researchers to provide explanations against recommendations. Therefore, in order to have a sufficient set of features, the content must either be in a form whose features can be extracted automatically by a computer (e.g., text). Research on automatic feature extraction methods are required, especially to obtain features from multimedia data.
- Researchers have been working on generating accurate recommendations. However, presenting these accurate recommendations with explanations in a way that attracts the customer more effectively is an open issue. Explanations need an attractive and usable recommender system interface that would display all the recommendation's information in an organized way that is useful to minimize the customer's decision making time.
- This proposed recommender system allows the customer to interact with it to provide feedback on the recommendations. Specifically, customers can make a suggestion on the justifications that are based on a product's features in order to revise their recommendations. Future work can include a recommender system that is able to

revise the recommendation for the active customer once the suggestions or feedback has been provided.

7. References

- [1] H. Shimazu. Expertclerk: Navigating shoppers' buying process with the combination of asking and proposing. In proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, San Francisco, CA , pp. 1443-448, ACM, 2001.
- [2] D. Cosley, S. K. Lam, I. Albert, J. A. Konstan, and J. Riedl. Is Seeing Believing? How Recommender Systems Influence Users' Opinions. In Proceedings of CHI 2003: Human Factors in Computing Systems, New York, NY, pp. 585-592, ACM, 2003.
- [3] A. Edmunds, and A. Morris. The problem of information overload in business organisations: a review of the literature. In International Journal of Information Management, Volume 20, Issue 1, pp. 17-28, Elsevier Science, 2000.
- [4] D. W Oard. The State of the Art in Text Filtering. In User Modeling and User-Adapted Interaction, Volume 7, Issue 3, pp. 141-178, Springer, 1997.
- [5] J. B. Schafer, J. Konstan, and J. Riedl. E-Commerce Recommendation Applications. In the Journal of Data Mining & Knowledge Discovery, vol. 5, pp. 115-153, Springer, 2001.
- [6] G. Linden, B. Smith, and J. York. Amazon.com Recommendations: Item-to-Item Collaborative Filtering. In Internet Computing, Volume 7, Issue 1, pp.76-80, IEEE, 2003.
- [7] M. Pazzani, J. Muramatsu, and D. Alexander Billsus. Syskill&Webert: Identifying interesting web sites. In Proceedings of the National Conference on Artificial Intelligence, Irvine, CA, pp.69-77, 1996.
- [8] R. Burke. Hybrid Recommender Systems: Survey and Experiments. In User Modeling and User-Adapted Interaction, Volume 7, Issue 3, pp. 331-370, Kluwer Academic Publishers, 2002.
- [9] R. Burke. The FindMe Approach to Assisted Browsing. In IEEE Expert, Volume 12 , Issue 4 , pp. 32-40, IEEE, 1997.
- [10] J. B. Schafer, J. Konstan, and J. Riedl. Recommender Systems in E-Commerce. In Proceedings of the 1st ACM conference on Electronic commerce, New York, ACM, 1999.
- [11] E. Vozalis, and K. G. Margaritis. Analysis of recommender systems algorithms. In Proceedings of the Sixth Hellenic-European Conference on Computer Mathematics and its Applications, 2003.
- [12] M. Perm, R. J. Mooney, and R. Nagarajan. Content boosted collaborative filtering for improved recommendations. In Proceedings of Eighteenth National Conference on Artificial Intelligence, Edmonton, Canada, 2002.
- [13] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Recommender Systems for Large-scale E-Commerce: Scalable Neighborhood Formation Using Clustering. In Proceedings of the Fifth International Conference on Computer and Information Technology, 2002.
- [14] M. A. Ghazanfar, and A. Prugel-Bennett. A scalable, accurate hybrid recommender system. In Proceedings of International Conference on Knowledge Discovery and Data Mining, Phuket, pp. 94-98, IEEE, 2010.
- [15] R. Burke. Hybrid Recommender Systems: Survey and Experiments. In User Modeling and User-Adapted Interaction, Volume 12, Issue 4, pp. 331 - 370, 2002.

