Republic of Iraq
Ministry of Higher Education
And Scientific Research
Al-Nahrain University
College of Science

Design and Implementation for Recommender System

A Thesis
Submitted to the College of Science, Al-Nahrain University
In Partial Fulfillment of the Requirements for
The Degree of Master of Science in Computer Science

By
Zena Hussain Fahad

(B.Sc. 2005)

Supervisor
Dr.Taha S.Bashagha

2010 م *1431 هـ*

بِسْمِ اللهِ الرَّحمٰنِ الرَّحِيمِ

يُؤتِي الحِكمَةَ مَن يَشَآءُ ۚ وَمَن يُؤتَ
الحِكمَةَ فَقَد أُوتِيَ خَيرًا كَثِيرًا ۗ وَمَا يَذَّكَّرُ
إِلَّا أُولُوا الأَلبَابِ

صَدَقَ اللهُ العَلِيُّ العَظِيم

البقرة ٢٦٩

Supervisor Certification

I certify that this thesis was prepared under my supervision at the Department of Computer Science/College of Science/Al-Nahrain University, by Zena Hussain Fahad as partial fulfillment of the requirements for the degree of Master of Science in Computer Science.

Signature:
Name: **Dr. Taha S. Bashaga**
Title: **Lecturer (supervisor)**
Date: / / **2010**

In view of the available recommendations, I forward this thesis for debate by the examination committee.

Signature:
Name: **Dr. Taha S. Bashaga**
Title: **Head of the Computer Science Department, Al- Nahrain University.**
Date: / / **2010**

Dedication

To the memory of my greatest father who was give me and teach me a lot, where his death was the biggest loss to me.

To the brave and kind woman, my mother for every thing produced to me.

To the terrific gift of the destiny, my dear husband for his continual love, support, consoling and patience.

To the sun of my life, my lovely son Kamal.

To my beloved brothers and sisters for their big help.

You are the cause of my success

Zena

Acknowledgement

First, I would like to thank God, for all the blessing that have given us .

Second, I would like to express sincere gratitude and appreciation and deepest thanks to my supervisor the Head of Computer Science Department Dr. Taha S. Bashaga for his humanist behavior about my special circumstances and his insistence to help me, also for his continued invaluable guidance and for honest effort given throughout the progress of this research.

Finally, my very special thanks to my dear family and my friends for their continuous supports and encouragement during the period of my studies.

Zena

Abstract

Recommender systems have been introduced to provide a solution to navigating the huge volume of information already available and growing at an explosive rate. The amount of information available in electronic form, such as news, movies, books, advertisements and other online information is overwhelming us. Recommender systems are computer-based techniques that can be utilized to efficiently provide personalized services in many e-business domains.

In this thesis, recommender system has been designed by mixing two main types of recommender systems (content based on personal profile and collaborative based). This type of system producing recommendations for its users in two stages. In the first stage, searching about active user's neighborhood is done to compute the similarity with the active user. The similarity is computed in two steps, the first step is to compute personal similarity using content based technique, depending on the personal features only. The second step is a conditional step that is if the user has enough rating then the similarity computed using collaborative filtering technique depending on the user ratings (rating similarity) in addition to personal similarity computed by the first step. In the second stage a list of new items is recommended from highly rated items by nearest neighbor users, with or without predictions on the acceptance of the list by the user.

The content based part in which a personal similarity is computed a weight for each personal feature is required. So in this work, a survey has been made to obtain initial value for impact ratio (weight) for the effectiveness of each feature. Then the computation of these ratios is updated from time to time according to the given new users information. These updates are made according to Mean Absolute Error (MAE) between the real ratings and prediction of ratings.

List of Abbreviations

Abbreviations	***Full words***
CF	Collaborative Filtering
CB	Content Based
DB	Data base
KB	Knowledge Based
MAE	Mean Absolute Error
MRS	Movies Recommender System
PCC	Pearson Correlation Coefficient

List of Tables

List of Figures

Table of Contents

Chapter One
General Introduction

Chapter One
General introduction

1.1 The problem of information overload

The mass of content available on the World Wide Web (WWW) raises important questions over its effective use. With largely unstructured pages authored by a massive range of people on a diverse range of topics, simple browsing has given a way to filter as a practical way to manage web-based information, and this normally means search engines.

Search engines are effective at filtering pages to match explicit queries. Unfortunately, people find articulating what they want as a search query difficult, especially if they are forced to use a limited vocabulary such as keywords. The result is large lists of search results that contain a handful of useful pages, defeating the purpose of filtering in the first place.

Semantic web offers the potential for help, allowing more intelligent search queries based on web pages marked up with semantic metadata. Semantic web technology is very dependent, however, on the degree to which web pages are annotated by their authors. Annotation requires a degree of selflessness in authors, since the annotations provided will only help others searching their pages. Because of this, and the huge number of web pages that require annotation, in the foreseeable future it's likely that most web pages will remain unannotated. The semantic web will thus only be of partial benefit to the problem of formulating explicit search queries [Mid03].

1.2 Recommender systems

People find articulating what they want hard, but they are very good at recognizing it when they see it. This insight has led to the utilization of relevance feedback, where people rate web pages as interesting or not interesting and the system tries to find pages that match the "interesting", positive examples and do not match the "not interesting", negative examples. With sufficient positive and negative examples, modern machine-learning techniques can classify new pages with impressive accuracy; in some cases text classification accuracy exceeding human capability has been demonstrated.

Capturing user preferences is a problematic task. Simply asking the users what they want is too intrusive and prone to error, yet monitoring behavior unobtrusively and then finding meaningful patterns is difficult and computationally time consuming. Capturing accurate user preferences is however, an essential task if the information systems of tomorrow are to respond dynamically to the changing needs of their users [Mid03].

Total information overload becomes increasingly severe in the modern times of omnipresent mass-media and global communication facilities, exceeding the human perception's ability to dissect relevant information from irrelevant. Consequently, since more than 64 years significant research efforts have been striving to conceive automated filtering systems that provide humans with desirable and relevant information only. During the last two decade, recommender systems have been gaining momentum as another efficient means of reducing complexity when searching for relevant information. Recommenders intend to provide people with suggestions of products they will

appreciate, based upon their past preferences, history of purchase, or demographic information or other types of information [Zie05].

A recommender system consists of three elements as shown in figure (1.1). Many ***recommendation contents*** which are presented to users have to be made. Then, ***users' preferences or behavioral data*** on these contents must be gathered. Finally, it needs to choose type of ***recommendation technique*** about how to analysis these user data and select the optimal content to each user [Seo03].

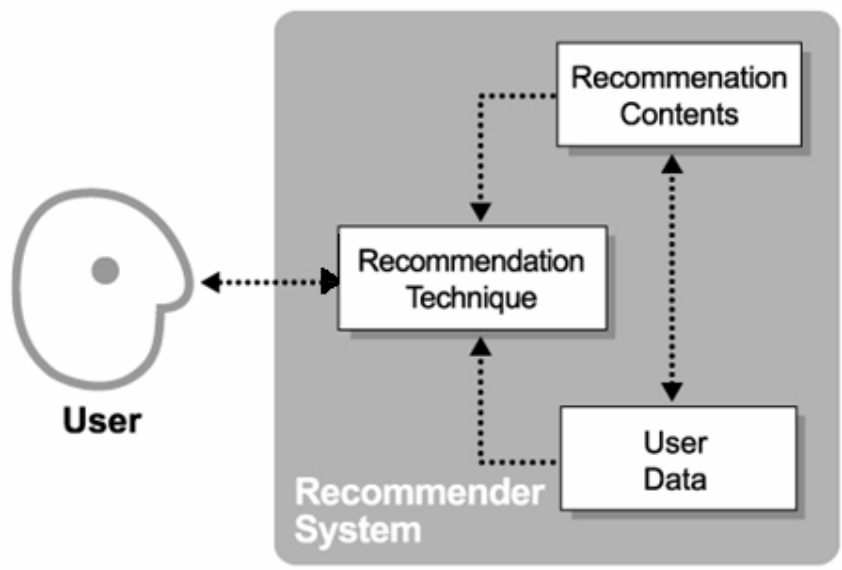

Figure 1.1 Constitution of Recommender System

The objective of collecting user information is to build a profile that describes a user interests, role in an organization, entitlements, and purchases or other information. The most common techniques are explicit profiling and implicit profiling [Abd06]:

- ***Explicit profiling:*** asks each visitor to fill out information or questionnaires by a specific form. This technique has the advantage of letting users tell the recommender system directly what they want.

- ***Implicit profiling:*** tracks the visitor's behavior. This technique is generally transparent to the user. The browsing is usually tracked by saving specific user identification and behavior information in log file which keeps the user hits or by a cookie files that is kept at the browser and updated at each visit. For example, Amazon.com logs each customer's buying history and, based on that history, recommends specific purchases.

Recommender systems apply data analysis techniques to the problem of helping users to find the items they would like to purchase at E-Commerce sites by producing a predicted likeliness score or a list of top–N recommended items for a given user (active user who the recommendations made to him). Items recommendation can be made using different methods. Recommendations can be based on demographics of the users, overall top selling items, or past buying habit of users as a predictor of future items these techniques are *Collaborative based recommender system, Content-based recommender system* and the modern recommender systems intend to mix the two ways this new technique called *Hybrid recommender system* [Sar01].

1.2.1 Collaborative Based Recommender Systems

Collaborative filtering has become a popular method for delivering recommendations to individuals on a wide range of items, most typically books, movies, music, and news articles. The basic idea behind collaborative filtering is to automate word-of-mouth. Collaborative filtering works by finding individuals with similar tastes, and making recommendations based on their likes and dislikes. This relies on the idea that if two individuals have similar tastes on number of items, they are likely to have similar tastes on other items as well [Woo04]. Contents

analysis is ignored and only the user's preference data on the considered contents are relevant where this step considered as strength in CF (collaborative filtering). But, it is difficult to apply to content which have few or no preference data from the users [Seo03].

1.2.2 Content-based Recommender System

Content-based recommendation technology has its roots in information retrieval and information filtering. Each item in a database is characterized by a set of attributes, known as the content profile. Such a profile is used to determine if the item is "similar" to the item that a user has preferred in the past and therefore its appropriateness for recommendation. The content profile is constructed by extracting a set of features from an item**.** In domains such as text documents and electronic products, keywords of a document or physical features of a product are used to build such item profiles and often no further extraction is needed. [Jon07]. Content can be recommended without any preference data from the users. But, it is difficult to apply to content that is hard to analyze and classify [Seo03].

1.2.3 Hybrid Recommender Systems

Hybrid approaches are geared towards unifying collaborative and content-based filtering under one single framework, leveraging synergetic effects and mitigating inherent deficiencies of either paradigm. Consequently, hybrid recommenders operate on both product rating information and descriptive features. In fact, numerous ways for combining collaborative and content-based aspects are conceivable. Most widely adopted among these, however, is the so-called "collaboration via content" paradigm where content-based profiles are built to detect similarities among users [Zie05].

1.3 User profiling in recommender systems [Mid03]

User profiling is typically either knowledge-based or behavior-based. Knowledge-based approaches engineer static models of users and dynamically match users to the closest model. Questionnaires and interviews are often employed to obtain this user knowledge. Behavior-based approaches use the user's behavior as a model, commonly using machine-learning techniques to discover useful patterns in the behavior. Some sort of behavioral logging is employed to obtain the data necessary from which to extract patterns in behavior.

The user profiling approach used by recommender systems is behavior-based, commonly using a binary, two-class model to represent what users find interesting and uninteresting. Machine-learning techniques are then used to find potential items of interest with respect to the binary model. There are a lot of effective machine-learning algorithms based on two classes.

1.4 Related works

One can classify recommender systems products as ***commercial*** and ***Academic recommender systems.***

1.4.1 Commercial system

I - Amazon.com

First, a user has to sign up as a new customer (if he/she was not already a customer) by filling personal information. Each customer will have a virtual cart that contains items the customer selected to buy. Amazon.com is structured with an information pages for each item, giving details of the text and purchase information. First, Amazon shows a list of top-selling items, either they are frequently purchased by customers or highly recommended from other customers. Amazon also

encourages direct feedback from customers about items they have purchased. Customers rate books they have read on a 5-point scale from "hated" to "loved." After rating a sample of items, customers may request recommendations for items that they might like. At that point, a half dozen non-rated texts are presented that correlate with the user's indicated tastes [Bas04]. Amazon.com use variations of CF (collaborative filtering) techniques to suggest products to their customers [Zie05].

II- LIBRA

Combines a content-based approach with machine learning to make book recommendations. The content-based approach differs from collaborative filtering in that it carries out analysis on the contents of the items being recommended. Furthermore, each user is treated individually - there is no sense of "community" which forms the basis of collaborative filtering. It also uses Bayesian text-categorization machine learning techniques to build a model for each user's preferences relative to the content of the items. The key advantage is explanations can be very easily produced. However a content-based approach is inappropriate when the items being considered are in a non-textual form such as images, and video or music clips [Mid03].

III- MovieLens

A well-known research movie recommendation website makes use of collaborative filtering technology to make its suggestions. This technology captures user preferences to build a profile by asking the user to rate movies. It searches for similar profiles (i.e. users that share the same or similar taste) and uses them to generate new suggestions. One shortcoming that most websites using collaborative filtering suffer from

is that they do not have any facility to provide explanations of how recommendations are derived [Mid03].

IV- My Personal Shopper

Landsend.com, the leading clothing company of the US (United States), has adopted a content-based approach to their website service. In their recommender system ("My Personal Shopper"), users are categorized into various types based on their preference for a series of clothes displayed in website. Then, the recommender system analyzes Landsend's products and recommend what each user would like. This type of recommendation is based on analyzing and classifying contents [Seo03].

V- Reel.com

Is a commercial system that recommends movies based on customer reviews. The customers enter their movie requirements (genre, viewing format, price etc.) and a set of recommendations is computed based on the habits of other customers by using CF (Collaborative Filtering) techniques [Mid03]. So this system considered as collaborative based recommender system.

1.4.2 Academic research recommender systems.

I- "Constructing User Profiles for Collaborative Recommender System" [Li 04]

In this paper, clustering technique is applied in the collaborative recommender framework to consider semantic contents available from the user profiles. The authors also suggest methods to construct user profiles from rating information and attributes of items to accommodate

user preferences. Further, they show that the correct application of the semantic content information obtained from user profiles does enhance the effectiveness of collaborative recommendation.

II- "A Content-Based Approach to Collaborative Filtering" [Woo04]

This paper presents a method of combining typical collaborative filtering techniques with content-based analysis of the items in order to provide accurate recommendations for a wide range of situations.

III- "Recommender system for ECommerce Data"[Bas04]

In this M.Sc. thesis, a recommender system is built that uses different recommendation methods to advice a customer the best items that suit his/her interest from a selected site. In a typical recommender system people provide recommendations as inputs which the system then aggregates and direct to appropriate recipients. Method accuracy depends on several factors, these factors are: the speed of that method, the accuracy which is different from customer to another and the psychological conditions of the recommender system users.

IV- "A Collaborative Filtering Algorithm and Evaluation Metric that Accurately Model the User Experience" [Mcl04]

In this article the writers empirically demonstrate that two of the most acclaimed CF recommendation algorithms have flaws that result in a dramatically unacceptable user experience. In response, they introduce a new Belief Distribution Algorithm that overcomes these flaws and

provides substantially richer user modeling. The Belief Distribution Algorithm retains the qualities of nearest-neighbor algorithms which have performed well in the past, yet produces predictions of belief distributions across rating values rather than a point rating value. In addition, they illustrate how the exclusive use of the mean absolute error metric has concealed these flaws for so long, and they propose the use of a modified Precision metric for more accurately evaluating the user experience.

V- "A Hybrid Collaborative Filtering Recommender System Using a New Similarity Measure" [Ahn07]

In This paper the author presents a hybrid recommender system using a new heuristic similarity measure for collaborative filtering that focuses on improving performance under cold-start conditions where only a small number of ratings are available for similarity calculation for each user. The new measure is based on the domain-specific interpretation of rating differences in user data. Experiments using three datasets show the superiority of the measure in new user cold-start conditions.

1.5 Aim of thesis

This research aimed to design a recommender system to directed users for suitable choices that meet their interest among huge amount of information to reduce their confusion.

The system tried to be easy and enjoyable for active user by using simple interfaces and asking active user (just in his first visit) about reasonable number of general personal information that all users can gave them. Also if active user has or give enough number of ratings the system will produce a top N-recommendations list with predictions on these recommendations and from these ratings the system also can use some scaling to be trusted by the user. In this work N is (10) items.

1.6 Thesis outlines

This is the summary of the contents of the subsequent chapters of this thesis:

- **Chapter two:** this chapter presents the theoretical background and analysis of recommender system and its types, important features and benefits for recommender system.
- **Chapter three:** this chapter presents the proposed system architecture, the analysis phase of building the proposed Recommender System, the algorithms that used to implement this system, and the implementation interfaces.
- **Chapter four:** this chapter explores conclusions of this work, and the suggestions for future works.

