NOTE TO USERS

This reproduction is the best copy available.

uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Abdulmajeed Alkhaldi
AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.Sc. (Systems Science)
GRADE / DEGREE

School of Engineering
FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

A Recommender System Using Tag-based Collaborative User Model

TITRE DE LA THÈSE / TITLE OF THESIS

Abdulmotaleb El Saddik
DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

Amiya Nayak

Shervin Shirmohammadi

Gary W. Slater
Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

A Recommender System using Tag-based Collaborative User Model

by

Abdulmajeed Alkhaldi

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the M.Sc. degree in
Systems Science

School of Information Technology and Engineering
Faculty of Engineering
University of Ottawa

Your file Votre référence
ISBN: 978-0-494-74173-3
Our file Notre référence
ISBN: 978-0-494-74173-3

Abstract

Internet users are overwhelmed by a huge media amount available online. Therefore there is a need of an automated way to make compelling recommendations to users according to their needs. There have been many research efforts to reduce that huge amount of content to what the user really needs or prefers. Recommender systems assisting users in easily finding the useful information, are a main research topic that serves this area. According to techniques recommender systems employ, they are mainly classified into three categories: a collaborative-based filtering, content-based filtering, and hybrid filtering. Collaborative filtering relies on the collaboration of users by capturing their judgments on items, and then recommends these items to users with similar taste. Content-based filtering takes advantage of content of a user's preferred items and recommends new items that have similar content. Hybrid filtering takes advantage of both collaborative and content- based filtering and might be in a different ways. No matter what the technique is used, recommender systems require an accurate user model that can reflect a user's characteristics, preferences, and topics of interest. In addition, the systems should take into account users who newly join the systems and thus has presented few opinions, commonly referred to as the cold start users problem. In our research, by leveraging user-generated tags, we propose the topic-driven enriched user model (EM), which is a new way of modeling a user's topics of interest in collaboration with other similar users, in order to improve the recommendation quality and alleviate the cold start user problem. We also present how the proposed model is applied to item recommendations by using locally weighted naive Bayes approach. For evaluating the performance of our model, we compare experimental results with a user model based on user-based collaborative filtering, a user model based on an item-based collaborative filtering, and a vector space model. The experimental results shows that EM outperforms the three algorithms in both recommendation quality and the cold start situation.

Acknowledgements

"To my very loved family: Haya, Saad (my parents) and Bodoor (my sweet wife)."

Special Thanks to:

Prof. Dr.-Ing. Abdulmotaleb El Saddik, *my supervisor.*

Dr.Heung-Nam Ken, *my co-supervisor.*

The Capital Market Authority of Saudi Arabia, *my sponser.*

MCR and DISCOVER labs' members, especially Dr.Atef Alamri, Dr.Mohammad Eid, Eng.Fawaz Alsuliman, Eng.Ali Karimi and Eng.Mohammad Alhamid.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Since the early days of the modern computers, even before the Internet era, the need on the information filtering and retrieval methods was presented [2]. After the Internet has been introduced, information overload on line became a big concern. The Internet contains a huge amount of content and the user has a limited time and attention for that. That emerges different techniques of information retrieval, indexing and filtering. In recent years, social media content is also rapidly groing. For example, as of 2008, YouTube[1] receives 65000 videos daily [3]. This number has increased to be 300000 videos daily as of 2009 [4]. In turn, the need of a solution for filtering out media content relevant to user needs has raised as well. Different Techniques from information retrieval field have been adopted in this field as well as new techniques. *Recommender systems* have been proposed as one of solutions. A typical recommender system is a web application that captures the users preferences and recommend new content to them based on their preferences. Recommender systems goal is to recommend the right content (we call it item) for the right user. Although social media content and e-commerce applications are the biggest targets of the recommender systems, recommender system domains can vary from recommending media content (e.g. videos, music, news etc) to the human relationships recommendation such as recommending new friends on line. Recommendation systems are often classified into three types: contend-based, collaborative and hybrid. Recommender systems have been widely used for commercial purposes such as *Amazon*[2] and research purposes such as *MovieLens*[3].

