Development of Recommender System by Using Fuzzy Set Theory

A thesis presented in partial fulfillment of the requirements for the degree of

Master of Information Science in the Department of CIS

Near East University, Lefkosa, North Cyprus

By

Nameer Talal Momani

JUNE, 2008

Abstract

Recently the e-commerce applications contain tremendous information, huge number of items to buy from, numerous brands, and different options to choose from. This has lead to an information overload problem in which it is very difficult for customers to choose what item is the best for them to buy, at the same time make it difficult for vendors to promote their products and to increase profits.

The recommendation systems are introduced to solve this problem, by guiding the customers to find what information or products could be the best for them. This problem is also can be introduced as a social problem, because often people refer to experts to describe what do they need, and according to their description the experts will provide the advice that could help them the most.

In order to solve this problem, the already done algorithms in the field of recommendation systems will be investigated in details. Advantages and disadvantages for the previous work will be taken into account. After that the proposed method will build on that work to eliminate or reduce some of the problems and challenges the current systems suffer from.

This thesis will represent a new recommendation system that uses old methods in a new and innovative way. The proposed method will use the fuzzy set theory to eliminate some problems with the old applied recommendation methods.

The findings of the new fuzzy system were promising, because it really solved some serious problems in the old methods used in the recommendation process.

ÖZET

Son zamanlarda e-ticaret uygulamaları çok büyük enformasyon içermekte, çok sayıda satın alınacak ürün, sayısız marka ve değişik tercihler sunmaktadır. Bu müşteriler için aşırı yükleme sorunu doğurmakta ve müşterilerin doğru ürünü seçmelerini zorlaştırmakta, satıcılar içinde ürünlerinin pazarlamasını zorlaştırmakta ve karlarını düşürmektedir.

Öneri sistemleri bu problemin çözümü için takdim edilmiştir ve bu sistemler müşterilerin en doğru kararı vermelerini sağlayacak, onların kendileri için en doğru ürünü seçmelerini sağlayacaktır. Ayrıca önerme sistemleri bir sosyal problem olarak ta karşımıza çıkar şöyle ki birçok insan ihtiyaçlar için seçim konusunu eksperlere bırakır ve eksperler onların ihtiyaçları doğrultusunda karar verir.

Bu problemin çözümü için öneri sistemleri konusunda hazırlanmış algoritmalar detaylarda incelenecektir. Avantajları ve dezavantajları hesaba katılacaktır. Problemleri elemek için önerilen yöntemler kullanılacak ve sistemin kalitesini düşüren etmenler azaltilacaktir.

Bu tez yenilikçi bir yöntemle önerme sistemleri hakkında sunum için hazırlanmıştır. Buna göre var olan önerme sistemleri yerine karmaşık takım teorisi kullanılarak yeni bir sistem geliştirilmiş olacaktır.

Son olarak yeni karmaşık sistem konusu umut vericidir çünkü eski önerme sisteminin muzdarip olduğu birçok sorunu elimine etmiş olacaktır.

Acknowledgment

I would like to thank my brother Professor Dr. Munther Momani for his endless support.

I am very thankful for my parents for their love, patience, and care.

Thanks to Mr. Kamal Momani for making all this possible for me.

Special thanks to my advisor Dr. Ilham Hüseyinov for supervising this thesis, and encouraging me in my work.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

A	
AAAI	American Association for Artificial Intelligence
B	
B2B	Business to Business
B2C	Business to Consumer
C	
CB	Content-Based
CF	Collaborative Filtering
D	
DVD	Digital Versatile Disk
H	
HB	Heuristic-Based or Memory-Based
I	
IDF	Inverse Document Frequency
ITWP	Intelligent Techniques for Web Personalization
L	
LSI	Latent Semantic Indexing
M	
MB	Model-Based
R	

RS(s)	Recommender System(s)
S	
SVD	Singular Value Decomposition
T	
TF	Term Frequency

Chapter 1 Introduction

Product selection and recommendation is now more than additional features to the e-commerce websites, the majority of the current recommendation systems try to understand the customers and learn from their opinions and behavior over the web. One of the most important features for any system is to be user-friendly, which means the customer doesn’t have to face difficulties when using the system. Some of the recommendation systems don’t even require any effort from the customer side, this can be achieved by tracing the user behavior over the web without asking him to fill in tedious forms and information. On the other hand in order to help the customers find the products they need we need some explicit information to be taken directly from them, to make it as easy as possible these information should be simple and comprehensible. For example most of the recommendation systems obtain the users feedback about the products by rating them, this simple technique which is very popular currently can help the recommendation system to understand what the customer would like the most. By using this technique we can observe that one user may find a product wonderful but the other may have completely different point of view, and this is also very good information to help us understand the customers. For sure there are many recommendation systems with different algorithms and approaches one of them emulate the other in specific domain and vice versa.

The more accurate the results of the recommendation system the more confidence we get from customers. The user can be tolerant if he was recommended a cheap or ordinary item (e.g. DVD), but if the user wants to buy something more important, and more expensive like Laptops, the recommendation system must be more accurate, and dependable. In another words this is one of the challenges of the current recommendation systems which is the quality of recommendation.

If we think about the recommendation, we can find out that we use it in our daily life. When people wants to buy something and they are confused what to choose, the first thing they do is to ask their friends or relatives (persons they trust) to help them decide what is good for them according to how do they express what do they need. Most of the current recommendation systems are trying to simulate the real life situation to find the best solution for customers. Some of the recommendation systems as we will see are called word-of-mouth, which is an indication that people would recommend products they bought to others in case they were satisfied, and the other people who were recommended to buy those products would do the same. Still one problem which is customers are providing an advise according to their preferences or according to their loyalty to some brand, that’s why some times you can’t count on even your close friends. This is another reason for the increasing importance of recommendation systems. Although current systems are a plus to the e-commerce websites that are using them they still have to be improved, so far there is no perfect system or even excellent system, they still suffer from poor or inaccurate recommendations, difficulty to scale to new changes, giving very optimistic or very pessimistic recommendations, insufficient information about the customers and consequently misunderstanding of their needs, all of these are some of the challenges to the current recommendation systems.

1.1. Introduction

In this chapter the RSs in e-commerce will be introduced, the categories of different recommendation systems will be defined according to how recommendations are made. Also the importance of intelligent methods in recommendation process will be described. After that the common problems and challenges in current RSs will be mentioned and defined briefly. The objectives of the project will be to solve some of the aforementioned challenges. After defining the objectives, the methodology to achieve these objectives will be listed. At the end of this chapter a brief description about the content of the other chapters will be presented.

1.2. Recommender Systems in E-commerce

Recommender systems are software applications that aim to support users in their decision-making while interacting with large information spaces. They recommend items of interest to users based on preferences they have expressed, either explicitly or implicitly. The ever-expanding volume and increasing complexity of information on the Web has therefore made such systems essential tools for users in a variety of information seeking or e-commerce activities. Recommender systems help overcome the information overload problem by exposing users to the most interesting items, and by offering novelty, surprise, and relevance. Recommender technology is hence the central piece of the information seeking puzzle. Major e-commerce sites such as Amazon and Yahoo are using recommendation technology in ubiquitous ways. Many new comers are on their way and entrepreneurs are competing in order to find the right approach to use this technology effectively. (1) RSs were introduced to solve the problem of products' information overload, they can be used to efficiently provide personalized product catalogs to specific users or customers, RSs will help the customer choose the products most probably he likes, at the same time help the business generate more profits because of customer satisfaction.

Recommender systems are usually classified into the following categories, based on how recommendations are made:

- **Content-based (CB) recommendations**: The user will be recommended items similar to the ones the user preferred in the past;
- **Collaborative recommendations**: The user will be recommended items that people with similar tastes and preferences liked in the past;
- **Hybrid approaches**: These methods combine collaborative and CB methods. (2)

For each recommendation system there is more than one algorithm and technique, these algorithms will be investigated in details from many aspects (e.g. performance, advantages and disadvantages), sometimes the same algorithm might use different mathematical formulas in its' processing, for example in the collaborative recommendations two popular formulas can be used to find the similarities between two users which are *correlation and cosine-based*. The hybrid approaches as mentioned above try to combine the CB and collaborative approaches, but this doesn't mean that the two approaches are combined. It should be stressed that this approach is not exactly a hybrid approach, in the sense that both CB and collaborative filtering components are kept separately, without affecting each other. This fact allows the system to benefit from individual advances occurring in either component, since there exist no interdependencies. Furthermore, this approach is more easily extensible to additional filtering methods by allowing each method to be added as a separate component which contributes to the weighted average with a separate weight. (3)

1.3. Importance of Artificial Intelligent Methods in Recommender Systems

The web is now an integral part of numerous applications in which a user interacts with a company, government, employer, or an information provider. However, the potential of the web is hampered by the enormity of the content available and the diverse expectations of its user base. These challenges, in turn, have driven the increasing need to more intelligent, personalized, and adaptive web services or applications, such as e-commerce recommender systems. Businesses have come to realize the potential of these personalized and adaptive systems in order to increase sales and to retain customers. Likewise, Web users have come to rely on such systems to help them in more efficiently finding items of interest in large information spaces. The two AAAI 2007 workshops, the Fifth Workshop on Intelligent Techniques for Web Personalization (ITWP'07) and the Workshop on Recommender Systems in E-Commerce, joined forces to address a host of issues and challenges in the design, implementation, deployment, and evaluation of web personalization and recommendation solutions, both from a research as well as from a practical perspective. The topics covered by the two workshops included user or customer behavior modeling, preference elicitation, scalable and effective recommendation algorithms, personalized search and information access, data mining and web mining for personalization, trust and security in recommender systems, the use of semantics and ontologies in recommendation and web personalization, and the evaluation of recommender systems. (4)

Recently it was proved that Artificial Intelligence (AI) techniques can be very helpful if they are embedded in the e-commerce websites from many aspects, AI techniques are extensively used in the development of e-commerce systems also. The field of e-commerce can be classified as B2C (Business to Consumer) e-commerce and B2B (Business to Business) e-commerce, in terms of AI techniques involved in this field. (5), we can think of the RS as an adviser which helps the customer to decide what product or item to purchase, AI is used in advising the users on the items they want to examine or purchase through the Internet. This kind of advice is necessary because there are no real persons to advice the customers in the Internet. This advice is helpful in navigating a large range of product descriptions. (5), now many popular websites are using the RS to help their customers choose from different available product options, this is helpful for the customers which also leads to more profits for the companies that are using this technique.

1.4. Problems of Current Recommender Systems

The current implemented RSs suffer from different problems and that’s why this field is problem-rich, here we will give a brief description about the problems and challenges for the current RSs.

Challenges and Problems

- **Quality of Recommendations**: this problem arises when the RS provide products that the customer doesn’t like.
- **Sparsity:** this problem arises when the number of users is limited, then it is difficult to locate neighbors for each item, this results into weak recommendation, and this problem often happens in the collaborative methods.
- **Cold Start Problem:** An item cannot be recommended unless a user has rated it before. This problem applies to new items and also to obscure items and is particularly harmful to users with eclectic tastes. (3)
- **Unusual User Problem:** It is also known as the Gray Sheep problem. It refers to individuals with opinions that are “unusual”, meaning that they do not agree or disagree consistently with any group of people. Those individuals would not easily benefit from recommender systems since they would rarely, if ever, receive accurate predictions (3)
- **Limited Content Analysis**: this problem is related with the CB systems. Content-based techniques are limited by the features that are explicitly associated with the objects that these systems recommend. Therefore, in order to have a sufficient set of features, the content must either be in a form that can be parsed automatically by a computer (e.g., text) or the features should be assigned to items manually. (2)
- **Overspecialization:** When the system can only recommend items that score highly against a user’s profile, the user is limited to being recommended items that are similar to those already rated.

1.5. Objectives of The Project

The main objective of this thesis is to use intelligent fuzzy techniques to solve the following problems:

- **The information overload:** the product or service information available over the e-commerce website might be huge, which makes the customer unable to navigate them all, and the problem is after navigating some of these information the customer might be interested in one product only, so why don't we provide the customers with only the information they need and keeping them from this tedious task.
- **Customers lack experience of buying specific kinds of products:** sometimes even if the information are not huge the customers lack the experience to determine which product is best for them, for example if the customer wants to buy a laptop and he is not familiar with this kind of product, he will be confused which laptop to choose because of the lacking of technical experience, some websites (e.g. DELL) are giving hints or tips about the technical specifications for computer related products, but also this might be not good enough to help the customer choose.
- **If some product is recommended then why? What makes the customer convinced that this product is the best for him?** In some e-commerce websites the system recommends a product for the customer but the customer does not understand the real reasons behind this recommendation which makes the customer distrusts the recommendation.

So the purpose from this research is to minimize the burden for the customers to find the exact products they need by minimizing their effort and time. Also to provide acceptable reasons to why some product is recommended, we will assume that the customer is naive, has no experience in buying some kinds of products. If these problems are solved this will relieves both the customers and the vendors.

1.6. Methodology of The Project

In order to reach an efficient RS we will follow the following steps:

- Compare the most common recommendation algorithms, so that we can benefit from these existing algorithms and try to develop them in a way that improves the quality of these algorithms and eliminates all or some of their limitations
- The comparison of the existing algorithms or techniques will be done in depth and specially from a technical point of view
- We will show and investigate each recommendation process, data flows, applied mathematical formulas, and any other significant issue
- Also we will explain these methods by citing technical figures from different resources
- After this comparison is done we will start building on it, and show possible alternatives or enhancements to the compared methods
- We may not confine the research on the most common recommendation methods (e.g. Collaborative Filtering (CF)) but we may recover other related systems like CB and hybrid recommendation approaches, another important thing that we should take into account is the different types of e-commerce like B2B (Business to Business)
- The recommendation systems can be suitable for a specific e-commerce type but for another type it might not be, that's why this research is also trying to develop a recommendation which is the best for a specific e-commerce type, and this type as we will see is B2C.

After a thorough investigation in the work already done in the field of recommendation systems, a new algorithm will be presented, that will overcome this work (in specific domain), for example the proposed algorithm will be better in terms of performance or accuracy, also this algorithm will be applied using a suitable programming language only to explain the meaning and capabilities of this new algorithm.

1.7. Thesis Outline

This thesis consists of the following chapters:

- **Chapter 2: Literature Review** introduces the current generations of the recommendation systems, and the algorithms used in these systems, some detailed examples are also given to illustrate the algorithms' process

- **Chapter 3: System Analysis and Design** introduces the proposed recommendation system, with step by step example to illustrate the recommendation process

- **Chapter 4: System Implementation** introduces the technical issues of the system with screen shots that illustrates how to use the system

- **Chapter 5: Evaluation and lessons learnt** introduces an evaluation and discussion of the proposed system and future work to improve it

Chapter 2 Literature Review

2.1. Introduction

Before we propose our own method, it is helpful to overview the field of the RSs, and describe the current generations of recommendation methods, also it would be helpful to describe the limitations of RSs and discuss possible improvements.

Recommender systems have become an important research area since the appearance of the first papers on collaborative filtering in the mid-1990s there has been much work done both in the industry and academia on developing new approaches to recommender systems over the last decade. The interest in this area still remains high because it constitutes a problem-rich research area and because of the abundance of practical applications that help users to deal with information overload and provide personalized recommendations, content, and services to them. Examples of such applications include recommending books, CDs, and other products at Amazon.com, movies by MovieLens, and news at VERSIFI Technologies (formerly AdaptiveInfo.com). Moreover, some of the vendors have incorporated recommendation capabilities into their commerce servers. (2).

Now we can give a description about the state-of-the-art in the most common recommendation systems, which are as mentioned above (CB, Collaborative, and Hybrid)

2.2. Algorithms Used in Recommender Systems

2.2.1. Content-Based Methods

In content-based recommendation methods, the utility[1] *u(c,s)* of item *s* for user *c* is estimated based on the utilities *u(c,s_i)* assigned by user *c* to items $s_i \in S$ that are "similar" to item *s*. For example, in a movie recommendation application, in order to recommend movies to user *c*, the content-based recommender system tries to understand the commonalities among the movies user *c* has rated highly in the past (specific actors, directors, genres, subject matter, etc.). Then, only the movies that have a high degree of similarity to whatever the user's preferences are would be recommended. The content-based approach to recommendation has its roots in information retrieval, and information filtering research. Because of the significant and early advancements made by the information retrieval and filtering communities and because of the importance of several text-based applications, many current content-based systems focus on recommending items containing textual information, such as documents, Web sites (URLs), and Usenet news messages. The improvement over the traditional information retrieval approaches comes from the use of user profiles that contain information about users' tastes, preferences, and needs. The profiling information can be elicited from users explicitly, e.g., through questionnaires, or implicitly—learned from their transactional behavior over time. (2)

More formally, let *Content(s)* be an item profile, i.e., a set of attributes characterizing item *s*. It is usually computed by extracting a set of features from item *s* (its content) and is used to determine the appropriateness of the item for recommendation purposes. Since, as mentioned earlier, CB systems are designed mostly to recommend text-based items, the content in these systems is usually described with keywords. For example the Syskill & Webert system represents

[1] Utility is the usefulness of item *s* to user *c*

documents with the 128 most informative words. The “importance” (or “informativeness”) of word k_j in document d_j is determined with some weighting measure w_{ij} that can be defined in several different ways, Syskill & Webert is intended to be used to find unseen pages that are relevant to the user’s interests. To evaluate the effectiveness of the learning algorithms, it is necessary to run experiments to see if Syskill & Webert’s prediction agrees with the user’s preferences. Therefore we use a subset of the rated pages for training the algorithm and evaluate the effectiveness on the remaining rated pages. For an individual trial of an experiment, we randomly selected *n* pages to use as a training set, and reserved the remainder of the data as a test set. From the training set, we found the 128 most informative features[2], and then recoded the training set as feature vectors to be used by the learning algorithm. The learning algorithm created a representation for the user preferences. Next, the test data (i.e., all data not used as training data) was converted to feature vectors using the features found informative on the training set. Finally, the learned user preferences were used to determine whether pages in the test set would interest the user. (6)

One of the best-known measures for specifying keyword weights in Information Retrieval is the *term frequency/inverse document frequency (TF-IDF)* measure that is defined as follows: Assume that N is the total number of documents that can be recommended to users and that keyword k_j appears in n_i of them. Moreover, assume that $f_{i,j}$ is the number of times keyword k_i appears in document *dj*. Then, $TF_{i,j}$, the TF (or normalized frequency) of keyword k_i in document d_j, is defined as

$$TF_{i,j} = \frac{f_{i,j}}{\max_z f_{z,j}} \tag{2.1}$$

Where the maximum is computed over the frequencies $f_{z,j}$ of all keywords k_z that appear in the document d_j. However, keywords that appear in many documents are not useful in distinguishing between a relevant document and a nonrelevant one.

[2] Informative features means informative words

Therefore, the measure of inverse document frequency *(IDF_i)* is often used in combination with simple term frequency $TF_{i,j}$. The inverse document frequency for keyword k_i is usually defined as

$$IDF_i = log\frac{N}{n_i}. \tag{2.2}$$

Then, the *TF-IDF* weight for keyword k_i in document d_j is defined as

$$w_{i,j} = TF_{i,j} \times IDF_i \tag{2.3}$$

and the content of document d_j is defined as *Content(d_j) = (w_{1j},……………..,w_{kj}).*

Example: let's assume that we have 6 documents, Doc A through Doc F, with term-occurrences as follows:

Doc A	care, cat, mug
Doc B	care, care, care, cat, cat, cat, mug, mug, mug
Doc C	cat, cat, cat, cat, cat, cat, cat, cat, cat
Doc D	care, cat, dog, dog, dog, dog, dog, dog, mug
Doc E	care, cat, dog
Doc F	care

TF weights

From this initial specification, we can make a number of observations:

1. The **length** of a document, $l_{\mathbf{d}_i}$, is the total number of term-occurrences in it. The length of Doc A is 3, the length of Doc B is 9, and so on. In other words,

   $l_{\text{docA}} = 3$

   $l_{\text{docB}} = 9$

   $l_{\text{docC}} = 9$

   $l_{\text{docD}} = 9$

   $l_{\text{docE}} = 3$

   $l_{\text{docF}} = 1$

2. The total number of **term-occurrences** in the collection, $f_{\mathbf{D}}$, is the sum of the document lengths: $f_{\mathbf{D}} = \sum_{i=1}^{N} l_{\mathbf{d}_i}$. In other words, for this collection,

   $f_{\mathbf{D}} = 34$

3. The total number of term-types[3] in the collection is 4: these term-types, in alphabetical order are "care", "cat", "dog" and "mug". Each document is made up of a different number of occurrences of each of these term-types. Doc A, for instance, is made up of 1 occurrence of the term-type "care", 1 occurrence of the term-type "cat", 0 occurrences of the term-type "dog", and 1 occurrences of the term-type "mug". Doc B, on the other hand, is made up of 3 occurrences of the term-type "care", 3 occurrences of the term-type "cat", 0 occurrences of the term-type "dog",

[3] Here we mean by term-types term keywords

and 3 occurrences of the term-type “mug”. From now on in this example, for brevity, we’ll use the word “term” to mean “term-type”, and “occurrence” to mean “term-occurrence”.

4. We can present this information in the form of a **vector** for each document. Each document vector consists of an ordered list of values, each value indicating the number of occurrences of a particular term. So, for Doc A, the vector is <1, 1, 0, 1>, and for Doc B, the vector is <3, 3, 0, 3>. Note that the order of terms represented by the values is the same in each vector: this is essential if the vectors are to be **compared**, either with each other or with query vectors. We can write:

Doc A	= <1, 1, 0, 1>
Doc B	= <3, 3, 0, 3>
Doc C	= <0, 9, 0, 0>
Doc D	= <1, 1, 6, 1>
Doc E	= <1, 1, 1, 0>
Doc F	= <1, 0, 0, 0>

5. The term-frequency matrix, where the rows represent documents, the columns represent terms, and the individual values represent individual *normalized* term frequencies is shown below, the individual values here are calculated according to equation (2.1):

	care	cat	dog	mug
Doc A	1/1	1/1	0/1	1/1
Doc B	3/3	3/3	0/3	3/3
Doc C	0/9	9/9	0/9	0/9
Doc D	1/6	1/6	6/6	1/6
Doc E	1/1	1/1	1/1	0/1
Doc F	1/1	0/1	0/1	0/1

Table 2.1 Normalized term-frequency matrix

IDF weights

The next thing we can do is calculate the *IDF* weights for each term. *IDF* stands for **inverse document frequency**, but can also be conceptualized as a **within-collection frequency** weight, to contrast with *TF* which is a within-document frequency weight. The *IDF* weight for a particular term does not vary from document to document, whereas the *TF* weight for a particular term may well be very different for different documents (as we saw earlier). The equation we need to use to calculate the *IDF* weight for a term k_i is equation (2.2). So, to calculate a value for IDF, we need firstly to divide N by n_i, then take the logarithm of the result. The reason we take the logarithm of N/n_i, rather than just using N/n_i on its own, is so that we don't get such high values for IDF whenever we're in a situation where N is very large and n is relatively small (which is often the case). Some methods use a formula for *IDF* that takes a logarithm to base 2, rather than to base 10; other methods use a formula for *IDF* that adds 1 to each final value (this is to ensure that you don't end up with values of 0 for terms that actually appear in every document in the collection, since $\log_{10}1 = 0$). It really doesn't make much difference which formula you use.