Chapter Two
Theoretical Background

Chapter Two
Theoretical Background

2.1 Introduction

Since the advent of the Web, there is very large amount of information have gone online. Millions of people around the world now have access to this global information resource. But, how do all of these people find the information they are most interested in from among all of those information? And, how do they find the other people they would most like to communicate with, work with, and play with? Increasingly, people are turning to recommender systems to help them to find the information that is most valuable to them. Recommender systems support a broad range of applications, including recommending movies, books, and even pets [Mil04]. Recommender systems have changed the way people shop online [Mil03]. Recommender systems apply knowledge discovery techniques to the problem of making personalized product recommendations during a live customer interaction [Sar00]. Recommender systems are a category of software that make personalized recommendations of goods, services, and people [Mo 01]. Recommender systems are characterized with “individualized” that separate them from search engines, which focus on the “matching”: the system is supposed to return all those items that match the query ranked by degree of match [Li 04]. Recommender systems have a single focus: predicting what items or pieces of information that a user will find interesting or useful. Predictions from a recommender system are personalized based on each user's individual profile, which generally contains relevance or interest judgments of

previously seen items. Recommender systems are centralized systems accessed by multiple users [Her00]. A recommender system can be viewed as a mapping of users and items to a set of utility values (or interest scores). The view of recommendation as a prediction task comes from the fact that this mapping is not, in general, defined on the whole domain of user-item pairs, and thus requires the system to estimate the interest values for some elements of the domain[Mob07]. The purpose of a recommender system is to eliminate the need for browsing the entire item space by presenting the user with items of interest early on [Nat07]. Recommender systems attempt to profile user preferences and model the interaction between users and products. Increasingly, their excellent ability to characterize and recommend items within huge collections represents a computerized alternative to human recommendations [Bel07].

2.2 Profile representations

A recommendation is "non-personalized" if it doesn't depend on a user profile. For example, a basic search engine providing a list of web pages makes a nonpersonalized recommendations. Another example is when all users browsing a given product on an e-Commerce web site see the same recommendations [Lem05].

Profile representations falls into three types, which are not mutually exclusive. Ratings-based, Content-based, and Knowledge-based representations [Mid03].

2.2.1 Ratings-based representations *(Relevance feedback)* [Mid03]

When users receive recommendations it is common to elicit feedback on how interesting the recommendations are to the needs of the user. This type of feedback is called relevance feedback.

Relevance feedback is elicited by offering the user a rating scale for each recommendation; the choice is commonly either "interesting" and "not interesting" or a 1 to 5-point scale of interest (Fairly awful, Bad, Good, Very good, Excellent). The representation of relevance feedback is thus a set of recommended items and the associated interest values provided by each user. Relevance feedback is often incomplete since users are often reluctant to invest time and effort to provide the feedback.

Relevance feedback can be acquired implicitly, allowing inference from observed user behavior. The problem with implicit feedback is that the assumptions made to allow inference often introduce errors. For example, a user may read an initially interesting looking document, only to find it was actually not interesting after all when its details are known; if all documents that are read are inferred to be interesting this situation would clearly introduce an error into the relevance feedback acquired. Implicit feedback is commonly in a positive/negative form, implied by clear positive or negative actions in an attempt to reduce the number of assumptions required.

A balance must be made between interrupting the user to acquire high quality explicit feedback and unobtrusive methods to obtain lower quality implicit feedback. Exactly how much interruption users will tolerate will depend upon the specific application domain.

2.2.2 Content-based representations [Mid03]

Most content-based analysis is performed on textual documents such as web pages, newspaper articles or document abstracts. The reason for this is that textual documents easily break down into individual words, whereas video and audio sources require sophisticated analysis to decompose into useful sub-components. All content-based recommender systems work with textual content.

I. Term-frequency vector representation **[Mid03]**

The most common abstraction of a textual document in the machine-learning context is a term-frequency (TF) vector. Terms consist of single words or phrases, and the frequency count is simply the number of times a term appears within the document text. To create a term-frequency vector the terms within a document are counted and the frequency values stored in an n-dimensional vector. The number of dimensions of the vector is the number of unique terms within a document.

It is common to reduce the dimensionality of term-frequency vectors to improve processing efficiency. Common terms, called stop words, are removed since they have little discriminating power as all documents contain them; examples of stop words are “and”, “if” and “the”. The removal of stop words is normally performed using a standard stop list, removing all terms that match the stop list.

Low frequency terms are also removed, since they too have little discriminating power, often appearing in just one document; an example of low frequency term is a web URL.

Another dimensionality reduction technique commonly employed is to stem terms. This involves removing suffixes so that basically similar

words are grouped together; an example would be to use the stemmed term "recommend" for the terms like "recommender", "recommendation" and "recommends".

In practice, stemming, stop lists and low frequency term removal are all applied to reduce the dimensionality of the term vectors as much as possible. Term-frequency representations are often called "bag of words" representations, since the structure of the document is lost. It has been shown that the loss of structural information such as sentences and paragraphs does not significantly degrade the performance of subsequent analysis and classification.

Recommender systems usually normalize the frequency data based on the length of the document, and some systems weight individual terms in favour of the more discriminating ones. This avoids larger documents always having highly weighted terms.

II. Binary class profile representation [Mid03]

The most common profile representation for content-based recommender systems is the binary class profile, representing user interests as a set of positive and negative examples. The positive, or "interesting", examples are represented as a collection of term-frequency vectors of documents that the user has rated as "interesting". The negative, or "not interesting", examples are likewise represented. This binary class representation is very suitable for a great many machine-learning techniques.

Since relevance feedback is required to obtain the sets of positive and negative examples, a ratings-based profile is often additionally implemented to create a hybrid recommender system.

III. Multi-class profile representation using an ontology

[Mid03]

The alternative to the binary class representation is a multi-class representation. Rather than simply having positive and negative classes, ontology of classes can be created that map to domain concepts such as newspaper topics like "sport". A user's profile is thus represented in terms of which classes they are most interested in, abstracting away from the specific examples of interest. When relevance feedback is acquired, examples of interest are classified according to the classes within the ontology, and the user's interest in that class recorded.

Multi-class classification is considerably more complex than binary class classification. Having more than two classes reduces the number of examples available for each class, thus reducing the accuracy of the machine-learning technique employed. In addition, since classes are shared between users, there will be a loss of information about individual user interests when compared to a binary representation where each user has their own set of examples; sharing examples does allow for a larger training set, however. These factors are the reason why very few recommender systems adopt this approach.

Most ontologies are created manually by a knowledge engineer and domain experts. They thus capture the relevant classes within a domain and relationships between them. It is possible to create classes automatically using clustering machine-learning algorithms. Clustering finds similar term frequency vectors and groups them together to make a class. Classes created by clustering, however, have no domain knowledge associated with them, making useful inference from them difficult.

2.2.3 Knowledge-based profile representation [Mid03]

Knowledge-based profile representations appear in the user modelling literature. Typically these approaches require questionnaires and interviews with users to acquire information about their requirements before a profile can be built. Profiles consist of asserted facts about a user in a knowledge-base, from which inferences can be drawn about user stereotypes and interests. Knowledge-based profiles are often used in the related fields of agent and intelligent tutoring systems.

2.3 Types of recommender systems

From an algorithmic point of view recommender systems fall into three general categories [Mob07]:

1. Knowledge-based systems.
2. Content filtering systems.
3. Collaborative filtering systems.

Advanced recommender systems tend to combine collaborative and content-based filtering, trying to mitigate the drawbacks of either approach and exploiting synergetic effects. These systems have been coined "***Hybrid Systems***" [Zie05].

Recommender systems have (i) background data, the information that the system has before the recommendation process begins, (ii) input data, the information that user must communicate to the system in order to generate a recommendation, and (iii) an algorithm that combines background and input data to arrive at its suggestions. On this basis, it can be distinguish three different recommendation techniques as shown in Table 2.1 Assume that **I** is the set of items over which recommendations might be made, **U** is the set of users whose preferences are known, **u** is the active user for whom

recommendations need to be generated, and **i** is some item for which the recommender system would tried to predict u's preference.

Table 2.1: Recommendation Techniques [Bur02]

Technique	**Background**	**Input**	**Process**
Collaborative	Ratings from users in **U** on items in **I**.	Ratings from active user (**u)** on some items In **I**.	Identify users in **U** similar to **u**, and extrapolate from their ratings on the suitable item i for active user.
Content-based	Features of all items in **I** or personal features of all users in **U.**	Features of items in **I** selected by active user **u**.	Generate a classifier that fits **u**'s rating behavior and use it on **i**.
Knowledge-based	Features of all items in **I**. Knowledge of how these items meet a user's needs.	A description of **u**'s needs or interests.	Infer a match between **i** and **u**'s need.

2.3.1 Knowledge based recommenders (KB)

This type of recommendation attempts to suggest objects based on inferences about a user's needs and preferences. Knowledge-based approaches have knowledge about how a particular item meets a particular need of the user, and can therefore reason about the relationship between a need and a possible recommendation [Ho 06]. Knowledge based recommenders rely either on explicit domain knowledge about the items or knowledge about the users (such as demographic characteristics) to derive relevant recommendations. Many such systems rely on manually or automatically generated knowledge-based decision rules that are used to recommend items to users who satisfy constraints specified by the stored rules. Like most rule-based systems, this type of personalization relies heavily on knowledge engineering by system designers to construct a rule

base in accordance to the specific characteristics of the domain or based on market research. The user profiles are generally obtained through explicit interactions with users. Some research has focused on machine learning techniques for classifying users into one of several categories based on their Demographic attributes, and therefore, automatically deriving decision rules that can be used for personalization [Mob07]. The Personal Logic recommender system offers a dialog that effectively walks the user down a discrimination tree of product features [Bur00]. It does not have to gather information about a particular user because its judgments are independent of individual tastes. These characteristics make knowledgebase Recommenders not only valuable systems on their own, but also highly Complementary to other types of recommender systems [Bur00].

2.3.2 Content-based Filtering (CB)

For more than three decades, computer scientists have been addressing the problem of information overload by designing software technology that automatically recognizes and categorizes information. Such software automatically generates descriptions of each item's content, and then compares the description of each item to a description of the user's information need to determine if the item is relevant to the user's need. The descriptions of the user's interest needs are either supplied by the user, such as in a query, or learned from observing the content of items the user consumes. These techniques called **content-based** because the software performs filtering based on software analysis of the content of the items analyzed. Text search engines are a prime example of content-based filtering. Many text search engines use a technique called term-frequency

indexing in term frequency indexing, documents and user information needs are described by vectors in a space with one dimension for every word that occurs in the database. Each component of the vector is the frequency that the respective word occurs in the document or the user query. The document vectors that are found to be the closest to the query vectors (computed using the dot-product) are considered the most likely to be relevant to the user's query. Most information filtering and information retrieval systems today are built using entirely content-based information retrieval technology. Other examples of content-based filtering are Boolean search indexes, where the query is a set of keywords combined by Boolean operators; probabilistic retrieval systems, where probabilistic reasoning is used to determine the probability that a document meets a user's information need; and natural language query interfaces, where queries are posed in natural sentences [Her00]. Content-based algorithms are principally used when documents are to be recommended, such as web pages, publications, jokes or news. The agent maintains information about user preferences either by initial user input about his interests during the registration process or by rating documents. Recommendations are formed by taking into account the content of documents and by filtering in the ones that better match the user's preferences and logged profile [Pap04]. Content-based recommender systems are classifier systems derived from machine learning research. For example, the NewsDude news filtering system is a recommender system that suggests news stories the user might like to read. These systems use supervised machine learning to induce a classifier that can discriminate between items likely to be of interest to the user and those likely to be uninteresting [Bur00]. A variety of algorithms have been proposed for analyzing the content of text documents and finding regularities in this

content that can serve as the basis for making recommendations. Many approaches are a specialized versions of classification learners, in which the goal is to learn a function that predicts which class a document belongs to (i.e., either liked or not-liked). Other algorithms would treat this as a regression problem in which the goal is to learn a function that predicts a numeric value (i.e., the rating of the document). There are two important sub-problems in designing a content-based filtering system. The first is finding a representation of documents. The second is to create a profile that allows for unseen documents to be recommended. All of the content-based approaches represent documents by the "important" words in the documents [Paz98].

Content-based recommender systems suffer from many limitations:

- New user problem: for the system to understand and accurately match a user's preferences, the user has to rate a sufficient number of products;
- Limited content analysis: due to the limited features that are explicitly associated with products in the recommendation system;
- Over-specialization: the system cannot recommend products that are different from anything the user has rated before, since the system can only find products that score highly against a user preferences [Rou06].

2.3.3 Collaborative filtering systems (CF)

Content-based filtering only works when dealing with domains where feature extraction is feasible and attribute information readily available. Collaborative filtering (CF), on the other hand, uses content less representations and does not face that same limitation [Zie05]. So In recent

years, **Collaborative filtering** (**CF**) has been developed to address areas where content-based filtering is weak. CF systems are different from traditional computerized information filtering systems in that they do not require computerized understanding or recognition of content. In a CF system, items are filtered based on user evaluations of those items instead of the content of those items [Her00]. Generally, in the collaborative filtering, the content analysis is ignored and only other user's opinions on the considered content are considered relevant. Therefore, the collaborative filtering approach is especially interesting for content for which content analysis is weak or impossible. However, the performance of the collaborative filtering approach relies on the available user preference data for the considered content and therefore fails when few or no opinions are known [Seo03]. Figure 2.1 shows the schematic diagram of the collaborative filtering process. CF algorithms represent the entire $m \times n$ user-item data as a ratings matrix, A. Each entry $a_{i,j}$ in A represent the preference score (ratings) of the i th user on the j th item. Each individual ratings is within a numerical scale and it can be 0 as well to a special case indicating that the user has not yet rated that item [Sar01]. The CF Ingredients (***input data***) are:

1. List of **m Users** $U = \{u_1, u_2, \ldots, u_m\}$ and a list of **n Items** $I = \{i_1, i_2, \ldots, i_n\}$.
2. Each user u_i has a **list of items** I_{ui} he/she expressed their **opinion** about (can be a null set)
3. **Explicit opinion** a rating score (numerical scale)
4. Sometime the rating is **implicitly** – purchase records
5. **Active user** u_a for whom the CF prediction task is performed
6. A **metric** for measuring **similarity between users**
7. A method for selecting a **subset of neighbors** for prediction

8. A method for **predicting a rating** for items not currently rated by the active user [Ric07]. $I_{u_i} \subseteq I$, it is possible for I_{u_i} to be a *null-set*. There exists a distinguished user $u_a \in U$ called the *active user* for whom the task of a collaborative filtering algorithm is to find an item likeliness that can be of two forms.

- **Prediction** is a numerical value, $P_{a,j}$, expressing the predicted likeliness of item $i_j \notin I_{u_a}$ for the active user u_a. This predicted value is within the same scale (e.g., from 1 to 5) as the opinion values provided by u_a.
- **Recommendation** is a list of N items, $I_r \subset I$ that the active user will like the most. Note that the recommended list must be on items not already purchased by the active user, i.e. $I_r \cap I_{u_a} = \Phi$. This interface of CF algorithms is also known as *Top-N recommendation* [Sar01].

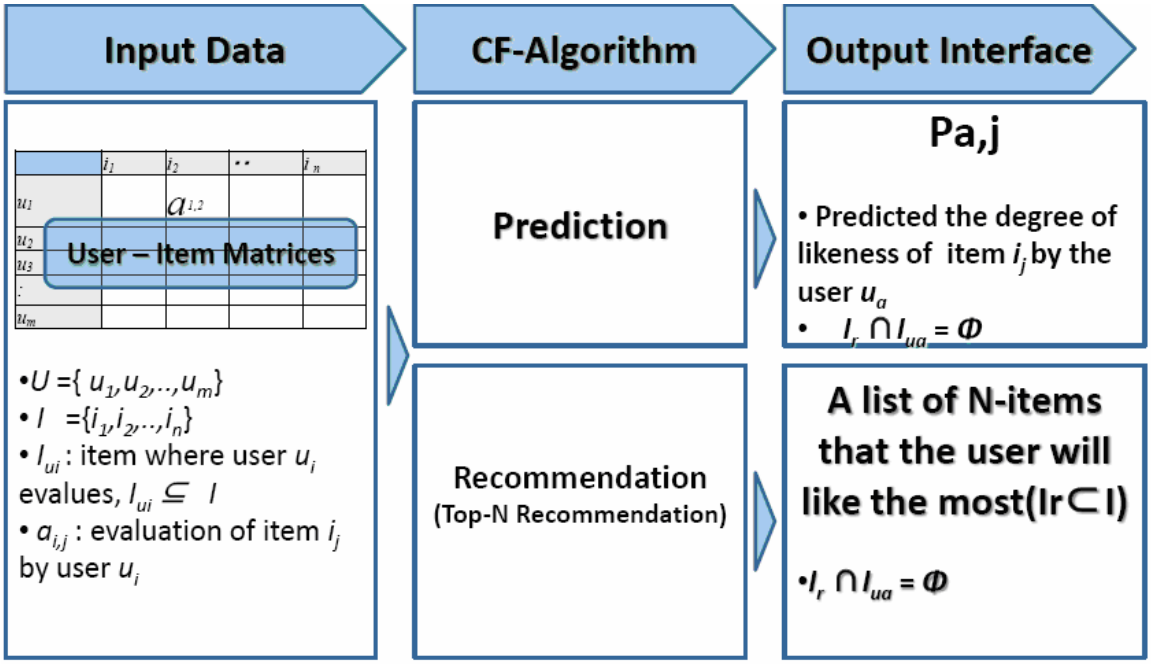

Figure 2.1 Collaborative Filtering Process [Ume08]

The collaborative based filtering recommendation technique proceeds in these steps:

1. For a **target/active user** (the user to whom a recommendation has to be produced) the set of his ratings is identified
2. The **users more similar** to the target/active user (according to a similarity function) are identified (neighbor formation)
3. The **products bought by these similar users** are identified
4. For each one of these products a **prediction** - of the rating that would be given by the target user to the product – is generated
5. Based on this predicted rating a set of **top N** products are Recommended [Ric07].