Content-based recommending approach was inspired by the early information re-

[1]http://www.youtube.com
[2]http://www.amazon.com
[3]http://www.movielens.org

trieval and filtering methods [2]. In this approach, each item's content is extracted to be used for describing that item. The item description is represented in different formats depends on the items themselves or on the design method. For example, for a news articles recommender system, the description might be the whole article text. It can be also some important keywords within the article. Another example is a movie recommender system where an item's description might be the movie's genre, the cast and the movie's script. However, nowadays with web 2.0, social tagging has been used widely for items content description. In that case, each item is tagged by a number of tags (either words or short phrases) that reflect the item's content. A recommender system that uses content-based filtering approach looks at the user's previously viewed items and then tries to find items similar to the viewed items. The similarity of items is based on the similarity of their content descriptions. A common way of calculating the similarity is using cosine similarity[5]. A one obvious question here is : What about a new user who have not viewed any item yet? The answer may depend on the system design; however, a common used ways is to ask the new user to fill up a survey about their favorite topics or to recommend popular items, which preferred by most of the users.

Collaborative filtering approach, on the other hand, mainly relies on the collaboration between users. Unlike machines, human are able to distinguish between items that have similar descriptions even though they actually belong to different fields. For example, an item in amazon.com tagged by "Red,Hot,Chili,Peppers" might be an Indian cuisine cookbook or a music CD of a famous rock band under that name. If amazon.com uses a pure content-based filtering, it might recommend variety of spices and herbs to customer X who bought that music album. However, a recommender system that uses a collaborative filtering approach considers the history of user X as well as the history of like minded users of user X. In addition, it also recall those users' judgments about items , called ratings. Each user can give their feedback about an item by giving ratings. Rating can be in different formats depending on the system design and requirements. A common format is the 5 stars where the user gives a number of stars to an item based on how much he liked that item. In this way, the system is capturing users' topics of interest by knowing which items are preferred by which user. Then, the systems recommends new items to a user based on the items preferred by other users with similar taste. Although collaborative filtering is more efficient than the content-based one, it suffers from *cold start users problem*. The cold start problem describes a new user that joins a recommender system and has presented few opinions (e.g., rating and tagging). With these situations, it is hard for the system to capture the user preferences and consequently

to make high quality recommendations.

Both approaches (i.e. Content-based and Collaborative filtering) have their own advantages and weaknesses. For that reason, a hybrid approach is introduced. A hybrid approach combines collaborative and content-based filtering in order to reduce each one's limitation. Different hybrid recommender systems combine the two approaches in different ways and levels. Many researchers and commercial systems preferred to use the hybrid approach, the current focus of our research. Hybrid approach, in general, leads to better recommendations [6] especially when combining collaborative filtering with content-based filtering [7]. We propose a new approach of hybrid recommender systems in that we use both collaborative and Content-based characteristics by taking advantage of ratings and social tagging for modeling users' interests.

1.1 Motivation

Social media has been changing the way people find information, share knowledge and communicate with each other. This social phenomenon has transformed the masses, who were only information consumers via mass media, to be producers of information. However, as rich information is shared through social media sites, the huge amount of information that has not previously been available is increasing exponentially with daily additions. In tune with that, it is becoming increasingly more difficult for users to find the most attractive items. Therefore, recommender systems require more accurate a user model that can reflect not only users preferences but also a new type of social media. In recent years, a number of modern services, such as Flickr, YouTube, Delicious, Facebook and so on, allow users to freely annotate their items with any kind of descriptive words, so called tags. One of our motivations is the current trend of user-generated tags. Since users tend to actively use tags to annotate items that they are interested in, as well as, a set of aggregated tags on an item is rich and compact enough to characterize and describe the same main concepts of the item [8] [9] [10], a user model can profit by those tags in addition to ratings. In general, ratings of a user for items reflect how relevant or interesting the items are to him/her. In addition, user-generated tags of an item are concise to the users understanding, and hence, capture content of the item. Therefore, it is worth examining how we can connect the users ratings and the tags assigned by other users in respect to the item. As mentioned before, a notable challenge in recommender systems is how to solve the cold start users problem. With respect to the cold start users, they should be encouraged to continuously provide their opinions because they