So, the *IDF* weights for each term in the collection can be calculated and expressed in the form of a single **inverse document-frequency vector** as follows:

$IDF_{i,j} = <\log_{10}(6/5), \log_{10}(6/5), \log_{10}(6/2), \log_{10}(6/3)>$

$= <0.079, 0.079, 0.477, 0.301>$

Combined W weights

The third step is to calculate combined *W* weights, i.e., ***TF.IDF* weights** for each term in each document. What we need to do, is this. For each term in each document, multiply the *TF* value for that term by the corresponding *IDF* value for that term. We end up with a matrix of values again, each row representing a document and each column representing a term, but this time the values aren't *TF* weights, they're *TF.IDF* weights. So, the *W* (i.e., *TF.IDF*) vectors look like this:

$W_{docA} = <1 \times 0.079, 1 \times 0.079, 0 \times 0.477, 1 \times 0.301>$

$= <0.079, 0.079, 0.00, 0.301>$

$W_{docB} = <3 \times 0.079, 3 \times 0.079, 0 \times 0.477, 3 \times 0.301>$

$= <0.237, 0.237, 0.00, 0.903>$

$W_{docC} = <0 \times 0.079, 9 \times 0.079, 0 \times 0.477, 0 \times 0.301>$

$= <0.00, 0.711, 0.00, 0.00>$

$W_{docD} = <1 \times 0.079, 1 \times 0.079, 6 \times 0.477, 1 \times 0.301>$

$= <0.079, 0.079,2.862, 0.301>$

$W_{docE} = <1 \times 0.079, 1 \times 0.079, 1 \times 0.477, 0 \times 0.301>$

$= <0.079, 0.079, 0.477, 0.00>$

W_{docF} = <1 × 0.079, 0 × 0.079, 0 × 0.477, 0 × 0.301>

= <0.079, 0.00, 0.00, 0.00>

Now so far we have the weights for the keyword for every document[4] which means the importance for these keywords to the user, these weights will be used to find the similarity between what the user preferred in the past and the new items profile, there are many techniques to help us determine how similar two vectors are, but here we will consider the most-commonly used equation in this category which is the cosine similarity measure (7):

$$u(c,s) = \cos(\vec{w}_c, \vec{w}_s) = \frac{\vec{w}_c . \vec{w}_s}{\|\vec{w}_c\|_2 \times \|\vec{w}_s\|_2} \tag{2.4}$$

$$= \frac{\sum_{i=1}^{K} w_{i,c} w_{i,s}}{\sqrt{\sum_{i=1}^{K} w_{i,c}^2} \sqrt{\sum_{i=1}^{K} w_{i,s}^2}}$$

The way we use the cosine similarity is like this.

In the above formula we want to find the similarity between the profile of user *c* (items he liked in the past) and the profile of a list of items *s*

1. We will compare the query (item profile *s*) and the document (user profile *c*). (Remember that we have to repeat the process, and calculate a new value of *u(c,s),* for every user profile *(c)*/item profile *(s)* pair. Suppose, for example, we want to compare Query 1 and Doc A.

   Query 1 cat

(For simplicity we take only one term in Query 1)

[4] We can think of a document here as a user profile

2. We take the W vector for the query and the W vector (i.e., the TF.IDF vector) for the document.

   The W vector for Query 1 looks like this:

   $W_{\text{query 1}}$ = <0, 1, 0, 0>

   vector values based on computing TF weights (step 4). And the *W* vector for Doc A looks like this:

   W_{docA} = <0.079, 0.079, 0.00, 0.301>

3. We multiply together the corresponding W values for each term. In the example, you end up with a vector that looks like this:

   $W_{\text{query 1}} \times W_{\text{docA}}$ = <0 × 0.079, 1 × 0.079, 0 × 0.00, 0 × 0.301>

   = <0.00, 0.079, 0.00, 0.00>

4. Then we add together (i.e., sum) the values in that vector. The result we get is the value of the top half of the cosine formula: this is the so-called inner product or dot product of the two original vectors. In the example,

   $\sum_{i=1}^{K} w_{i,c} w_{i,s}$ = 0.00 + 0.079 + 0.00 + 0.00

   = 0.079

5. Moving now to the bottom half of the formula, we first need to calculate the squares of the W values in the query vector. In the example, you get a vector that looks like this:

   $w_{i,s}^2$ = <0 × 0, 1 × 1, 0 × 0, 0 × 0>

   = <0.00, 1.00, 0.00, 0.00>

6. Then we need to sum the values in that vector. In the example,

$$\sum_{i=1}^{K} w_{i,s}^{2} = 0.00 + 1.00 + 0.00 + 0.00$$

$$= 1.00$$

7. Similarly, you need to calculate the squares of the W values in the document vector. In the example, you get a vector that looks like this:

$$w_{i,c}^{2} = <0.079 \times 0.079, 0.079 \times 0.079, 0.00 \times 0.00, 0.301 \times 0.301>$$

$$= <0.006241, 0.006241, 0.00, 0.090601>$$

8. Then you need to sum the values in that vector. In the example,

$$\sum_{i=1}^{K} w_{i,c}^{2} = 0.006241 + 0.006241 + 0.00 + 0.090601$$

$$= 0.103083$$

9. Next, we take the square root of the results of steps (6) and (8) together. In the example,

$$\sqrt{\sum_{i=1}^{K} w_{i,s}^{2}} = 1.00$$

$$\sqrt{\sum_{i=1}^{K} w_{i,c}^{2}} = 0.32106$$

10. Now we multiply the square roots of the result of step (9). This is the result of the bottom half of the formula. In the example,

$$\sqrt{\sum_{i=1}^{K} w_{i,c}^{2}}\ \sqrt{\sum_{i=1}^{K} w_{i,s}^{2}} = 0.32106 \times 1.00$$

$$= 0.32106$$

11. Finally, you need to divide the result of step (4) by the result of step (10). This is the value of the cosine similarity! In the example,

$$\frac{\sum_{i=1}^{K} w_{i,c} w_{i,s}}{\sqrt{\sum_{i=1}^{K} w_{i,c}^2}\sqrt{\sum_{i=1}^{K} w_{i,s}^2}} = 0.079/0.32106$$

$$= 0.246$$

So, after all that, we can say that the degree of similarity between Query 1 and Document A is 0.246, on a scale of 0 to 1, where 1 represents complete similarity. We just gave this example to clarify how we can recommend new items to user according to the similarity among these items and the items the user preferred in the past.

The cosine similarity technique is not the only one used in the CB recommendation, other techniques have been used, for example Bayesian classifiers (8), learning user profiles (6), machine learning techniques, decision trees and artificial neural networks.

CB Limitations

CB recommendation systems need to be improved, below we describe the limitations of these systems.

- Either the items must be of some machine parsable form (e.g. text), or attributes must have been assigned to the items by hand. With current technology, media such as sound, photographs, art, video or physical items cannot be analyzed automatically for relevant attribute information. Often it is not practical or possible to assign attributes by hand due to limitations of resources.
- Content-based filtering techniques have no inherent method for generating serendipitous finds. The system recommends more of what the user already has seen before (and indicated liking). In practice, additional hacks are often added to introduce some element of serendipity.
- Content-based filtering methods cannot filter items based on quality, style or point-of-view. For example, they cannot distinguish between a well written and a badly written article if the two articles happen to use the same terms. (9)

- Overspecialization: the problem with overspecialization is not only that the content-based systems cannot recommend items that are different from anything the user has seen before. In certain cases, items should not be recommended if they are too similar to something the user has already seen, such as a different news article describing the same event. The diversity of recommendations is often a desirable feature in recommender systems. For example, it is not necessarily a good idea to recommend all movies by Woody Allen to a user who liked one of them (2)
- New user problem: The user has to rate a sufficient number of items before a CB RS can really understand the user's preferences and present the user with reliable recommendations. Therefore, a new user, having very few ratings, would not be able to get accurate recommendations.

2.2.2. Collaborative Methods

Unlike content-based recommendation methods, collaborative recommender systems (or collaborative filtering systems) try to predict the utility of items for a particular user based on the items previously rated by *other users*. More formally, the utility $u(c,s)$ of item s for user c is estimated based on the utilities $u(c,s_j)$ assigned to item s by those users $c_j \in C$ who are "similar" to user c. For example, in a movie recommendation application, in order to recommend movies to user c, the collaborative recommender system tries to find the "peers" of user c, i.e., other users that have similar tastes in movies (rate the same movies similarly). Then, only the movies that are most liked by the "peers" of user c would be recommended. (2), currently there are many websites which are using the collaborative methods, GroupLens (10), Video Recommender (11), and Ringo (9) were the first systems to use collaborative filtering algorithms to automate prediction. Other examples of collaborative recommender systems include the book recommendation system from Amazon.com, the PHOAKS system that helps people find relevant information on the WWW, and the Jester system that recommends jokes. (2)

Algorithms for collaborative recommendations can be grouped into two general classes (12): *memory-based (or Heuristic-Based (HB)) and Model-Based (MB)*.

HB algorithms

Memory-based algorithms essentially are heuristics that make rating predictions based on the entire collection of previously rated items by the users. That is, the value of the unknown rating $r_{c,s}$ for user *c* and item *s* is usually computed as an aggregate of the ratings of some other (usually, the *N* most similar) users for the same item *s*. (2)

There are several approaches to apply the HB method but below in the example we will use the most common used one.

Example:

Let's assume that we have 4 users and 9 items (books), as shown in table (2.2)

	B1	B2	B 3	B 4	B 5	B 6	B 7	B 8	B 9
Nameer	2	1	1	4		4	3	4	3
Ahmad	1		1	5	5	4	4	3	
Khaled	4	5	5	2		3	3	2	
Majed	5	4		1	3	2		2	

Table 2.2 Users-Items Matrix

Where B stands for book, we want to use the HB algorithm to predict the values for the unrated items (empty cells), we can do that by following these steps:

1. Determine similarities between users

2. Predict missing ratings

For step one we use Pearson Correlation equation as shown below

$$s_{a,b} = \frac{\sum_{i \in I_a \cap I_b} (v_{ai} - \bar{v}_a)(v_{bi} - \bar{v}_b)}{\sqrt{\sum_{i \in I_a \cap I_b} (v_{ai} - \bar{v}_a)^2 \sum_{i \in I_a \cap I_b} (v_{bi} - \bar{v}_b)^2}} \qquad (2.5)$$

Where	*a,b*	users
	i	item
	I_a	set of items rated by user *a*
	v_{ai}	rating of user *a* for item *i*
	$\bar{v}_a$	average rating of user *a*

For step two we use the weighted deviations from the average equation (similarities are weights)

$$p_{ai} = \bar{v}_a + \frac{\sum_{b \in U_i} s_{ab} (v_{bi} - \bar{v}_b)}{\sum_{b \in U_i} |s_{ab}|} \qquad (2.6)$$

Where U_i the set of users who have rated item *i*.

Here we use the weighted deviations from average instead of the weighted sum, because the ordinary weighted sum does not take into account that different users may use the rating scale differently (e.g. some users rank the best item as 3 out of 5 and some other users rank the item 5 out of 5), but the weighted deviation from average does.

Now the first step is to subtract each user's average from his rating as shown in the table below

	B1	B2	B 3	B 4	B 5	B 6	B 7	B 8	B 9
Nameer	-0.8	-1.8	-1.8	1.2		1.2	0.2	1.2	0.2
Ahmad	-2.3		-2.3	1.7	1.7	0.7	0.7	-0.3	
Khaled	0.6	1.6	1.6	-1.4		-0.4	-0.4	-1.4	
Majed	2.2	1.2		-1.8	0.2	-0.8		-0.8	

Table 2.3 Subtracted-Average User-Item Matrix

For example the subtracted average for user Nameer on B1 is computed as the following 2- ((2 + 1 + 1 + 4 + 4 + 3 + 4 + 3)/8) = -0.8, the other values are computed the same way.

Now we want to compute the similarity between users, for simplicity we will find the similarity only between two users (Khaled and Majed), according to Pearson correlation equation (2.5) and to table (2.3)

$$\frac{0.06 \cdot 2.2 + 1.6 \cdot 1.2 + (-1.4) \cdot (-1.8) + (-0.4) \cdot (-0.8) + (-1.4) \cdot (-0.8)}{\sqrt{(4.8 + 1.4 + 3.2 + 0.6 + 0.6)(0.4 + 2.6 + 2.0 + 0.2 + 2.0)}} \approx 0.83$$

As we can see here the similarity between users Khaled and Majed are based on the items they *both* rated, we can compute the other similarity between other users the same way, after doing these calculations we end up with table (2.4).

Similarity	Nameer	Ahmad	Khaled	Majed	B3
Nameer	1	0.78	-0.96	-0.85	-1.8
Ahmad	0.78	1	-0.74	-0.77	-2.3
Khaled	-0.96	-0.74	1	0.83	1.6
Majed	-0.85	-0.77	0.83	1	

Table 2.4 Users' Similarities

As mentioned above that our purpose is to find the missing rating, so to find (predict) the missing rating for user Majed on B3, we need to do another step, according to formula (2.6).

$$2.8 + \frac{(-0.85) \cdot (-1.8) + (-0.77) \cdot (-2.3) + 0.83 \cdot 1.6}{0.85 + 0.77 + 0.83} \approx 4.7$$

This means that the predicted rating (based on the scale from 1 to 5) for user Majed on B1 item is 4.7, which means Majed most likely well find B1 (book1) interesting.

Another approach to find the similarities between two users based on the items they both rated is the cosine-based which is the same as the cosine similarity used in the CB method, however in the CB recommendation systems cosine-similarity is used to measure the similarity between vectors of TF-IDF weights as mentioned earlier, whereas, in collaborative systems, it measures the similarity between vectors of the actual user-specified ratings. There are many extensions to the HB algorithms for example according to (12) a number of modifications to the standard HB algorithms can improve performance, some of these extensions

- **Default Voting**: Default voting is an extension to the correlation algorithm. It arose out of the observation that when there are relatively few votes, for either the active user or the matching user, the correlation algorithm will not do well because it uses only votes in the intersection of the items both individuals have voted on. If we assume some default value as a vote for titles for which we do not have explicit votes, then we can form the match over the union of voted items, where the default vote value is inserted into the formula for the appropriate unobserved items.
- **Inverse User Frequency**: In applications of vector similarity in information retrieval, word frequencies are typically modified by *the inverse document frequency*. The idea is to reduce weights for commonly occurring words, capturing the intuition that they are not as useful in identifying the topic of a document, while words that occur less frequently are more indicative of topic. We can apply an analogous transformation to votes in a CF database, which we term *inverse user frequency*. The idea is that universally liked items are not as useful in capturing similarity as less common items.

Also some algorithms use the HB methods to compute similarities between *items* instead of *users*, an example of this approach is (13)

MB algorithms

MB CF algorithms provide item recommendation by first developing a model of user ratings. Algorithms in this category take a probabilistic approach and envision the CF 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 (12) formulates a probabilistic model for CF problem. The clustering model treats CF as a classification problem (14) (12) 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 (15)

For example we will take a look at (12) to clarify the idea behind the MB methods:
From a probabilistic perspective, the CF task can be viewed as calculating the expected value of a vote, given what we know about the user. For the active user, we wish to predict votes on as-yet unobserved items. If we assume that the votes are integer valued with a range for 0 to *m* we have:

$$p_{a,j} = E(v_{a,j}) = \sum_{i=0}^{m} \Pr\left(v_{a,j} = i \middle| v_{a,k,} k \in I_a\right) i \qquad (2.7)$$

Where the probability expression is the probability that the active user will have a particular vote value for item *j* given the previously observed votes. Two alternative probabilistic models for CF will be described, *cluster models* and *Bayesian networks*.

Cluster Models

One plausible probabilistic model for CF is a Bayesian classifier where the probabilities of votes are conditionally independent given membership in an unobserved class variable *C* taking on some relatively small number of discrete values. The idea is that there are certain groups or types of users capturing a common set of preferences and tastes. Given the class, the preferences regarding the various items (expressed as votes) are independent. The probability model relating joint probability of class and votes to a tractable set of conditional and marginal distributions is the standard “naive" Bayes formulation:

$$\mathrm{pr}\left(C = c, v_{1,} \ldots\ldots, v_n\right) = \mathrm{p}r\ (C = c) \prod_{i=1}^{n} \mathrm{pr}\ (v_i | C = c)$$

The left-hand side of this expression is the probability of observing an individual of a particular class and a complete set of vote values. It is straightforward to calculate the needed probability expressions for equation (2.7) within this framework. This model is also known as a multinomial mixture model. The parameters of the model, the probabilities of class membership Pr($C = c$), and the conditional probabilities of votes given class Pr ($v_i \mid C = c$) are estimated from a training set of user votes, the user database.

Since we never observe the class variables in the database of users, we must employ methods that can learn parameters for models with hidden variables, Such as EM algorithm. We can choose the number of classes by selecting the model structure that yields the largest (approximate) marginal likelihood of the data. To approximate the marginal likelihood (16) can be used.

Bayesian Network Model

An alternative model formulation for probabilistic CF is a Bayesian network with a node corresponding to each item in the domain. The states of each node correspond to the possible vote values for each item. We also include a state corresponding to “no vote” for those domains where there is no natural interpretation for missing data.

We then apply an algorithm for learning Bayesian networks to the training data, where missing votes in the training data are indicated by the “no vote” value. The learning algorithm searches over various model structures in terms of dependencies for each item. In the resulting network, each item will have a set of parent items that are the best predictors of its votes. Each conditional probability table is represented by a decision tree encoding the conditional probabilities for that node.

There have been several other MB collaborative recommendation approaches proposed in the literature. A statistical model for CF was proposed in (17), and several different algorithms for estimating the model parameters were compared, including K-means clustering and Gibbs sampling. Other CF methods include a Bayesian model (18), a probabilistic relational model (19), a linear regression (13), and a maximum entropy model (20). More recently, a significant amount of research has been done in trying to model the recommendation process using more complex probabilistic models. Other probabilistic modeling techniques for RSs include probabilistic latent semantic analysis (21), (22) and a combination of multinomial mixture and aspect models using generative semantics of Latent Dirichlet Allocation (23). Similarly, (24) also use probabilistic latent semantic analysis to propose a flexible mixture model that allows modeling the classes of users and items explicitly with two sets of latent variables. Furthermore, (25) use a simple probabilistic model to demonstrate that CF is valuable with relatively little data on each user, and that, in certain restricted settings, simple CF algorithms are almost as effective as the best possible algorithms in terms of utility.

Collaborative Methods Limitations

According to (26)

- **New User Problem:** this problem is the same as in CB methods as mentioned above, To be able to make accurate predictions, the system must first learn the user’s preferences from the ratings that the user makes. If the system does not show quick progress, a user may lose patience and stop using the system.
- **New Item Problem (Recurring Startup Problem):** New items are added regularly to RSs. A system that relies solely on users’ preferences to make predictions would not be able to make accurate predictions on these items. This problem is particularly severe with systems that receive new items

regularly, such as an online news article recommendation system. Therefore, until the new item is rated by a substantial number of users, the RS would not be able to recommend it.

- **Sparsity:** In any RS, the number of ratings already obtained is usually very small compared to the number of ratings that need to be predicted. Effective prediction of ratings from a small number of examples is important. Also, the success of the collaborative RS depends on the availability of a critical mass of users. For example, in the movie RS, there may be many movies that have been rated by only few people and these movies would be recommended very rarely, even if those few users gave high ratings to them. Also, for the user whose tastes are unusual compared to the rest of the population, there will not be any other users who are particularly similar, leading to poor recommendations
- **Scaling Problem:** RSs are normally implemented as a centralized web site and may be used by a very large number of users. Predictions need to be made in real time and many predictions may potentially be requested at the same time. The computational complexity of the algorithms needs to scale well with the number of users and items in the system.

2.2.3. Hybrid Methods

Several recommendation systems use a hybrid approach by combining collaborative and CB methods, which helps to avoid certain limitations of CB and collaborative systems. Different ways to combine collaborative and CB methods into a hybrid RS can be classified as follows:

1. Implementing collaborative and content-based methods separately and combining their predictions,

2. Incorporating some content-based characteristics into a collaborative approach,
3. Incorporating some collaborative characteristics into a content-based approach
4. Constructing a general unifying model that incorporates both content-based and collaborative characteristics. (2)

Also according to (27), Hybrid RSs combine two or more recommendation techniques to gain better performance with fewer of the drawbacks of any individual one. Most commonly, CF is combined with some other technique in an attempt to avoid the ramp-up problem. Table (2.5), below shows some of the combination methods that have been employed.

- **Weighted**: A weighted hybrid recommender is one in which the score of a recommended item is computed from the results of all of the available recommendation techniques present in the system. For example, the simplest combined hybrid would be a linear combination of recommendation scores. It initially gives collaborative and CB recommenders equal weight, but gradually adjusts the weighting as predictions about user ratings are confirmed or disconfirmed. The benefit of a weighted hybrid is that all of the system's capabilities are brought to bear on the recommendation process in a straightforward way and it is easy to perform post-hoc credit assignment and adjust the hybrid accordingly. However, the implicit assumption in this technique is that the relative value of the different techniques is more or less uniform across the space of possible items. From the discussion above, we know that this is not always so, a collaborative recommender will be weaker for those items with a small number of raters.