Collaborative recommendation addresses the shortcomings of content-based systems [Mo 01]**.**The idea behind this method is that, it may be of benefit to one's search for information to consult the behavior of other users who share the same or relevant interests and whose opinion can be trusted [Pap04]. CF analyzes relationships between users and interdependencies among products, in order to identify new user-item associations [Bel07].

A collaborative filtering system (CF) stores the preferences and opinions of the thousands of users of the system. These opinions are recorded as ratings by users for items. When an active user would like a recommendation, the system finds users with similar taste and uses their opinions to generate a recommendation [Mcl04].

Two general classes of CF algorithms have been widely investigated. ***Memory-based*** CF ***(user-based)***, which is the most prevalent approach, operates over the entire user preference database to make predictions. In contrast ***Model-based*** algorithms ***(item-based)*** use the preference database to infer a model, which is then applied for predictions [Yu 02].

I. Memory-based Approaches

The memory-based approaches are among the most popular prediction techniques in collaborative filtering. The basic idea is to compute the active user's predicted vote of an item as a weighted average of votes by other similar users or ***K nearest neighbors*** **(KNN)** [Xue05]. Several systems use statistical techniques to provide personal recommendations of documents by finding a group of other users, known as ***Neighbors*** that have a history of agreeing with the target user. Usually, neighborhoods are formed by applying proximity measures such as the Pearson correlation between the opinions of the users. These are called nearest-***neighbor techniques***. Figure 2.2 depicts the neighborhood formation using a ***nearest-neighbor technique*** in a very simple two dimensional space. Notice that each user's neighborhood is those other users who are most similar to him, as identified by the proximity measure [Sar99].

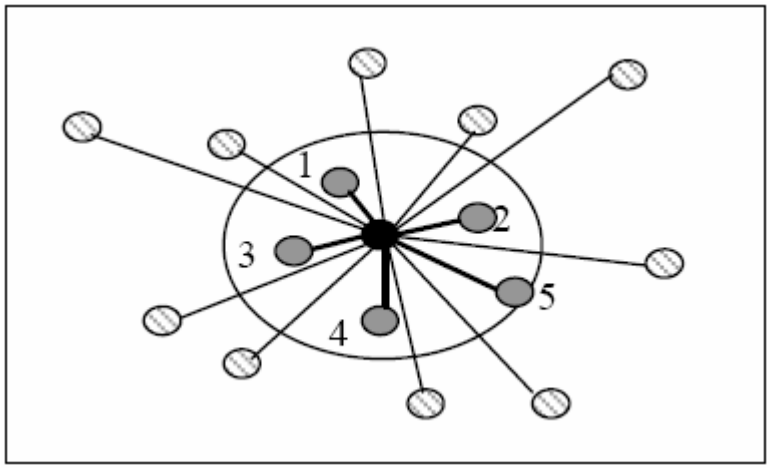

Figure 2.2: Illustration of the neighborhood formation process. The distance between the target user and every other user is computed and the closest-k users are chosen as the neighbors (for this diagram k = 5) ***[Sar99].***

Collaborative filtering based on *k*-nearest-Neighbor (*k*NN) approach involves comparing the active record for a target user with historical records of other users in order to find the top *k* users who have similar tastes or interests. The mapping of a visitor record to its neighborhood could be based on similarity in ratings of items, access to similar contents or pages, or purchases of similar items. This neighborhood is then used to recommend items not already accessed or purchased by the active user [Bha07].

The primary advantages of KNN algorithms are:

1. Ability to predict a rating value for a single item.
2. The best average accuracy in multi-value data (in terms of predicting the rating value).
3. Good performance with large datasets (accomplished with sampling).
4. Simple to understand, explains, implement, and maintain [Mcl04].

KNN is not widely-used in web personalization for efficiency reasons – it does not scale well to extremely large numbers of profiles [Bha07].

However, neighborhood-based methods raise some concerns:

1. By definition, the interpolation weights sum to one, this may cause over fitting. Suppose that an item has no useful neighbors rated by a particular user. In that case, it would be best to ignore the neighborhood information, staying with the current data normalization. Nevertheless, the standard neighborhood formula uses a weighted average of ratings for the uninformative neighbors.
2. Neighborhood methods may not work well if variability differs substantially among neighboring items (users) [Bel07].

The most commonly used memory-based algorithm is ***the Pearson Correlation Coefficient*** **(PCC)** algorithm. The PCC algorithm generally achieves higher performance than other similarity method [Xue05].

The User Nearest Neighbor algorithm based on the Pearson r Correlation coefficient was used in some of the earliest collaborative filtering systems, yet it remains a popular baseline algorithm, since it is easy to implement and demonstrates high accuracy when measured with mean absolute error. This algorithm may call as the ***User-User*** algorithm, because at its core is the computation of similarity between each pair of users [Mcl04]. A common way to measure the similarity between two users is the Pearson correlation between ratings of items which both users voted for. However, if two users have just a few common ratings, the Pearson correlation is a bad estimate for their similarity.

- **Pearson Correlation Coefficient (PCC)**

User-based collaborative filtering engaging PCC was used in a number of recommendation systems, since it can be easily implemented and can achieve high accuracy when comparing with other similarity computation methods. In user-based collaborative filtering, PCC is employed to define the similarity between two users ***a*** and ***u*** based on the items they rated in common: [Ma07]

$$sim(a,u) = \frac{\sum_{i \in I(a) \cap I(u)} (r_{a,i} - \bar{r}_a) \cdot (r_{u,i} - \bar{r}_u)}{\sqrt{\sum_{i \in I(a) \cap I(u)} (r_{a,i} - \bar{r}_a)^2} \cdot \sqrt{\sum_{i \in I(a) \cap I(u)} (r_{u,i} - \bar{r}_u)^2}} \quad ...(2.1)$$

Where ***Sim* (*a, u*)** denotes the similarity between user ***a*** and user ***u***, and ***i*** belongs to the subset of items which user ***a*** and user ***u*** both already rated. $r_{a,i}$ is the rate user ***a*** gave item ***i***, and $\bar{r}_a$ represents the average rate of user ***a***. From this definition, user similarity ***Sim* (*a, u*)** is ranging from [0, 1], and a larger Value means users *a* and *u* are more similar [Ma07].

II. Model-based approaches

The item-based approach looks into the set of items the target user has rated and computes how similar they are to the target item and then selects most similar items. At the same time their corresponding similarities are also computed. Once the most similar items are found, the prediction is then computed by taking a weighted average of the target user's ratings on these similar items. Model-based collaborative filtering algorithms provide item recommendation by first developing a model of user ratings. Algorithms in this category take a probabilistic approach and envision the collaborative filtering process as computing the expected value of a user prediction, given his/her ratings on other items. The model building process is performed by different *machine learning* algorithms such as **Bayesian network, clustering,** and **rule-based** approaches. The Bayesian network model formulates a probabilistic model for collaborative filtering problem. Clustering model treats collaborative filtering as a classification problem and works by clustering similar users in same class and estimating the probability that a particular user is in a particular class *C*, and from there computes the conditional probability of ratings. The rule-based approach applies association rule discovery algorithms to find association between co-

purchased items and then generates item recommendation based on the strength of the association between items [Sar01].The nearest-neighbor algorithms introduced above finds users who have rated the active item and have interests similar to the active user. An alternate approach ***Item Nearest Neighbor (ITEM)*** is to find items rated by the active user that are similar to the item being predicted (the active item). several different algorithms that used similarities between items, rather than users, to compute predictions. These algorithms all assume that the active user's ratings for items related to the active item are a good indication of the active user's preference for the active item [Mcl04].

One critical step in the item-based collaborative filtering algorithm is to compute the similarity between items and then to select the most similar items. The basic idea in similarity computation between two items *i* and *j* is to first isolate the users who have rated both of these items and then to apply a similarity computation technique to determine the similarity *sim (i, j)* symbol as $S_{i,j}$. Figure 2.3 illustrates this process, here the matrix rows represent users and the columns represent items. There are number of different ways to compute the similarity between items. Some of these methods are cosine-based similarity, correlation-based similarity and adjusted-cosine similarity [Sar01].

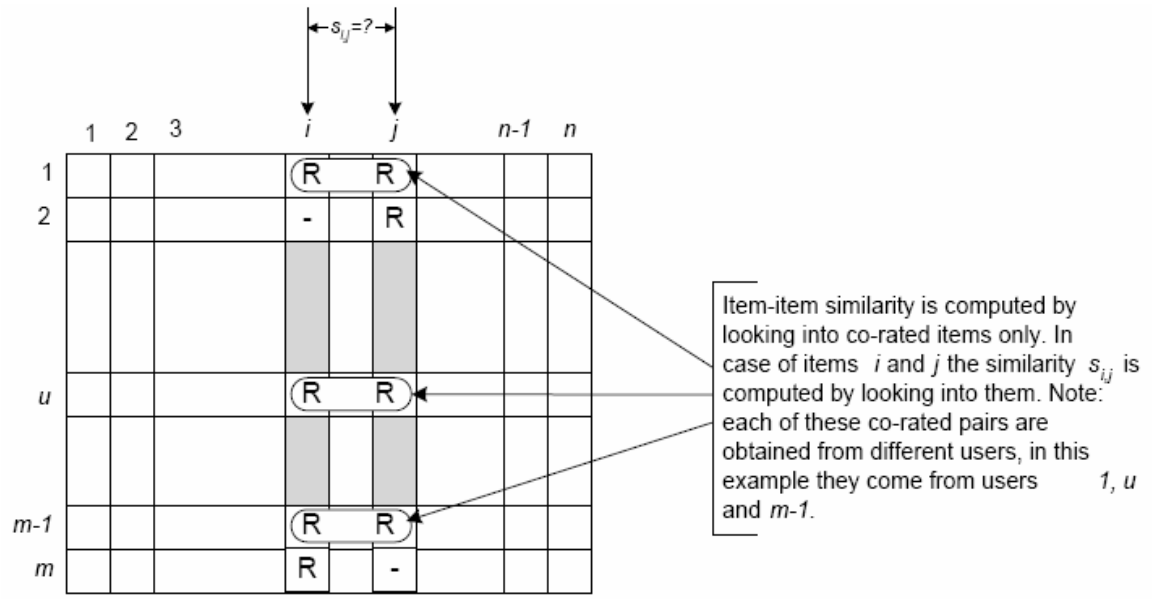

Figure 2.3: Isolation of the co-rated items and similarity computation

a. Cosine-based Similarity

In this case, two items are thought of as two vectors in the ***m*** dimensional user-space. The similarity between them is measured by computing the cosine of the angle between these two vectors. Formally, in the ***m*** × ***n*** ratings matrix, similarity between items ***i*** and ***j***, denoted by ***sim(i, j)*** is given by

$$sim\ (i, j) = \cos(\vec{i}, \vec{j}) = \frac{\vec{i} \cdot \vec{j}}{\|\vec{i}\|_2 * \|\vec{j}\|_2} \quad ...(2.2)$$

where "·" denotes the dot-product of the two vectors. $\vec{i} \cdot \vec{j}$ are vectors of rating on both items.

b. Correlation-based Similarity

In this case, similarity between two items ***i*** and ***j*** is measured by computing the ***Pearson-r*** correlation ***corr i, j.*** To make the correlation computation accurate, isolate the co-rated cases (i.e., cases where the users rated both items ***i*** and ***j*** as shown in figure (2.3)) . Let the set of users who both rated ***i*** and ***j*** are denoted by ***U,*** then the correlation similarity is given by

$$sim(i,\ j) = \frac{\sum_{u \in U} (R_{u,i} - \overline{R}_i)(R_{u,j} - \overline{R}_j)}{\sqrt{\sum_{u \in U} (R_{u,i} - \overline{R}_i)^2} \sqrt{\sum_{u \in U} (R_{u,j} - \overline{R}_j)^2}} \quad ...(2.3)$$

[Sar01][Nat07].

Here $R_{u,i}$ denotes the rating of user *u* on item *i* , $\overline{R}_i$ is the average rating of the *i* -th item[Sar01].

c. Adjusted Cosine Similarity

One fundamental difference between the similarity computation in user-based CF and item-based CF is that in case of user-based CF the similarity is computed along the rows of the matrix but in case of the item-based CF the similarity is computed along the columns i.e., each pair in the co-rated set corresponds to a different user Figure (2.3). Computing similarity using basic cosine measure in item-based case has one important drawback–the difference in rating scale between different users is not taken into account. The adjusted cosine similarity offsets this drawback by

subtracting the corresponding user average from each co-rated pair. Formally, the similarity between items *i* and *j* using this scheme is given by

$$sim(i, j) = \frac{\sum_{u \in U} (R_{u,i} - \overline{R}_u)(R_{u,j} - \overline{R}_u)}{\sqrt{\sum_{u \in U} (R_{u,i} - \overline{R}_u)^2} \sqrt{\sum_{u \in U} (R_{u,j} - \overline{R}_u)^2}} \quad ...(2.4)$$

Here $\overline{R}_u$ is the average of the *u*-th user's ratings [Sar01].

2.3.4 Challenges of User-based Collaborative Filtering Algorithms

User-based collaborative filtering systems have been very successful, but their widespread use has revealed some potential challenges such as:

• **Sparsity.** In practice, many commercial recommender systems are used to evaluate large item sets (e.g., Amazon. com recommends books and CDnow.com recommends music albums). In these systems, even active users may have purchased well under 1% of the items (1% of 2 million books is 20, 000 books). Accordingly, a recommender system based on nearest neighbor algorithms may be unable to make any item recommendations for a particular user. As a result the accuracy of recommendations may be poor.

• **Scalability.** Nearest neighbor algorithms require computation that grows with both the number of users and the number of items. With millions of users and items, a typical web-based recommender system running existing algorithms will suffer serious scalability problems [Sar01]. Therefore, inorder to bring recommendation algorithms successfully on the web, and

succeed in providing recommendations with acceptable delay, sophisticated data structures and advanced, scalable architectures are required [Pap04].

- **The Cold-start Problem**

One difficult problem commonly faced by recommender systems is the cold-start problem, where recommendations are required for new items or users for whom little or no information has yet been acquired. Poor performance resulting from a cold-start can deter user uptake of a recommender system. This effect is thus self-destructive, since the recommender never achieves good performance since users never use it for long enough [Mid02]. An item cannot be recommended unless a user has rated it. This problem applies to new and obscure items and is particularly detrimental to users with eclectic taste [Pap04]. Although collaborative filtering has been very successful in both research and practice areas, it can not recommend new items to users without any history in the system and completely denies any information that can be extracted from semantic contents of items. For this reason, hybrid recommender systems have been provided, which can exploit both user preferences and semantic contents [Li 04].

There are two types of cold-start problem.

1. The ***new-system cold-start*** problem is where there are no initial ratings by users, and hence no profiles of users. In this situation most recommender systems have no basis on which to recommend, and hence perform very poorly [Mid02].
2. The ***new-user cold-start*** problem is where the system has been running for a while and a set of user profiles and ratings exist, but no information is available about a new user. Most recommender systems perform poorly in this situation too [Mid02]. This problem can be addressed using

hybrid recommendation approaches, or other alternative approaches that use techniques based on product popularity, product entropy, user personalization, or their combinations to determine the most informative products (to the system) the new user should rate[Rou06].

Collaborative recommender systems fail to help in cold-start situations, as they cannot discover similar user behaviour because there is not enough previously logged behaviour data upon which to base any correlations. Content-based and hybrid recommender systems perform a little better since they need just a few examples of user interest in order to find similar items [Mid02].

2.4 Advantages and disadvantages collaborative and content-Based filtering [San00]

The advantages and disadvantages of both collaborative and content based filtering are summarized by the following:-

1. For CB in each time a new item enters into the system, it can be recommended (an analyses of its content can be made), but in CF there's no way to recommend an incoming item until it has been voted by a minimum number of users.
2. In CB the filtering quality is not affected by the number of registered users and the number and the quality of recommendations made by them, but in CF the filtering quality depends on the number of registered users and the number and the quality of recommendations made by them.
3. The content-based analyses of an item (document) can only manage textual documents. It's hard to analyze by its content information such as images, the style or the layout of a document, or semantic information,

while CF based on the opinions made by other users. Users are entities with huge power of semantically and visual analysis.

4. In CB the system only search similar topics to the items retrieved before. Is hardly to get new interesting topics to the registered users, while in CF the taste of the users usually changes in time, in a very dynamic way. Users commonly recommend new topics.

2.5 Hybrid Recommender Systems

The two approaches (the collaborative and the content-based) have their respective strengths and weaknesses. There have been numerous systems developed to take the hybrid approach which uses the strength of one approach to overcome the limitation of the other [Jon07]. However, such approaches increase the complexity of the matching problem as they usually use more expressive rating language.

The requirements of real-time recommendations add additional challenges to the already complicated matching problem. The information filtering process in recommender systems requires an efficient matching algorithm with high throughput and scalability. For algorithms to be efficient, they have to achieve a good balance between effectiveness and performance. Clearly, the magnitude of this problem increases with respect to the number of users and products (attributes), as matches must be done in real-time. A recommender system must ensure the timely prediction of ratings upon demand [Rou06]. Proposed approaches to the hybrid system which combine content-based and collaborative filters together can be categorized into three groups. The first one is the linear combination of results of collaborative and content-based filtering. The second group is the sequential combination of content-based filtering and collaborative filtering.

In these systems, firstly, content-based filtering algorithm is applied to find users, who share similar interests. Secondly, collaborative algorithm is applied to make predictions. The last one is the mixed combination, both the semantic contents and ratings are applied to make recommendations [Li 04].