- [16] Puntheeranurak. A Multi-Clustering Hybrid Recommender System. In Proceedings of Seventh International Conference on Computer and Information Technology, Tokyo, pp. 223-228, IEEE,2007.
- [17] M. A. Ghazanfar, and A. Prugel-Bennett. Fulfilling the Needs of Gray-Sheep Users in Recommender Systems, A Clustering Solution. In Proceedings of International Conference on Information Systems and Computational Intelligence, Harbin, China, Eprints, 2011.
- [18] P. Massa, and Paolo Avesani. Trust-aware Recommender Systems. In Proceedings of ACM conference on Recommender systems, New York, NY, ACM, 2007.
- [19] N. Tintarev. Explanations of recommendations. In Proceedings of the ACM conference on Recommender systems, pp. 203-206, ACM,2007.
- [20] P. Symeonidis, A. Nanopoulos, and Y. Manolopoulos. Providing justifications in Recommender System. In IEEE Transactions on Systems, MAN, and Cybernetics—Part A: Systems and Humans, vol. 38, Issue 6, pp: 1262 - 1272, IEEE,2008.
- [21] G. Adomavicius, and A. Tuzhilin. Toward the next generation of recommender system: a survey of the state-of-the-art and possible extensions. In IEEE Transactions on Knowledge and Data Engineering, vol,17, issue 6. pp. 734-749, IEEE,2005.
- [22] N. Tintarev, and J. Masthoff. A Survey of Explanations in Recommender Systems. In Proceedings of IEEE conference on data engineering, Istanbul, 2007.
- [23] L. Chen, and P. Pu. A Cross-Cultural User Evaluation of Product Recommender Interfaces. In Proceedings of ACM conference on Recommender systems, New York, pp, 75-82, ACM,2008.
- [24] C. D. Manning, P. Raghavan, and H. Schtze. Introduction to Information Retrieval. New York: Cambridge University, 2008.
- [25] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-Based Collaborative Filtering Recommendation Algorithms. In Proceedings of the 10th International Conf on World Wide Web, New York,Pages 285-295, ACM,2001.
- [26] S. Chen, T. Luo, W. Liu, and Y. Xu. Incorporating similarity and trust for collaborative filtering. In Proceedings of the the International Conference on Fuzzy Systems and Knowledge Discovery, Beijing, 2009.
- [27] Q. Li, and B. Man Kim. Clustering Approach for Hybrid Recommender System. In Proceedings of the International Conference on Web Intelligence, South Korea, IEEE , 2003.
- [28] G. Adomavicius, S. RAMESH, S. SHAHANA, and T. ALEXANDER. Incorporating Contextual Information in Recommender Systems Using a Multidimensional Approach. In ACM Transactions on Information Systems, vol. 23, Issue 1, pp. 103-145, ACM, 2005.
- [29] G. Adomavicius, and A. Tuzhilin. Context-Aware Recommender Systems. pp. 1-34, Springer, 2001.
- [30] F. Ricci. Context-Aware Music Recommender Systems. In Proceedings of the WWW – AdMIRe'12 Workshop, Lyon, France, 2012.
- [31] B. Chikhaoui. An Improved Hybrid Recommender System by Combining Predictions. In Proceedings of the International Conference on Advanced Information Networking and Applications, Biopolis, pp. 644-649,IEEE,2011.
- [32] T. Tran. Designing Recommender Systems for E-Commerce: An Integration Approach. In Proceedings of the 8th international conference on Electronic commerce, New York, pp. 512 - 518. ACM,2006.