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

Table 2.5 Hybridization Methods

- **Switching:** A switching hybrid builds in item-level sensitivity to the hybridization strategy: the system uses some criterion to switch between recommendation techniques. The DailyLearner system uses a content/collaborative hybrid in which a CB recommendation method is employed first. If the CB system cannot make a recommendation with sufficient confidence, then a collaborative recommendation is attempted. This switching hybrid does not completely avoid the ramp-up problem, since both the collaborative and the CB systems have the “new user” problem. However, DailyLearner’s CB technique is nearest-neighbor, which does not require a large number of examples for accurate classification. What the collaborative technique provides in a switching hybrid is the ability to cross genres, to come up with recommendations that are not close in a semantic way to the items previous rated highly, but are still relevant. For example, in the case of DailyLearner, a user who is interested in the Microsoft anti-trust trial might also be interested in the AOL/Time Warner merger. Content matching would not be likely to recommend the merger stories, but other users with an interest in corporate power in the high-tech industry may be rating both sets of stories highly, enabling the system to make the recommendation collaboratively. DailyLearner's hybrid has a "fallback" character – the short-term model is always used first and the other technique only comes into play when that technique fails. Switching hybrids introduce additional complexity into the recommendation process since the switching criteria must be determined, and this introduces another level of parameterization. However, the benefit is that the system can be sensitive to the strengths and weaknesses of its constituent recommenders.

- **Mixed:** Where it is practical to make large number of recommendations simultaneously, it may be possible to use a "mixed" hybrid, where recommendations from more than one technique are presented together. The PTV system (28) uses this approach to assemble a recommended program of television viewing. It uses CB techniques based on textual descriptions of TV shows and collaborative information about the preferences of other users. Recommendations from the two techniques are combined together in the final suggested program. The mixed hybrid avoids the "new item" start-up problem: the CB component can be relied on to recommend new shows on the basis of their descriptions even if they have not been rated by anyone. It does not get around the "new user" start-up problem, since both the content and collaborative methods need some data about user preferences to get off the ground, but if such a system is integrated into a digital television, it can track what shows are watched (and for how long) and build its profiles accordingly. Like the fallback hybrid, this technique has the desirable "niche-finding" property in that it can bring in new items that a strict focus on content would eliminate. The PTV case is somewhat unusual because it is using recommendation to assemble a composite entity, the viewing schedule. Because many recommendations are needed to fill out such a schedule, it can afford to use suggestions from as many sources as possible. Where conflicts occur, some type of arbitration between methods is required – in PTV, CB recommendation take precedence over collaborative responses. Other implementations of the mixed hybrid, ProfBuilder (29), present multiple recommendation sources side-by-side. Usually, recommendation requires ranking of items or selection of a single best recommendation, at which point some kind of combination technique must be employed.

- **Feature Combination:** Another way to achieve the content/collaborative merger is to treat collaborative information as simply additional feature data associated with each example and use CB techniques over this augmented data set. For example, (14) report on experiments in which the inductive rule learner Ripper was applied to the task of recommending movies using both user ratings and content features, and achieved significant improvements in precision over a purely collaborative approach. However, this benefit was only achieved by hand-filtering content features. The authors found that employing all of the available content features improved recall but not precision. The feature combination hybrid lets the system consider collaborative data without relying on it exclusively, so it reduces the sensitivity of the system to the number of users who have rated an item. Conversely, it lets the system have information about the inherent similarity of items that are otherwise opaque to a collaborative system.
- **Cascade:** Unlike the previous hybridization methods, the cascade hybrid involves a staged process. In this technique, one recommendation technique is employed first to produce a coarse ranking of candidates and a second technique refines the recommendation from among the candidate set. The restaurant recommender EntreeC (27), is a cascaded knowledge-based and collaborative recommender. Like Entree, it uses its knowledge of restaurants to make recommendations based on the user's stated interests. The recommendations are placed in buckets of equal preference, and the collaborative technique is employed to break ties, further ranking the suggestions in each bucket. Cascading allows the system to avoid employing the second, lower-priority, technique on items that are already well-differentiated by the first or that are

sufficiently poorly-rated that they will never be recommended. Because the cascade’s second step focuses only on those items for which additional discrimination is needed, it is more efficient than, for example, a weighted hybrid that applies all of its techniques to all items. In addition, the cascade is by its nature tolerant of noise in the operation of a low-priority technique, since ratings given by the high-priority recommender can only be refined, not overturned.

- **Feature Augmentation:** One technique is employed to produce a rating or classification of an item and that information is then incorporated into the processing of the next recommendation technique. For example, the Libra system (30) makes CB recommendations of books based on data found in Amazon.com, using a naïve Bayes text classifier. In the text data used by the system is included “related authors” and “related titles” information that Amazon generates using its internal collaborative systems. These features were found to make a significant contribution to the quality of recommendations. The GroupLens research team working with Usenet news filtering also employed feature augmentation (31). They implemented a set of knowledge-based “filterbots” using specific criteria, such as the number of spelling errors and the size of included messages. These bots contributed ratings to the database of ratings used by the collaborative part of the system, acting as artificial users. With fairly simple agent implementations, they were able to improve email filtering. Augmentation is attractive because it offers a way to improve the performance of a core system, like the NetPerceptions’ GroupLens Recommendation Engine or a naive Bayes text classifier, without modifying it. Additional functionality is added by intermediaries who can use other techniques to augment the data itself. Note that this is different from feature combination in which raw data from different sources is combined. While both the cascade and augmentation techniques sequence two recommenders, with the first recommender having an influence over the second, they are fundamentally quite different. In an augmentation hybrid, the features used by the second recommender include the output of the first one, such as the ratings contributed by GroupLens' filterbots. In a cascaded hybrid, the second recommender does not use any output from the first recommender in producing its rankings, but the results of the two recommenders are combined in a prioritized manner.

- **Meta-level:** Another way that two recommendation techniques can be combined is by using the model generated by one as the input for another. This differs from feature augmentation: in an augmentation hybrid, we use a learned model to generate features for input to a second algorithm; in a meta-level hybrid, the entire model becomes the input. The first meta-level hybrid was the web filtering system Fab (32). In Fab, user-specific selection agents perform CB filtering using Rocchio's method to maintain a term vector model that describes the user's area of interest. Collection agents, which garner new pages from the web, use the models from all users in their gathering operations. So, documents are first collected on the basis of their interest to the community as a whole and then distributed to particular users. In addition to the way that user models were shared, Fab was also performing a cascade of collaborative collection and CB recommendation, although the collaborative step only created a pool of documents and its ranking information was not used by the selection component. A meta-level hybrid that focuses exclusively on recommendation is described by (33) as "collaboration via content". A CB model is built by Winnow (34) for each user describing the features that predict restaurants the user likes. These models, essentially vectors of terms and weights, can then be compared across users to make predictions. More recently, (35) have used a two-stage Bayesian mixed-effects scheme: a CB naive Bayes classifier is built for each user and then the parameters of the classifiers are linked across different users using regression. LaboUr (36) uses instance-based learning to create CB user profiles which are then compared in a collaborative manner. The benefit of the meta-level method, especially for the content/collaborative hybrid is that the learned model is a compressed representation of a user’s interest, and a collaborative mechanism that follows can operate on this information-dense representation more easily than on raw rating data.

2.3. Memory-based VS. Model-based Filtering Algorithms

According to (37) *MB* CF algorithms usually take a probabilistic approach, envisioning the recommendation process as the computation of the expected value of a user rating, when his past ratings on other items are given. They achieve that by developing a model of user ratings, sometimes referred to as the user profile. The development of such a model is primarily based on the original user-item matrix, *R*, but once the model is trained, matrix *R* is no longer required for recommendation generation. The advantage of this approach is that, because of the much more compact user model, it avoids the constant need of a possibly huge user-item matrix to make recommendations. This would certainly lead to systems with lower memory requirements and rapid recommendation generation. Nevertheless, the model building step, which is equivalent to the neighborhood formation step in plain CF algorithms, is executed off-line since the user model is expensive to build or update. As a result, it is recomputed only after sufficient changes have occurred in the user-item matrix, for example, once per week. *MB* filtering algorithms include CF as a Machine Learning Classification Problem, Personality Diagnosis and Bayesian Network Model. On the other hand, *HB* CF algorithms are basing their predictions on the original (or probably reduced, through statistical methods like SVD/LSI) user-item matrix, *R*, which they keep in memory throughout the procedure. This results in greater memory requirements and probably not so fast recommendations. Yet, the predictions are always in agreement with the most current user ratings. There is no need for off-line updating, which would probably have caused a performance bottleneck. *HB* filtering algorithms include the basic CF algorithm, Item-based CF and the Algorithm using SVD/LSI for Prediction Generation.

2.4. Summary

As it was mentioned in this chapter, there has been much research done on recommendation technologies over the past several years that have used a broad range of statistical, machine learning, information retrieval, and other techniques that have significantly advanced the state-of-the art in comparison to early RSs that utilized collaborative and CB heuristics. As was discussed above, RSs can be categorized as being 1) CB, collaborative, or hybrid, based on the recommendation approach used, and 2) HB or MB, based on the types of recommendation techniques used for the rating estimation. Table 2.6 (2) classifies the RSs research and approaches:

Recommendation Approach	**Recommendation Techniques**	
	Heuristic-based	Model-based
Content-based	Commonly used techniques: • TF-IDF (information retrieval) • Clustering Representative research examples: • Lang 1995 • Balabanovic & Shoham 1997 • Pazzani & Billsus 1997	Commonly used techniques: • Bayesian classifiers • Clustering • Decision trees • Artificial neural networks Representative research examples: • Pazzani & Billsus 1997 • Mooney et al. 1998 • Mooney & Roy 1999 • Billsus & Pazzani 1999,2000 • Zhang et al. 2002
Collaborative	Commonly used techniques: • Nearest neighbor (cosine, correlation) • Clustering • Graph theory Representative research examples: • Resnick et al.1994 • Hill et al. 1995 • Shardanand & Maes 1995 • Breese et al. 1998 • Nakamura & Abe 1998 • Aggarwal et al. 1999 • Delgado & Ishii 1999 • Pennock & Horwitz 1999 • Sarwar et al. 2001	Commonly used techniques: • Bayesian networks • Clustering • Artificial neural networks • Linear regression • Probabilistic models • Representative research examples: • Pazzani & Billsus 1998 • Breeze et al. 1998 • Ungar & Foster 1998 • Chien & George 1999 • Getoor & Sahami 1999 • Pennock & Horwitz 1999 • Goldberg et al. 2001 • Kumar et al. 2001 • Pavlov & Pennock 2002 • Shani et al. 2002 • Yu et al. 2002,2004 • Hofmann 2003,2004 • Marlin 2003 • Si & Jin 2003
Hybrid	Combining content-based and collaborative components using: • Linear combination of predicted ratings • Various voting schemes • Incorporating one component as a part of the heuristic for the other Representative research examples: • Balabanovic & Shoham 1997 • Claypool et al. 1999 • Good et al. 1999 • Pazzani 1999 • Billsus & Pazzani 2000 • Tran & Cohen 2000 • Melville et al. 2002	Combining content-based and collaborative components by: • Incorporating one component as a part of the model for the other • Building one unifying model Representative research examples: • Basu et al. 1998 • Condliff et al. 1999 • Soboroff & Nicholas 1999 • Ansari et al. 2000 • Popescul et al. 2001 • Schein et al. 2002

Table 2.6 Classification of RSs research

Chapter 3 System Analysis and Design

3.1. Introduction

In this chapter a new system for product selection and recommendation will be discussed in details. The proposed system is concerned with recommending laptops to customers. According to the user information that will be gathered and to the features of the laptops a list of recommendations will be provided for each customer.

The new proposed method will use the fuzzy set theory. First, the original user-item matrix that was mentioned in the previous chapters will be converted to a fuzzy matrix in a process called Fuzzificaiton. The next step will be to normalize the fuzzified matrix in order to have the same contribution from each user. After normalization, a new fuzzy-based similarity technique will be used to find the similarity among users according to their ratings. And the last step will be generating predictions for the missing ratings according to the results of the previous steps.

Below a detailed investigation will be shown about each step in this process.

3.2. Constructing A Fuzzy User-Item Matrix

In this stage of the process a user-item matrix will be constructed, the rows of this matrix will represent the users involved in the system and the columns represent the items from which a list of recommendations will be provided. The values of each row column combination represent the rating value, for example let’s assume we have the following user-item matrix

	I 1	I 2	I 3	I 4	I 5	I 6	I 7	I 8	I 9
U 1	0.8	0.2	0.5	0.2		0.3	0.4	0.6	0.9
U 2	0.3		0.9	0.7	0.6	0.7	0.4	0.5	
U 3	0.9	0.5	0.6	0.4		0.9	0.3	0.4	
U 4	0.2	0.7		0.8	0.4	0.8		0.5	
U 5	0.3	0.8		0.2		0.8	0.6		0.7

Table 3.1 Constructing User-Item Matrix

This matrix represents the rating of each user *U* to the available items *I*, in this case we have 5 users and 9 items (laptops for our system), for example *U*1 rated *I*1 by the value 0.8 and *U*2 rated *I*3 by the value 0.9. There are some missing values in this matrix, which means the corresponding user didn't rate that item(s), for example *U*1 didn't rate *I*5, and *U*5 didn't rate *I* 3, *I*5, and *I*8.

Another important thing in the above matrix is that the rating values are represented as fuzzy values, each row in the above matrix can be represented as a fuzzy set. (Fuzzy set definition) Let X be a nonempty set. A fuzzy set A in X is characterized by its membership function $\mu_A : X \to [0, 1]$ and $\mu_A(x)$ is interpreted as the degree of membership of element x in fuzzy set A for each $x \in X$. A is completely determined by the set of pair

$A = \{(x, \mu_A(x))|x \in X\}$.

If $X = \{x_1, . . . , x_n\}$ is a finite set and A is a fuzzy set in X then we often use the notation $A = \mu_1/x_1 + \cdot \cdot \cdot + \mu_n/x_n$ where the term μ_i/x_i, i = 1, . . . , n signifies that μ_i is the grade of membership of x_i in A and the plus sign connects the elements (does not mean summation), the closer the value of $\mu_A(x)$ is to 1 the more x belong to A. (38), now simply put we can think about these values in the above matrix as the degree of satisfaction (usually referred as the degree of membership), so if we take the combination of $U1$on $I1$ we can think of it as, the degree of satisfaction of item $I1$ to user $U1$ is 0.8, and the rest of the combinations are considered the same way, the closer the value is to 1 the more the customer is satisfied about the product.

There is an important question here how do we fill the fuzzy values in the user-item matrix, are we going to ask the user to fill in these fuzzy values? Does the user know anything about fuzzy? of course we will not assume that, instead first we should determine the scale of the rating for example 1 – 5, 1 is the minimum rating value and 5 is the maximum rating value, after that we convert these values into fuzzy values that can be processed using fuzzy rules, this process is known as Fuzzification. In order to accomplish that the following membership function is used

$$\mathrm{M}(\mathrm{ri,j}) = \frac{\mathrm{ri,j} - \min R}{\max R - \min R} \qquad (3.1)$$

where $r_{i,j}$ is rating of user i on item j, and R is the rating scale used in the dataset.

There is something worth noting in equation (3.1) which is flexibility, because any rating scale could be converted to fuzzy value without losing the meaning of the original number, as an example we will convert rating values with different scales to fuzzy values.

Example: assume we have three rating scales 1-5, 1-10, 20-30, the first number of each scale is the minimum value and the last number is the maximum value, also assume that user UA rated four items, each time with different rating scale, now for each rating scale we will convert the rating numbers to fuzzy values using equation (3.1), to make sure the numbers don't lose their meaning

	I 1	I 2	I 3	I 4
UA	1	2	5	4

1-5 rating scale

	I 1	I 2	I 3	I 4
UA	0	0.25	1.0	0.75

Fuzzified 1-5 rating scale

	I 1	I 2	I 3	I 4
UA	1	7	10	6

1-10 rating scale

	I 1	I 2	I 3	I 4
UA	0	0.67	1	0.56

Fuzzified 1-10 rating scale

	I 1	I 2	I 3	I 4
UA	23	20	29	30

20-30 rating scale

	I 1	I 2	I 3	I 4
UA	0.3	0	0.9	1

Fuzzified 20-30 rating scale

Now after fuzzifing the original rating scales, it’s obvious that the converted values still have the same meaning, for example, from each rating scale take the least scale and compare it to its corresponding value in the fuzzified scale, you will see that the corresponding value is also the least value in terms of fuzzy sets (the set of values between 0 and 1), the same thing is applied for the largest values in the scale and even to the in between values.

In the proposed system this technique will be used in order to be more flexible and to adjust the output according to the rating scale, so that any rating scale could be used without affecting the original values, now what we are going to do with these values will be explained in details in the next stages of the process.

3.3. Fuzzy Matrix Normalization

Continuing from the previous stage, the next step is normalization of fuzzified values, this step is normally taken to ensure that each row, in our case each user has the same contribution to the prediction calculations, to clarify this idea we will take table 3.1, normalize it then finding the predictions for the missing ratings, the normalization of fuzzy values is done by dividing each value by the sum of the entire row so that the normalized values add to 1.

	I 1	I 2	I 3	I 4	I 5	I 6	I 7	I 8	I 9
U1	0.21	0.05	0.13	0.05		0.08	0.10	0.15	0.23
U2	0.07		0.22	0.17	0.15	0.17	0.10	0.12	
U3	0.23	0.13	0.15	0.10		0.23	0.08	0.10	
U 4	0.06	0.21		0.24	0.12	0.24		0.15	
U 5	0.09	0.24		0.06		0.24	0.18		0.21

Table 3.2 Normalization of Table 3.1

3.4. Computing Users Similarities Based on Fuzzy Sets Similarity

Like in most CF algorithms it is necessary to find the similarity among users, the idea here is that if two users rated the same items in similar manner (with the same or similar rating values), both users will most likely have similar taste or preferences, so we can predict missing ratings for each user by finding the most similar users to him, and based on the ratings of the most similar users we provide predictions for the missing ratings.

The question is how to find the similarities among users, as it was mentioned above there were already proposed algorithms to solve this problem, like Pearson correlation and cosine similarity, but the proposed method will not use these algorithms, because they have many limitations:

(1) Very limited number of co-ratings under data sparsity.

(2) If the number of co-rated items is 1, COR cannot be calculated and COS results in 1 regardless of differences in individual ratings.

(3) If all the available ratings of a given user are flat, e.g. <1, 1, 1>, <3, 3,

3> or <4, 4, 4>, COR cannot be calculated between the user (and, hence, often regarded as 0) and another since the denominator part of the correlation formula becomes zero.

(4) If two vectors are on the same line, e.g. vectors <2, 2> and <3, 3>, COS results in 1 regardless of the difference between the two.

(5) Both COR and COS can be sometimes misleading, where very different users may appear to be very similar to each other by the similarity measures, and vice versa (39)

The similarity measure that will be used here is based on the similarity between two fuzzy sets, as it was mentioned above, we can consider the ratings for each user as a fuzzy set, and hence to find the similarity between two users we can use the fuzzy similarity measure as the following:

$$\sigma(\mu_{1,}\mu_2) = \frac{|\mu_{1\cap}\mu_2|}{|\mu_{1\cup}\mu_2|} \tag{3.2}$$

According to (3.2), the similarity between two fuzzy sets is given by the ratio of two quantities: the cardinality of the intersection of the fuzzy sets and the cardinality of the union of the fuzzy sets. The intersection of two fuzzy sets is defined by the minimum operator:

$(\mu_1 \cap \mu_2)$ (p) = min $\{\mu_1(p), \mu_2(p)\}$.

The union of two fuzzy sets is defined by the maximum operator:

$(\mu_1 \cup \mu_2)$ (p) = max $\{\mu_1(p), \mu_2(p)\}$ The cardinality of a fuzzy set (also called "σ-count") is computed by summing up all its membership values:

$|\mu|=\sum_{p=1}^{p} \mu(p)$. Hence, the similarity of two users is defined as:

$$s(u_{1,}u_2) = \frac{\sum_{p=1}^{p} \min\{\mu_{u_1,p}, \mu_{u_2,p}\}}{\sum_{p=1}^{p} \max\{\mu_{u_1,p}, \mu_{u_2,p}\}} \tag{3.3}$$

Where *u* denotes user and *p* denotes item that both users have rated.

This similarity measure is reflexive (i.e.(u,u)=1) and symmetric (i.e.(u,v)= S(v,u)). Unlike the angle cosine, it gives a much more acceptable evaluation of the similarity between two users.

Now let's take a simple example to see how equation (3.3) works

	I 1	I 2	I 3	I 4	I 5
U A	0.3		0.9	1.0	
U B	0.2		0.8	0.9	
U C	1.0		0.1	0.2	

$$\mathrm{S(UA, UB)} = \frac{0.2 + 0.8 + 0.9}{0.3 + 0.9 + 1.0} = 0.86$$

Assume the rating values in the above table are already fuzzified, it's obvious that user UA and user UB have rated the same items with similar values, so the similarity result between them is high (closer to 1) which is 0.86, on the other hand it's easy to notice that UA and UC have rated the same items with dissimilar values, now the similarity between them should be less, let's see

$$\mathrm{S(UA, UC)} = \frac{0.3 + 0.1 + 0.2}{1.0 + 0.9 + 1.0} = 0.21$$

As the result indicates the similarity between UA and UC is slight.

If we want to compute the similarity among users for table 3.2, we get the following users similarity matrix:

	U1	U 2	U 3	U 4	U 5
U1	1	0.54	0.64	0.37	0.5
U 2	0.54	1	0.63	0.75	0.59
U 3	0.64	0.63	1	0.58	0.60
U 4	0.37	0.75	0.58	1	0.70
U 5	0.5	0.59	0.60	0.70	1

Table 3.3 Users-Similarity Matrix

For sure the similarity between the user and himself is 1, the users-similarity matrix diagonal will always be 1, and as we can see this is a symmetric matrix. The most similar users in the above matrix are *U*2 and *U*4, on the contrary the most dissimilar users are *U*1 and *U*4.