2.6 The set of features important to a recommender *system:* [Mid03]

There are five main issues a recommender system must address:-

2.6.1 Knowledge acquisition technique: - must be employed to gather information about the user from which a profile can be constructed. This knowledge is processed to provide the basis for an individual's user profile; it must thus be represented in a convenient way. Knowledge can either be implicitly or explicitly acquired from the user.

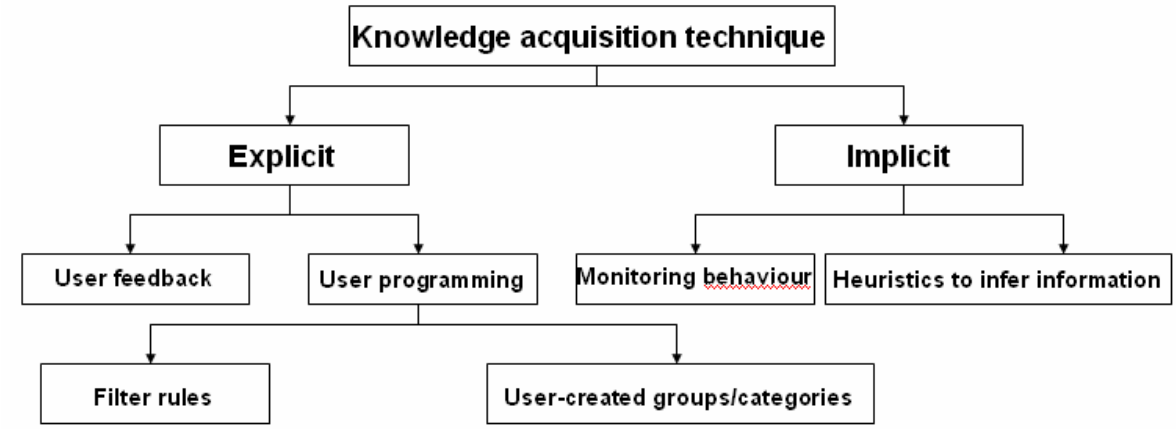

Figure (2.4) knowledge acquisition techniques

1. **Implicit knowledge acquisition: -** is often the preferred mechanism since it has little or no impact on the user's normal work activity.

I. Monitoring behavior: - System observes users using it and records this behavior. Unobtrusive monitoring of the user discovers behavioral data about the user's normal work activity over a period of time; this data can be used to infer preferences for frequently occurring items.

II. Heuristics to infer information: - Rules are used to infer information about users Heuristics can also be employed to infer facts from existing data. Implicitly acquired knowledge requires some degree of interpretation to understand the user's real goals; this is an inherently error prone process, reducing overall confidence in any resulting user profiles.

2. Explicit knowledge acquisition requires the user to interrupt their normal work to provide feedback or conduct some sort of programming of the system. Explicit knowledge is generally high confidence information, since it is provided by the users themselves and not acquired from indirect inference.

I. User feedback: - Users provide explicit feedback e.g. item relevance, item examples, etc. Feedback types include item relevance, interest and quality.

II. User programming:-occurs when the user is asked to create filter rules, either visually or via a programming language, or to tell the system about groups or categories of items that exist in the domain.

- ***Filter rules:***- Users provide filter rules to the system
- ***User-created groups/categories:***- Users define system groups or categories

2.6.2 Shared information: - There must be a knowledge source from which items can be recommended. Recommender systems allow information to be shared amongst users to enhance the overall recommendation performance; this shared information must be clearly defined.

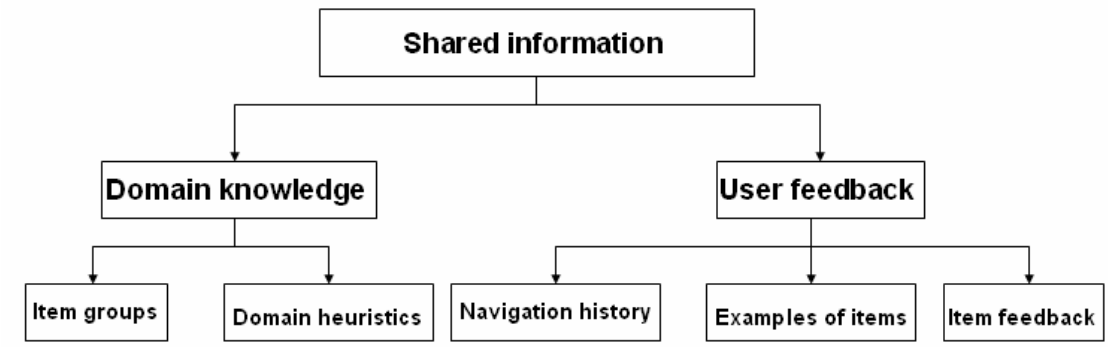

Figure (2.5) shared information

1. **User feedback:-**can be shared for the purpose of recommendation. If collaborative filtering is to be used, other users' feedback on unseen items can be used as a basis for recommendations for a particular user.

   Examples of interesting items can be shared between similar users to increase the size of the training set and hence improve classification accuracy. Previous navigation patterns are also useful to share, as they allow new users to receive the benefit from other people's previous mistakes and successes.

I. **Item feedback:-**item feedback is used by the system to help other users

II. **Examples of items:-** System pools examples of items to form a collective training set

III. **Navigation history:-** System uses recorded navigation histories to help other users

2. **Domain knowledge: -** can also be shared, since it is normally programmed in and hence available to the system from the start. Categorizations of items can be used to provide order to a domain, and common sets of domain heuristics, potentially part of a knowledge base, can be useful when computing recommendations.

I. Item groups / categorizations: - System shares communal groups and categories, whether defined by the system or other users

II. Domain heuristics: - System shares a set of domain filter rules between all users

2.6.3 Profiles representation: -

Profiles can be represented as a feature vector in a vector-space model. This is a standard representation and allows easy application of machine-learning techniques when formulating recommendations. For content-based recommendation the features in the vectors might be the word frequencies of interesting documents, while for collaborative filtering the features could be the keywords commonly used by users in their search queries. Navigation trails can be used to represent time-variant user behaviors. If some initial knowledge engineering has been conducted there may also be knowledge about the users available to a profile.

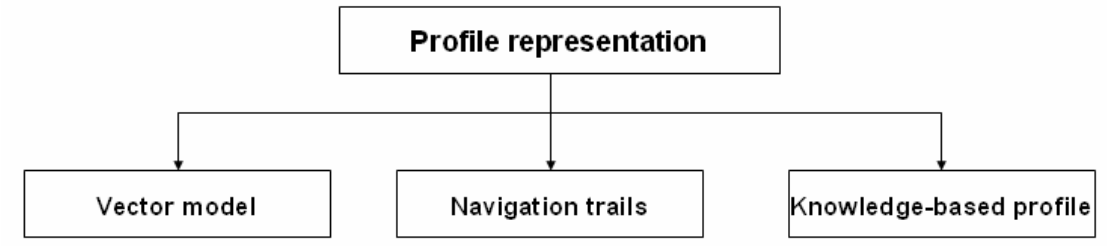

Figure (2.6) profile representation

I. Vector model:- System uses vectors to model for documents or interest profiles

II. Navigation trails:- System holds a navigation history e.g. a web browsing history

III. Knowledge-based profile:- System uses knowledge-based profiles

2.6.4 Knowledge source: -

The domain itself will contain sources of information to be recommended to the users. These could be from a database held by the recommender system, such as movie titles, or available dynamically via the web, such as links from the currently browsed page or web pages crawled from a web site. Systems can also rely on external events, such as incoming emails, to provide items for recommendation.

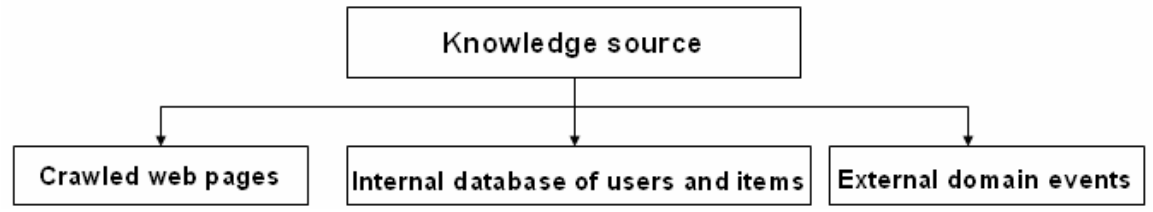

Figure (2.7) knowledge source

I. Internal database of items and users:- System recommends from an internal database of items

II. Crawled web pages:- System crawls the web for items to recommend

III. External domain events:- Events occur that trigger recommendation e.g. an email arrives

2.6.5 Recommendation techniques:-There is a wide variety of recommendation techniques employed today, with most techniques falling into three broad categories.

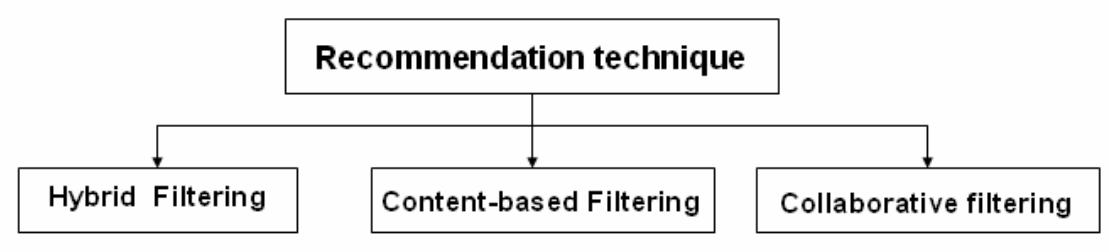

Figure (2.8) recommendation techniques

2.7 Methods of Collecting user Preferences

Regardless of the algorithmic approach to personalized recommendation, the data for user profiles must be collected either implicitly or explicitly [Mob07].

2.7.1 An explicit rating

Identifies the preference of a user to a specific item. A user is prompted by the agent's interface to provide ratings for items so as to improve his model. The more ratings the user provides, the more accurate the recommendations provided to him are. Ratings range from 1 to 5 with 1 expressing greatest aversion to the item and 5 expressing greatest liking to the item. Explicit ratings are logged by the system and form the user's model [Pap04]. However, this creates an additional load on users of the systems [Her00].

2.7.2 An implicit rating

Identifies the preference of a user to specific categories. They use here the term "implicit" somewhat excessively so as to express that a user is

never actually prompted to express his preference to categories [Pap04]. It learn preferences by gathering implicit ratings. Implicit ratings are ratings that are not entered by the user, but are inferred from observation of the user's actions [Her00].

2.8 Prediction computing [Her00]

A prediction engine collects ratings and uses collaborative filtering technology to provide predictions. An active user provides the prediction engine with a list of items, and the prediction engine returns a list of predicted ratings for those items. Most prediction engines also provide a recommendation mode, where the prediction engine returns the top predicted items for the active user from the database. The problem space can be formulated as a matrix of users versus items, with each cell representing a user's rating on a specific item. Under this formulation, the problem is to predict the values for specific empty cells (i.e. predict a user's rating for an item). In collaborative filtering, this matrix is generally very sparse, since each user will only have rated a small percentage of the total number of items. Table 2.2 shows a simplified example of a user-rating matrix where predictions are being computed for movies.

Table 2.2: Collaborative filtering can be represented as the problem of predicting missing values in a user-item matrix. This is an example of a user-item rating matrix where each filled cell represents a user's rating for an item. The prediction engine is attempting to provide Nathan a prediction for the movie 'Titanic.'

	Star Wars	Hoop Dreams	Contact	Titanic
Joe	5	2	5	4
John	2	5	0	3
Al	2	2	4	2
Nathan	5	1	5	?

The most prevalent algorithms used in collaborative filtering are what call the neighborhood-based methods. In neighborhood-based methods, a subset of appropriate users is chosen based on their similarity to the active user, and a weighted aggregate of their ratings is used to generate predictions for the active user. Other algorithmic methods that have been used are Bayesian networks, singular value decomposition with neural net classification and induction rule learning. As an example of a neighborhood based method, consider Table 2.2 again. A prediction is made to know how Nathan will like the movie "Titanic." Joe is Nathan's best neighbor, since the two of them have agreed closely on all movies that they have both seen. As a result, Joe's opinion of the movie Titanic will influence Nathan's prediction the most. John and Al are not as good neighbors because both of them have disagreed with Nathan on certain movies. As a result, they will influence Nathan's predictions less than Joe will.

Neighborhood-based methods can be separated into three steps.

1. Weight all users with respect to similarity with the active user.
2. Select a subset of users to use as a set of predictors (possibly for a specific item)

3. Normalize ratings and compute a prediction from a weighted combination of selected neighbors' ratings.

   Within specific systems, these steps may overlap or the order may be slightly different [Her00].

2.8.1 Prediction Algorithms

Prediction algorithms try to guess the rating that a user is going to provide for an item. It will refer to this user as *active user* a_u and to this item as *active item* a_i. These algorithms take advantage of the logged history of ratings and of content associated with users and items in order to provide predictions [Pap04].

I. Random Prediction Algorithms

The random prediction algorithm represents the worst case of prediction algorithm, since instead of applying a sophisticated technique to produce a prediction it generates a random one. They refer to the random prediction algorithm so as to have a reference point at how much better results they obtain by the utilization of more sophisticated ones [Pap04].

II. User-based Prediction Algorithms Description

User-based prediction algorithms are based on user's average rating and an adjustment to it, as given by equation 2.5.

prediction = user _ average + adjustment

$$adjustment = \frac{\sum_{b \in U_i} sim_{a,b}(R_{b,i} - \overline{R_b})}{\sum_{b \in U_i} |sim_{a,b}|}$$ or more formally

$$P_{a,i} = \overline{R_a} + \frac{\sum_{b \in U_i} sim_{a,b} (R_{b,i} - \overline{R_b})}{\sum_{b \in U_i} |sim_{a,b}|} \quad \ldots(2.5)$$

Where $P_{a,i}$ is the predicted rating of active user a on item i ,U_i set of users that have rated item i, $\overline{R_a}$ the average ratings of active user , $sim_{a,b}$ the similarity between active user a and user b.

Adjustment is most often a weighted sum that integrates user-based or item-based similarity measures. Since prediction arises as the sum of the two, improvements can be considered in both operators [Pap04].

III. Item-based Prediction Algorithms Description

The item-based prediction algorithms are referred to the algorithms that are based on item's average rating and an adjustment to it, as given by equation 2.6.

prediction = item _ average + adjustment …(**2.6**)

Adjustment is most often a weighted sum that integrates user-based or item-based similarity measures. Since prediction arises as the sum of the two, improvements can be considered in both operators [Pap04].

2.9 Metrics to evaluate collaborative filtering

There are two key dimensions on which the quality of a prediction algorithm can be measured, namely *coverage* and *accuracy*[Pap04].

2.9.1 Coverage Metric

Coverage is a measure of the percentage of items for which a recommendation agent can provide predictions. A basic coverage metric is the percentage of items for which predictions are available. Coverage can be reduced by defining small neighborhood sizes or by sampling users to calculate predictions. A prediction is impossible to be computed in case that very few people rated an item or in case that the active user has zero correlations with other users [Pap04].

2.9.2 Accuracy Metrics

Several metrics have been proposed for assessing the accuracy of collaborative filtering methods. The main category is *statistical accuracy metrics* [Pap04].

2.9.2.1 Statistical Accuracy Metrics

Statistical accuracy metrics evaluate the accuracy of a filtering agent by comparing the numerical prediction values against user ratings for the items that have both predictions and ratings. Some of them frequently used are *Mean Absolute Error (MAE), Root Mean Squared Error (RMSE)* and *correlation* between ratings and predictions. All of the above metrics were computed on result data and generally provided the same conclusions [Pap04].

- **Mean Absolute Error (MAE)**

Mean absolute error measures the average absolute deviation between a predicted rating and the user's true rating [Her00]. mean absolute error (MAE) has been used to e valuate the performance of CF algorithms. MAE works well for measuring how accurately the algorithm predicts the rating of a randomly selected item [Mcl04].*Mean Absolute Error (MAE),* which is a

measure of the deviation of recommendations from their true user-specified values. For each ratings-prediction pair < *rating=qi*, *prediction=pi* >,this metric treats the absolute error between them i.e., |*pi* − *qi*| equally. The MAE is computed by first summing these absolute errors of the *N* corresponding ratings-prediction pairs and then computing the average. Formally,

$$MAE = \frac{\sum_{i=1}^{N} |p_i - q_i|}{N} \quad ...(2.8)$$

, The lower the MAE, the more accurately the recommendation engine predicts user ratings **[Sar00]**. There are two advantages to mean absolute error. First, the mechanics of the computation are simple and easily recognized by all. Second, the properties of mean absolute error are well studied from statistical point of view. Properties that provide a means for testing the significance of difference between the mean absolute errors of two systems [Her00]. Three measures related to mean absolute error are **mean squared error**, **root mean squared error**, and **normalized mean absolute error**. The first two variations square the error before summing it. The result is more emphasis on large errors. For example, an error of one point increases the sum of error by one, but an error of two points increases the sum by four. The third related measure, normalized mean absolute error is mean absolute error normalized with respect to the range of rating values, in theory allowing comparison between prediction runs on different datasets (although the utility of this has not yet been investigated) [Her03]. However, in particular, MAE has the following negative characteristics:

- During offline analysis, if no rating is available for a recommended item, then that recommendation has no affect on MAE.
- An error of size e has the same impact on MAE regardless of where that error places the item in a top-N ranking.
- Highly accurate predictions on many mediocre or bad items (perhaps rated 3 out of 5) can drown out poor performance on highly ranked items.
- MAE cannot evaluate algorithms that produce ranked recommendations but do not produce predicted rating values [Mcl04].