- [33] F. Lorenzi. A Multiagent Knowledge-Based Recommender Approach With Truth Maintenance. In Proceedings of the ACM conference on Recommender systems, New York, Pp. 195-198, ACM,2007.
- [34] L. Martínez. REJA: A Georeferenced hybrid recommender systems for restaurants. In Proceedings of the International Joint Conferences on Web Intelligence and Intelligent Agent, Milan, Italy, IEEE,2009.
- [35] S. Puntheeranurak, and H.Tsuji. A Multi-Clustering Hybrid Recommender System. In Proceedings of the International Conference on Computer and Information Technology, Aizu-Wakamatsu, Fukushima, pp. 223 - 228, IEEE,2007.
- [36] M. Rey-López¹, A. Belén Barragáns-Martínez, A. Peleteiro, and F. A. Mikic-Fonte. moreTourism: Mobile Recommendations for Tourism. In Proceedings of the International Conference on Consumer Electronics, Las Vegas, NV, pp. 347 - 348, IEEE,2011.
- [37] L. Iaquinta, A. Lisa Gentile, P. Lops, M. de Gemmis and G. Semeraro. A Hybrid Content-Collaborative Recommender System Integrated into an Electronic Performance Support System. In Proceedings of the Seventh International Conference on Hybrid Intelligent Systems, Kaiserslautern, pp. 47 - 52, IEEE, 2007.
- [38] B. Chikhaoui, M. Chiazzaro, and S. Wang. An Improved Hybrid Recommender System by Combining Predictions. In Proceedings of the International Conference on Advanced Information Networking and Applications, Biopolis, pp. 644 - 649, IEEE, 2011.
- [39] D. Jannach, and G. Friedrich. Tutorial: Recommender Systems. In Proceedings of the International Joint Conference on Artificial Intelligence, Barcelona, 2011.
- [40] (January 05, 2013). Introduction to Database View. Available: <http://www.mysqltutorial.org/introduction-sql-views.aspx>
- [41] (29-04-2012). 3-Tier Architecture. Available: <http://channukambalyal.tripod.com/NTierArchitecture.pdf>
- [42] (03-07-2012). Internet Movie Database. Available: <http://www.imdb.com/>
- [43] (Sunday, January 06, 2013). Managing Database Index in MySQL. Available: <http://www.mysqltutorial.org/mysql-create-drop-index.aspx>
- [44] (Sunday, January 06, 2013). Optimization and Indexes. Available: <http://dev.mysql.com/doc/refman/5.5/en/optimization-indexes.html>
- [45] (04-05-2012). Microsoft support. Available: <http://support.microsoft.com/kb/196271>
- [46] (02-05-2012). Mahout. Available: <https://cwiki.apache.org/confluence/display/MAHOUT/Overview>
- [47] C. Liu, C. Sun, and J. Yu. The Design of an Open Hybrid Recommendation System for Mobile Commerce. In Proceedings of the International Conference on Communication Technology, Hangzhou, pp. 129 - 134, IEEE,2008.
- [48] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, San Francisco, pp. 43-52, ACM,1998.
- [49] R. Jizba. Measuring Search Effectiveness. Creighton university, 2007.
- [50] (02-07-2012). Survey Monkey. Available: <http://www.surveymonkey.com>

Appendices

Appendix A

This appendix contains JSP and HTML code from the various developed web pages that performs the main functionalities in the proposed system. The following code validates the customer login credentials from the database:

```
<%
    String user = request.getParameter("user");
    String pass = request.getParameter("pass");
    Connection conn = null;
    String url = "jdbc:mysql://localhost:3306/";
    String dbName = "hybridrecommender";
    String driver = "com.mysql.jdbc.Driver";
    String userName = "root";
    String password = "raza";
    Statement st;
    HashMap ratedMoviesRDateMap = new HashMap();
    try {
        Class.forName(driver).newInstance();
        conn = DriverManager.getConnection(url + dbName, userName, password);
        String sql = "select * from users where user_id='" + user + "' and password='" +
pass + "'";
        ResultSet rs;
        st = conn.createStatement();
        rs = st.executeQuery(sql);
        int count = 0;
        while (rs.next()) {
            count++;
        }
        if (count > 0) {
            out.println("<h2>");%>
<font size="+2" color="red">
            Login Successful
        <%
        } else {

            response.sendRedirect("index.jsp");
        }
    }%>
</font>
```