3.5. Generating Predictions for Missing Ratings

Now in this stage, predictions for missing ratings in the users-item matrix should be generated and based on those predictions, recommendations will be generated. The prediction equation (3.4) that will be used to find the missing predictions is similar to equation (2.6) with slight differences. Instead of using the subtracted average, the normalized rating rb will be used. Also the similarity will always be positive, so there is no need to use the absolute value for the similarity.

$$p_{ai} = \bar{v}_a + \frac{\sum_{b \in U_i} s_{ab} \times rb}{\sum_{b \in U_i} s_{ab}} \tag{3.4}$$

Where *a,b* users, *i* item, $\bar{V}_a$ average rating of user a, U$_i$ set of users that have rated item *i*, *s* similarity.

According to equation (3.4), to find the missing rating for user $U1$ item $I5$ combination in table 3.2, we get the following result:

$$P_{1,5} = 0.49 + \frac{(0.54 \times 0.15) + (0.37 \times 0.12)}{0.54 + 0.37} = 0.63$$

The other missing ratings could be found the same way.

After explaining the main stages of the proposed method to recommend items for users, below is a complete example to make the process more clear and comprehensible.

Example: assume we have the below user-item matrix, we want to predict missing ratings for laptops for each user, after that recommend the top *N* predictions (laptops), also assume that the rating scale is 1-10.

	Laptop 1	Laptop 2	Laptop 3	Laptop 4	Laptop 5	Laptop 6	Laptop 7	Laptop 8	Laptop 9
User 1	1	5	3	5		8	10	1	2
User 2	7		2	1	1	3	10	6	
User 3	2	1	4	6		7	8	10	
User 4	6	2		4	2	9		3	
User 5	7	3		8		2	1	7	1

Table 3.4 Original User-Item Ratings with Scale 1-10

Step 1: according to equation (3.1) the corresponding fuzzified matrix for table 3.4 is given below:

	Laptop 1	Laptop 2	Laptop 3	Laptop 4	Laptop 5	Laptop 6	Laptop 7	Laptop 8	Laptop 9
User 1	0.0	0.44	0.22	0.44		0.78	1.0	0.0	0.11
User 2	0.67		0.11	0.0	0.0	0.22	1.0	0.56	
User 3	0.11	0.0	0.33	0.56		0.67	0.78	1.0	
User 4	0.56	0.11		0.33	0.11	0.89		0.22	
User 5	0.67	0.22		0.78		0.11	0.0	0.67	0.0

Table 3.5 Fuzzification of Table 3.4

Step 2: normalization of table 3.5. The corresponding normalized matrix given below:

	Laptop 1	Laptop 2	Laptop 3	Laptop 4	Laptop 5	Laptop 6	Laptop 7	Laptop 8	Laptop 9
User 1	0.0	0.15	0.07	0.15		0.26	0.33	0.0	0.04
User 2	0.26		0.04	0.0	0.0	0.09	0.39	0.22	
User 3	0.03	0.0	0.10	0.16		0.19	0.23	0.29	
User 4	0.25	0.05		0.15	0.05	0.40		0.10	
User 5	0.27	0.09		0.32		0.04	0.0	0.27	0.0

Table 3.6 Normalization of Table 3.5

Step 3: according to equation (3.3) the corresponding similarity matrix for table 3.6 is given below:

	User 1	User 2	User 3	User 4	User 5
User 1	1	0.34	0.48	0.44	0.17
User 2	0.34	1	0.44	0.41	0.39
User 3	0.48	0.44	1	0.41	0.40
User 4	0.44	0.41	0.41	1	0.44
User 5	0.17	0.39	0.40	0.44	1

Table 3.7 Similarity of Users According to Table 3.6

Step 4: according to equation (3.4) the prediction for the missing ratings will be calculated as given below:

$$P_{1,5} = 0.37 + \frac{(0.34 \times 0.0) + (0.44 \times 0.05)}{0.34 + 0.44} = 0.40$$

$$P_{2,2} = 0.37 + \frac{(0.34 \times 0.15) + (0.44 \times 0.0) + (0.41 \times 0.05) + (0.39 \times 0.09)}{0.34 + 0.44 + 0.41 + 0.39} = 0.55$$

$$P_{2,9} = 0.37 + \frac{(0.34 \times 0.04) + (0.39 \times 0.0)}{0.34 + 0.39} = 0.39$$

$$P_{3,5} = 0.49 + \frac{(0.44 \times 0.0) + (0.41 \times 0.05)}{0.44 + 0.41} = 0.51$$

$$\mathrm{P}_{3,9} = 0.49 + \frac{(0.48 \times 0.04) + (0.40 \times 0.0)}{0.48 + 0.40} = 0.51$$

$$\mathrm{P}_{4,3} = 0.37 + \frac{(0.44 \times 0.07) + (0.41 \times 0.04) + \ (0.41 \times 0.10)}{0.44 + 0.41 + \ 0.41} = 0.74$$

$$\mathrm{P}_{4,7} = 0.37 + \frac{(0.44 \times 0.33) + (0.41 \times 0.39) + \ (0.41 \times 0.23) + (0.44 \times 0.0)}{0.44 + 0.41 + \ 0.41 + 0.44} = 0.61$$

$$\mathrm{P}_{4,9} = 0.37 + \frac{(0.44 \times 0.04) + (0.44 \times 0.0)}{0.44 + 0.44} = 0.39$$

$$\mathrm{P}_{5,3} = 0.35 + \frac{(0.17 \times 0.07) + (0.39 \times 0.04) + \ (0.40 \times 0.10)}{0.17 + 0.39 + 0.40} = 0.42$$

$$\mathrm{P}_{5,5} = 0.35 + \frac{(0.39 \times 0.0) + \ (0.44 \times 0.05)}{0.39 + 0.44} = 0.38$$

After finding the predictions for the missing ratings, we can choose the top *N* predictions and recommend the corresponding laptops, for example for user 4 we can recommend laptop number 3, because it has the highest prediction value. In real life applications the number of items, users and missing ratings will be much more than the given example, that's why we can choose from a large set of predictions of missing ratings then providing the highest values.

Step 5: converting the prediction values to the original scale.
We can think of this step as defuzzification, for example to convert the prediction $\mathrm{P}_{5,3}$ to the original scale we can use equation (3.1) as follows

$$\frac{\mathrm{ri,j} - 1}{10 - 1} = \ 0.42 \ \Rightarrow \mathrm{ri,j} = 4.78$$

3.6. Why to Use Fuzzy Method

The browsing behavior of users on the web is highly uncertain and fuzzy in nature. Each time the user accesses the site, s/he may have different browsing goals. Furthermore, the same user in the same session may have different goals and interests at different times. It is inappropriate to capture such overlapping interests of the users in crisp clusters. This makes fuzzy clustering algorithms more suitable for usage mining (40) In fuzzy clustering objects which are similar to each other are identified by having high memberships in the same cluster. "Hard" clustering algorithms assign each object to a single cluster that is using the two distinct membership values of 0 and 1. In web usage profiling, this "all or none" or "black or white" membership restriction is not realistic. Very often there may not be sharp boundaries between clusters and many objects may have characteristics of different classes with varying degrees. The main advantage of fuzzy clustering over hard clustering is that it yields more detailed information about the underlying structure of the data.

3.7. Summary

The recommendation process that is described in this chapter mainly depends on the initial matrix, which is users items matrix, this matrix represents the provided ratings from each user to some or all of the available products, in real life the missing ratings will be more because we are dealing with information overload problem. After the users provide their ratings the system starts to manipulate those initial values compare them with each other, finding the similarities between them and then generating predictions for the missing ratings.

Chapter 4 System Implementation

This chapter is about the system implementation, a brief description will be given about the purpose of the system, also a screen shots will be given to illustrate the use of the system.

4.1 Introduction

This system is an E-Commerce project which is dedicated to sell notebooks, personal computers and monitors. These products can be from different manufacturers (brands), the user can navigate through the website and take a look at the available products. After that the user can add a specific product to his shopping cart and after specific period he can check out the products in his cart or he can choose the Buy Now option which allows him to immediately buy specific product, the user can customize his product by choosing from different available options (Customization), and for each option there is different price. Before the user checks out or confirms his order he can review the order, change it, he can change the shipment and payment information, and he can update the quantity for any product to a new value which is less than or equal the quantity available in the stock for that product, also in this project there is an administration page which allows the administrator to add new product, update or delete specific product, show new messages for each user according to his orders (Personalization). The administrator also can read the users' feedback from his page.

As it was mentioned above our purpose is to recommend notebooks to users according to the process described in chapter 3, for the purpose of this thesis the recommendation process will be generated dynamically, which means the inputs for the process will be chosen from the admin, the rating scale as well, the values of the initial users items matrix will be generated randomly each time the admin will change the inputs, this is done to take different scenarios of the aforementioned recommendation process and also for test purposes.

4.2. Home Page

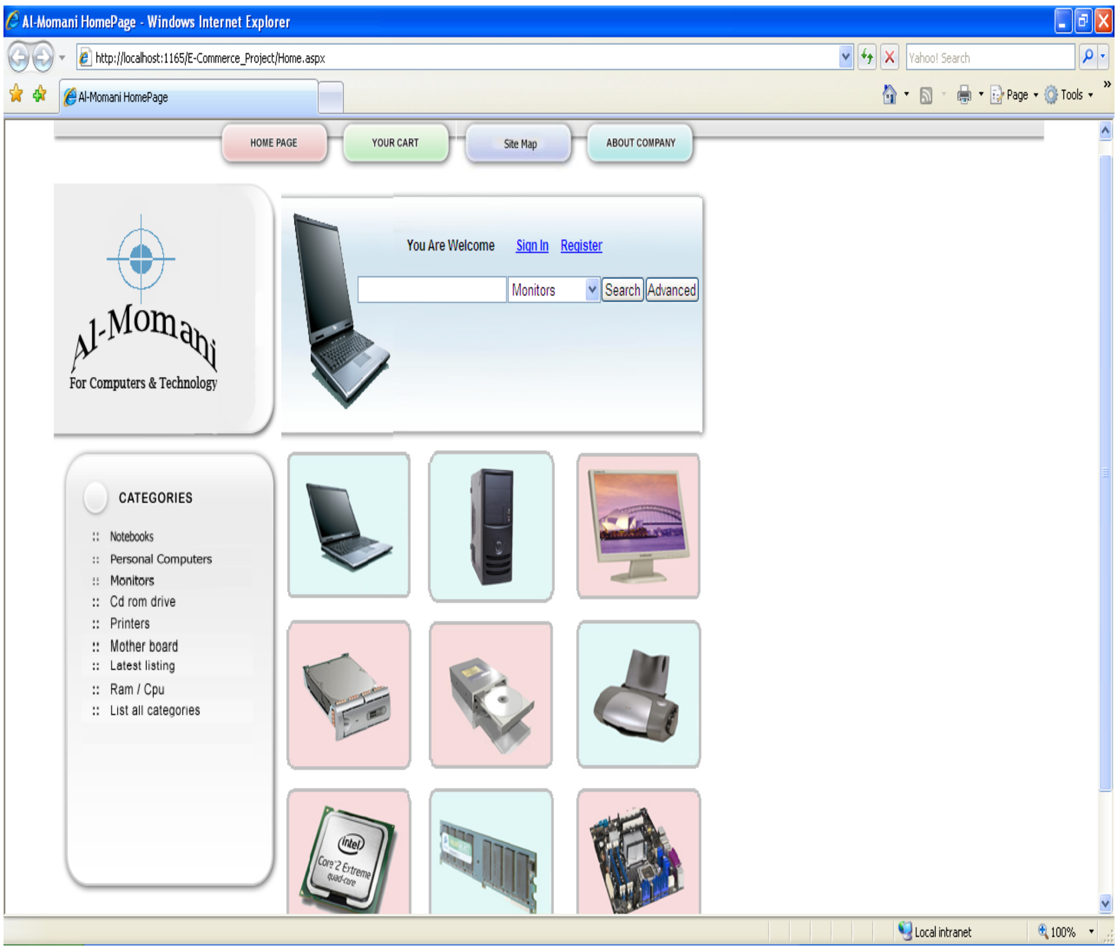


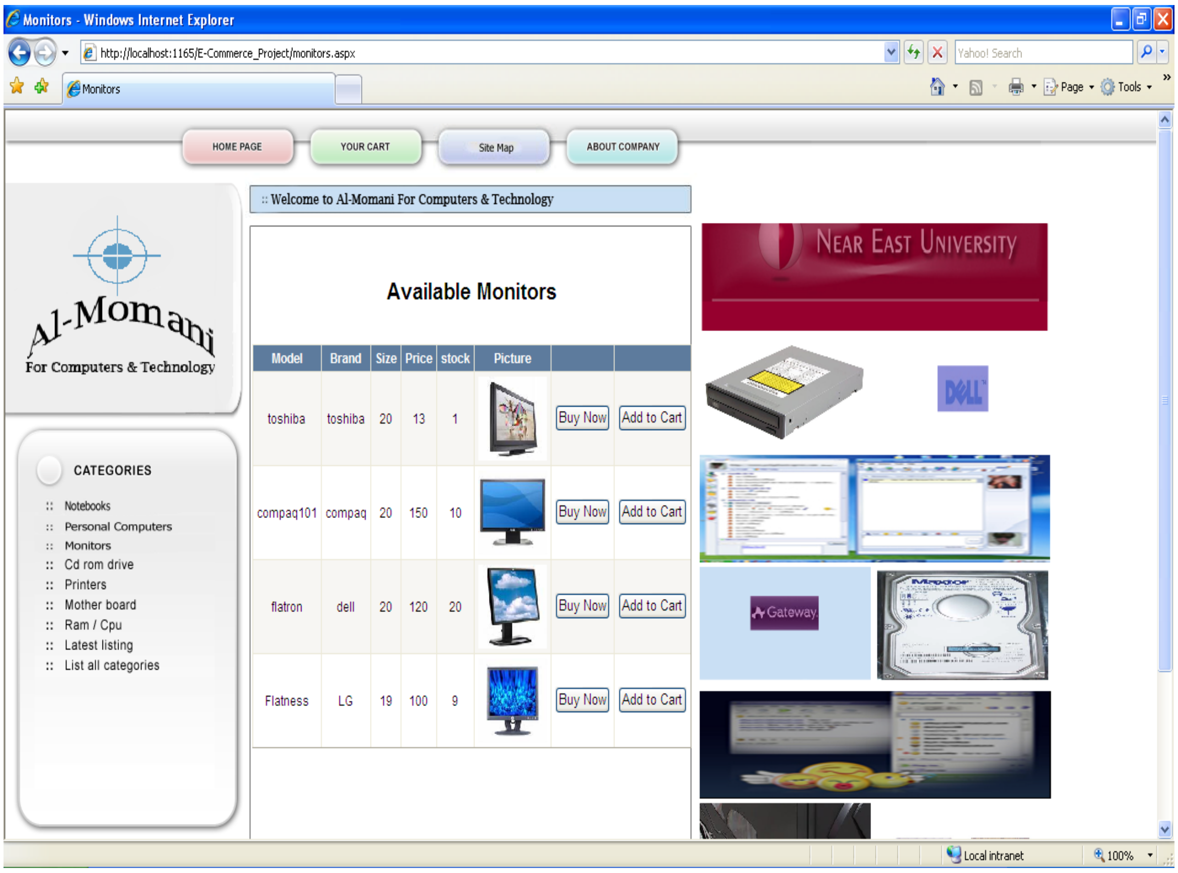
Figure 4.1 Home Page

From the home page the user can do any of the following:-

- Search for a specific product by choosing from the category dropdown list (Monitors, Notebooks, or PCs) and entering key words into the blank text box and then clicking the Search button.
- Advanced Search, by clicking on the advanced button three options will be available category, brand, and price starting at, the user can choose any combinations of these.
- Navigate available notebooks by clicking on the notebook picture or by clicking on the notebook link on the lift side.
- Navigate available personal computers by clicking on the personal computer picture, or by clicking on the personal computers link on the lift side.
- Navigate available monitors by clicking on the monitor picture, or by clicking on the monitors link on the lift side.
- Register or create a new account.
- Signs in for his account.
- Views his shopping cart.
- Views the site map, to view the website structure.

4.3. Available Products

When the user clicks on any product picture or link (Notebooks, Personal Computers, or Monitors) a list of available products will be shown and the order for these products will be according to the newest product added, so the latest product added will be shown first. For example when the user clicks on the monitors link, available monitors page will be appear as shown in Fig. 4.2.



:: Welcome to Al-Momani For Computers & Technology

Available Monitors

Model	Brand	Size	Price	stock	Picture		
toshiba	toshiba	20	13	1		Buy Now	Add to Cart
compaq101	compaq	20	150	10		Buy Now	Add to Cart
flatron	dell	20	120	20		Buy Now	Add to Cart
Flatness	LG	19	100	9		Buy Now	Add to Cart

Figure 4.2 Available Products

4.4. How to Buy

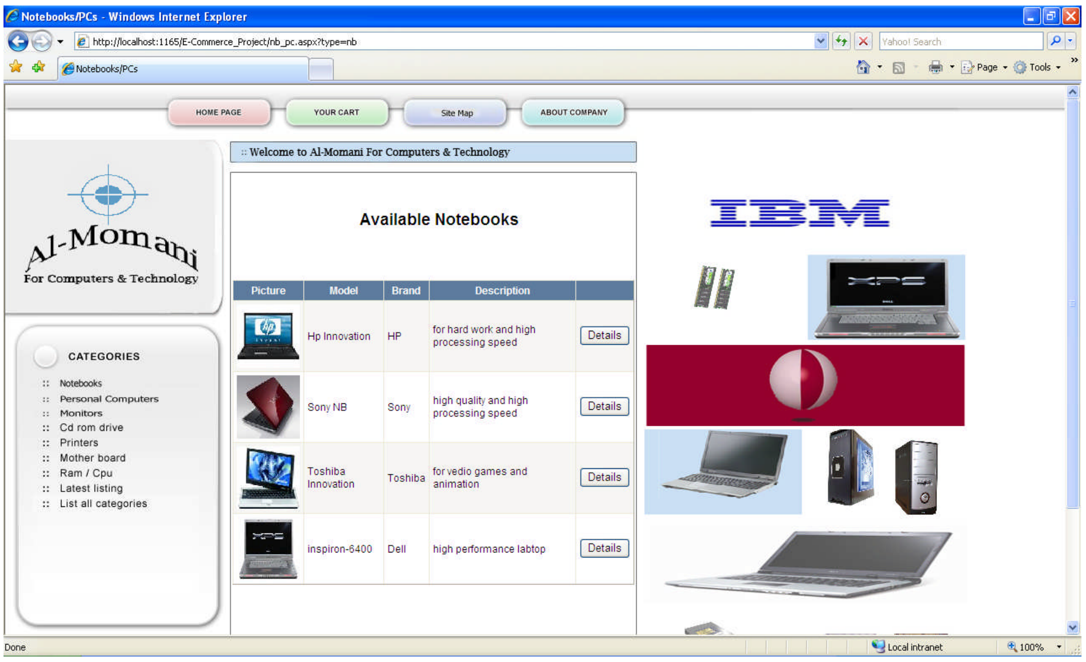
The user has two options to buy a product which are:-

- Buy Now
- Add to Cart

Buy Now Example:-

We assume here that the customer will buy a notebook using this option

When the user clicks on the notebooks link, the page in Fig. 4.3 will be displayed, this page will contain the available notebooks.

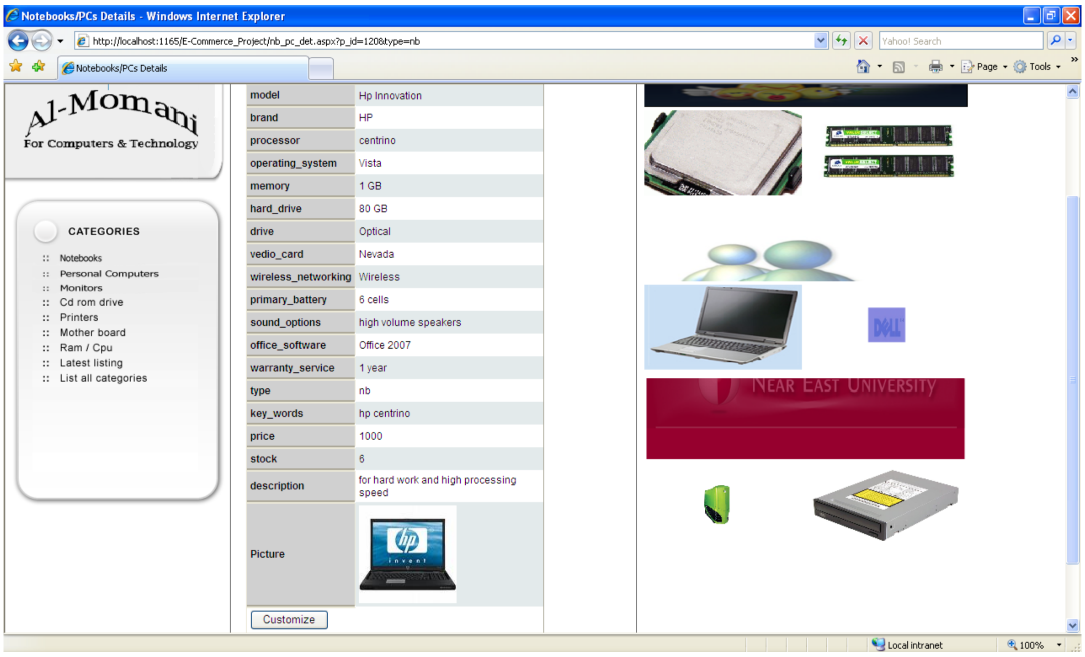


Available Notebooks

Picture	Model	Brand	Description	
	Hp Innovation	HP	for hard work and high processing speed	Details
	Sony NB	Sony	high quality and high processing speed	Details
	Toshiba Innovation	Toshiba	for vedio games and animation	Details
	inspiron-6400	Dell	high performance labtop	Details

Figure 4.3 Available Noebooks

Now if the user wants to view the details for this product he simply clicks the details button, after the user clicks the details button the details for the selected notebook will be displayed as shown in Fig. 4.4.