Chapter three

Proposed Hybrid Recommender System

Chapter three
Proposed Hybrid Recommender System

3.1 Introduction

This chapter is dedicated to present the design considerations, which were taken throughout the construction stage of the proposed recommender system. The system deals with active user who may be a ***registered user*** (already registered in the system) or may be ***new user*** (visiting the system for the first time). The system tries to reduce user's confusion to get what he wants from the huge number of items (of some specified domain). Active user patience and also the simplicity in the implementation of the proposed system have been given the highest priority. Some of theoretical concepts discussed in chapter two were utilized to design proposed system, and some others are suggested in this work.

3.2 System structure

The main target of this research is to build a hybrid recommender system that mixing between two main recommendation techniques (CB and CF), trying to reduce the drawbacks of either approaches and exploiting the advantages of each of them. The proposed system first apply content-based filtering by using the content of the user profile (personal features), where it will search about users who are similar to active user in his personal features (personal profile) depending on personal features and impact ratio (weight) of these features. Then if

active user is new or already registered user but have not enough ratings, the personal similarity and the set of items genres of his/ her personal interest will be used to produce the list of recommendations where it's elements will be selected from the domain of highly rated items of nearest neighbors and non of them are repeatedly recommended to such user. This step represents the used of Collaborative Filtering (CF) depending on the opinions of similar users. Though if active user has enough ratings, then the similarity between this user and other registered users will be computed using memory-based collaborative filtering by Pearson Correlation Coefficient (PCC) as given in equation (2.1) which needs two conditions to be satisfied :

1. The personal similarity must be greater than or equal to a threshold (80%) of similarity to reduce the computations, and to reduce the number of similar persons with active user and get closer to him to maximize the accuracy.
2. The users (obtained from condition 1) who shared at least (40%) of items rated with active user are the only considered users, others are neglected.

These percentages constant is obtained by trial and error in testing the system performance. These constants can be change according to the application.

Then the recommendations list will be produced from highly rated items of nearest neighbors, and predictions on this recommendations list will be computed according to equation (2.5).

From the above discussion and as discussed in (2.4) this research will use the following features to design a recommender system as summarized by figure (3.1).

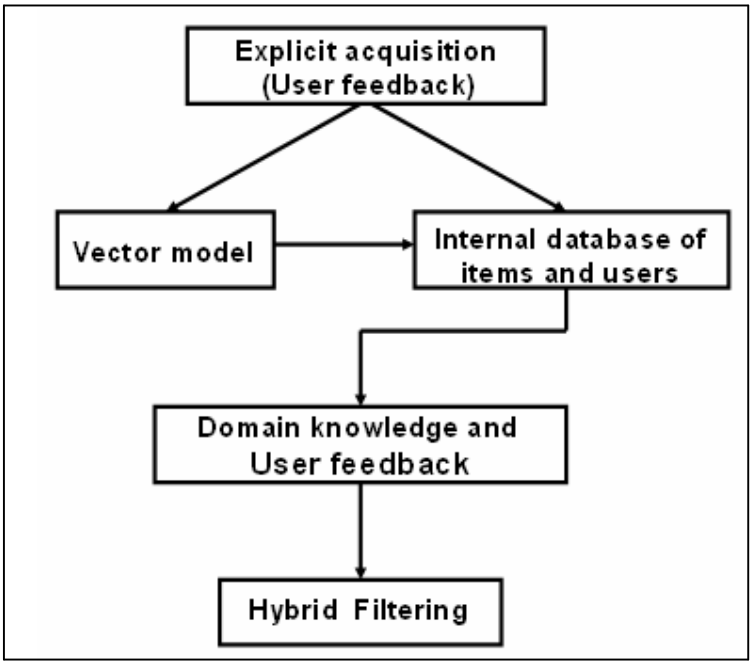

Figure (3.1) the features of proposed system

1. ***Knowledge acquisition techniques:*** An explicit knowledge technique is used to obtain personal information about active user and information about his interest by giving rating on items from recommendations list of last visit or by selecting the items genres of his best interest.

2. ***Shared information:*** The shared information between users will be (items, items ratings, and personal features); therefore, whole or most of the information of the database (DB) will be shared.

3. ***Profile representation:*** Vector model representation will be used to represent the profiles of users and items.

4. ***Knowledge source:*** Knowledge domain as users profiles, items profiles, ratings profiles attached with user profile, and the list of recommendations of previous visit for each registered user.

5. ***Recommendation technique:*** The recommendation technique used is hybrid technique between CF and CB. CB will be used first to find the similarity between active user and other users obtained from the personal features and the items that are interested by the active user. Then if active user has acceptable ratings the CF will be used to find similarity between him and other users obtained from their ratings on shared items by using PCC. Then there are two types of recommendations lists (with prediction and without prediction) where ***Prediction*** is a numerical value expressing the predicted likeliness of recommended item for the interest of the active user, this predicted value is within the same scale (e.g., from 1 to 5) as the opinion values provided by any registered user. The idea of the collaborative will be used to compute each of them, where, the system will select the items with high ratings of nearest neighbors such that they are not recommended previously to the active user. These features are explained in details in the following sections.

3.3 The analysis phase of building the proposed Recommender System

In the analysis phase, many problems appeared which are requiring a solution. In this work an appropriate solution has been suggested to solve these problems:

3.3.1 The new-system cold-start problem

At the start point the DB for any recommender system contains information about items only, but there are no profiles of users and hence no user/items ratings. There are two solutions to such problems

1. Select randomly a list of recommended items from the genres preferred by the active user.

2. The active user enforced to selects manually a list of items from the genres he preferred.

Then by either case, the user gives ratings on this list of items. Then information of the recommender system will be accumulated gradually and the performance of the system will be enhanced by the time.

In the case of this work there is no initial DB to start with, so most recommender systems use already made DB. So a DB from MovieLens is used to test the current system. This DB was presented by GroupLens which contains personal information about 3410 users and about 3883 movies. Every user gives at least rate on 20 movies. Frequently there is a possibility for such a DB to updated for the user part only.

3.3.2 Sparsity of matrix representation

In using collaborative filtering a matrix representation for users/items rating is highly required. But there are a huge number of items and users, so it is unreasonable that each user gives rating on all items. Hence there are a huge number of cells in this matrix remain empty or zeros cells. To overcome such problem, a suggested solution is made which is given in two stages, they are:

1- First, the rating similarity will be computed for users who are satisfying (80%) of personal similarity. This step will reduce the number of comparisons and make the computation to be more accurate, as well as the reduction of space complexity

2- The matrix representation of user/item ratings required a space size (no. of users × no. of items) which is of a huge size, but it is not possible all users rates on all items, so there are a huge number of cells without values or zero values.

Now instead of using such matrix, a table of ratings for each user in his profile is attached. This table consists of two columns; the first contains the item ID (no.) and the second column for ratings on these items, and that means the space complexity during the running time of the program reduce by a factor of $(\frac{2}{\text{no. of users}})$ of the matrix representation form.

3.3.3 The problem of new user

All recommender system using collaborative filtering (CF) have problems in dealing with the new users, because new users didn't have any ratings on items of interest. The collaborative filtering requires rating of user on items to compute similarity between active user and other registered users. This problem will not occur for the new users only but also for the lazy users who did not give rating or they have no enough ratings on items recommended to them previously. In this work there, are two stages to overcome such problem, these are:

1- The first stage, new user register to the system by giving the full personal information by filling a given formal form. Then he must choose a set of preferred items genres (or class). And then choose some items (at least 10 items) from the interested genres and rate on them. So if the chosen items (with ratings) are enough for calculation then the system will deal with him as an old user, otherwise the system will perform poorly and gives bad set of recommendations, because the result depends only on the personal features similarity, so the second stage will be highly required for any new user to help in minimizing errors in computing personal similarity.

2- The second stage is to compute impact ratio of personal features on the items to be selected, that means using the personal information above

and get benefit from them to compute the personal similarity. But it is difficult to know which personal feature is more effective in chosen items genres, so a method for computing impact ratio (or feature weight) is required.

A suggested solution is made by constructing a survey (Appendix A) on a randomly selected samples of 300 persons of different (gender, age, occupation, education) and a set of items genres of the selected domain (the domain here is a set of movies). The personal features selected are not depending on the opinion of the designer as for real system, but on the available (MovieLins) requirements of DB.

The available DB contains features of (gender, age and occupation) only, and calculation is made for these features only because of their availability.

Now the impact ratio (weight) of each personal feature is computed in four steps as follows:

1. Find ratio of all genres for each faction in each feature

For each faction in each feature selecting all its users and then compute ratio of each genres in this faction where ratio for each genre in each faction computed as follows :

$$RG(F_{li}, G_j) = \frac{\text{frequency of } (G_j) \text{ in } (F_{li})}{K} \qquad ...3.1$$

Where $RG(F_{li}, G_j)$ = ratio of current item genre (G_j) in current faction (F_{li}).

$\text{frequency of } (G_j) \text{ in } (F_{li})$ = number of persons in (F_{li}) chosen current genre (G_j)

l =feature number, i=faction number, j=genre number.

l = (1… nof), where nof= number of features.

i= (1 ... s(**l**)), where s(**l**)= number of factions in feature **l**.

j= (1 ... m), where m= number of genres.

K= number of persons in faction (i)/ feature (**l**).

2. Find the mean difference between any two factions in same feature

Now using MAE to compute the difference between any two factions in the same feature to compute the effect of such faction on the selection of genres by the following equation.

$$Dif(F_{lx}, F_{ly}) = \frac{\sum_{i=1}^{M} \left| RG(F_{lx}, G_i) - RG(F_{ly}, G_i) \right|}{M} \quad \dots 3.2$$

Where $Dif(F_{lx}, F_{ly})$ =mean of the difference between any two different factions in feature (**l**),

Dif = array of 2D of mean of differences between any two different factions.

M =number of genres.

x, y= any two factions in current feature, where $x \neq y$

x= (1…s(**l**)-1), y= (x+1…s(**l**)), i=(1…m), **l**=(1… nof)

$RG(F_{lx}, G_i)$ = ratio of current item genre (G_i) in current faction (F_{lx})

For example in gender feature there are two factions (Male, Female), (**l**=1, 2) then the result appear to be as shown in table (3.2):-

Table (3.1) show the difference between two factions

FACTIONS IN FEATURE (L)	**ACTION**	***ANIMATION***	**…….**	**WESTREN**
Female	0.23	0.13		0.08
Male	0.52	0.32		0.01
Difference	0.29	0.19		0.07

$$\text{Dif(male, female)} = \frac{0.29 + 0.19 + \ldots + 0.07}{18}$$

3. Find average difference between all factions for each feature

If there are more than or equal two factions, it must be compute difference between each faction with all other factions and then summing difference between all factions and divide result on number of comparison to obtain the average difference of current feature

$$\text{ADF}_{\mathbf{l}} = \frac{\sum_{i=1}^{s(\mathbf{l})-1} \sum_{j=i+1}^{s(\mathbf{l})} Dif(F_{\mathbf{l}i}, F_{\mathbf{l}j})}{(s(\mathbf{l})(s(\mathbf{l}) - 1)/2)} \qquad \ldots 3.3$$

Where $\text{ADF}_{\mathbf{l}}$ = The average of all differences $\text{Dif}(F_{\mathbf{l}i}, F_{\mathbf{l}j})$ between any two factions (i, j) in feature ($\mathbf{l}$).

$s(\mathbf{l})$ = number of factions in feature $\mathbf{l}$

Dif ($F_{\mathbf{l}i}$, $F_{\mathbf{l}j}$) = difference between factions (i and j) in feature $\mathbf{l}$.

4. Find impact ratio of each feature comparing with all other features

Then finally from equation (3.3) above one can compute the weight (impact ratio) for the effect of each feature on personal interest of selecting genres which can be given by the following equation:-

$$W_{\mathbf{l}} = \frac{\text{ADF}_{\mathbf{l}}}{\sum_{z=1}^{nof} \text{ADF}_z} \qquad \ldots 3.4$$

$W_{\mathbf{l}}$ =weight (impact ratio) of feature $\mathbf{l}$, $\mathbf{l}$ = feature number,

$\text{ADF}_{\mathbf{l}}$ = the average of all differences $\text{Dif}(F_{\mathbf{l}i}, F_{\mathbf{l}j})$ between any two factions (i, j) in feature ($\mathbf{l}$).

So the calculation of impact ratio (weight) is summarized by algorithm (3.1).

Algorithm(3.1) : computation of the impact ratio (weight) for each feature

Precondition: information in all the polls form
Output/action: impact ratio of each feature.

```
{RG= array of 2D of genres ratios in each faction }
{s= array of number of factions in each feature.}
{Dif = array of 2D of mean of differences between any two different factions}
{ADF= array of average differences between any two factions.}
{W=array of weights of all features.}
**********************************************
M=18
For l =1 to nof                          {nof= number of all features}
    For i=1 to s(l)                      { s(l)= number of factions in feature l }
        For j=1 to m                     {m=number of genres}
            Genc=0                       {Genc=genre counter}
            For v=1 to k                 {k= number of persons in faction i}
                If G_j ∈ set of genres preferred by person(v) then
                    Genc= Genc +1
                End if
            End for v
            RG(F_li,G_j)= Genc/k
        End for j
    End for i
    temp=0                               {temp=integer variable}
    sum_Dif=0                            {sum_Dif= integer variable}
    sum_all_ ADF =0                      {sum_all_ADF= integer variable}
    For x=1 to s(l)-1
        For y=x+1 to s(l)
            For j=1 to m
                temp=temp+abs(RG(F_lx,G_j)-RG(F_ly,G_j)
            End for j
            Dif(F_lx,F_ly)=temp/m        {Dif=array of real value}
            Sum_Dif = Sum_Dif +Dif(F_lx,F_ly)
        End for y
    End for x                                          To be continue
```

```
ADF_l =sum_Dif/( s(l)(s(l)-1)/2)
        Sum_all_ADF=sum_all_ADF+ADF_l
End for l
For    l =1 to nof
        W_l  = ADF_l/sum_all_ADF                {W=array of real value}
End for l
```

From the above algorithm the Impact Ratios (weights) for example obtained for the age and gender features are (53% and 47%) respectively. Other features are not computed for many reasons. The education feature is not available in the given DB and that is difficult to add such feature. The occupation feature is given as a long list of factions in the used DB, which is not compatible with the given list in the survey list, and that required a very large sample size of the survey to cover all of these factions and that is not possible in this study case. So the last two features are not considered.

Each application required different set of features and may be for each feature different set of factions. In the present application the features used are depend on their availability in the used DB and the accuracy depend on such data.

The impact ratios of personal features will be used just in the first stages of system application. The personal information will be accumulated gradually by the repeated use for the system (after each n of users using the system), therefore by time the system will be learning and can use these accumulated information to compute more real and accurate impact ratios. The impact ratios will be computed offline from time to time as the number of users increases and the previous computation will be neglected.

The computed Impact Ratio (weight) will be used to compute ***personal similarity*** between active user and other users in DB by equation (3.5).

$$\text{personalsimilarity}(a,u) = \sum_{l=1}^{nof} DF_l(a,u) * W_l \qquad ...3.5$$

Where $DF_l(a,u)$ = difference between active user *(a)* and user *(u)* for feature l

W_l =weight (impact ratio) of feature l

l = feature number, l = (1 … nof), nof= number of all features

And equation (3.6) computes difference between active user and any other user.

$$DF_l(a,u) = 1 - \frac{|F_{la} - F_{lu}|}{F_{ls(l)} - 1} \qquad ...3.6$$

F_{la} =faction which active user belong to it in feature l

F_{lu} =faction which user (u) belongs to in feature l

$F_{ls(l)}$ =last faction no. in the feature l

Feature l contains s(l) factions

For example if active user from age faction=1(under 18) and the other user from age faction 3(from 25 to 34) then the difference between them computed as follows:-

$$DF_{age}(a,u) = 1 - \frac{|1-3|}{7-1} = 0.7$$

Then if active user's gender is male (1) and other user's gender is female (2) then the difference between them

$$DF_{gender}(a,u) = 1 - \frac{|1-2|}{2-1} = zero$$

Then $\text{personal similarity}(a,u) = 0.7 * 0.53 + 0 * 0.47 = 0.371$

Now to compute ***recommendations list***, just take users whose personal similarity with active greater than or equal to specific threshold (80% in current system), who are considered as active user's nearest neighbors. These users will be sorted in ascending order according to their similarity. Then check items of the first user in similarity list to choose his items which satisfy higher rating value in rate scale such that the active user not purchasing them yet; finally check if they have at least one genre of active user's genres. If there is any matching between candidate item's genres and active user's genres add this item to recommendation list, continue until full the first 80% of recommendation list and then sort items from item which satisfy the largest number of active user's genres.

The above computation will be used also for registered users who have not enough ratings, whom are considered as new users. But for active registered users their similarity will be two parts first one will be personal similarity and the second part will be rating similarity as shown in equation (3.7).

$$\text{Total } sim(a,u) = \text{personal similarity} * 0.5 + \text{rating similarity} * 0.5 \quad ...3.7$$

3.3.4 The problem of new item

Each recommender system using collaborative filtering will face the problem of new item, where there are no or not enough ratings about those items, so system can not recommend them. In this work such problem will be solved in two stages, as follows:

1- First, in separate list or the last part of recommendation list (the last 20%) will be recommended from those new items which satisfy the all or at least one of the active user's preferred genres, and indicate that these items are new items. Algorithm (3.2) explains this process.

2- Second, these items will be rated in the login stage, where, when the registered user login, the list of recommendations from last visit will show to him to give his ratings on them, therefore the ratings on the new items will be accumulated by time.