The following code registers the new customer into the system and also stores the rating information in the system

```
<%
```

```

String user = request.getParameter("user");
String pass = request.getParameter("pass");
Connection connection = null;
Class.forName("com.mysql.jdbc.Driver").newInstance();
connection=
DriverManager.getConnection("jdbc:mysql://localhost:3306/hybridrecommender", "root",
"raza");
Statement statement = connection.createStatement();
String sql = ("INSERT INTO users (user_id, password) VALUES (" + user + "," +
pass + ") ");
statement.executeUpdate(sql);
%>
<h2>
Registration Successful!
<hr />
<br />
<font color="purple">
Please Rate Movies</font>
</h2>
<h3> <font color="purple">Rating Scale: 1 to 5</font></h3>
<h4> <font color="purple">1: Less Favorite</font></h4>
<h4> <font color="purple">5: Most Favorite</font></h4>
<form name="frmRating" action="rating_result.jsp">
<%
ResultSet resultset = statement.executeQuery("select distinct
movies.movie_id,movies.movie_title,ratings.rating,movies.action,movies.comedy,movies.ro
mance from movies,ratings where ratings.movie_id=movies.movie_id and (ratings.rating=5
or ratings.rating=1) and (movies.action=1 or movies.comedy=1 or movies.romance=1) and
ratings.user_id between 1 and 20");
int i = 1;
while (resultset.next()) {
session.setAttribute("userid", user);
String movieId = resultset.getString("movies.movie_id");
String movieTitle = resultset.getString("movies.movie_title");
out.println(movieTitle + "<input type=text id=mRating" + i + " name=mRating" +
i + "><br>");
out.println("<input type=hidden id=mId" + i + " name=mId" + i + " value=" +
movieId + "><br>");
i++;
%>
<%
out.println("</br>");
}
%>
<input type="submit">
</form>

```

The following code first validates a customer's credentials and if it is successful then displays his rating products (movies) and also displays his favorite product types.

```
<%
    String user = request.getParameter("user");
    String pass = request.getParameter("pass");
    Connection conn = null;
    String url = "jdbc:mysql://localhost:3306/";
    String dbName = "hybridrecommender";
    String driver = "com.mysql.jdbc.Driver";
    String userName = "root";
    String password = "raza";
    Statement st;
    HashMap ratedMoviesRDateMap = new HashMap();
    try {
        Class.forName(driver).newInstance();
        conn = DriverManager.getConnection(url + dbName, userName, password);
        String sql = "select * from users where user_id='" + user + "' and password='" +
pass + "'";
        ResultSet rs;
        st = conn.createStatement();
        rs = st.executeQuery(sql);
        int count = 0;
        while (rs.next()) {
            count++;
        }
        if (count > 0) {
            out.println("<h2>");%>
<font size="+2" color="red">
            Login Successful
            <%
                } else {
                    response.sendRedirect("index.jsp");
                }
            %>
        </font>
        <%
            out.println("</br>");
            out.println("<hr />");
            ResultSet rs1 = st.executeQuery("select
users.user_id,movies.movie_title,ratings.rating,movie_rdate,movies.movie_id from
users,movies,ratings where users.user_id= ratings.user_id and
ratings.movie_id=movies.movie_id and ratings.rating >=4 and users.user_id='" + user + "'");
            int count1 = 0;
            %>
            <TABLE cellpadding="15" border="1" style="background-color: #ffffcc;">
                <font size="+2" color="purple">
```

```

        Your rated movies
    </font>
    <%
        out.println("</h2>");
        out.println("</br>");
        while (rs1.next()) {
            ratedMoviesRDateMap.put(rs1.getString("movie_id"),
rs1.getDate("movie_rdate"));

        %>
        <TR>
            <TD><%=rs1.getString("movies.movie_title")%></TD>
        </TR>
        <% }%>
    </TABLE>
    <%
        } catch (Exception e) {
            out.println(e.toString());
        }
        ItemBased itemobj = new ItemBased();
        try {
            itemobj.itemBased(user);
            if (itemobj.Reclist_Item.size() > 0) {
                recommendationsPackage.Reasoning res_obj = new
recommendationsPackage.Reasoning();
                res_obj.Reason(user);
                res_obj.ReasoningMoviesType(user);
                ArrayList Actionlist = new ArrayList(), Comedylist = new ArrayList(),
Romancelist = new ArrayList();
            %>
            <font size="+2" color="purple">
                Your Most Favorite Movie Type
            </font>
            <%
                out.println("</br>");
                out.println(res_obj.mostFavoriteMovietype + " movies");

            %>