Notebooks/PCs Details - Windows Internet Explorer

http://localhost:1165/E-Commerce_Project/nb_pc_det.aspx?p_id=120&type=nb

Al-Momani For Computers & Technology

CATEGORIES

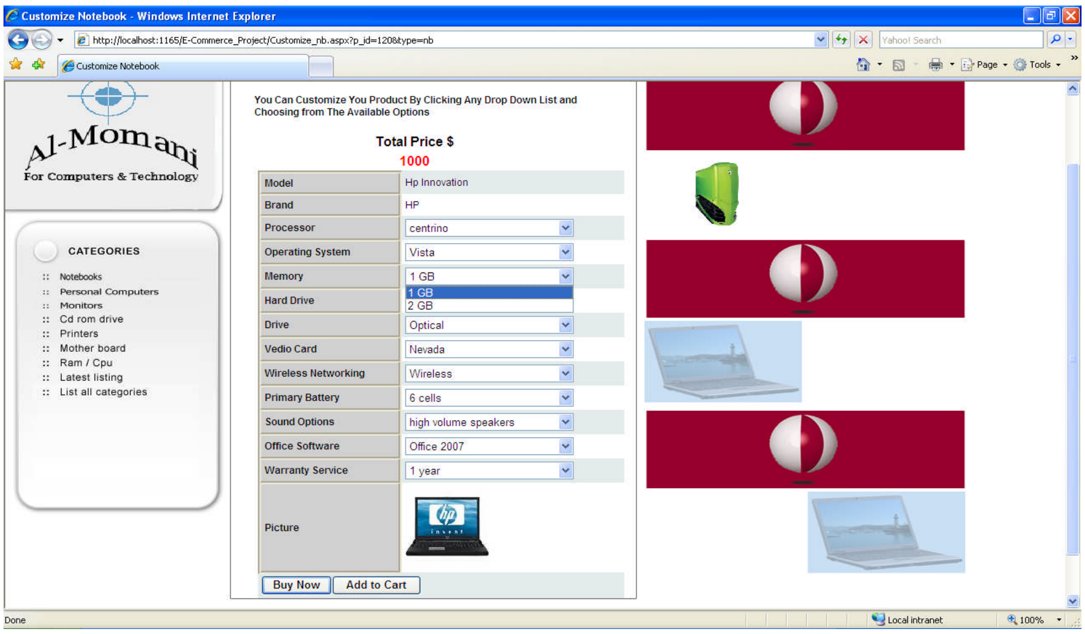
- Notebooks
- Personal Computers
- Monitors
- Cd rom drive
- Printers
- Mother board
- Ram / Cpu
- Latest listing
- List all categories

model	Hp Innovation
brand	HP
processor	centrino
operating_system	Vista
memory	1 GB
hard_drive	80 GB
drive	Optical
vedio_card	Nevada
wireless_networking	Wireless
primary_battery	6 cells
sound_options	high volume speakers
office_software	Office 2007
warranty_service	1 year
type	nb
key_words	hp centrino
price	1000
stock	6
description	for hard work and high processing speed
Picture	

Customize

Figure 4.4 Notebook Details

In this page at the end of the details there is a customize button, this button will transfer the user to the customization page as shown in Fig. 4.5.


Customize Notebook - Windows Internet Explorer

http://localhost:1165/E-Commerce_Project/Customize_nb.aspx?p_id=120&type=nb

Al-Momani For Computers & Technology

CATEGORIES

- Notebooks
- Personal Computers
- Monitors
- Cd rom drive
- Printers
- Mother board
- Ram / Cpu
- Latest listing
- List all categories

You Can Customize You Product By Clicking Any Drop Down List and Choosing from The Available Options

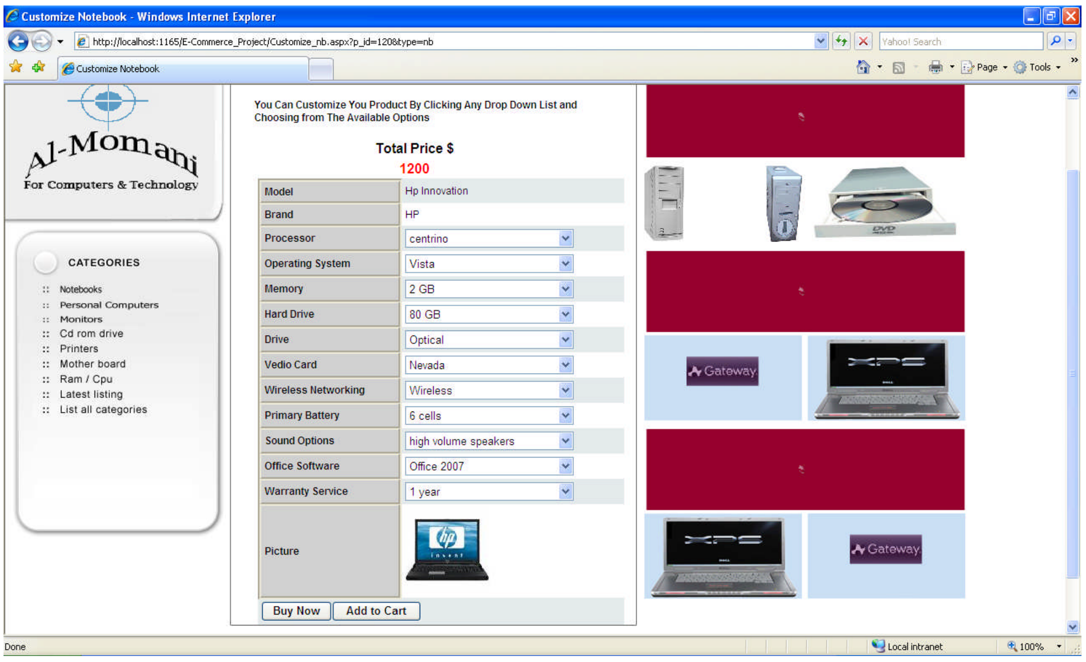
Total Price $
1000

Model	Hp Innovation
Brand	HP
Processor	centrino
Operating System	Vista
Memory	1 GB (1 GB, 2 GB)
Hard Drive	
Drive	Optical
Vedio Card	Nevada
Wireless Networking	Wireless
Primary Battery	6 cells
Sound Options	high volume speakers
Office Software	Office 2007
Warranty Service	1 year
Picture	

Buy Now | Add to Cart

Figure 4.5 Customization

In this page the user will see a dropdown list for each of the specifications, for example the user might want to choose 2 GB of memory instead of 1 GB so the user can click on the list beside the memory and change the memory from 1 GB to 2 GB after that the total price will be changed accordingly, as show in Fig. 4.6.



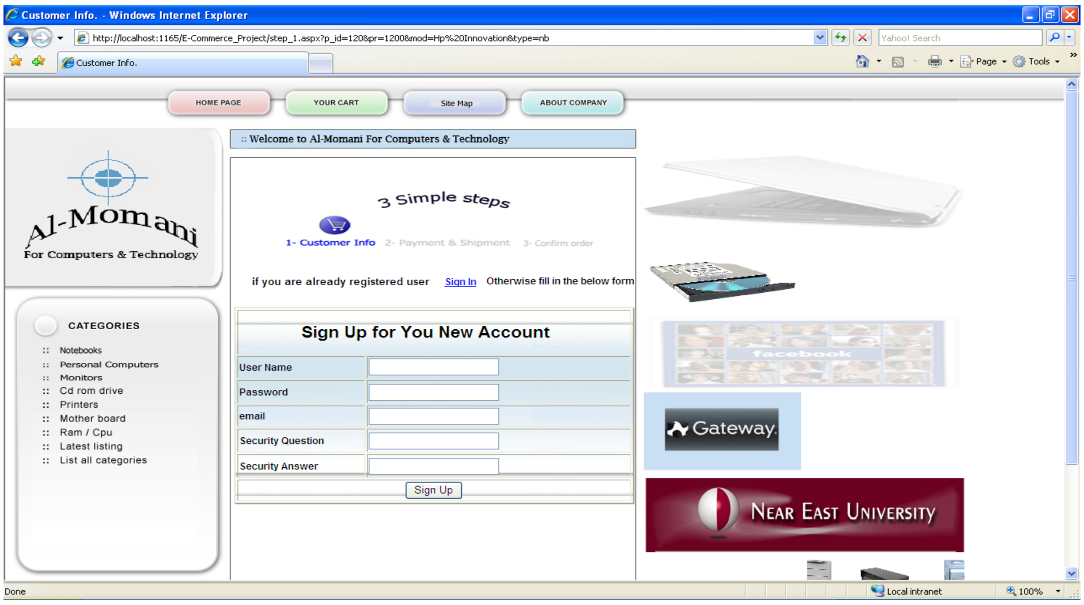
You Can Customize You Product By Clicking Any Drop Down List and Choosing from The Available Options

Total Price $
1200

Model	Hp Innovation
Brand	HP
Processor	centrino
Operating System	Vista
Memory	2 GB
Hard Drive	80 GB
Drive	Optical
Vedio Card	Nevada
Wireless Networking	Wireless
Primary Battery	6 cells
Sound Options	high volume speakers
Office Software	Office 2007
Warranty Service	1 year
Picture	

Figure 4.6 Customization (Cont.)

Now in the above page there are two buttons at the end of the page, our purpose in this example is to show how to Buy Now, when the user clicks on this button, the user will be transferred to step 1, as shown in Fig. 4.7, if the user is registered then he can click the Sign In link, otherwise the user can creates his new account by filling the form shown in the figure.


Customer Info. - Windows Internet Explorer

http://localhost:1165/E-Commerce_Project/step_1.aspx?p_id=120&pr=1200&mod=Hp%20Innovation&type=nb

:: Welcome to Al-Momani For Computers & Technology

3 Simple steps

1- Customer Info 2- Payment & Shipment 3- Confirm order

if you are already registered user Sign In Otherwise fill in the below form

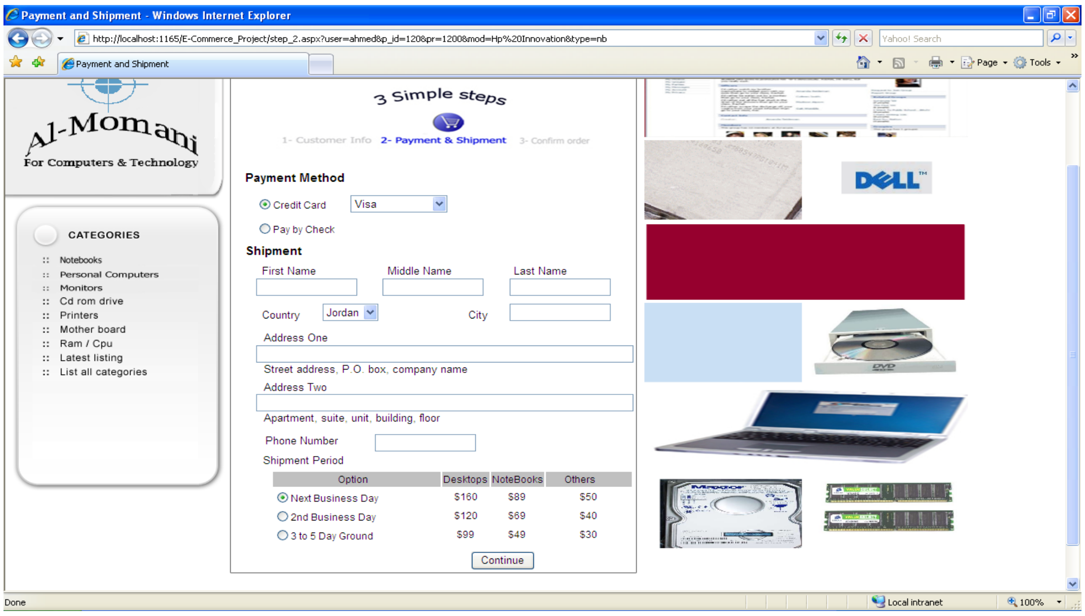
Sign Up for You New Account

User Name	
Password	
email	
Security Question	
Security Answer	

Sign Up

Figure 4.7 Customer Info. (Step 1)

After the user signs in the user will be transferred to step 2 which is Payment & Shipment as shown in Fig. 4.8, in this page the user can fill his payment method, his address, and his shipment period.


Payment and Shipment - Windows Internet Explorer

http://localhost:1165/E-Commerce_Project/step_2.aspx?user=ahmed&p_id=120&pr=1200&mod=Hp%20Innovation&type=nb

3 Simple steps

1- Customer Info 2- Payment & Shipment 3- Confirm order

Payment Method

[✓ Credit Card] Visa

Pay by Check

Shipment

First Name Middle Name Last Name

Country Jordan City

Address One

Street address, P.O. box, company name

Address Two

Apartment, suite, unit, building, floor

Phone Number

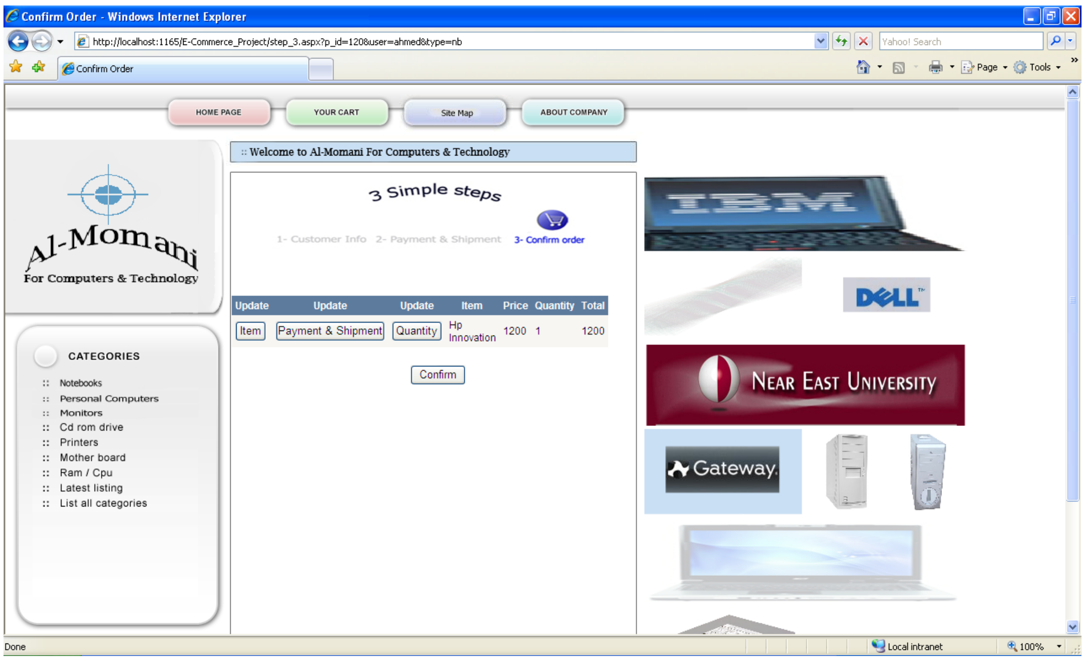
Shipment Period

Option	Desktops	NoteBooks	Others
[✓ Next Business Day]	$160	$89	$50
2nd Business Day	$120	$69	$40
3 to 5 Day Ground	$99	$49	$30

Continue

Figure 4.8 Payment & Shipment (Step 2)

When the user clicks the continue button he will be transferred to the final step which is confirm order as shown in Fig. 4.9, from this page the user can update his item by clicking on the item button, update his payment and shipment by clicking on the Payment & Shipment button, and update the quantity by clicking on the quantity button, after doing the necessary modifications the user can click the confirm button to confirm his Order.

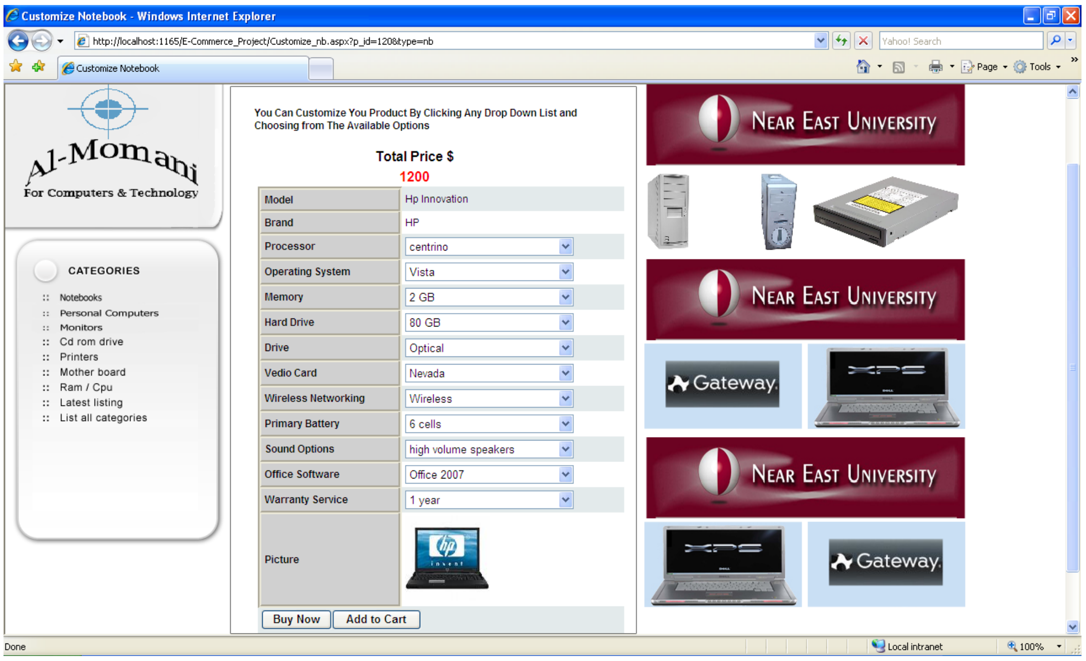


Update	Update	Update	Item	Price	Quantity	Total
Item	Payment & Shipment	Quantity	Hp Innovation	1200	1	1200

Figure 4.9 Confirm Order (Step 3)

4.5. Add to Cart Example:-

Let's assume that the user wants to add a notebook to his cart, in Fig. 4.10 the user simply clicks the Add to Cart button.



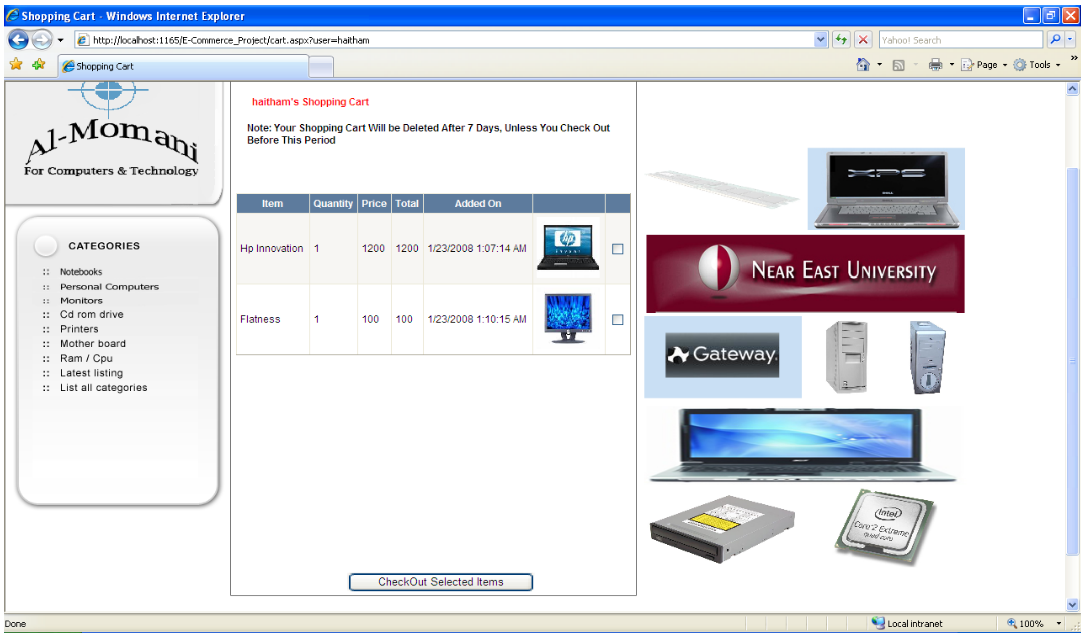
You Can Customize You Product By Clicking Any Drop Down List and Choosing from The Available Options

Total Price $
1200

Model	Hp Innovation
Brand	HP
Processor	centrino
Operating System	Vista
Memory	2 GB
Hard Drive	80 GB
Drive	Optical
Vedio Card	Nevada
Wireless Networking	Wireless
Primary Battery	6 cells
Sound Options	high volume speakers
Office Software	Office 2007
Warranty Service	1 year
Picture	

Figure 4.10 Add to Cart

After that the user will be transferred to the Sign In page as in the previous example but here after the user signs in he will be transferred to his shopping cart as shown in Fig. 4.11.



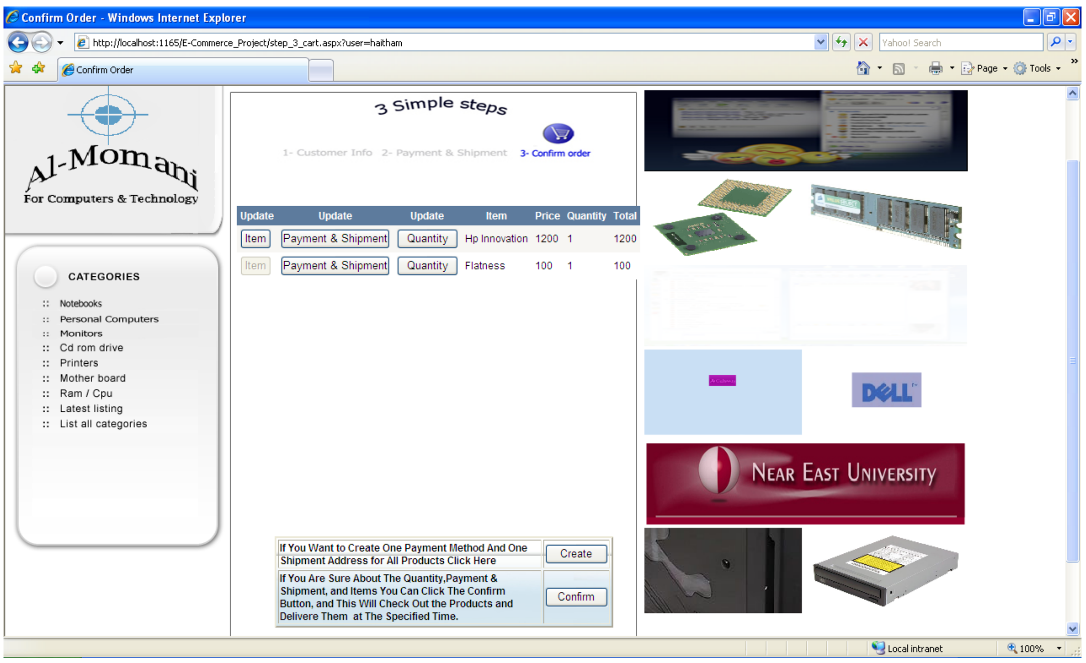
haitham's Shopping Cart

Note: Your Shopping Cart Will be Deleted After 7 Days, Unless You Check Out Before This Period

Item	Quantity	Price	Total	Added On		
Hp Innovation	1	1200	1200	1/23/2008 1:07:14 AM		☐
Flatness	1	100	100	1/23/2008 1:10:15 AM		☐

Figure 4.11 Shopping Cart

Now the user can check the items he wants to check out and then pressing the *Checkout Selected Items* button. After that the user will be transferred to step 2 (Payment & Shipment) and then to step 3 (Confirm order) but with one difference here, that in step 3 the user can create one Payment Method and one Shipment Address for all the products if he wants to, and that can be done by clicking the create button as shown in Fig. 4.12.