do not have enough rating information. However, inaccurate recommendations from the insufficiency of the users historical information lead them to undermine the credibility of the system, and thus, cause their deviation from the system. Therefore, a differentiated strategy is necessary to improve recommendation quality for cold start users. To enrich a model for those users, it might be useful to enrich some tags from other users with similar tastes. Besides the cold start users, general users may also obtain benefits in terms of diversity. If a user model exclusively relies on tags which are used by a user or annotated in his/her rated items, it is hard to recommend novel items different from anything the user has rated before [11] , As for social tagging, users assign diverse tags to items, that is, users have multiple interests. Therefore, when building a user model, we should take multiple interests of users into consideration. Since user-generated tags can provide a high-level abstraction on content of an item, the aggregated tags can be regarded as a particular topic for the item. In this sense, if multiple tags are frequently annotated together with each other in highly rated items of a user, the user would be interested in a particular topic described by the co-occurred tags. Moreover, the user is likely to prefer similar topics that other users with similar tastes are interested in. To address the discuss issues, in this thesis, we introduce a new method of building a user model that can represent a users diverse preferences and thus can be exploited to recommender systems. We connect tags and ratings as a way to infer a users topics of interest. In addition, to provide the user model with more diversity, valuable topics in terms of both the user likes and dislikes are enriched in collaboration with other similar users.

1.2 Problem Statement

Internet users are overwhelmed by the huge amount of social media content in the Internet. Recommender systems help the users to overcome that overwhelming. However, they suffer from the cold start and overspecialization problems. A recommender system should be able to recommend appropriate items to users with non-clear topics of interest. In addition, a recommender system should be able to recommend diverse items to a user. We would like to improve the recommendation quality especially for cold start users as well as to recommend diverse items. By using social tagging for items content description and using users' explicit items ratings as users' judgments on items, we create a hybrid recommender system. In addition, we model user's topics of interest using social tags and we consider both relevant and irrelevant topics of interest. By enriching user model

using similar users' models, we aim to reduce the cold start users problem, increase the quality of the recommendations and recommend diverse items.

1.3 Thesis Contribution

In this thesis work, we have the following contributions

1. Design and development of a collaborative user model by leveraging user-generated tags in that the model reflects topics of interest with diversity.
2. Design and development of an algorithm to enrich the user model in collaboration with other similar users.
3. Design and development of an algorithm to identify two sets of similar users with user-generated tags in terms of both relevant topics and non-relevant topics.
4. Design and development of a recommender system incorporated with our user model to enhance the recommendation quality particularly for cold start users.

1.4 Author's publications

1. Heung-nam Kim, Abdulmajeed Alkhaldi, and Abdulmotaleb El Saddik. Collaborative User Modeling with Tags for Tag-based Recommender Systems. Submitted to ACM Transactions on Intelligent Systems and Technology (ACM TIST) journal.
2. Md. Abdur Rahman, Abdulmajeed Alkhaldi, Jongeun Cha, and Abdulmotaleb El Saddik. Adding haptic feature to youtube. In Proceedings of the international conference on Multimedia, MM 10, pages 1643 – 1646, New York, NY, USA, 2010. ACM.

1.5 Thesis Organization

This thesis is organized as follows:

- Chapter 2 provides background information and related work. In the background information we cover the literature review of recommender systems' concepts and

techniques including collaborative filtering, content-based filtering, cold start problem, social tagging and naive bayes. We also review some studies closely related to our work.

- Chapter 3 provides detailed descriptions of the proposed system: Recommender System using Tag-based Collaborative User Model. In this chapter we illustrate the system architecture and design. We also show how to build the user model with collaboration from other similar users and how to apply our model to item recommendations via a probabilistic approach.

- Chapter 4 covers the system implementation. We will briefly talk about the data set and some technical issues regarding it. Also, we will demonstrate our database design and the system's modules designs and implementation as well as the implemented system's screen shots.

- Chapter 5 provides evaluation metrics (users Precision and items Ranking Scores) and some benchmark algorithms for recommender systems. We test our system using the mentioned measures. Also, we present the performance of our approach through experimental evaluations in comparison with earlier work. We analyze the test results in details.