Algorithm (3.2): compute recommendation for new items.

Precondition: item table and rating table
Output/action: the last part of recommendation list.

```
{Rec_List= array of recommendation items}
{ n= no. of all items in DB}
{new= integer variable to save  the number of new items who satisfy at least one of preferred genres of active user}
{m= no. of items in Rec_list}
                    ***************************
For i= 1 to n
        If no. of users  rating on  item (i) < 10 then
                For j= 1 to m                                   {m= no. of active genres}
                        If item(i) ∈ active gener(j) then
                                flag=flag+1                     {flag=integer counter}
                         End if
                End for j
                 If flag > 1 then
                         Add item(i) to Rec_List
                         new=new+1
                        If (new/m)≤ 0.2 then
                                Exit for i
                       End if
                End if
        End if
End for i
```

3.4 The design steps

There are five steps to design the proposed system (knowledge acquisition, shared information, profile representation, knowledge source and recommendations technique).

3.4.1 Knowledge acquisition

The system gives the users the ability to update his information in the data base (DB) by changing or updating any possible part of his personal record and his interest or the preferred genres. The knowledge is also acquired from the rating given by the user on the items purchased in the previous visit by refusing item(s) or accepting these items with the degree of acceptance. Further more the user must gives a list of interested genres in the first visit to the system which might be changed during any future visits. Registered user can obtain his recommendation list either from previous ratings or from new ratings (treated as new user). From this information one can compute the similarity with others and the impact ratio between time and time. So the system can deduce for example that some features are not necessarily, and can be neglected and others can be strongly accepted. These can summarize by figure (3.2).

3.4.2 Shared information

The using of content-based filtering needs to shared personal features between active user and other users to compute personal similarity, also because using collaborative filtering the rating information needs to be shared between users to find neighborhood for active user where it need to know shared items (item feedback and items groups) to compute recommendation list and rating similarity.

s

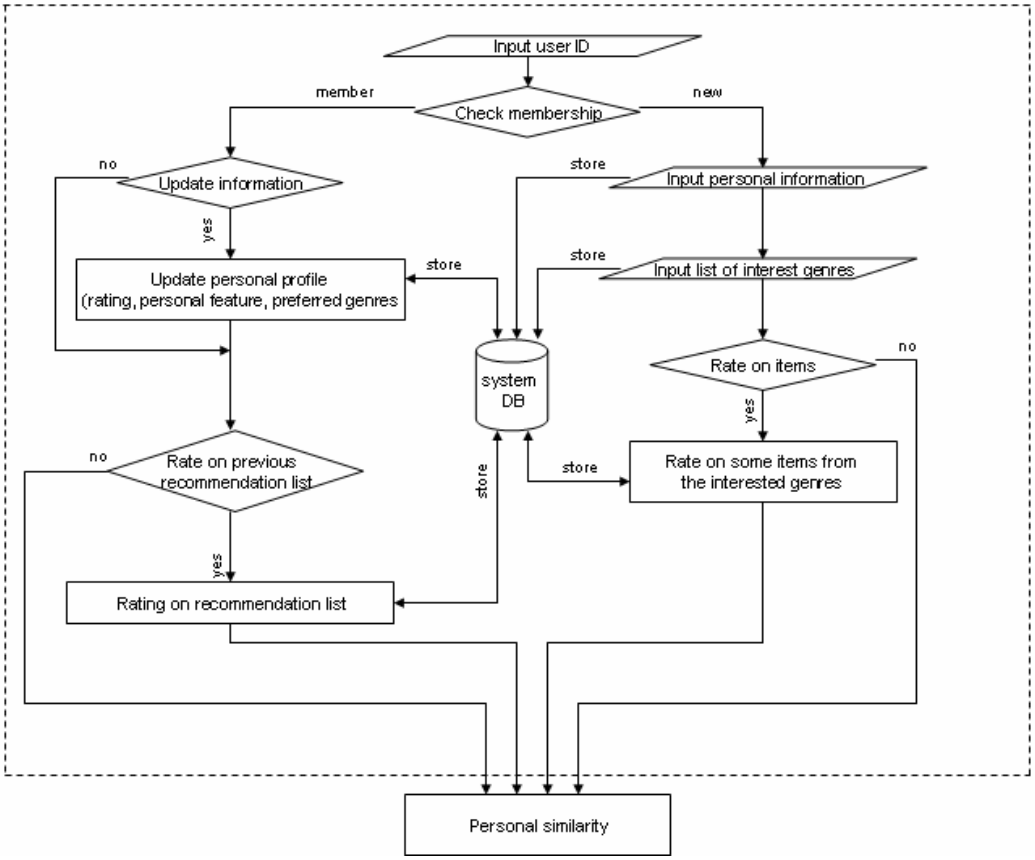

Figure (3.2) login for active user

3.4.3 Profile representation

Each user will be represented by record of two vectors (vector model representation), first vector contain a set of personal features ($feature_1$ … $feature_n$) and each feature represented as table of factions. The second vector contain ratings of user, this vector could be empty or partially empty for new users or lazy registered users. Also this record contains a set of user's preferred items genres.

Each item will be represented by a vector model also. This vector contains details (features) of this item (depending on application). In current work the contents of item profile will not be used in computation

of similarity, but it will used in describing items in the recommendations list only.

3.4.4 Knowledge source

The main information source for the system is internal database of users and items; either constructed or already used DB. This DB contain a list of items information and a list of personal information with a set of items genres (types) preferred by the active user. Also a third part of information is ratings on items for each user.

3.4.5 Recommendations technique

The proposed system is designed by mixing two main techniques (collaborative and content-based). A memory based collaborative filtering (CF), which is the most prevalent approach operates over the entire users preference part of the DB (the ratings of the user on the items), to make recommendation. The method chosen is the Pearson Correlation Coefficient (PCC) method given by equation (2.1), because it is generally achieves higher performance than other methods in computing similarities (rating similarity). While the content based filtering is used to compute the personal similarity depending on the vector of personal features and the impact ratios for these features. Figure (3.3) summarized these operations.

3.4.5.1 Personal similarity

The personal similarity is computed first which is depends on the personal features and impact ratio of these features only, which is given by equation (3.5) and the detail explained in algorithm (3.3).

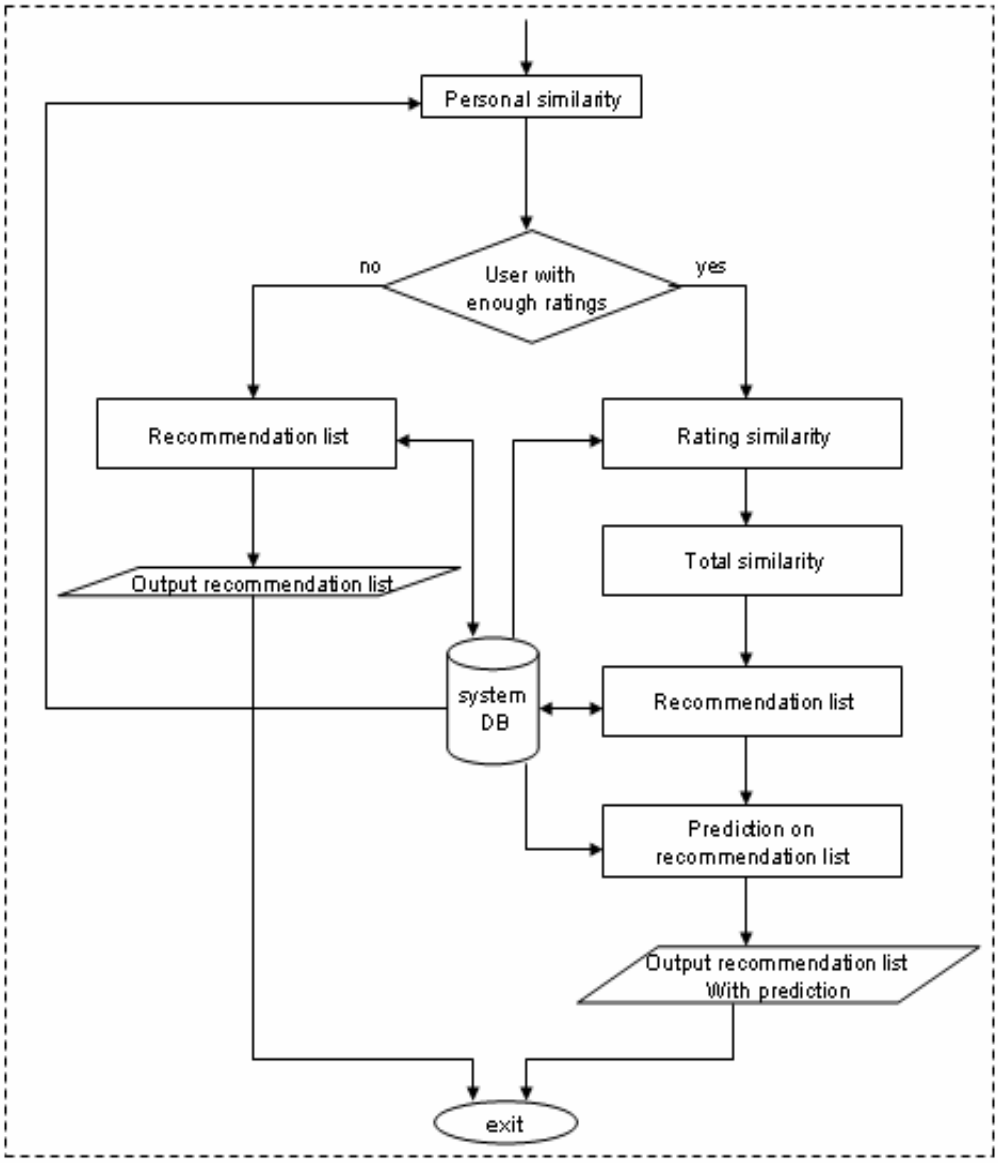

Figure (3.3) the proposed system

3.4.5.2 Rating similarity

The rating similarity is computed for the active user who has enough ratings on items with the users who have greater than (80%) of personal similarity with active user and shared more than (40%) of the rated items with active user. This similarity is computed using the Pearson Correlation Coefficient given by equation (2.1). The system first will search about users who is (rated) shared by (40%) of items rated by

active user these users will called as co-users. The algorithm (3.4) explain how find co-users. Then algorithm (3.5) explain how find rating similarity.

Algorithm (3.3):- compute personal similarity between active user and others

Precondition: active user personal profile, Impact ratios (w), personal profile for all users in DB

Output/action: list of users who satisfy 80% of personal similarity with active user

*{F_{la} =faction of active user for feature **l** }*
*{F_{lu} =faction of user (u) for feature **l** }*
*{$F_{ls(1)}$ =last faction of feature **l** }*
*{F_{l1} =first faction of feature **l** }*
{DF=array of 1D to save difference between features of active user and any other user}
Personal_sim=array of 1D to save personal similarity between active user and users who satisfy 80% of personal similarity}

```
**********
lik=0                {lik= integer variable to save no. of likeminded users}
For x= 1 to all users
    sum=0                                  {sum=temporary real variable
                                            to save personal similarity}
     For l = 1 to nof                      {nof=number of personal features}
          DF(l)= 1- (abs(F_la -F_lu ))/(F_ls(1) -1)
                sum = sum+(DF(l)*W(l)
     End for l
     If sum ≥ 0.8 then
          Personal_sim(x)= sum
          lik=lik+1
          Users_id(lik)=x
     End if
End for x
Sort Personal_sim list and User_id list according to their similarity in
descending order
```

Algorithm (3.4):- finds co-users for active user (users who satisfy 40% of shared items with active user.

Input: the number of users similar in personality (whose achieve 80% of personal similarity), their rating profile, user_id list and rating profile of active user.

Out put/Action: co-users list, co-rated items and average ratings for each co-user and active user himself.

```
{Co-user=array of records}, {Co-user.id=id of user who satisfy 40% of shared items}, {Co-usrer.co-rate=array to save shared items},Co-user.no=no. of shared items}, {Avrg=array to save average for co-users and active user}
{active-sum=integer variable to accumulate ratings of active user}
        ****************************************
Y=0                                              {y=number of co-users}
Active-sum=0
For i= 1 to lik        {lik= no. of users who satisfy 80% of personal similarity}
      x=0                                {x=number of co-rated (shared)  items}
      For j= 1 to w                              {w= number of active user's items}
            sum=0   {sum=integer variable to accumulate ratings of user i}
            For k= 1 to s                    {s=number of all  items for user i}
                  sum=sum+ rating of item_k
                  If item_j =item_k  then
                        x=x+1
                        co-rated(x)=item_j
                  End if
            End for k
            If (x/w)≥0.4 then
                  y=y+1
                  Co-user(y).Id=user-id(i)
                        For z=1 to x
                              Co-user(y).co-rated(z)=co-rated(z)
                        End for z
                  Co-user(y). no=x
                  Avrg(y)=sum/s
            End if
      End for j
End for i
For j=1 to w
      active-sum=active-sum+ rating of item_w
End for j
Avrg(active user)= Active-sum/w
lik=y
```

Algorithm (3.5):- compute rating similarity

Precondition: co-users list, co-rated items and average ratings for each co-user and active user himself.
Output/Action: rating similarity list.

{u=co-user(i).id}
{a=active user Id}
{ R_{uj} = rating of user $_u$ on co-user(i).co-rated(x) }
{R_{aj} = rating of active user on co-user(i).co-rated(x) }
{up=real variable to compute couching of ratings similarity equation}
{down=real variable standing of ratings similarity equation}

****

For i= 1 to lik *{lik= no. of co-users}*
up=0
down=0 *{u=active user ID}*
For j= 1 to x *{x=no of co-rated items}*
up=up+((R_{aj} -avrg(a))(R_{uj} -avrg(u)))*
*down=down+(sqrt(R_{aj} -avrg(a))^2 *(sqrt(R_{uj} -avrg(u))^2*
End for j
rating_sim(i)= up/down
End for i
Sort rating_sim from highest similarity

3.4.5.3 Total similarity

The total similarity is computed by using both personal and rating similarity, and given by equation (3.7), and described by algorithm (3.6).

Algorithm (3.6):- compute total similarity

Input: personal similarity list and rating similarity list.
Output/Action: final similarity list.

{u= co-user(i).id }

****

For i= 1 to lik

*sim(u)=personal_sim(u)*0.5+rating_sim(u)*0.5*

End for i

Sort sim list from highest similarity

3.4.5.4 The computation of recommendation list

There are two cases to be considered. The first case is that if the active user or lazy registered users who have no enough ratings then the personal similarity and the preferred genres are used to compute the recommendation list of items. Where the elements of recommendation list will be taken from the high rated items of nearest similar user and each of these items must be achieve at least one of the preferred genres of active user. This process explained in algorithm (3.7).

Algorithm (3.7): computation of the recommendation using personal similarity and preferred genres of active user only.

Precondition: users whom satisfy 80% of personal similarity, preferred genres of active user.
Output/action: Recommendations list for new user or lazy registered users whom have not enough ratings.

```
{Flag1=Boolean variable to check if candidate item is shared between active user and comparable user}
{Rec-list=array of 1D to save recommendation list}
{Flag2=integer variable to accumulating the no. of active user's preferred genres found in candidate item}
*******************************************
x=0                                          {x= No. of items in Rec-list}
For i = 1 to lik          {lik=users whom satisfy 80% of personal similarity}
    For j= 1 to n              {n=no. of high rated items by user (i)}
        flag1=false
        For w= 1 to m                        {m=no. of active user items}
            If item_j =item_w  then
                flag1  = true
                Exit for w
            End if
        End for w
                                  {checking if item_j purchased by active user}
        If flag1 = false then
            For s = 1 to z
                                  {z=no. of preferred genres for active user}
                If item_j ∈ Genre_s  then          To be continue
```

```
                                        flag2_j = flag2_j + 1
                                    End if
                                End for s
                            End if
                            If flag2_j > 0 then
                                x = x+1
                                rec-List_x = item_j
                            End if
                            If x ≥ 10 then
                                Exit for i
                            End if
                        End for j
End for i
Sort rec-list from the item whose satisfy the highest no. of active user's
preferred genres.
```

The second case is the recommendation list using the total similarity, and the items in the list taken from items of higher ratings from nearest user to the active user. The computation of recommendation list from second case will be given in algorithm (3.8). Then the prediction of each item in this recommendation list is given by algorithm (3.9). Which is useful in calculation of the Mean Absolute Error (MAE).

Algorithm (3.8):- computation of recommendation list use total similarity (personal and rating similarity).

Precondition : users in total similarity list
Output: top N-recommendation.

```
{Rec-list= recommendation list}
{u=co-user(i).id}
            ****************************************
While m < 10                                    {m=number of items in rec_list}
    For i= 1 to lik          {lik= no. of similar users in total similarity list}
        For j= 1 to h(u)          {h(u)=no of high rated items of user u}
            flag=false
            for w= 1 to x                        {x= items of active user}
                if item_j = item_w then
                    flag = true                              To be continue
```

```
                        exit for w
                    end if
                end for w
                if flag=false then
                    rec_list(m)=item_j
                    m=m+1
                    if m ≥ 10 then
                        exit while loop
                    end if
                end if
            end for j
        end for i
end while loop
```

Algorithm (3.9):- compute prediction for recommendation list

Precondition: recommendation list that formed depending on total similarity.
Output/Action: prediction for each item in recommendation list.

```
{u=co-user (j).Id}
{a=active user Id}
        ****************************************
For i= 1 to m                       {m= no. of items in recommendations list}
    For j= 1 to x   {x= no. of users in total similarity list who rated item_i }
        up=up+ sim(u)*(R_ui - avrg(u))   { R_ui =rating of user_u on item_i }
        down=down+abs(sim(u))
    End for j
    prd(i)=avrg(a)+(up/down)            {prd(i)= prediction on item_i }
End for i
Sort rec_list from high prediction
```

3.4.5.5 The Mean Absolute Error (MAE)

The mean absolute error is a measure of the deviation of recommendations from their true user-specified values. It is computed using equation (2.8). By taking the predictions on the last

recommendations list and the user ratings on them, for many users. from this information the (MAE) can be computed as in algorithm (3.10).