Update	Update	Update	Item	Price	Quantity	Total
Item	Payment & Shipment	Quantity	Hp Innovation	1200	1	1200
Item	Payment & Shipment	Quantity	Flatness	100	1	100

If You Want to Create One Payment Method And One Shipment Address for All Products Click Here	Create
If You Are Sure About The Quantity,Payment & Shipment, and Items You Can Click The Confirm Button, and This Will Check Out the Products and Delivere Them at The Specified Time.	Confirm

Figure 4.12 Checking out numerous products

4.6. Administration Page

The administration page as the name implies is for administrators and its look like Fig. 4.13.

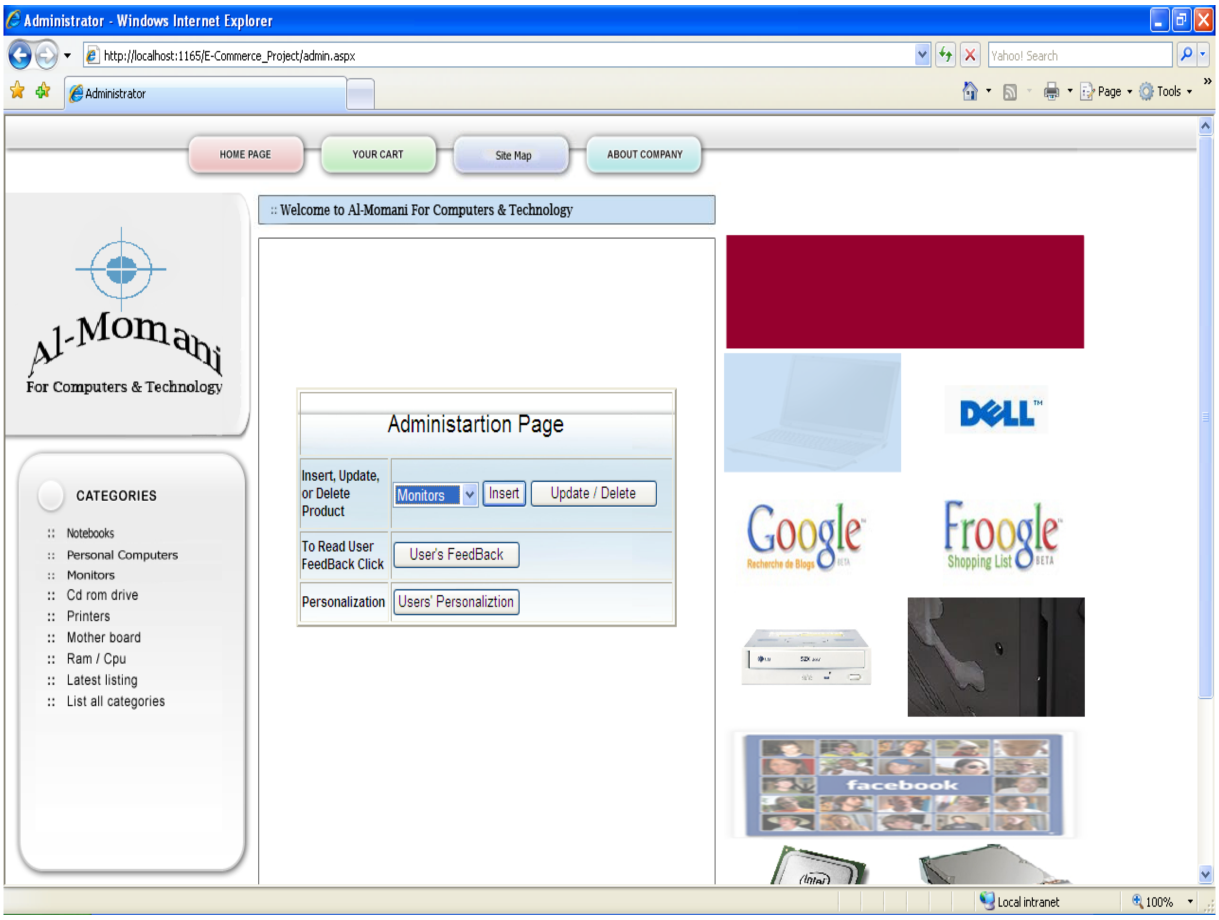


Figure 4.13 Adminstration Page

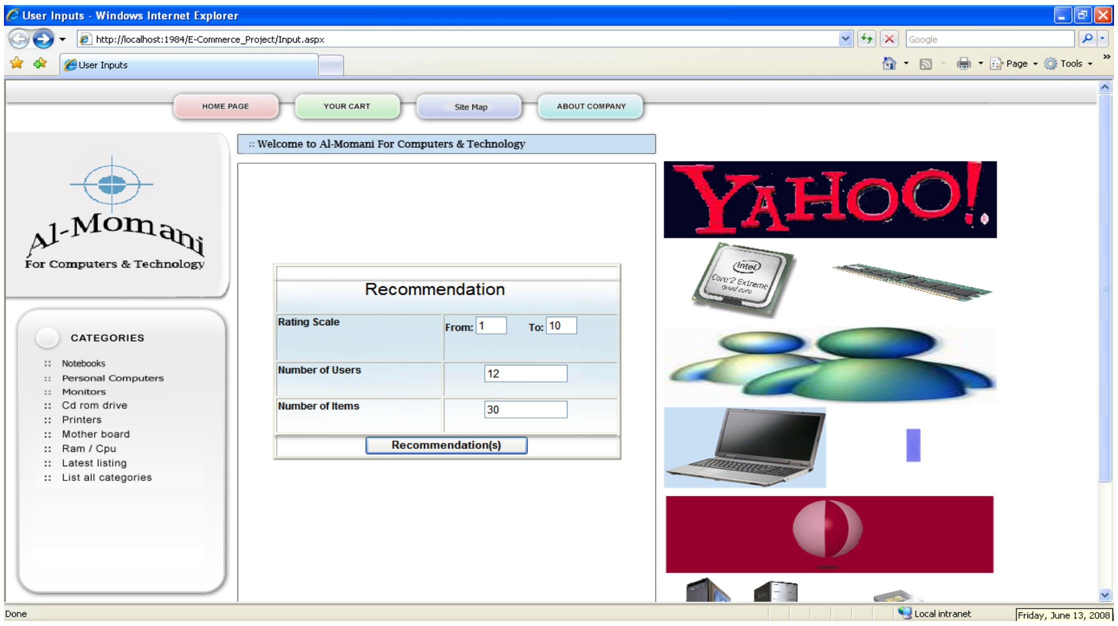
From this page the admin can insert, update or delete a product depending on the selected value in the dropdown list, the administrator also can view the user's feedback, and also the administrator can create a new message for a specific user depending on his orders (*Personalization*).

4.7. Recommendation Process

It should be mentioned here that the recommendation process that is used in the system is the same process explained in chapter 3, as it was mentioned the original user item matrix will contain the actual users' ratings, with some missing ratings for each user, usually the user is asked to rate specific number of items and in this case the rest of the items will be unrated, also at the beginning there won’t be a sufficient number of users or items to complete the recommendation process, therefore for the purpose of this thesis, every piece of information needed will be generated randomly and every time with different values, to enable us to see the results of the proposed system using different situations.

4.7.1. Defining the users item matrix dimensions and rating scale

In Fig. 4.14 we will define the dimension of the users items matrix, the rating scale as well, for example as shown in the figure the rating scale is 1-10, so the minimum rating value will be 1 and the maximum will be 10, the number of users is 12 which means the number of rows of the dynamically generated matrix will be 12, the number of items is 30, which means the number of columns of the matrix will be 30, after that we simply click the recommendations button to see the results.



Recommendation

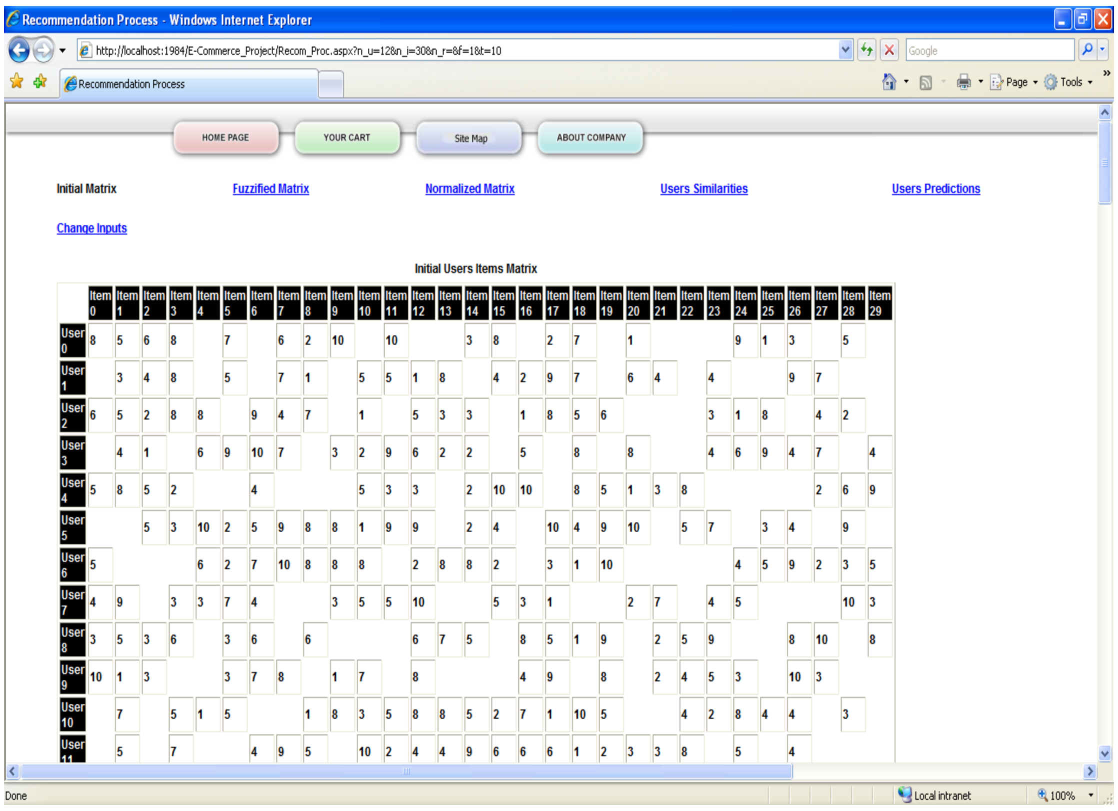
Rating Scale	From: 1 To: 10
Number of Users	12
Number of Items	30

Recommendation(s)

Figure 4.14 Defining matrix dimensions

4.7.2. The recommendation process

The initial matrix: this matrix in Fig. 4.15 is generated according to user inputs in the previous section, the rating values in the body of the matrix is generated dynamically for each user, also the missing ratings for each user are generated dynamically, each time we change the inputs or refresh the page the matrix will be filled with different values

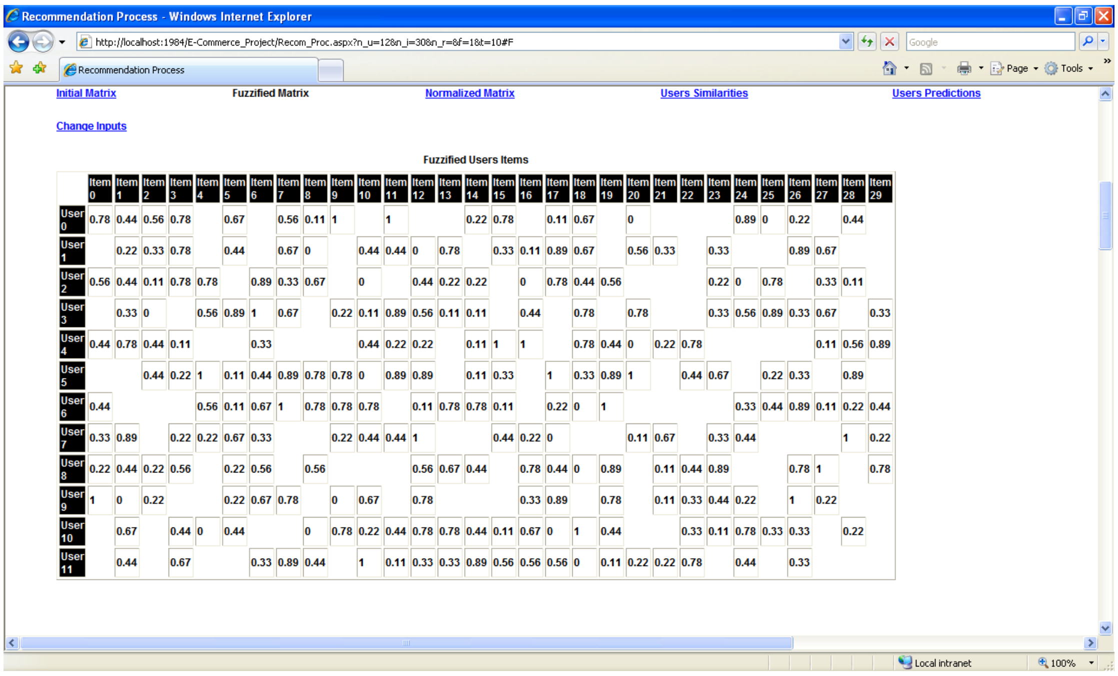


Initial Users Items Matrix

	Item 0	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6	Item 7	Item 8	Item 9	Item 10	Item 11	Item 12	Item 13	Item 14	Item 15	Item 16	Item 17	Item 18	Item 19	Item 20	Item 21	Item 22	Item 23	Item 24	Item 25	Item 26	Item 27	Item 28	Item 29
User 0	8	5	6	8		7		6	2	10		10			3	8		2	7		1				9	1	3		5	
User 1		3	4	8		5		7	1		5	5	1	8		4	2	9	7		6	4		4			9	7		
User 2	6	5	2	8	8		9	4	7		1		5	3	3		1	8	5	6				3	1	8		4	2	
User 3		4	1		6	9	10	7		3	2	9	6	2	2		5		8		8			4	6	9	4	7		4
User 4	5	8	5	2			4				5	3	3		2	10	10		8	5	1	3	8					2	6	9
User 5			5	3	10	2	5	9	8	8	1	9	9		2	4		10	4	9	10		5	7		3	4		9	
User 6	5				6	2	7	10	8	8	8		2	8	8	2		3	1	10					4	5	9	2	3	5
User 7	4	9		3	3	7	4			3	5	5	10			5	3	1			2	7		4	5				10	3
User 8	3	5	3	6		3	6		6				6	7	5		8	5	1	9		2	5	9			8	10		8
User 9	10	1	3			3	7	8		1	7		8				4	9		8		2	4	5	3		10	3		
User 10		7		5	1	5			1	8	3	5	8	8	5	2	7	1	10	5			4	2	8	4	4		3	
User 11		5		7			4	9	5		10	2	4	4	9	6	6	6	1	2	3	3	8		5		4			

Figure 4.15 Initial Matrix

Fuzzified matrix: according to equation (3.1) each value in the original matrix will be converted to the corresponding fuzzy value, to see the result of this step we simply click on the Fuzzified Matrix link, then the result will be as shown in Fig. 4.16.


Recommendation Process - Windows Internet Explorer
http://localhost:1984/E-Commerce_Project/Recom_Proc.aspx?n_u=12&n_i=30&n_r=8&f=1&t=10#F
Recommendation Process
Page
Tools
Local intranet
100%

Initial Matrix Fuzzified Matrix Normalized Matrix Users Similarities Users Predictions

Change Inputs

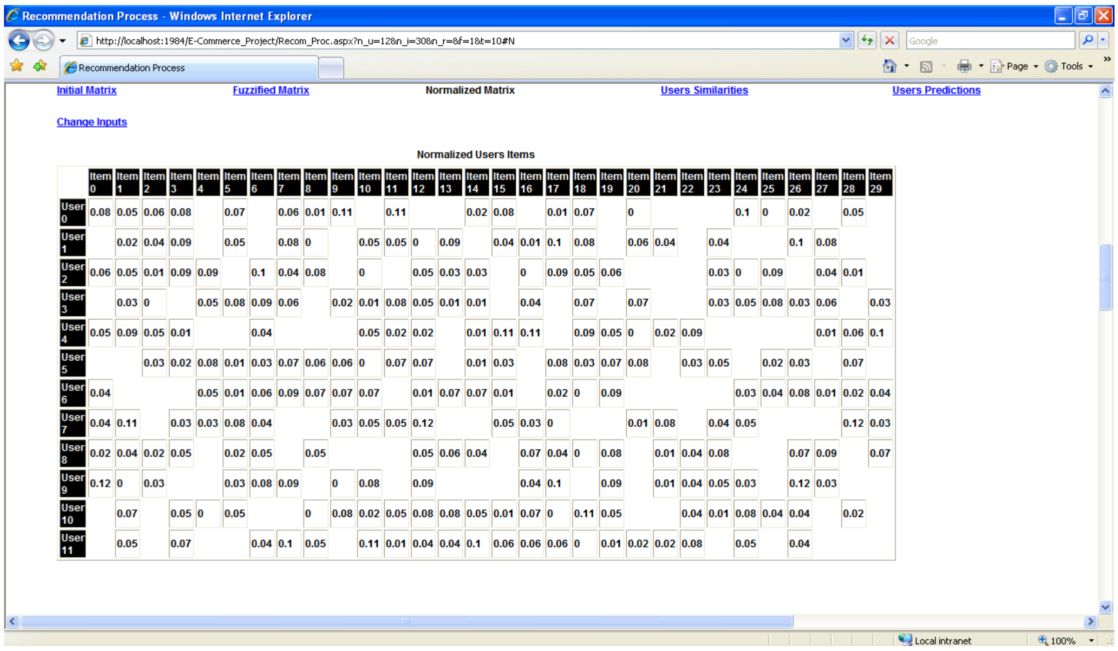
Fuzzified Users Items

	Item 0	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6	Item 7	Item 8	Item 9	Item 10	Item 11	Item 12	Item 13	Item 14	Item 15	Item 16	Item 17	Item 18	Item 19	Item 20	Item 21	Item 22	Item 23	Item 24	Item 25	Item 26	Item 27	Item 28	Item 29
User 0	0.78	0.44	0.56	0.78		0.67		0.56	0.11	1		1			0.22	0.78		0.11	0.67		0				0.89	0	0.22		0.44	
User 1		0.22	0.33	0.78		0.44		0.67	0		0.44	0.44	0	0.78		0.33	0.11	0.89	0.67		0.56	0.33		0.33			0.89	0.67		
User 2	0.56	0.44	0.11	0.78	0.78		0.89	0.33	0.67		0		0.44	0.22	0.22		0	0.78	0.44	0.56				0.22	0	0.78		0.33	0.11	
User 3		0.33	0		0.56	0.89	1	0.67		0.22	0.11	0.89	0.56	0.11	0.11		0.44		0.78		0.78			0.33	0.56	0.89	0.33	0.67		0.33
User 4	0.44	0.78	0.44	0.11			0.33				0.44	0.22	0.22		0.11	1	1		0.78	0.44	0	0.22	0.78					0.11	0.56	0.89
User 5			0.44	0.22	1	0.11	0.44	0.89	0.78	0.78	0	0.89	0.89		0.11	0.33		1	0.33	0.89	1		0.44	0.67		0.22	0.33		0.89	
User 6	0.44				0.56	0.11	0.67	1	0.78	0.78	0.78		0.11	0.78	0.78	0.11		0.22	0	1					0.33	0.44	0.89	0.11	0.22	0.44
User 7	0.33	0.89		0.22	0.22	0.67	0.33			0.22	0.44	0.44	1			0.44	0.22	0			0.11	0.67		0.33	0.44				1	0.22
User 8	0.22	0.44	0.22	0.56		0.22	0.56		0.56				0.56	0.67	0.44		0.78	0.44	0	0.89		0.11	0.44	0.89			0.78	1		0.78
User 9	1	0	0.22			0.22	0.67	0.78		0	0.67		0.78				0.33	0.89		0.78		0.11	0.33	0.44	0.22		1	0.22		
User 10		0.67		0.44	0	0.44			0	0.78	0.22	0.44	0.78	0.78	0.44	0.11	0.67	0	1	0.44			0.33	0.11	0.78	0.33	0.33		0.22	
User 11		0.44		0.67			0.33	0.89	0.44		1	0.11	0.33	0.33	0.89	0.56	0.56	0.56	0	0.11	0.22	0.22	0.78		0.44		0.33			

Figure 4.16 Fuzzified Matrix

As we can see now the original ratings values are converted to fuzzy values of the range from 0 to 1, the 0 value corresponds to the lowest rating scale in our example is 1, and the value 1 corresponds to the highest scale which is 10

Normalized Matrix: to ensure that each user has the same contribution to the prediction calculations, the next step is to normalize the fuzzified matrix, simply we click the Normalized Matrix link, the result will be shown in Fig. 4.17


Recommendation Process - Windows Internet Explorer
http://localhost:1984/E-Commerce_Project/Recom_Proc.aspx?n_u=12&n_i=30&n_r=8&f=1&t=10#N
Recommendation Process
Local intranet
100%