- Chapter 6 summarizes the current work as well as presents future vision of the research work.

Chapter 2

Background Information and Related Work

2.1 Recommender Systems

Recommender system research area has appeared in the mid-1990s. Recommender systems nowadays are basic tools in e-commerce or information access applications [11]. Recommender system takes people recommendations of items as an input. Then it accumulates recommendations from different users in order to deliver items to the right receivers . Resnick et al. expanded the term to include " any system that produces individualized recommendations as output or has the effect of guiding the user in a personalized way to interesting or useful objects in a large space of possible options "[12]. In a typical recommender system, a relationship between users and items is identified based on the user rating. This is known as user-item matrix. In normal cases, user does not rate all the items. Therefore some cells in user-items matrix are empty. One of main tasks of recommender system is to predict the values of those empty cells based on the rated items by a certain user and other similar taste user ratings. Then, the recommender system will recommend the non-rated item(s) in a descent order of the expected rating(s) [1]. For example, Figure 2.1 shows a user-item rating matrix in a movie recommender system. Let us assume that we only have four users and five movies as in the figure. Apparently, users Bodoor and Nami have a similar taste and Nami has not rated the movie *Inception*. However, since Bodoor has already rated the same movie with a high rate: 4, the recommender system would expect Nami to give a similar rate to the movie [11]. That is what a "typical" recommender system works;however,

there have been a lot of researches in recommender systems techniques since the mid nineties. Two emergent paradigms have appeared since then. Content-based filtering

	Inception	Avatar	Iron Man	Mother
Abdul	5	1	1	5
Nami		5	2	4
Bodoor	4	4	1	4
Sandro	3	4	2	

Figure 2.1: Empty cells in a user-item rating matrix of a movies recommender system[11]

and Collaborative filtering are the two most popular techniques that are used in recommender systems. Other techniques exist such as knowledge-based, utility-based and demographic filtering. Furthermore, there is also the hybrid method which combines two or more filtering methods [6].

2.1.1 Content-based Filtering

According to Herlocker et al. content-based filtering is the selection of "the right information for the right people by comparing representations of content contained in the documents to representation of content that the user is interested in" [13]. Content-based recommender systems have been derived from information filtering systems. In fact, they are a special case of information filtering [14]. In order to recommend items to the a user (we call him target user), a content-based recommender system should analyze the content of items to be recommended such as the content of a book, the script and the cast of a movie and so on. In case of items that do not have textual content such as clothes, the recommender system analyze the item's description. Then it should look at the target user's previously preferred items and try to find similar items in order to recommend them. For example, in a newsgroup, if a user read a post contains the term "sport cars", he is probably interested in other article containing the same term so the system will try to recommend other articles that contain the same term. In order for the system to track the user's previously preferred items, a user profile should be built. User profile contains user's preferences which are either extracted from user's activities history such as viewing items, purchasing items etc or from a model constructed by the user himself through filling up some questionnaires about his own topics of interest. The profile, which contains user's preferred topics,is being used by the system when it comes

to recommend new items Simply the system selects some new items by comparing their content with the user's profile content The top matching items will be recommended [15]

2.1.2 Collaborative Filtering

The term *Collaborative Filtering* (CF) was introduced since around a decade even though the concept itself has exist for hundreds of years in human history Humans share their valuation of goods among each other According to J Ben Schafer [16], *Collaborative Filtering* is " the process of filtering or evaluating items using the opinions of other people" Collaborative filtering is considered as one of the most powerful techniques used for recommender systems [17] Systems that recommend items using *Collaborative Filtering* are often called *Automated Collaborative Filtering* (ACF) an ACF system collects users' ratings for items in a certain domain then tries to find other users with a similar taste Then the system uses that information to suggest new items for users [13], predict a user's judgment on a certain item or restrict recommendations on certain group of items [16] A famous early example of an ACF system is the GroupLens project on Usenet news group by Konstan et al [18], which was introduced on 1992 In Usenet project, users give ratings to the news groups articles and then get recommendations for new articles [18] The same group has also developed an ACF system for movies and they called it Movielens [13] In this system, users give ratings to movies and receive new movies recommendation as well as a prediction of what a user rating for a movie would be [19] Ringo system by MIT Media-Lab is an other example of ACF system which recommends music for users and help them to keep in touch with other users who have the same taste in music [20]