```

The following code displays the recommendations and justifications for the active customer.

```

<font size="+2" color="red">
    Recommended Movies For You </font>
    <%
        String suggestedMovieIds = "-1";
        for (int i = 0; i < res_obj.AllType_list_suggested.size(); i++) {
            suggestedMovieIds += ", " + res_obj.Allid_list_suggested.get(i);

```

```

    }
    String sqlm = "SELECT movie_id,movie_rdate FROM movies WHERE
movie_id in (" + suggestedMovieIds + ")";
    ResultSet rsm;
    Statement stm = conn.createStatement();
    rsm = stm.executeQuery(sqlm);
    HashMap suggestedMoviesRDateMap = new HashMap();
    while (rsm.next()) {
        suggestedMoviesRDateMap.put(rsm.getString("movie_id"),
rsm.getDate("movie_rdate"));
    }
    %>
    <br />
    <font size="+2" color="purple">
        Movie type: Romance Movies

    </font>

    <%
        out.println("<br>");
        int movieCount = 0;
        movieCount = 0;
        for (int m = 0; m < res_obj.AllType_list_suggested.size(); m++) {
            if (res_obj.AllType_list_suggested.get(m) == "romance") {
                Romancelist.add(res_obj.Allid_list_suggested.get(m));
                // Getting average of recommended movie
                String sql = "SELECT movie_title,round(avg(rating))as ratings FROM
ratings,movies WHERE ratings.movie_id=movies.movie_id and movies.movie_id=" +
res_obj.Allid_list_suggested.get(m) + """;
                ResultSet rs;
                st = conn.createStatement();
                rs = st.executeQuery(sql);
                while (rs.next()) {
                    movieCount++;
                }
            }
        }
    %>
    <TABLE cellpadding="15" border="1" style="background-color: #ffffcc;"
width="100%">
    <TR>
    <TD><center><%out.println(movieCount + ": " +
rs.getString("movie_title"));%></center>
    </TD>
    </TR>
    <TR>
    <TD>
        <font size="+2" color="purple">
            Reasoning Behind To Recommend This Movie
        <br>

```

```

</font>
<font size="+2" color="red">
    Popularity Among Other Users
</font>
<br></font>
<%
    out.println("Average Rating of this Movie is:");
    out.println(rs.getString("ratings") + "/5");
    }
%>
<br>
<font size="+2" color="red">
    Content-Based Knowledgeable Justifications<br>
</font>
<ul>
    <%
        try {
            out.println("<li>Recommending you this movie because you like
romnace movies</li>");
            String likedActors = "";
            for (int j = 0; j < Romancelist.size(); j++) {

                for (int i = 0; i < res_obj.Actor_list_suggested.size(); i++) {
                    if
(res_obj.ID_list_suggested.get(i).toString().equals(Romancelist.get(j).toString())) {
                        //out.println("Recommending you the movie" +
res_obj.ID_list_suggested.get(i) + " because you like the " +
res_obj.Actor_list_suggested.get(i).toString());
                        if (likedActors.equals("")) {
                            likedActors += res_obj.Actor_list_suggested.get(i).toString();
                        } else {
                            likedActors += ", " +
res_obj.Actor_list_suggested.get(i).toString();
                        }
                    }
                }
            }
            if (!likedActors.equals("")) {
                out.println("<li>Recommending you this movie because you like the
actor(s) " + likedActors + "</li>");
            }
            String likedDirectors = "";
            for (int j = 0; j < Romancelist.size(); j++) {
                for (int i = 0; i < res_obj.Director_list_suggested.size(); i++) {
                    if
(res_obj.ID_list_suggested.get(i).toString().equals(Romancelist.get(j).toString())) {
                        if (likedDirectors.equals("")) {