Initial Matrix Fuzzified Matrix Normalized Matrix Users Similarities Users Predictions

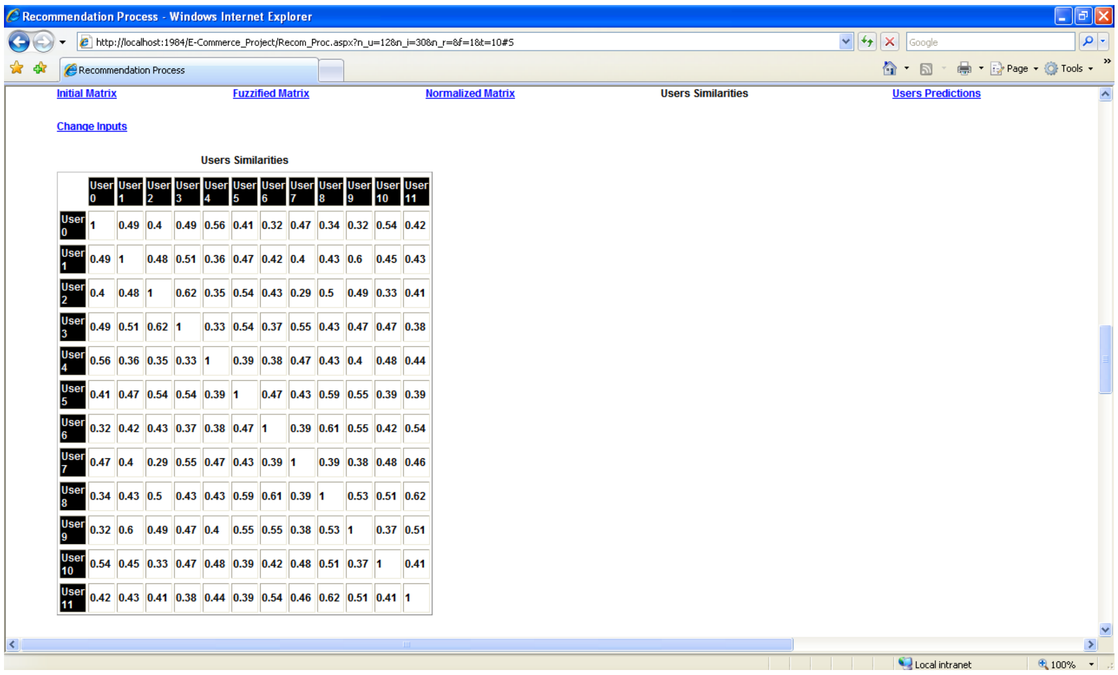
Change Inputs

Normalized Users Items

	Item 0	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6	Item 7	Item 8	Item 9	Item 10	Item 11	Item 12	Item 13	Item 14	Item 15	Item 16	Item 17	Item 18	Item 19	Item 20	Item 21	Item 22	Item 23	Item 24	Item 25	Item 26	Item 27	Item 28	Item 29
User 0	0.08	0.05	0.06	0.08		0.07		0.06	0.01	0.11		0.11			0.02	0.08		0.01	0.07		0				0.1	0	0.02		0.05	
User 1		0.02	0.04	0.09		0.05		0.08	0		0.05	0.05	0	0.09		0.04	0.01	0.1	0.08		0.06	0.04		0.04			0.1	0.08		
User 2	0.06	0.05	0.01	0.09	0.09		0.1	0.04	0.08		0		0.05	0.03	0.03		0	0.09	0.05	0.06				0.03	0	0.09		0.04	0.01	
User 3		0.03	0		0.05	0.08	0.09	0.06		0.02	0.01	0.08	0.05	0.01	0.01		0.04		0.07		0.07			0.03	0.05	0.08	0.03	0.06		0.03
User 4	0.05	0.09	0.05	0.01			0.04				0.05	0.02	0.02		0.01	0.11	0.11		0.09	0.05	0	0.02	0.09					0.01	0.06	0.1
User 5			0.03	0.02	0.08	0.01	0.03	0.07	0.06	0.06	0	0.07	0.07		0.01	0.03		0.08	0.03	0.07	0.08		0.03	0.05		0.02	0.03		0.07	
User 6	0.04				0.05	0.01	0.06	0.09	0.07	0.07	0.07		0.01	0.07	0.07	0.01		0.02	0	0.09					0.03	0.04	0.08	0.01	0.02	0.04
User 7	0.04	0.11		0.03	0.03	0.08	0.04			0.03	0.05	0.05	0.12			0.05	0.03	0			0.01	0.08		0.04	0.05				0.12	0.03
User 8	0.02	0.04	0.02	0.05		0.02	0.05		0.05				0.05	0.06	0.04		0.07	0.04	0	0.08		0.01	0.04	0.08			0.07	0.09		0.07
User 9	0.12	0	0.03			0.03	0.08	0.09		0	0.08		0.09				0.04	0.1		0.09		0.01	0.04	0.05	0.03		0.12	0.03		
User 10		0.07		0.05	0	0.05			0	0.08	0.02	0.05	0.08	0.08	0.05	0.01	0.07	0	0.11	0.05			0.04	0.01	0.08	0.04	0.04		0.02	
User 11		0.05		0.07			0.04	0.1	0.05		0.11	0.01	0.04	0.04	0.1	0.06	0.06	0.06	0	0.01	0.02	0.02	0.08		0.05		0.04			

Figure 4.17 Normalized Matrix

Users Similarities: according to equations (3.2) and (3.3) similarity among each user and the other users will be calculated as shown in Fig. 4.18.


Recommendation Process - Windows Internet Explorer
http://localhost:1984/E-Commerce_Project/Recom_Proc.aspx?n_u=12&n_i=30&n_r=&f=1&t=10#5
Recommendation Process
Initial Matrix
Fuzzified Matrix
Normalized Matrix
Users Similarities
Users Predictions
Change Inputs
Local intranet
100%

Users Similarities

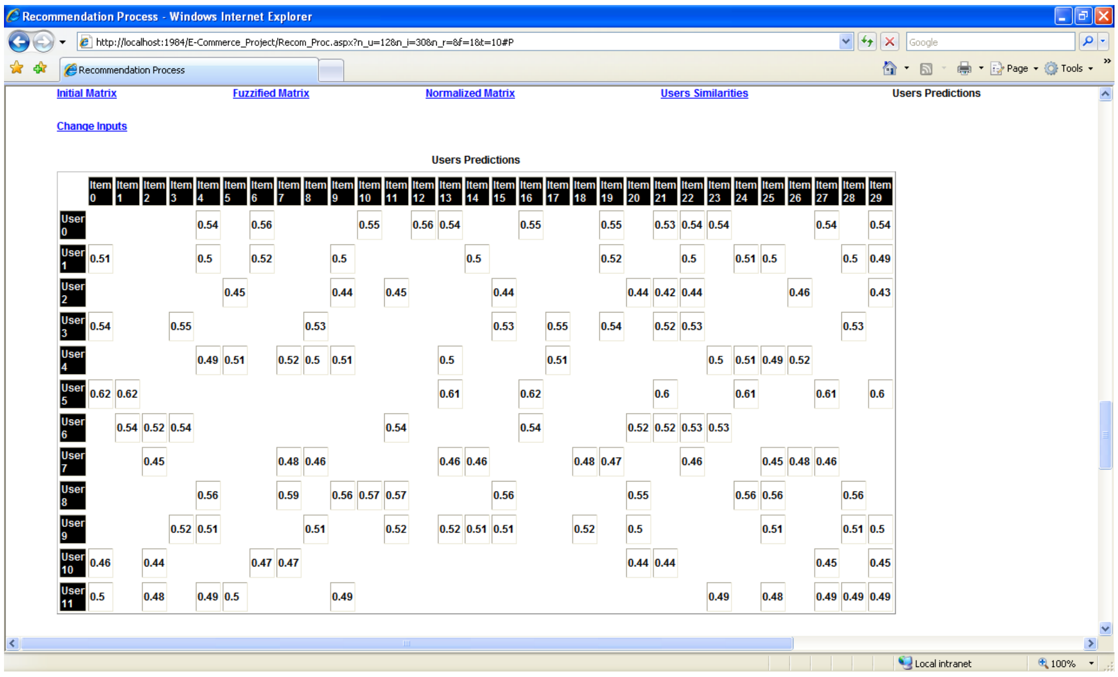
	User 0	User 1	User 2	User 3	User 4	User 5	User 6	User 7	User 8	User 9	User 10	User 11
User 0	1	0.49	0.4	0.49	0.56	0.41	0.32	0.47	0.34	0.32	0.54	0.42
User 1	0.49	1	0.48	0.51	0.36	0.47	0.42	0.4	0.43	0.6	0.45	0.43
User 2	0.4	0.48	1	0.62	0.35	0.54	0.43	0.29	0.5	0.49	0.33	0.41
User 3	0.49	0.51	0.62	1	0.33	0.54	0.37	0.55	0.43	0.47	0.47	0.38
User 4	0.56	0.36	0.35	0.33	1	0.39	0.38	0.47	0.43	0.4	0.48	0.44
User 5	0.41	0.47	0.54	0.54	0.39	1	0.47	0.43	0.59	0.55	0.39	0.39
User 6	0.32	0.42	0.43	0.37	0.38	0.47	1	0.39	0.61	0.55	0.42	0.54
User 7	0.47	0.4	0.29	0.55	0.47	0.43	0.39	1	0.39	0.38	0.48	0.46
User 8	0.34	0.43	0.5	0.43	0.43	0.59	0.61	0.39	1	0.53	0.51	0.62
User 9	0.32	0.6	0.49	0.47	0.4	0.55	0.55	0.38	0.53	1	0.37	0.51
User 10	0.54	0.45	0.33	0.47	0.48	0.39	0.42	0.48	0.51	0.37	1	0.41
User 11	0.42	0.43	0.41	0.38	0.44	0.39	0.54	0.46	0.62	0.51	0.41	1

Figure 4.18 Users' Similarities

Two important things should be noticed

- The main diagonal is always 1, because the similarity between the user and himself is always 1
- The users similarity matrix is a symmetric matrix, for example the similarity between user 2 and user 1 should be the same as the similarity between user 1 and user 2, which is what we got

Users Predictions: according to equation (3.4) prediction for each user for the *missing ratings* will be calculated after clicking on the Users Predictions link as shown in Fig. 4.19.


Recommendation Process - Windows Internet Explorer
http://localhost:1984/E-Commerce_Project/Recom_Proc.aspx?n_u=12&n_i=30&n_r=8&f=1&t=10#P
Recommendation Process
Initial Matrix
Fuzzified Matrix
Normalized Matrix
Users Similarities
Users Predictions
Change Inputs
Local intranet
100%

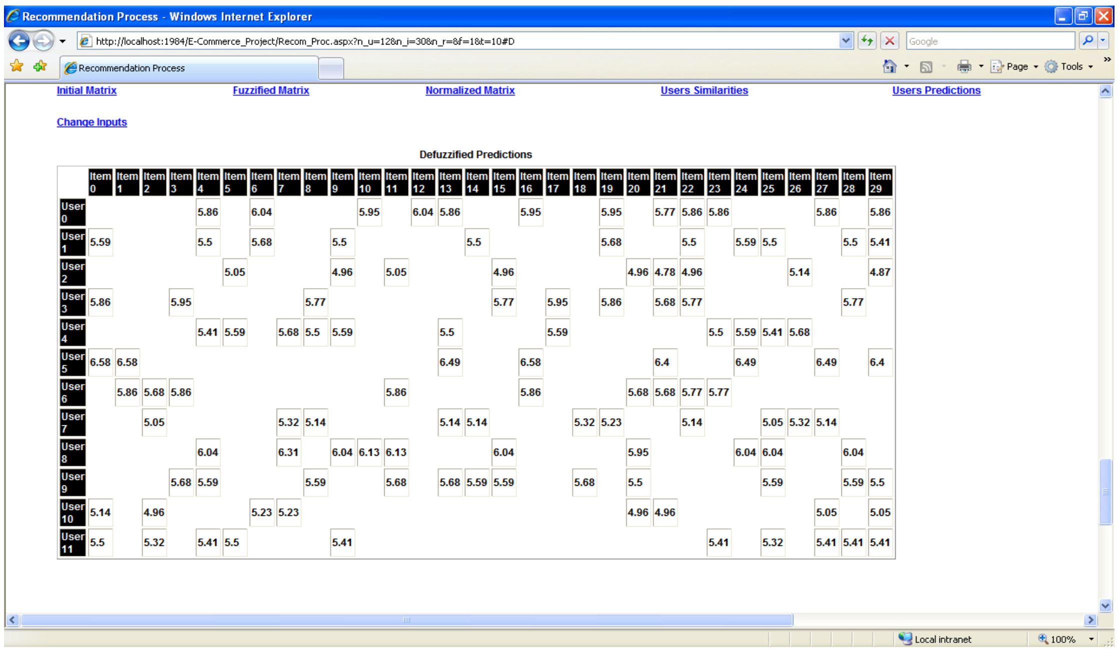
Users Predictions

	Item 0	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6	Item 7	Item 8	Item 9	Item 10	Item 11	Item 12	Item 13	Item 14	Item 15	Item 16	Item 17	Item 18	Item 19	Item 20	Item 21	Item 22	Item 23	Item 24	Item 25	Item 26	Item 27	Item 28	Item 29
User 0					0.54		0.56				0.55		0.56	0.54			0.55			0.55		0.53	0.54	0.54				0.54		0.54
User 1	0.51				0.5		0.52			0.5					0.5					0.52			0.5		0.51	0.5			0.5	0.49
User 2						0.45				0.44		0.45				0.44					0.44	0.42	0.44				0.46			0.43
User 3	0.54			0.55					0.53							0.53		0.55		0.54		0.52	0.53						0.53	
User 4					0.49	0.51		0.52	0.5	0.51				0.5				0.51						0.5	0.51	0.49	0.52			
User 5	0.62	0.62												0.61			0.62					0.6			0.61			0.61		0.6
User 6		0.54	0.52	0.54								0.54					0.54				0.52	0.52	0.53	0.53						
User 7			0.45					0.48	0.46					0.46	0.46				0.48	0.47			0.46			0.45	0.48	0.46		
User 8					0.56			0.59		0.56	0.57	0.57				0.56					0.55				0.56	0.56			0.56	
User 9				0.52	0.51				0.51			0.52		0.52	0.51	0.51			0.52		0.5					0.51			0.51	0.5
User 10	0.46		0.44				0.47	0.47													0.44	0.44						0.45		0.45
User 11	0.5		0.48		0.49	0.5				0.49														0.49		0.48		0.49	0.49	0.49

Figure 4.19 Users' Predictions

Here the values in the body of the matrix are the prediction for the missing ratings in the original matrix, and the missing values in the above page are not missing actually, but they are already rated.

Defuzzified Predictions: the last step in the recommendation process is to convert the prediction back to the specified rating scale as shown in Fig. 4.20


Recommendation Process - Windows Internet Explorer
http://localhost:1984/E-Commerce_Project/Recom_Proc.aspx?n_u=12&n_i=30&n_r=8&f=1&t=10#D
Recommendation Process
Page
Tools
Local intranet
100%

Initial Matrix | Fuzzified Matrix | Normalized Matrix | Users Similarities | Users Predictions

Change Inputs

Defuzzified Predictions

	Item 0	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6	Item 7	Item 8	Item 9	Item 10	Item 11	Item 12	Item 13	Item 14	Item 15	Item 16	Item 17	Item 18	Item 19	Item 20	Item 21	Item 22	Item 23	Item 24	Item 25	Item 26	Item 27	Item 28	Item 29
User 0					5.86		6.04				5.95		6.04	5.86			5.95			5.95		5.77	5.86	5.86				5.86		5.86
User 1	5.59				5.5		5.68			5.5					5.5					5.68			5.5		5.59	5.5			5.5	5.41
User 2						5.05				4.96		5.05				4.96					4.96	4.78	4.96				5.14			4.87
User 3	5.86			5.95					5.77							5.77		5.95		5.86		5.68	5.77						5.77	
User 4					5.41	5.59		5.68	5.5	5.59				5.5				5.59						5.5	5.59	5.41	5.68			
User 5	6.58	6.58												6.49			6.58					6.4			6.49			6.49		6.4
User 6		5.86	5.68	5.86								5.86					5.86				5.68	5.68	5.77	5.77						
User 7			5.05					5.32	5.14					5.14	5.14				5.32	5.23			5.14			5.05	5.32	5.14		
User 8					6.04			6.31		6.04	6.13	6.13				6.04					5.95				6.04	6.04			6.04	
User 9				5.68	5.59				5.59			5.68		5.68	5.59	5.59			5.68		5.5					5.59			5.59	5.5
User 10	5.14		4.96				5.23	5.23													4.96	4.96						5.05		5.05
User 11	5.5		5.32		5.41	5.5				5.41														5.41		5.32		5.41	5.41	5.41

Figure 4.20 Defuzzified Predictions

It should be noticed here that the defuzzified values should not be below the lowest scale rate in our case 1, and it should not exceed the highest value which is 10 in our case. Now if the we want to change the inputs for this process we simply click on the Change Inputs link, or if we want to keep the same inputs but with different generated values we click the refresh button.

4.8. Validation

In order to this website to work properly the website has many validations which are:-

- No two users are allowed to have the same user name.
- The user cannot add the same item to his cart instead he can update the quantity for his product.
- The quantity updated by the user cannot exceed the quantity available in the stock.
- In the administration page the admin cannot add the same option for any specification
- The lifetime for shopping cart cannot exceed seven days, after that if the user does not check out items in his shopping cart, his products will be deleted, and he will be given notification message.
- The rating scale in the recommendation process, the number of users and the number of items should always be greater than zero

4.9. Technology Used

The server side code was implemented using

- ASP.net 2 under Visual Basic.net language using Visual Studio 2005

The database tables and DML operations was implemented using

- SQL Server 2005

The website design and animation was implemented using

- Adobe Photoshop
- FlashMX
- DreamWeaver

4.10. Summary

Chapter 4 is dedicated to the technical implementation of the system proposed in chapter 3, but before that, a general description about how to use the system is given along with screenshots. This system enables the users to navigate available products, search for products based on different criteria, customizing the products according to the customer’s needs, and step by step online ordering, the users also can have their own account to check their shopping cart and to be able to update orders whenever they need. The administrator of the website can insert, update, or delete new products, view the feedback of customers, and give messages to customers according to their purchases. As it was mentioned above for the purpose of this thesis the recommendation process will be done with different input values, and with different rating scales, also each time the rating values and missing ratings will be generated dynamically.

Chapter 5 Evaluation and Lessons Learnt

5.1. Introduction

After designing and implementing the new proposed RS, it’s time to evaluate it, thinking of other methods to improve it, and trying to enhance the system by depending on other information that might be useful to increase its performance, the proposed system emulates other RSs in specific domains, on the other hand it is inferior to other systems from different domains, we can say that the proposed system is distinguishable from others in the domain of flexibility and accuracy, for example the system could use any rating scale to be more precise or less in expressing the users opinions for the available items, also the proposed system could make the predictions for the missing ratings for a large number of users and items at once, this is also good because the users might change their opinions about the items and give different ratings, in this case the only thing we need is clicking the button to generate new predictions with the new provided data, then reflecting those results into recommendations for each user.

The recommendation process that was described in chapter 4 enables us to take unlimited dimensions and scales for the initial users items matrix which can be used to take a clear understanding about the proposed system, and how the results will be different according to the corresponding different user inputs or ratings. As it is the case for most recommendation systems the more data available about users and items the more accurate would be the results, and if the initial data are insufficient the results might be very optimistic or very pessimistic, in general there is no perfect RS because of the challenges that were mentioned above, but there is one system which is better than the other in some situations and worse than others in different situations.

5.2. Discussion

The RSs are still problem rich and research active topics, the challenges in current RSs are already known, still there is no perfect solution, the question is does it worth all this effort, the answer is definitely yes, we can think of RSs as finding exactly what product you need to purchase from thousands or even more available options with tiny effort and time, which is very desirable for each one of us.

This thesis is an attempt to improve the quality of the RSs, the proposed method took benefit from the previous work done in this field, but this time with eliminating some of previous drawbacks, for example in the Pearson correlation if the number of the items both users have rated is 1, then in this case the similarity between the two users is undefined or cannot be defined, this problem is solved in the proposed method in this thesis, also if the ratings provided by users are of the same values the similarity between them could not be calculated, for example if user 1 rated items 1,2,3 with the values 2,2,2 respectively, and user 2 rated the same items with the values 5,5,5 the similarity between them will be inapplicable.

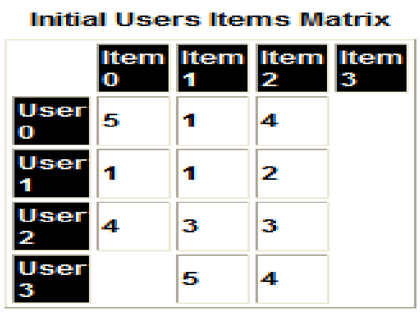
Other than the technical issues that might complicate the recommendation process, there are some problems that need more sophisticated solution, for example customers might be very similar to each other in terms of their taste to the products but they give different rating values, for example user 1 and user 2 have the same opinion about product *a*, but user 1 rated product *a* with the value 7 out of 10, and user 2 rated the same item with the value 10 out of 10, such a problem is usually ignored in recommendation algorithms.

Still there are problems that this thesis doesn’t handle, like the cold start problem which was mentioned early, in this chapter we will evaluate the proposed system by taking many scenarios and see if the results make sense, also an idea that might improve the performance of the proposed system will be mentioned.

5.3. Evaluation

In order to evaluate the proposed system different scenarios should be taken, to illustrate the results

- Assume for simplicity we have 2 users and 2 items, and the users didn't rate the same item(s), in this case the prediction value will be the same as rated value for each user, because in this case the prediction will be the average for each user
- Assume we have 4 users and 4 items as shown in Fig. 5.1.