2.1.3 Hybrid Recommender Systems

In order to improve the recommendation quality, the performance and other factors of the the recommending process, two different filtering techniques can be combined into hybrid filtering Burke [6] has surveyed the existing and the possible hybridization of filtering techniques Most of the times, collaborative filtering is being combined with another technique in order to overcome some of the weaknesses of both technique For example, combining collaborative filtering with content-based filtering, which is common, helps to reduce the overspecialization problem of the content-based technique as well as the sparsity problem in the collaborative filtering method as we will cover later in this

chapter. Hybridization can be in several ways. First, the weighted method in which items rating results of different filtering technique is combined as a single score used to give the recommendation. Second, the switching method in which the recommender system switches between different filtering techniques based on the situation. Third, the mixed method in which the recommender system uses different filtering techniques at the same time and present the results together. Fourth, the feature combination method in which one recommendation technique is used; however, the knowledge is derived from different knowledge sources and it is given as in input to that algorithm. Fifth, the cascade method in which the filtering process is done through two stages: coarse filtering using a certain filtering technique (stage 1). Then using another filtering technique to refine the results (stage 2). Sixth, the feature augmentation method in which a filtering technique is used to filter items. Then the result is also filtered using another technique. Seventh, the meta-level technique in which a filtering technique is used to create a model. Then this model is used as an input to another filtering technique [6]. However, some researchers classify the hybridization methods in less details as in Adomavicius et al. survey which classify them into only four classes [11].

2.1.4 Users' feedback

In order to capture users' preferences about some items, recommender systems facilitate users' feedbacks. User feedback might be implicit so the system is tracking users' activities and, by analyzing them, infers users' preferences [21]. One advantage of the implicit feedback is its low cost to the user. However, the inferred rating data from tracking users activities might be insufficient. For example, if the time spent at a product's page is taken as implicit rating, a user might spend a long time to read about a product then found it not suitable for her. In that case, it is inappropriate to say that user liked that product [16]. On the other hand, explicit rating is the most accurate when it comes to capturing users' preferences [22] [16]. Rating scales for rating items are vary based on the recommender system design requirements and goals. In general, there are three common rating scales: Unary, Binary and Integer. Unary scale is either liked an item or just does not know anything about it while the binary scale is either the user liked an item or disliked it. The binary item has been used by YouTube [1] and Yahoo! news [2]. Finally, the integer scale which allows a range of values between "I like" and "I do not

[1] http://www.youtube.com

[2] http://news.yahoo.com

like" which allows the user to express different level of ratings. The scale values might be in any range such as 1–5 or 1–10. The numeric values might be masked with word expressions such as "bad" for the value 1 and "great" for the value 5 and so on. This kind of scale is used by Amazon [3] and IMDb [4] which is owned by Amazon.

2.1.5 Social Tagging

Social tagging is defined by "the process by which many users add meta data in the form of keywords to shared content" [23]. Tags helps its creator to organize their items and helps the other users for either browsing or judging items since a tag can give an opinion such as "great" or "bad". In addition, tags can be used to describe items content. In fact, tags founded to be an efficient way to describe items content in for recommender systems due to their simplicity, multiple usability and popularity [24]. Also, tags are relatively cheap to create and to make use of. In some literatures, as we will cover in the related work section, tags by themselves can be directly used for recommendation [25] [24]. However, with a huge amount of items and tags and with the existing of misleading tags, there is a need to use rating information along with tags; when the social tagging is in free form (i.e. free text), the problem of tags noise arise. Many tags become misleading and\or not useful such as synonyms tags, personal tags or spammers tags [25]. Therefore, collaborative tagging is used to enhance the collaborative filtering recommender systems which makes it hybrid in that case since the items content are involved. In such systems, user can annotate items with tags and rate the items at the same time. Tags are treated as items content and ratings are treated as items judgments which results in a collaborative filtering\content-based filtering hybrid recommender system. When collaborative filtering is extended to include collaborative tags, the user-item rating matrix shown in figure 2.1 can be combined with a new matrix of user-tag. This process will result in three dimensional matrix as shown in figure 2.2 [1].