```

```

        likedDirectors +=
res_obj.Director_list_suggested.get(i).toString();
    } else {
        likedDirectors += " ," +
res_obj.Director_list_suggested.get(i).toString();
    }
}
}
}
if (!likedDirectors.equals("")) {
    out.println("<li>Recommending you this movie because you like the
director(s) " + likedDirectors + "</li>");
}
int releaseDateMatchCount = 0;
//for (int j = 0; j < Romancelist.size(); j++) {
String suggestedMovieId =
res_obj.Allid_list_suggested.get(m).toString();
Date rDateSuggestedMovie = (Date)
suggestedMoviesRDateMap.get(suggestedMovieId);
Set s = ratedMoviesRDateMap.keySet();
Iterator it = s.iterator();
while (it.hasNext()) {
    String ratedMovieId = (String) it.next();
    Date rDateRatedMovie = (Date)
ratedMoviesRDateMap.get(ratedMovieId);
    if (rDateSuggestedMovie.getYear() <= rDateRatedMovie.getYear() +
10 && rDateSuggestedMovie.getYear() >= rDateRatedMovie.getYear() - 10) {
        releaseDateMatchCount++;
    }
}
// }
if (releaseDateMatchCount > 0) {
    out.println("<li>Recomending you this movie because the release date
of this movie is within 10 years of " + releaseDateMatchCount + " other rated movies</li>");
}
%>
</ul>
<font size="+2" color="red">
    Context-Based Knowledgeable Justifications<br>
</font>
<ul>
<%
// Added a logic to display context-based justifications
String watchedTimeofweek = "";
for (int j = 0; j < Romancelist.size(); j++) {
    for (int i = 0; i < res_obj.Timeofweek_list_suggested.size(); i++) {

```

```

        if
(res_obj.ID_list_suggested.get(i).toString().equals(Romancelist.get(j).toString())) {
            if
(watchedTimeofweek.indexOf(res_obj.Timeofweek_list_suggested.get(i).toString()) == -1) {
                if (watchedTimeofweek.equals("")) {
                    watchedTimeofweek +=
res_obj.Timeofweek_list_suggested.get(i).toString();
                } else {
                    watchedTimeofweek += " ," +
res_obj.Timeofweek_list_suggested.get(i).toString();
                }
            }
        }
    }
}
if (!watchedTimeofweek.equals("")) {
    out.println("<li>Recommending you this movie because you have
watched the movie on a " + watchedTimeofweek + "</li>");
}

String watchedCompanion = "";

for (int j = 0; j < Romancelist.size(); j++) {
    for (int i = 0; i < res_obj.companion_list_suggested.size(); i++) {
        if
(res_obj.ID_list_suggested.get(i).toString().equals(Romancelist.get(j).toString())) {
            if
(watchedCompanion.indexOf(res_obj.companion_list_suggested.get(i).toString()) == -1) {
                if (watchedCompanion.equals("")) {
                    watchedCompanion +=
res_obj.companion_list_suggested.get(i).toString();
                } else {
                    watchedCompanion += " ," +
res_obj.companion_list_suggested.get(i).toString();
                }
            }
        }
    }
}
if (!watchedCompanion.equals("")) {
    out.println("<li>Recommending you this movie because you have
watched romnace movies with a " + watchedCompanion + "</li>");
}

// Ended a logic to display context-based justifications
} catch (Exception ex) {
    out.println("<br><i>Exception in generating reasonings</i> ");
}

```



```

        ex.printStackTrace();
    }
    %>
</ul>
</td>
</tr>

```

Appendix B

Appendix B mainly contains the code from the JAVA package that has two classes. The first class is 'itembased' that creates an item-based recommender and second class is 'reasoning' that determines the reasoning behind the recommendations.