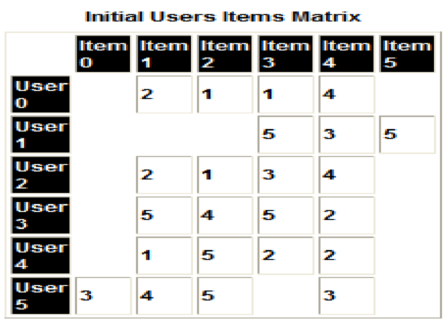
Initial Users Items Matrix

	Item 0	Item 1	Item 2	Item 3
User 0	5	1	4	
User 1	1	1	2	
User 2	4	3	3	
User 3		5	4	

Figure 5.1 Scenario 1

Assume also that we want to predict the missing rating for user 0 item 3 combination, in this case the similarity among users won't be useful because no user rated item 3, so the average will be taken here

- Assume we have 6 users and 6 items, as shown in Fig 5.2, and we want to predict the missing rating for users 5 item 5 combination



Initial Users Items Matrix

	Item 0	Item 1	Item 2	Item 3	Item 4	Item 5
User 0		2	1	1	4	
User 1				5	3	5
User 2		2	1	3	4	
User 3		5	4	5	2	
User 4		1	5	2	2	
User 5	3	4	5		3	

Figure 5.2 Scenario 2

In this case two factors will determine the prediction value, user 5’s average, and the similarity between user 5 and user 1, because user 1 is the only user who rated item 5, and this makes sense. We can have unlimited number of scenarios, but after testing the system it was clear that the more information provided by the user the more accurate are the results.

5.4. Future work

The proposed system in this thesis counts only on the users’ ratings to generate predictions for items that are not seen by the user, actually this is good because we want to relieve the user from providing much information to the system. On the other hand we can perform some additional tasks which will be hidden from the user to improve the system, these tasks could be including the features of the products in the recommendation process. For example if one user found a product very interesting, and there is a new a product which is very similar to the one the user likes, so why not recommending this product, it would be a good idea, but how to include the product features in the recommendation process, below we describe a new idea about how to incorporate the features of the product (notebooks in our case)

Constructing weights matrix

The idea here is to define the most important features of notebooks and give weight for each feature according to its importance.

	Weights										
	0.05	**0.3**	**0.2**	**0.05**	**0.1**	**0.05**	**0.05**	**0.05**	**0.05**	**0.03**	**0.02**
	Model	Brand	CPU	OS	RAM	H.D	Drive	V.C	Network	Battery	Office
I 1	D1	Dell	Intel	XP	DD	Sata	DVD	Neva _da	Wireless	6 Cells	XP
I 2	H1	HP	AMD	Vista	SD	S1	CD	VC1	N1	9 Cells	2007
I 3	S1	Sony	Dual	XP	DD	S2	CD/D VD	VC2	N2	6 Cells	XP
I 4	D2	Dell	Intel	XP	SD	Sata	DVD	Neva _da	Wireless	9 Cells	XP
I 5	H1	HP	AMD	Vista	SD	S1	CD	VC1	N1	9 Cells	2007

Table 5.1 Notebooks' features weights

For example in table 5.1 above to find the similarity between I1 (item 1) and I4 we find the features which are the same for both items and sum their weights, so the similarity between the two items would be 0.82, the closest the result is to 1 the more similar are the items, from the result we have just found we can see that items 1 and 4 are very similar, so if some user likes item 1 he will most likely like item 4, this idea can be added to the previously mentioned recommendation process somehow.

Appendix

➢ Read Me File

This part of the appendix describes the system requirements, the content of folders and subfolders attached in the CD , and how to deal with them.

➢ System Requirements:

First of all it should be mentioned that in order to use the implemented system, and to view the source code, the user should install Microsoft Visual Studio 2005 or later version. The below table lists the system requirements for installing Visual Studio 2005.

System Requirements for Installing Visual Studio 2005	
Processor	Minimum: • 600 megahertz (MHz) Pentium processor Recommended: • 1 gigahertz (GHz) Pentium processor recommended
Operating System	Visual Studio 2005 can be installed on any of the following systems: • Microsoft® Windows® 2000 Professional SP4 • Microsoft® Windows® 2000 Server SP4 • Microsoft® Windows® 2000 Advanced Server SP4 • Microsoft® Windows® 2000 Datacenter Server SP4 • Microsoft® Windows® XP Professional x64 Edition (WOW) • Microsoft® Windows® XP Professional SP2 • Microsoft® Windows® XP Home Edition SP2 • Microsoft® Windows® XP Media Center Edition 2002 SP2 • Microsoft® Windows® XP Media Center Edition 2004 SP2 • Microsoft® Windows® XP Media Center Edition 2005 • Microsoft® Windows® XP Tablet PC Edition SP2 • Microsoft® Windows Server™ 2003, Standard Edition SP1

	• Microsoft® Windows Server™ 2003, Enterprise Edition SP1 • Microsoft® Windows Server™ 2003, Datacenter Edition SP1 • Microsoft® Windows Server™ 2003, Web Edition SP1 • Microsoft® Windows Server™ 2003, Standard x64 Edition (WOW) • Microsoft® Windows Server™ 2003, Enterprise x64 Edition (WOW) • Microsoft® Windows Server™ 2003, Datacenter x64 Edition (WOW) • Microsoft® Windows Server™ 2003 R2, Standard Edition • Microsoft® Windows Server™ 2003 R2, Standard x64 Edition (WOW) • Microsoft® Windows Server™ 2003 R2, Enterprise Edition • Microsoft® Windows Server™ 2003 R2, Enterprise x64 Edition (WOW) • Microsoft® Windows Server™ 2003 R2, Datacenter Edition • Microsoft® Windows Server™ 2003 R2, Datacenter x64 Edition (WOW) Installation of Visual Studio 2005 on the Intel Itanium (IA64) is not supported.
RAM	Minimum: • 192 megabytes (MB) Recommended: • 256 MB
Hard Disk	Without MSDN: • 2 GB of available space required on installation drive • 1 GB of available space required on system drive With MSDN: • 3.8 GB of available space required on installation drive with a full MSDN install or 2.8 GB of available space required on installation drive with a default MSDN install. • 1 GB of available space required on system drive
CD or DVD Drive	Required
Display	Minimum: • 800 x 600 256 colors Recommended: • 1024 x 768 High Color - 16-bit
Mouse	Microsoft mouse or compatible pointing device

The database management system that will be used to store, and manipulate the system's information is Microsoft SQL Server 2005 Mobile Edition. The following table lists the hardware and software requirements for the three SQL Server Mobile environments: development, client, and server environment.

Environment	**Requirements**
Development environment	• Microsoft Visual Studio 2005 • One of the following operating systems: Microsoft Windows Server 2003, Windows XP Media Center Edition, Windows XP Professional, Windows XP Tablet PC Edition, Windows 2000 Professional SP4 or later versions, Windows 2000 Server SP4 or later versions • Microsoft Internet Explorer 6.0 or later versions Required to access SQL Server Mobile Books Online • Microsoft ActiveSync 4.0 or later versions Required to debug and deploy applications
Client environment	• Any device that runs Microsoft Windows CE 5.0, Microsoft Windows XP Tablet PC Edition, Microsoft Mobile Pocket PC 2003, Microsoft Mobile Version 5.0 Pocket PC, or Microsoft Mobile Version 5.0 Smart Phone. • Between 2 MB and 3 MB of available storage space, depending on processor type and components installed.
Server environment	• Microsoft SQL Server 2000 SP3a or later versions • SQL Server 2005 Intel or compatible Pentium 600 MHz or greater processor (recommended processor speed is1 GHz or greater), 256 MB RAM minimum (recommended RAM 512 MB or more), hard disk space 250 MB • IIS Supported on Windows Server 2003, Windows 2000 SP4 or later versions, and Windows XP 120 MB of available disk space on the server • Microsoft ActiveSync 4.0 or later versions Required when you use SQL Server 2005 Management Studio to manage SQL Server Mobile databases on connected devices. • Microsoft Internet Explorer 6.0 or later versions

➢ Folders and Subfolders

The attached CD will contain one folder, four subfolders, and one file. The folder name is E-Commerce_Project and the file name is User's Guide.

➢ E-Commerce_Project Folder:

This folder contains the ASP.NET server pages, the Visual Basic source files, and four subfolders. Below is a description to the subfolders' content

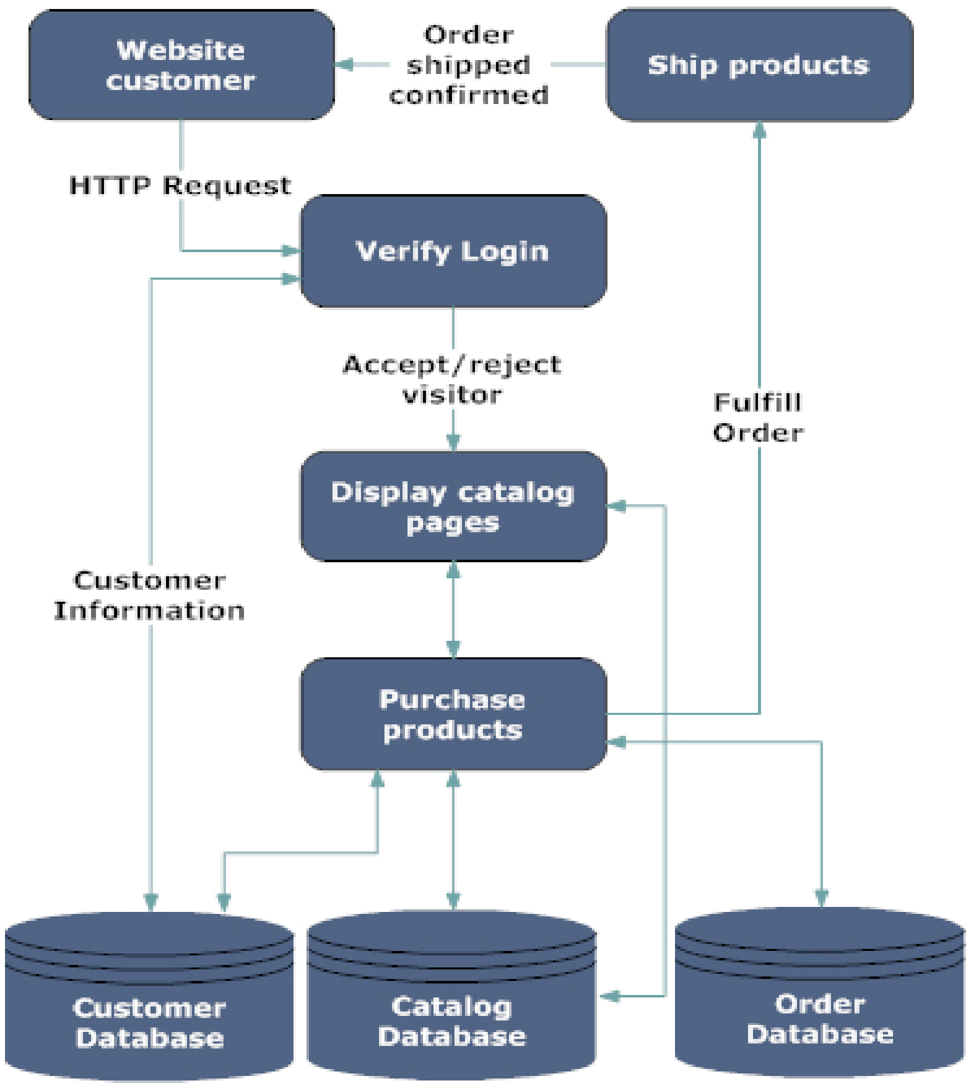
Subfolder	Description
Notebooks_PCs	This folder contains the uploaded images of notebooks and PCs. The administrator of the website will upload the images into this folder
Monitors	This folder contains the uploaded images of monitors. The administrator of the website will upload the images into this folder
images	This folder contains the images that are used in the design and interface of the website.
fonts	This folder contains the fonts that are used in the design and interface of the website.

➢ User's Guide File

This file is a document that guides the user through the navigation of the website. This file also contains many screen shots that illustrate how to use the system, with step-by-step examples. Also in this file there is a brief description about the technology used to implement the system.

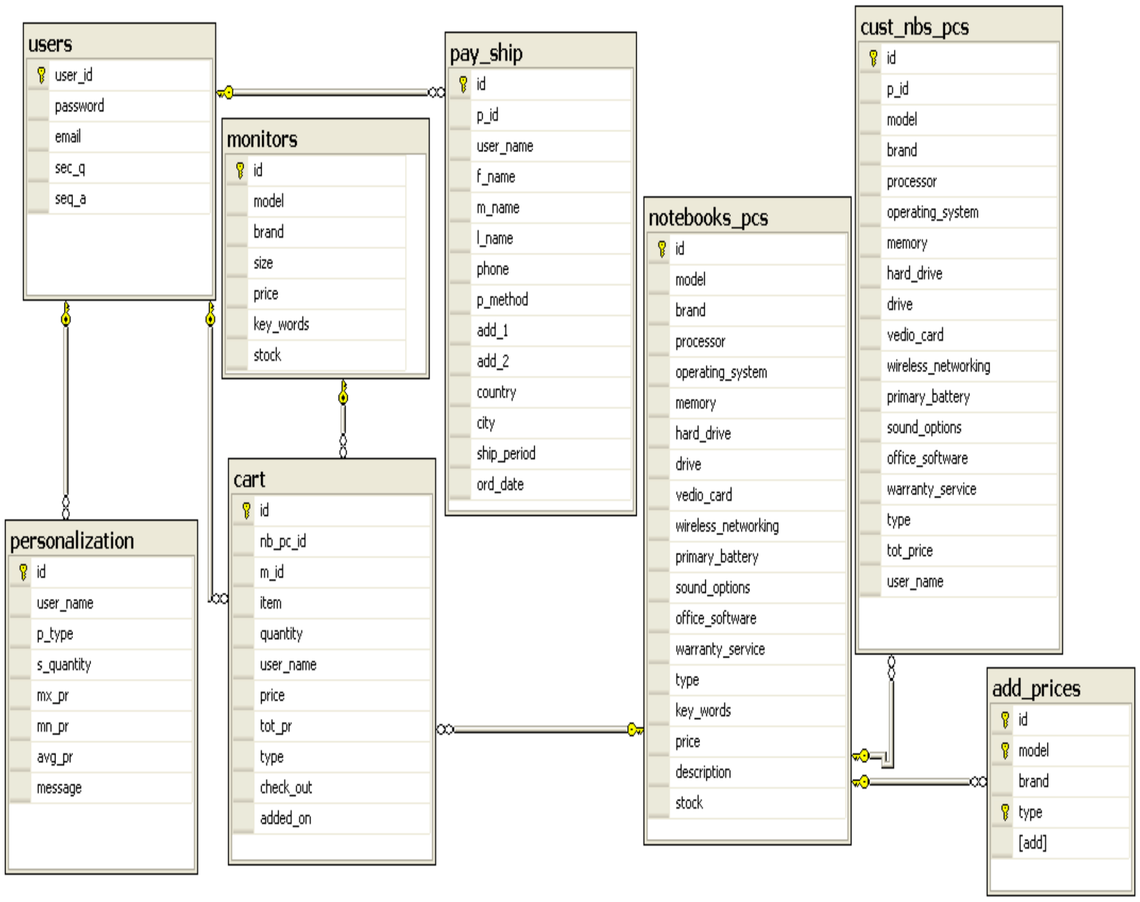
➢ Database Design

➢ Logic Design Diagram



➢ Database Tables and Relations

The below figure shows the used database tables with their relationships, the key notation represents the primary key for each table.



References

1. **ACM.** ACM Recommender Systems 2008. [Online] 2008. http://recsys.acm.org/.

2. *Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions.* **Adomavicius, G and Tuzhilin, A.** 6, 2005, IEEE Transactions on Knowledge and Data Engineering, Vol. 17.

3. **Vozalis, E, and, Konstantinos G.** Analysis of Recommender Systems' Algorithms. 2003.

4. **Press, AAAI.** Intelligent Techniques for Web Personalization and Recommender Systems in E-Commerce. [Online] 2007. http://www.aaai.org/Press/Reports/Workshops/ws-07-08.php.

5. *Intelligent Techniques for E-commerce.* **Prasad, B.** 2, s.l. : Journal of Electronic Commerce, 2003, Vol. 4.

6. *Learning and Revising User Profiles:The Identification of Interesting Web Sites.* **Pazzani, M and Billsus, D.** 1997, Machine Learning, Vol. 27.

7. **Baeza-Yates, R and Ribeiro-Neto, B.** *Modern Information Retrieval.* s.l. : Addison-Wesley, 1999.

8. **Mooney, R.J., Bennett, P.N. and Roy, L.** *Book Recommending Using Text Categorization with Extracted Information.* 1998. Technical Report.

9. *Social Information Filtering:Algorithms for Automating 'Word of Mouth'.* **Shardanand, U and Maes, P.** 1995. Human Factors in Computing Systems.

10. *GroupLens: Applying Collaborative Filtering to Usenet News.* **Konstan, J.A., et al.** 3, 1997, ACM, Vol. 40.

11. *Recommending and Evaluating Choices in a Virtual Community of Use.* **Hill, W., et al.** 1995. Human Factors in Computing Systems.

12. *Empirical Analysis of Predictive Algorithms for Collaborative Filtering.* **Breese, J.S., Heckerman, D. and Kadie, C.** 1998. Uncertainty in Artificial Intelligence.

13. *Item-Based Collaborative Filtering Recommendation Algorithms.* **Sarwar, B., Karypis, G. and Konstan, J., and Riedl,J.** 2001. 10th Int'l WWW.

14. *Recommendation as Classification: Using Social and Content-Based Information in Recommendation.* **Basu, C, Hirsh, H and Cohen, W.** 1998. National Conference on Artificial Intelligence.

15. *Analysis of Recommendation Algorithms for E-Commerce.* **Sarwar, B. M., et al.** 2000. Proceedings of the ACM EC'00 Conference.

16. **Cheeseman, P. and Stutz, J.** *Bayesian classification (Auto-Class): Theory and results.* s.l. : AAAI Press, 1995.

17. **Ungar, L.H. and Foster, D.P.** *Clustering Methods for Collaborative Filtering.* 1998. Technical Report.

18. *A Bayesian Model for Collaborative Filtering.* **Chien, Y.-H. and George, E.I.** 1999. Seventh Int'l Workshop Artificial Intelligence and Statistics.

19. *Using Probabilistic Relational Models for Collaborative Filtering.* **Getoor, L. and Sahami, M.** 1999. Workshop Web Usage Analysis and User Profiling.

20. *A Maximum Entropy Approach to Collaborative Filtering in Dynamic, Sparse, High-Dimensional Domains.* **Pavlov, D and Pennock, D.** 2002. Neural Information Processing.

21. *Collaborative Filtering via Gaussian Probabilistic Latent Semantic Analysis.* **Hofmann, T.** 2003. 26th Ann. Int'l ACM SIGIR.

22. *Latent Semantic Models for Collaborative Filtering.* **Hofmann, T.** 1, 2004, ACM Trans. Information Systems, Vol. 22.

23. *Modeling User Rating Profiles for Collaborative Filtering.* **Marlin, B.** 2003. Neural Information Processing Systems.

24. *Flexible Mixture Model for Collaborative Filtering.* **Si, L. and Jin, R.** 2003. 20th Int'l Conf. Machine Learning.

25. *Recommendation Systems: A Probabilistic Analysis.* **Kumar, R., et al.** 1, 2001, J. Computer and System Sciences, Vol. 63.

26. *Collaborative Learning for Recommender Systems.* **Lee, W.S.** 2001. Int'l Conf. Machine Learning.

27. *Hybrid Recommender Systems: Survey and Experiments.* **Burke, R.** 4, 2002, User Modeling and User-Adapted Interaction, Vol. 2002.

28. *A Personalized TV Listings Service for the Digital TV Age.* **Smyth, B. and Cotter, P.** 2000. Knowledge-Based.

29. *Collecting user access patterns for building user profiles and collaborative filtering.* **Wasfi, A.** 1999. International Conference of Intelligent User Interfaces.

30. **Mooney, R. J. and Roy, L.** Content-Based Book Recommending Using Learning for Text Categorization. *Workshop on Recommender Systems: Algorithms and Evaluation.* 1999.

31. *Using Filtering Agents to Improve Prediction Quality in the GroupLens Research Collaborative Filtering System.* **Sarwar, B.M., et al.** 1998. Conference on Computer Supported Cooperative Work.

32. *Fab: Content-Based, Collaborative Recommendation.* **Balabanovic, M. and Shoham, Y.** 3, 1997, ACM, Vol. 40.

33. **Pazzani, M.J.** A Framework for Collaborative, Content-Based and Demographic Filtering. *Artificial Intelligence Review.* 1999.

34. **Littlestone, N. and Warmuth, M.** The Weighted Majority Algorithm. *Information and Computation.* 1994.

35. **Condliff, M.K., et al.** Bayesian Mixed-Effects Models for Recommender Systems. *Workshop on Recommender Systems: Algorithms and Evaluation.* 1999.

36. *Learning User Interests through Positive Examples Using Content Analysis and Collaborative Filtering.* **Schwab, I, Kobsa, A and Koychev, I.** 2001, User Modeling and User-Adapted Interaction.

37. **Vozalis, E and Konstantinos, G.** *Analysis of Recommender Systems' Algorithms.* 2003.

38. **Aliev R.A., Mamedova G.A., Aliev R.R.** *Fuzzy Sets Theory and its application.* 1993.

39. *A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem.* **Juhn, Ahn, H.** 2008, Science Direct, pp. 37-51.

40. *Robust Fuzzy Clustering Methods to Support Web Mining.* **Joshi, A and Krishnapuram, R.** 1998.

41. **Laudon, K. and Traver, C.** *E-Commerce business.technology.society.* s.l. : Prentice Hall, 2008.

42. **Russell, S and Norvig, P.** Artificial Intelligence Amodern Approach. s.l. : Prentice Hall, 2003.

43. *Trust in Recommender Systems.* **Donovan, J and Smyth, B.** 2005, ACM.

44. *Fuzzy Logic Introduction.* **Hellmann, M.** 2001.

45. *Fuzzy Systems Technology: A Brief Overview.* **Ying, H.** 3, 2000, IEEE Circuits and Systems Society Newsletter, Vol. 11, pp. 28-37.

46. *Scalable Collaborative Filtering Using Cluster-based Smoothing.* **Xue, G, et al.** 2005. ACM SIGIR Conference.

47. **Claypool, M., et al.** *Combining Content-Based and Collaborative Filters in an Online Newspaper.* 1999. Technical Report.

48. **King, D, Viehland, D and Lee, J.** *Electronic Commerce.* s.l. : Prentice Hall, 2006.