2.2 Naive Bayesian Classification

Recommender systems use several methods for classification and prediction. Classification is a very basic task in recommender systems. One of the famous method for

[3] http://www.amazon.com
[4] http://www.imdb.com

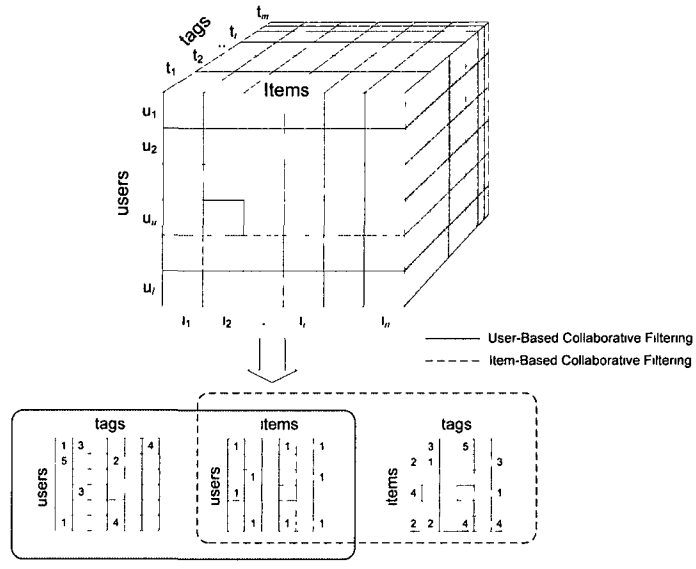

Figure 2.2: Involving tags in order to enhance the collaborative filtering [1]

classification is called Naive Bayesian Classifier. Naive Bayesian classifier (NBC) is simple yet powerful classification method. It is based on Bayesian Classification from Bayes' theorem which is represented by equation 2.1.

$$P(H \mid X) = \frac{P(X \mid H)P(H)}{P(X)}. \tag{2.1}$$

Where $P(H \mid X)$ is called the *posterior probability*, of H conditioned on X and similarly is $P(X \mid H)$ which is the *posterior probability*, of X conditioned on H. The probability $P(H)$ is is called the *prior probability*, of H. Like any classification method, in NBC, an algorithm learns from a data set of training data that consists of records of attributes assigned to classes labels [26]. Each record is called tuple and each tuple is assigned to a class by a class attribute. More formally, a tuple X is represented by attributes vector with n attributes $X = (x_1, x_2, \ldots, x_n)$. A set of m possible classes for a tuple X are represented by $C_1, \ldots, C_m$. A tuple X is assigned to a class C by the class attribute which is used for categorization [27]. NBC calculates the probability of of a tuple X being of a class C. That eventually gives ranking to more than one possible class. The class with the highest ranking (probability) would be selected for labeling a tuple [26]. More formally, in order for tuple X to belong to Class C_i, the posterior probability $P(C_i \mid X)$

should satisfy the condition in equation 2.2

$$P(C_i \mid X) > P(C_j \mid X) \text{ for } 1 \leq j \leq m, j \neq i. \tag{2.2}$$

By applying Bayes' theorem in equation 2.1, we have the following equation

$$P(C_i \mid X) = \frac{P(X \mid C_i)P(C_i)}{P(X)}. \tag{2.3}$$

In order to maximize $P(C_i \mid X)$ in equation 2.3, only $P(X \mid C_i)P(C_i)$ should be maximize since P(X) is constant for all the classes. In addition, there is a naive assumption is applied that says all the prior probability have the same value. More formally,

$$P(C_1) = P(C_2) = \ldots = P(C_m)$$

Therefore, we need only to maximize $P(X \mid C_i)$. and finally we get the formal naive bayesian classification method represented in equation 2.4.