Algorithm (3.10):- compute MAE

Precondition: random no. (n) of registered users who gives enough ratings on their last recommendation list, predictions on these rated items.

Output/Action: MAE.

```
{n= no of registered users who rate enough no.  of recommended items }
{r(m)=array to represent the no. of rated items in rec-list for each user}
{R_j  = real rating of registered user i on items j}
{prd(j)=prediction value on item j}
{sum=real variable to compute MAE for each user}
          ****************************************
For i= 1 to n
        Sum=o
       For j= 1 to r(m)
             Sum=sum+abs(R_j -prd(j))
       End for j
       Sum=sum/r(m)
      MAE=MAE+sum
 End for i
MAE=MAE/n
```

The system performance depends mainly on the computation of the (MAE) to evaluate its performance. Then if the (MAE) is greater than some threshold value, after (n) users using the system, the calculation of the impact ratio will be recalculated to enhanced the computation of personal similarity and then the recommendations list. This process can be done between time and time. During this process one can eliminate one or more of the personal features and see its effect and so on.

3.5 System implementation

In the first of the system implementation one need to reconstruct the used DB because it is not compatible with designed system in this work.

The original DB does not contain the preferred genres (classes) and the ratings stored in matrix form. For these reasons the DB spread in three parts. The first contains the personal profiles including the personal features, the preferred genres and the ratings attached to user profile also the recommendations of the last visit. The second part contains the items profiles. The third part is used to store the last computed Impact Ratios.

There are friendly used interfaces is designed and implemented to interact with the active user. These interface windows are fully described in appendix B.

Chapter Four

Conclusions and Suggestions for Future Works

Chapter Four

Conclusions and Suggestions for Future Works

4.1 Conclusions

The conclusions that can be drawn from this work are listed as follows:

1. The computation of the Impact Ratios is highly recommended which is one of the main ideas introduced in this work. These impact ratios are highly helpful to give the weight and effect of each personal feature in computing the similarities and then the recommendations list.
2. The use of CB filtering approach on personal features is also recommended, because the information available on the user is more effective.
3. The use of CB approach on personal similarity reduces the number of users in personal neighbors, to be used in the second part which the ratings similarity computed.

4.2 Suggestions for future works

There are many ideas to increases the performance of the recommender system.

1. Hybridizing the recommendation again by adding items similarity in addition to other similarities used in this work.
2. Increase reasonable the number of personal features which are suitable to the application of the system.
3. An expert required classifying the factions in each feature and that depends on the application also.
4. Introduce a machine learning approach to make the system more intelligent in re-computing the Impact Ratios and eliminating features or asking for other features, to make the system more stable.
5. Apply the proposed system on distributed data base, or Intranet, or on the internet

References

References

[Abd06] Ahmed Abdalaa, ***"Techniques for Personalization in E-learning***", M.Sc thesis, Department of Computer Science, Vrije University, Brussel, Belgium, 2006.

[Aga06] Nitin Agarwal, Ehtesham Haque, Huan Liu and Lance Parsons, ***" A Subspace Clustering Framework for Research Group Collaboration ",*** Supported by grants from Prop 301 (No. ECR A601) and CEINT 2006.

[Ahn07] Hyung Jun Ahn***, "A Hybrid Collaborative Filtering Recommender System Using a New Similarity Measure",*** Private Bag 3105, Hamilton 3240, New Zealand, 2007.

[Bas04] Aseel Basim , "***Recommender System for E-Commerce Data***", M.Sc. Thesis, Department of Computer Science, Al-Nahrain University, Baghdad, Iraq, 2004.

[Bel07] Robert M. Bell and Yehuda Koren, ***"Improved Neighborhood-based Collaborative Filtering",*** *KDDCup'07,* August 12, 2007, San Jose, California, USA. Copyright ACM 978-1-59593-834-3/07/0008, 2007

[Bha07] Runa Bhaumik, Robin Burke, Bamshad Mobasher,*"* ***Effectiveness of Crawling Attacks Against Web-based Recommender Systems",*** This work was supported in part by the National Science Foundation Cyber Trust program under Grant IIS-0430303. Copyright c 2007, Association for the Advancement of Artificial Intelligence (www.aaai.org).

[Bur00] Robin Burke, ***"Knowledge-based recommender systems",*** To Appear in the Encyclopedia of Library and Information Science, 2000

[Bur02] Robin Burke, ***"Hybrid Recommender Systems: Survey and Experiments"***, To appear in User Modeling and User-Adapted Interaction.2002

[Her00] Jonathan Lee Herlocker, ***" Understanding and Improving Automated Collaborative Filtering Systems",*** Ph.D Thesis, The Faculty of the Graduate School of the University of Minnesota, 2000

[Her03] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen and John T. Riedl, ***"Evaluating Collaborative Filtering Recommender Systems",*** This research was supported by the National Science Foundation under grants DGE 95-54517, IIS 96-13960, IIS 97-34442, IIS 99-78717, IIS 01-02229, IIS 01-33994, and by Net Perceptions, Inc. 2003

[Ho 06] Ai Thanh Ho, Esma Aïmeur, ***"DIGICAM: A NEED-BASED RECOMMENDER SYSTEM"***, ISBN: 972-8924-23-2 © 2006 IADIS.

[Jon07] Nicolas Jones and Pearl Pu, ***"User Technology Adoption Issues in Recommender Systems",*** Human Computer Interaction Group, Swiss Federal Institute of Technology, CH-1015 Lausanne, Switzerland, 2007

[Lee05] Choong Hee Lee, Junseok Hwang, ***"Private Information Shielding Service for Overcoming Privacy Risk in Recommender System",*** Research paper , Techno-Economics & Policy Program, Seoul National University, Seoul, Korea, 2005

[Lem05] Daniel Lemire and Sean McGrath, ***"Implementing a Rating-Based Item-to-Item Recommender System in PHP/SQL"***, Technical Report D-01, January 2005 (the report was edited on November 4th 2005 following comments by Chris Harris.

[Li 04] Qing Li and Byeong Man Kim, ***"Constructing User Profiles for Collaborative Recommender System"***, This work was supported by grant No. 2000-1-51200-008-2 from the Korea Science and Engineering Foundation, 2004.

[Ma 07] Hao Ma, Irwin King and Michael R. Lyu, "***Effective Missing Data Prediction for Collaborative Filtering"***, *SIGIR'07*, July 23–27, 2007, Amsterdam, The Netherlands. Copyright 2007 ACM 978-1-59593-597-7/07/0007

[Mcl04] Matthew R. Mclaughlin and Jonathan L. Herlocker, ***"A Collaborative Filtering Algorithm and Evaluation Metric that Accurately Model the User Experience"***, the definitive version was published in proceedings of the 27th Annual international conference on Research and Development in Information Retrieval, 2004.

[Mid02] Stuart E. Middleton, Harith Alani, Nigel R. Shadbolt, David C. De Roure, ***"Exploiting Synergy Between Ontologies and Recommender Systems",*** Semantic Web Workshop 2002 Hawaii, USA, 2002.

[Mid03] Stuart Edward Middleton, ***"Capturing knowledge of user preferences with recommender systems"***, Ph.D thesis, Faculty of Engineering and Applied Science, Department of Electronics and Computer Science, University of Southampton, May 2003.

[Mil03] Bradley N. Miller, Istvan Albert, Shyong K. Lam, Joseph A. Konstan John Riedl***, "MovieLens Unplugged: Experiences with a Recommender System on Four Mobile Devices",*** Research paper, GroupLens Research Group, University of Minnesota, Minneapolis, MN 55446, 2003.

[Mil04] Bradley N. Miller, Joseph A. Konstan, and John Riedl, ***"PocketLens: Toward a PersonalRecommender System",*** ACM Transactions on Information Systems, Vol. 22, No. 3, July 2004, Pages 437–476.

[Mob07] Bamshad Mobasher, ***"Recommender Systems",*** Research paper, 2007.

[Nat07] Tavi Nathanson, Ephrat Bitton and Ken Goldberg, ***"Eigentaste 5.0: Constant-Time Adaptability in a Recommender System Using Item Clustering",*** RecSys'07, October 19–20, 2007, Minneapolis, Minnesota, USA. Copyright 2007 ACM 978-1-59593-730-8/07/0010

[Pap04] Manos Papagelis and Dimitris Plexousakis, ***"Qualitative Analysis of User-based and Item-based Prediction Algorithms for Recommendation Agents",*** Institute of Computer Science, Foundation for Research and Technology - Hellas and Department of Computer Science, University of Crete, P.O. Box 2208, GR-71409 and P.O. Box 1385, GR-71110, Heraklion, Greece, 2004.

[Paz98] Michael J. Pazzani, ***"A Framework for Collaborative, Content-Based and Demographic Filtering",*** Irvine, CA 92697, 1998.

[Ric07] Francesco Ricci, ***"Collaborative Filtering and Evaluation of Recommender Systems"***, Research paper, 2007.

[Rou06] A. M. Roumani and D. B. Skillicorn, ***"A Dimensionality Reduction Technique for Collaborative Filtering",*** 27 November 2006, External Technical Report ISSN-0836-0227-2006-527, Department of Computing and Information Science Queen's University, Kingston, Ontario, Canada K7L 3N6 Document prepared December 4, 2006.

[San00] Ramon Sanguesa, Alberto Vazquez-Huerga and Javier Vazquez-Salceda, ***"Mixing Collaborative and Cognitive Filtering in Multiagent Systems",*** Campus Nord, Mòdul C-6, Despatx 204, C/Jordi Girona Salgado, 1-308034 Barcelona, SPAIN, 2000

[Sar99] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, John T. Riedl, ***"Application of Dimensionality Reduction in Recommender System -- A Case Study",*** Research paper, GroupLens Research Group / Army HPC Research Center,

Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455, +1 612 625-4002, 1999.

[Sar00] Badrul M. Sarwar, George Karypis, Joseph Konstan and John Riedl, ***"Recommender Systems for Large-scale E-Commerce: Scalable Neighborhood Formation Using Clustering",*** Research paper, GroupLens Research Group /Army HPC Research Center, Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455, USA, 2000.

[Sar01] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl, ***"Item-based Collaborative Filtering Recommendation Algorithms",*** Appears in WWW10, May 1-5, 2001, Hong Kong.

[Seo03] Jong-Hwan Seo, Kun-Pyo Lee, ***"Development of Website Design Personalization Service Using Design Recommender System",*** Dept. of Computer Graphics, Tongmyung University of Information Technology , 535Yongdang-dong, Busan and Dept of Industrial Design, Advanced Institute of Science and Technology, 373-1Guseong-dong, Yuseong-gu, Daejeon, KOREA, 2003.

[Sot05] Rosario Sotomayor, Joe Carthy and John Dunnion, ***"The Design and Implementation of an Intelligent Online Recommender System",*** American Association for Artificial Intelligence (www.aaai.org), 2005.

[Tos08] Andreas Toscher, Michael Jahrer, and Robert Legenstein, ***"Improved Neighborhood-Based Algorithms for Large-Scale Recommender Systems",*** 2nd Netflix-KDD Workshop, August 24, 2008, Las Vegas, NV, USA. Copyright 2008 ACM 978-1-60558-265-8/08/0008

[Ume08] Takashi Umeda, ***"Item-Based Collaborative Filtering Recommendation Algorithms***", Seminar, Deguchi Lab., 2008.

Mail: umeda07[at]cs.dis.titech.ac.jp

Web: http://umekoumeda.net/

[Mo 01] Wentao Mo, ***"A Referral-Based Recommender System for E-commerce"***, Msc thesis, Department of Computer Science, North Carolina State University, 2001.

[Woo04] Brandon Douthit-Wood, ***"A Content-Based Approach to Collaborative Filtering",*** CS 470: Applied Software Development Project, 2004.

[Xue05] Gui-Rong Xue1, Chenxi Lin, Qiang Yang, WenSi Xi, Hua-Jun Zeng, Yong Yu, Zheng Chen, ***"Scalable Collaborative Filtering Using Cluster-based Smoothing",*** SIGIR'05, August 15–19, 2005, Salvador, Brazil.Copyright 2005 ACM 1-59593-034-5/05/0008

[Yu 02] Kai Yu, Xiaowei Xu1, Jianhua Tao, Martin Ester and Hans-Peter Kriegel, ***"KInstance Selection Techniques for Memory-Based Collaborative Filtering"***The work was performed in Cooperate Technology, Siemens AG,2002

[Zie05] Cai-Nicolas Ziegler, ***"Towards Decentralized Recommender Systems",*** Phd thesis, Fakult¨at f¨ur Angewandte Wissenschaften, Institut f¨ur Informatik, Albert-Ludwigs-Universit¨at Freiburg, 2005.

Appendix A

Appendix A

الجنس	العمر	الوظيفة	التحصيل الدراسي	نوعية الافلام المفضلة (الاختيار واحد او اكثر)
ذكر ☐	تحت 18 ☐	الكاتب ☐	اعدادية فما دون ☐	اكشن ☐
أنثى ☐	24-18 ☐	أكاديمي / مربي ☐	دبلوم فني ☐	مغامرة ☐
	34-25 ☐	الفنان ☐	بكلوريوس ☐	صور متحرّكة ☐
	44-35 ☐	كتابية / إدارة ☐	ماجستير فما فوق ☐	الأطفال ☐
	49-45 ☐	طالب كليّة ☐		كوميديا ☐
	55-50 ☐	خدمات الزبائن ☐		تحقيقات جنائية ☐
	+56 ☐	طبيب / رعاية صحية ☐		برنامج وثائقي ☐
		مدير تنفيذي / إداري ☐		دراما ☐
		المزارع ☐		خيالي ☐
		صانع بيتي ☐		فلم سوداوي ☐
		المحامي ☐		رعب ☐
		المبرمج ☐		مسرحية موسيقية ☐
		متقاعد ☐		لغز ☐
		مبيعات / تسويق ☐		رومانسية ☐
		العالم ☐		خيال علمي ☐
		المهنة الحرة ☐		قصة مثيرة ☐
		تقني / مهندس ☐		حرب ☐
		تاجر / حرفي ☐		غربي ☐
		عاطل ☐		
		آخرى أو ليس محدّد ☐		

Table (A.1) survey questionnaire

Appendix B

Appendix B

Movies Recommender System implementation interfaces

The Main form contains three buttons as shown in Figure (B.1):

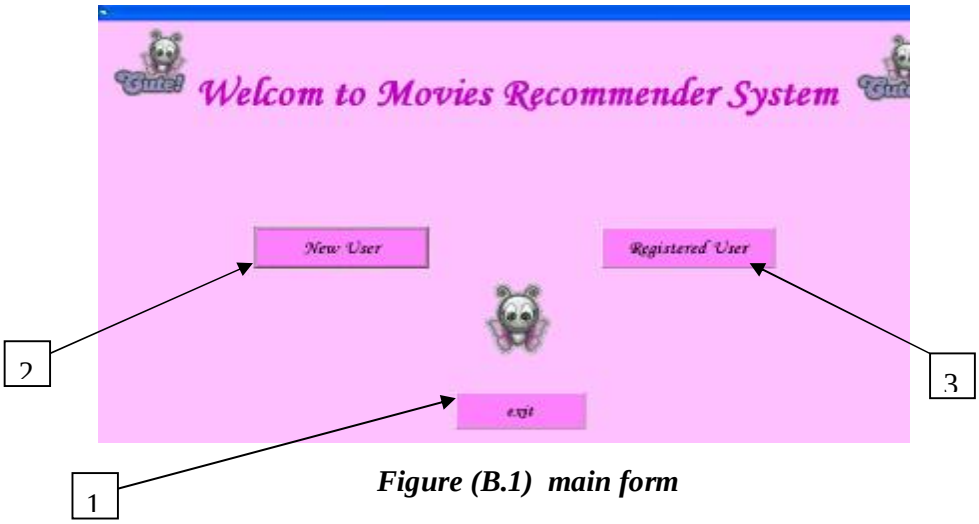

Figure (B.1) main form

1. ***Exit :-***exit from the program run
2. ***New user: -*** the new user form will be display to add new account for the user who visits the system for the first time as shown in figure (B.2).