Itembased

```

MySQLConnectionPoolDataSource dataSource = new MySQLConnectionPoolDataSource();
    dataSource.setUser("root");
    dataSource.setPassword("raza");
    dataSource.setServerName("localhost");
    dataSource.setPort(3306);
    dataSource.setDatabaseName("hybridrecommender");

    MySQLJDBCDataModel model = new MySQLJDBCDataModel(dataSource,
"ratings", "User_id", "movie_id", "rating", "rating_date");

    RecommenderBuilder recommenderBuilder = new RecommenderBuilder() {

        public Recommender buildRecommender(DataModel model) throws
TasteException {
            ItemSimilarity similarity = new PearsonCorrelationSimilarity(model);
            return new GenericItemBasedRecommender(model, similarity);
        }
    };
    int usr = Integer.parseInt(user);
    Recommender recommender = recommenderBuilder.buildRecommender(model);
    List<RecommendedItem> recomendations = recommender.recommend(usr,
maxRecomendationsReqd);
    for (int i = 0; i < recomendations.size(); i++) {
        System.out.println(recomendations.get(i).getItemID() + " Recommended Movie
ID");
        System.out.println(recomendations.get(i).getValue() + " Preference value");
        Long s = recomendations.get(i).getItemID();
        Reclist_Item.add(s);
    }

```

Reasoning

```
public void Reason(String user) {
    ItemBased item_obj = new ItemBased();
    try {
        item_obj.itemBased(user);
    } catch (Exception ex) {
        System.out.println(ex.toString());
    }
    int usr = Integer.parseInt(user);
    try {
        Class.forName(driver).newInstance();
        conn = DriverManager.getConnection(url + dbName, userName, password);
        System.out.println("Connected to the database");
        String sql;
        ArrayList Users_Actor_list = new ArrayList();
        ArrayList Users_Director_list = new ArrayList();
        ArrayList Users_Timeofweek_list = new ArrayList();
        ArrayList Users_Companion_list = new ArrayList();
        String actor_movie = null, director_movie = null, time_of_week_movie = null,
companion_movie = null;

        sql = "select actor,director,time_of_week,companion from feature_profile_v where
user_id =" +usr;
        st = conn.createStatement();
        rt = st.executeQuery(sql);

        System.out.println("user's actor,director,time of week and companion");
        while (rt.next()) {
            Users_Actor_list.add(rt.getString("actor"));
            Users_Director_list.add(rt.getString("director"));
            Users_Timeofweek_list.add(rt.getString("time_of_week"));
            Users_Companion_list.add(rt.getString("companion"));
        }
        for(int i=0;i<item_obj.Reclist_Item.size();i++) {
            sql = "select actor,director,time_of_week,companion from feature_profile_v where
movie_id =+"
                + item_obj.Reclist_Item.get(i);

            st = conn.createStatement();
            rt = st.executeQuery(sql);
            if (rt.next()) {
                actor_movie = rt.getString("actor");
                director_movie = rt.getString("director");
                time_of_week_movie=rt.getString("time_of_week");
                companion_movie=rt.getString("companion");
                System.out.println(actor_movie);
            }
        }
    }
}
```

```

        System.out.println(director_movie);
        System.out.println(time_of_week_movie);
        System.out.println(companion_movie);
    }
    boolean OneTimeAdded = false;
    if (Users_Actor_list.contains(actor_movie)) {
        Actor_list_suggested.add(actor_movie);
        ID_list_suggested.add(item_obj.Reclist_Item.get(i));
        OneTimeAdded = true;

    }
    if (Users_Director_list.contains(director_movie)) {
        Director_list_suggested.add(director_movie);
        if(!OneTimeAdded)
            ID_list_suggested.add(item_obj.Reclist_Item.get(i));

    }

    if (Users_Timeofweek_list.contains(time_of_week_movie)) {
        Timeofweek_list_suggested.add(time_of_week_movie);
        if(!OneTimeAdded)
            ID_list_suggested.add(item_obj.Reclist_Item.get(i));

    }

    if (Users_Companion_list.contains(companion_movie)) {
        companion_list_suggested.add(companion_movie);
        if(!OneTimeAdded)
            ID_list_suggested.add(item_obj.Reclist_Item.get(i));

    }

    }
    conn.close();
    System.out.println("Disconnected from database");
} catch (Exception e) {
    e.printStackTrace();
}
}

```