$$P(X \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i) = P(x_1 \mid C_i) \times P(x_2 \mid C_i) \times \ldots \times P(x_n \mid C_i). \tag{2.4}$$

NBC assumes the independence of attribute values of the same class. For example, a customer would considered a member of a Class "big screen TV buyer" if he has the attributes: male, middle-aged, sports fan and high-income. Although those attributes are dependent on each other, NBC method will look at them independently when trying to classify a customer as a "big screen TV buyer". When calculating the probability of a customer being of a class "big screen TV buyer", the mentioned attributes will treated as independent factors and therefore the method is considered "naive". Yet, besides being simple to implement, NBC method has a high accuracy level as well as fast performance especially when applied to huge datasets [27].

2.2.1 Locally Weighted Naive Bayes

Many techniques have been introduced for improving Naive Bayesian Classifier performance especially for areas where the attribute independence assumption does not appear to be reasonable. Many of the methods tend to reduce the naivety of NBC by involving another method of classification. For instance, Locally Weighted Naive Bayes (LWNB) has been introduced by Frank et al. [28] in order to improve the performance of NBC. By using a lazy learning method, LWNB builds a naive Bayesian model for the input

tuple and weight the model using the available training data around the new tuple The selection of the training data is data-dependent and based on the k-nearest neighbor algorithm The training data that used to build the new model is called the neighborhood since they have been selected specifically for their high weight or in other words for their closeness to the input tuple LWNB has successfully improved the NBC method [28]

2.3 Related work

Recommender systems have been researched for decades Research on recommender systems covers the different techniques of recommender system and different aspects regarding each one of them As we have covered before,recommender systems are commonly classified into three types in terms of information filtering method Content-based, Collaborative-based and Hybrid First, the content based recommender systems This method has been derived from the information retrieval and so its evaluation measures [11] There have been a lot of researches about the content-based recommender systems Recently, the researchers are focusing on how to learn a user's model (i e preferences) in order to give it as in input to the system along with a set of new items and get the recommended items as an output The recommended items are commonly given as the top N similar items to the the target user's model [15] One algorithm for learning a user's model is *Nearest Neighbor* which is the k most similar new items to the user's profile This algorithm has been used by [29] and [30] for recommender systems and by [31] and [32] for text classification Another algorithm is the relevance feedback algorithm (also called Rocchio's algorithm) which first is introduced by Rocchio [33] for information retrieval and has been used for content-based recommendation later As its name says, relevance feedback algorithm collects the users' feedbacks (ratings) about the recommended items and deploy the feedbacks to improve the next recommendations It is used by [14], [7] and [34] On the other hand there are several algorithms that follow probabilistic classification methods The most commonly used one is the naive Bayesian classifier which we have covered earlier in this chapter Naive classifier is preferred for its exceptional performance [15] Many researchers, including this thesis, have adopted this algorithm such as in [35], [36] and [26] Content-based filtering recommender systems have some shortcomings First, some content are not easy to be analyzed by an automated process Another side of the same coin is the incapability of capturing the user's taste For example, it hard to analyze the content of a movie without involving human's brain,two movies content might be too similar while a user likes one of them and does not like the

other one. Another example is when two articles are too similar in topic however one of them is well written, in the user's point of view, and the other is not. The problem of content analysis is common in content-based recommender systems especially in movies and articles domains [13]. Second, there is what so called the overspecialization problem. Content-based filtering keeps recommending items that are similar to user profile's items. That means, the content-based recommender system can only recommend items that are similar to the items that user has already experienced before. Therefore, the user might ended up getting recommendation from the same category while the diversity feature is always desirable [11]. The overspecialization of the content-based recommender systems makes then unable to provide serendipitous recommendations—recommending qualified items that are not similar in content to what the user has in his profile [13]. Third, content-based recommender systems suffer from the new user problem that is so called the cold start user problem. When there is a new user without much information in his profile, the recommender system is not able to figure out the users interest and therefore not able to give accurate recommendations for that user [11].