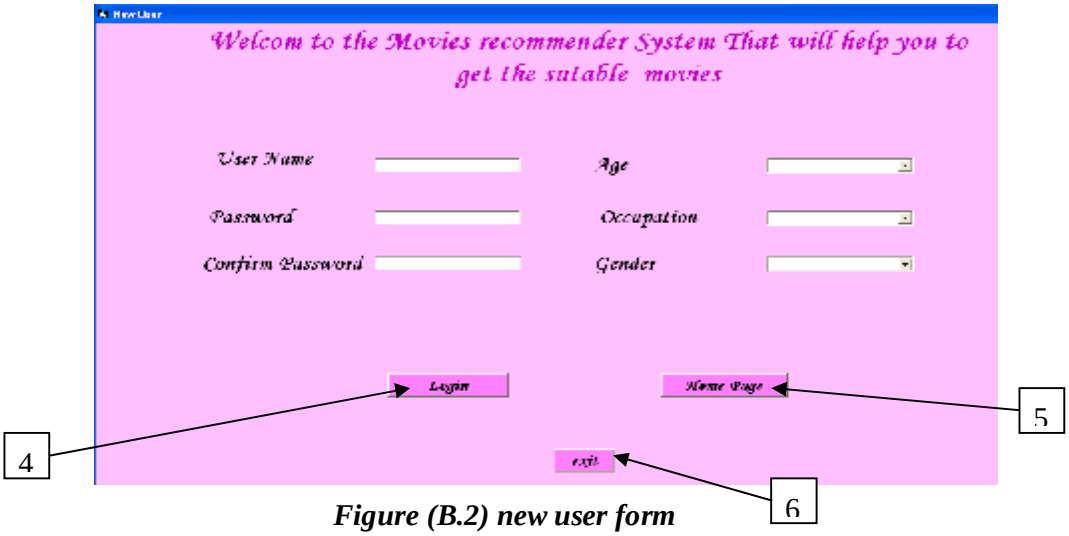

Figure (B.2) new user form

3. ***Registered user:-*** the repeated user form will be display for registered user to deal with system as shown in figure (B.3)

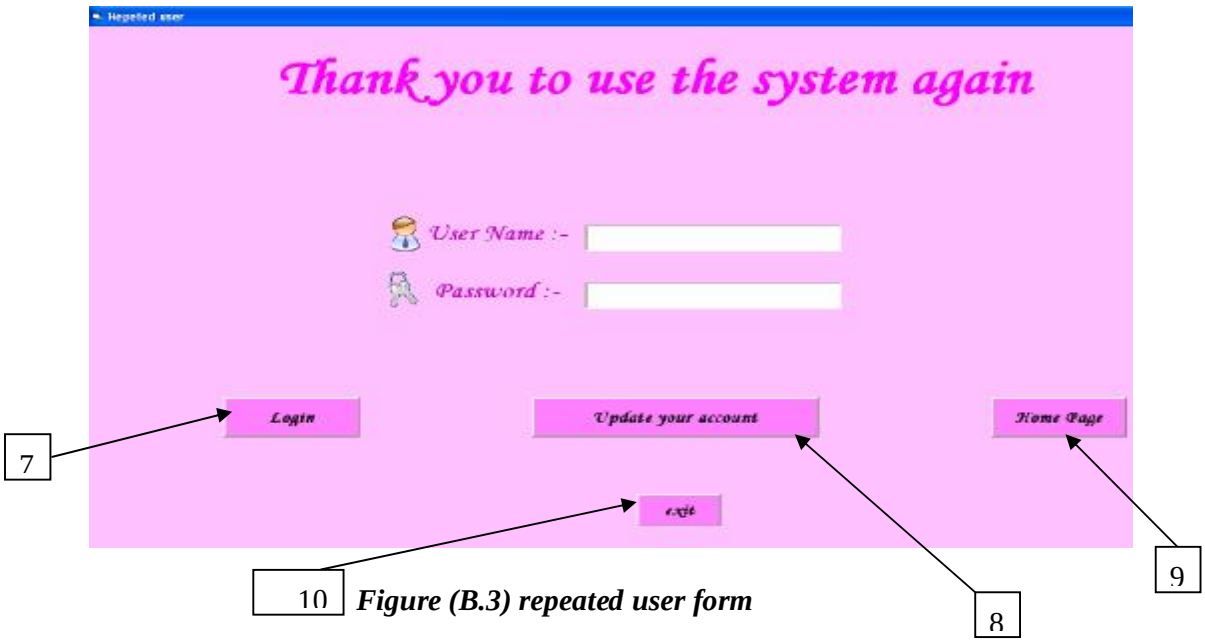

Figure (B.3) repeated user form

The new user form contains three buttons as shown in figure (B.2):-

4. ***Login:-***The movies types form will be display for new user to choose types of the movies which he prefers as shown in figure (B.4)

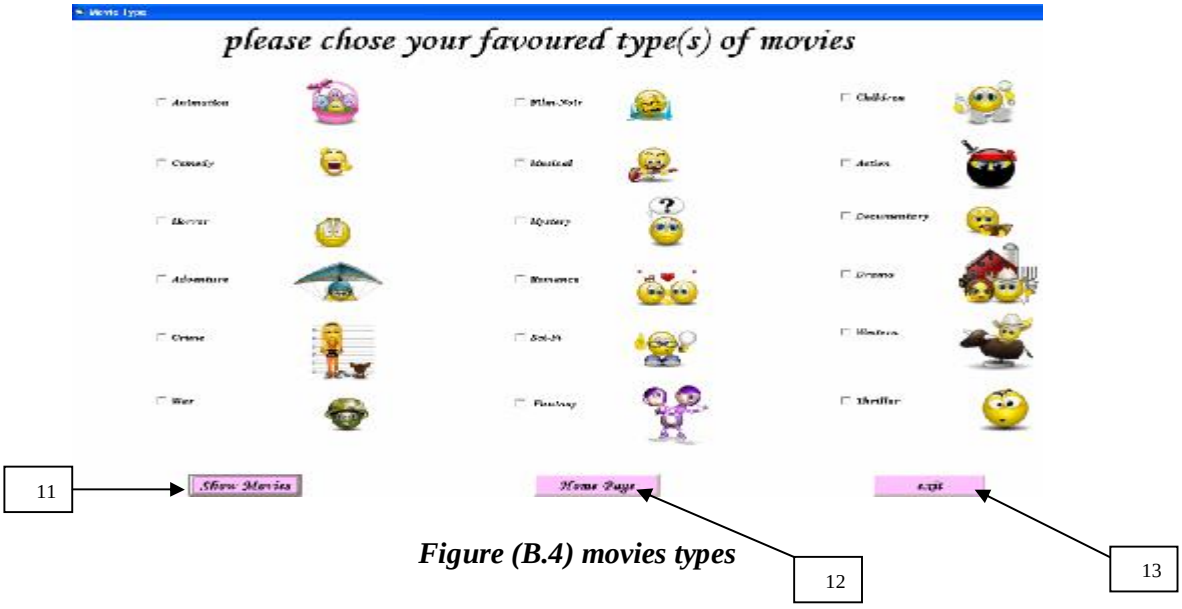

Figure (B.4) movies types

5. Home Page: - back to the main form

6. ***Exit:*** **-** end the program execution

The registered user form contains three buttons as shown in figure (B.3):-

7. Login: **-** a form will display to asking repeated user to rating his recommendation list from last visit as shown in figure (B.5).

Figure (B.5) last recommendation list

8. Update your account:-a frame will be display to ask repeated user about his new information to update his account as shown in figure (B.6).

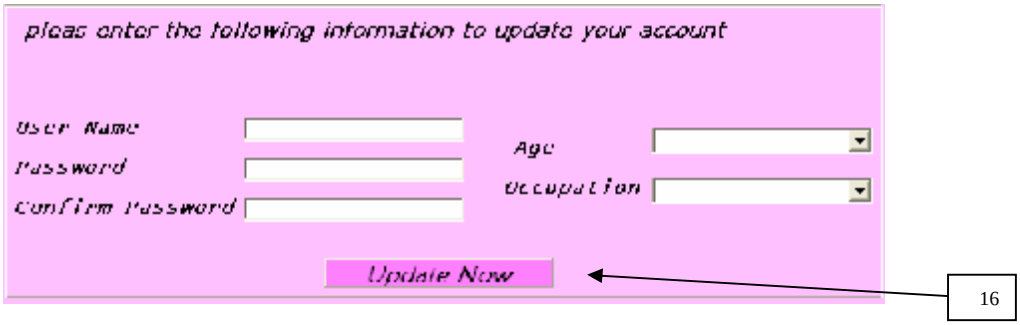

Figure (B.6) system ask user if he wants to update his account

9. Home page: **-** back to main form

10. Exit:-end the program execution

The movies types form contains the following buttons as shown in figure (B.4):-

11. show movies:- depending on types that active user chose them the system will display the list of movies available on data base as shown in figure (B.7)

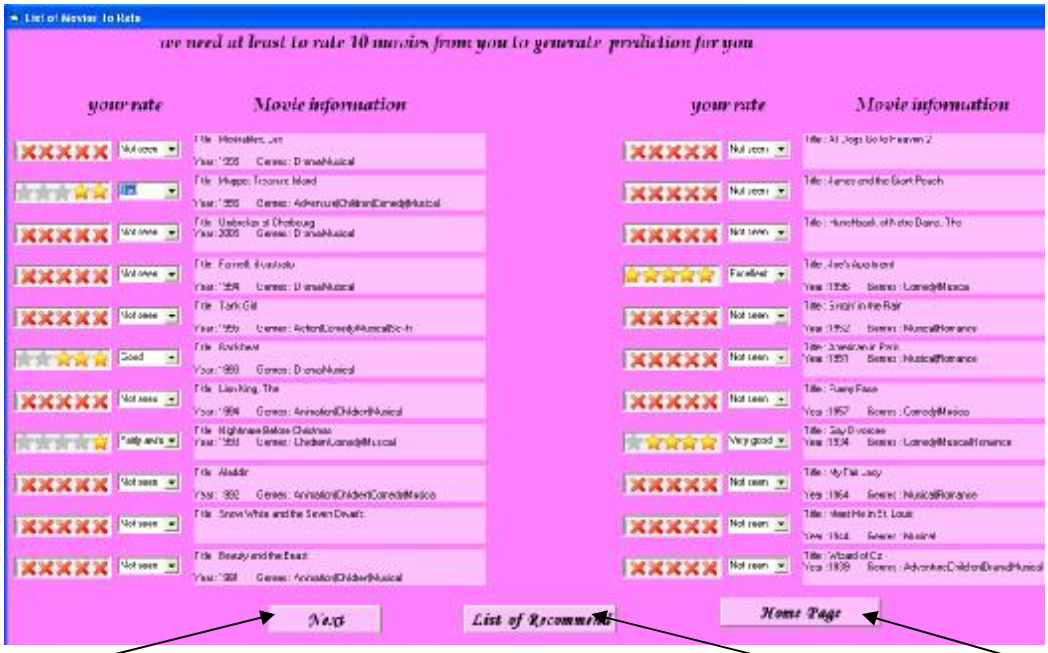

Figure (B.7) list of movies to rate

12. Home page: - back to main form

13. Exit:-end the program execution

The form that asks active user to rate the recommendations list from the last visit contains two buttons as shown in figure (B.5):-

14. Save: - this button will save repeated user's ratings on the recommendations list from the last visit, then display a frame to ask repeated user from where he wants his recommendations as shown in figure (B.8).

15. Cancel: - this button will display a frame to ask repeated user from where he wants his recommendations as shown in figure (B.8).

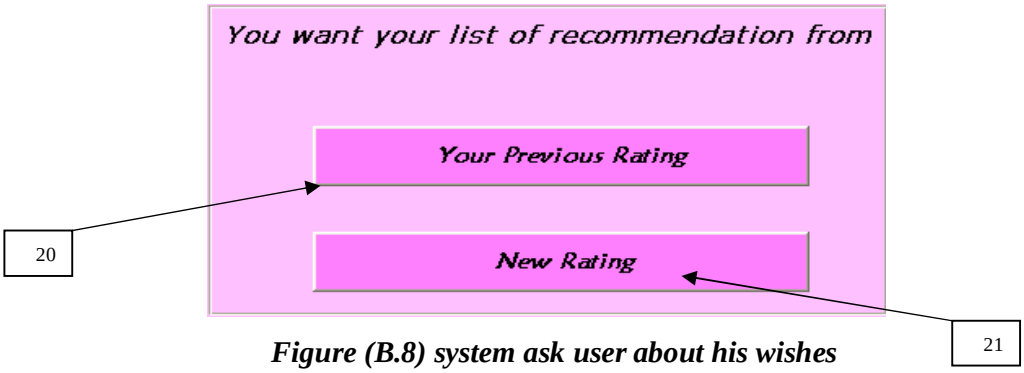

Figure (B.8) system ask user about his wishes

16. ***Update***: - updated the active user account after enter his new information as shown in figure (B.6).

The list of movies form contains three buttons as shown in figure (B.7):-

17. ***Next:*** - Ask active user if he wants to save his rating on current list of movies and then switch to the next list until there are no more movies available from the chosen types.
18. ***list of recommendations:-*** the system will produce a recommendations list for active user with prediction if he give more than 10 ratings on movies as shown in figure(B.9) else the system will produce recommendation list without prediction as shown in figure(B.10).

Figure (B.9) the list of recommendations with predictions

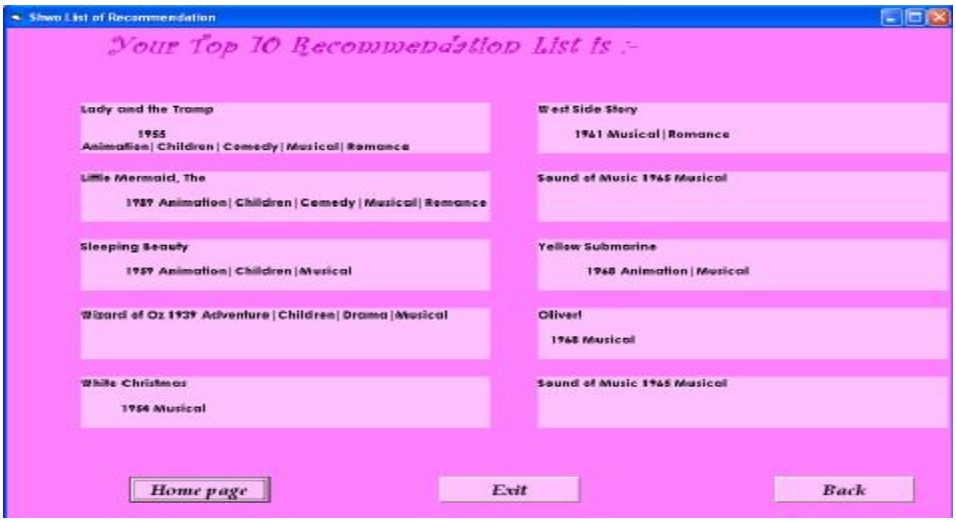

Figure (B.10) the list of recommendations

19. Home Page: - back to the main form.

The frame that ask active user from where he want his recommendation contains two buttons as shown in figure (3.8):-

20. Your previous rating:-this button will be display the list of recommendations and prediction on those recommendations as shown in figure (B.9)

21. New rate: - movies types form will be display as shown in figure (B.4).

The form that display recommendation list with predictions contains three buttons as shown in figure (B.9):-

22. Download:-in future work the system will download chosen movies from recommendation list.

23. Home page:-back to the main form.

24. Exit:-ending program execution.

الخلاصة

ان أنظمة التوصيات قدّمت لحلّ مشكلة البحث ضمن المعلومات المتزايدة بنسبة كبيرة. إنّ كمية المعلومات المتوفرة بالشكل الإلكتروني، مثل الأخبار، أفلام، كتب، إعلانات ومعلومات على الإنترنت اكتسحت المجتمع و سببت حيرة كبيرة للشخص الذي يبحث عن معلومات معينة. أنظمة التوصيات هي تقنيات معتمدة على الحاسوب يمكن أن تستعمل لتزويد الخدمات الشخصية بشكل كفوء في العديد من مجالات الأعمال الإلكترونية.

في هذه الإطروحة , نظام التوصيات المقترح صمّم بمزج النوعين الرئيسيين من انظمة التوصيات (اعتماد المحتوى بأستخدام المواصفات الشخصية و الترشيح التعاوني). هذا النوع من النظام الهجين ينتج التوصيات لمستعمليه في مرحلتين. في المرحلة الأولى يفتّش عن جيران المستعمل النشيط بحساب التشابه بين المستعمل النشيط و المستعملين الاخرين. إنّ التشابه يحسب في خطوتين، الخطوة الأولى تحسب التشابه الشخصي باستخدام تقنية (اعتماد المحتوى) بالإعتماد على الميزّات الشخصية و اوزانها المؤثرة فقط. اما الخطوة الثانية فهي خطوة شرطية حيث إذا المستعمل النشيط اعطى تقيمات على عدد كافي من المواد, عندها تقنية الترشيح التعاوني تستعمل لحساب التشابه حيث تعتمد على هذه التقيمات او التقديرات, بالأضافة إلى استخدام التشابه الشخصي الذي تم حسابه في الخطوة الأولى. في المرحلة الثانية قائمة من المواد الجديدة بالنسبة للمستعمل النشيط سيوصّى بها و يجب ان تكون مختارة من المواد المقدّرة بشكل عالي من قبل الجيران الأقرب له، هذه القائمة قد تكون مزودة في اغلب الاحيان بتنبؤات على مدى قبولها من قبل المستعمل النشيط.

في الجزء الذي يحسب فيه التشابه الشخصي هرت الحاجة لحساب وزن كلّ ميزّة شخصية. لذا في هذا العمل تم عمل مسح (استبيان) لحساب القيمة الأولية لنسبة التأثير (وزن) لكلّ ميزّة. ثمّ يعاد حساب هذه النسب بين وقت و اخر طبقا لمعلومات المستعملين الجديدة التي تدخل النظام. هذا التحديث يحصل بموجب نسبة الخطأ المطلق المتوسط المحسوب بين التقديرات الحقيقية و التنبؤ على هذه التقديرات.

جمهورية العراق
وزارة التعليم العالي و البحث العلمي
جامعة النهرين
كلية العلوم

تصميم و تنفيذ لنظام توصيات

رسالة مقدمة الى كلية العلوم, جامعة النهرين كجزء من متطلبات نيل درجة ماجستيرعلوم في علم الحاسوب

من قبل

زينه حسين فهد

(بكلوريوس 2005)

المشرف

د. طه سعدون باشاغا

2010 م 1431هـ