To cover up for content-based filtering drawbacks, researchers introduced the collaborative filtering method. As we covered earlier in this chapter, collaborative filtering method collect users' feedbacks about items and use them to recommend new items based on items preferred by similar taste users. By getting the human judgments involved, collaborative filtering method do not need more analyzing of items content since it has been analyzed and rated by real users. This way makes capturing items quality and user's taste easier and more efficient[13]. In addition, collaborative filtering based recommender systems are able to provide serendipitous recommendations; they recommend items from similar taste users without looking at their content. By focusing on user-user similarities instead of item-item content similarities, collaborative filtering method is able to solve the overspecialization problem [13] [7]. There have been a lot of researches about collaborative filtering based recommender systems. Generally, collaborative filtering methods are divided into two categories based on the technique being used: *Memory-based* (*aka heuristic-based*) which requires the users, ratings and items data to be in memory all the time and *Model-based* which periodically computes and learn users models and patterns offline [5] [16].

Memory-based algorithms use the whole previous rating data of users in order to predict the new items ratings of the target user. Those users are the most N similar users to the target user who have rated the candidate items of the target user. For memory-based technique, researchers in this category are focusing in the following as-

pects [37] First aspect is similarity computation which concerns how the user-user or item-item similarity is calculated Second aspect is the prediction and recommendation computation which concerns about how to predict the ratings of the target user for the candidate items by looking at the ratings of similar users for those items This aspect is used by recommender systems that are interested in rating prediction and\or use the prediction for recommendation Third aspect is top-N recommendation calculation, in which the system gives scores to the candidate items based on the target user's and his similar users' tastes Then, the recommender system recommend the top N items [38] [37]

Regarding the first aspect, *cosine-based similarity* approach was used by this thesis as well as [5], [38], [39], [36] and [24] In that approach, each user, item is converted to a vector and then the similarity is the cosine of the angel formed by each two vectors [40] Another common similarity calculating approach is called the *correlation-based similarity*, in which the similarity between users or items is calculated using *Pearson* correlation approach or any other correlation based approach [37] This approach used by many researchers including [41], [42], [43] and [20]

For the second aspect, which is the most critical step in collaborative filtering based recommender systems, a numerical value of how much rating the target user would give a candidate item is calculated The predicted item rating scale is the same scale of normal rating (e g 1–5) In order to calculate the predicted rating, a set of most similar users (neighbors) to the target user are collected then a weighted average of their ratings for the candidate item is used to calculate the item rating prediction of the target user [37] [13] [38] Several ways of calculating the the weighted average exists For example, the simplest and most naive way is to take the average of the neighbors' ratings for the candidate items as used by [44] A simple weighted average is used by [38] for prediction In addition, some research added what they called "Boosted similarity" to the simple weighted average that used by [38] [45] Some other research included the target user similarity to the candidate item along with user-user similarity between the target user and users who rated the candidate item in the prediction calculation [41]

However, calculating the predicted rating does not necessarily means that the system is ready to make a recommendation While predicting items ratings and recommending items are two main tasks of the collaborative filtering based recommender systems, the two tasks have different requirements If a certain system gives good prediction that does not necessarily mean it would give good recommendations [16] Item recommendation is the third aspect researched in this field and it is being referred as *Top-N Recommenda-*

tions. The recommendation process can be done using different algorithms; however, in general it is the process of analyzing the user-item matrix, determining the relationship between user and items then use it to recommend a set of items with different scores. Items then are ordered based on their scores and introduced as a recommendation list. Recommendation algorithms are divided into two categories: *user-based* and *item-based.* The former works by collecting the k nearest neighbors of the target user, collect their rated items, weight and average them then use them as new recommendation to the target user [37] [46]. Many researchers have adopted this algorithm including [18], [43], [7] and [47]. However, this kind of algorithm faces some practical challenges. First, calculating user-user similarities is time consuming sepecially in e-commerce systems when there are millions of users. Second, is what so called the sparsity problem. In real life systems, rating data of the users is sparse; Many users do not have enough common items with other users or do not have common items at all [16] [37] [46]. Therefore, the *item-based* algorithms have been introduced to address the *user-based* problems. Item based algorithm works in a similar way to users-based however, it concerns about item-item similarity. The algorithm, for each item, collect all the similar items, identify the k most similar items, then take a subset of the k most similar items as a candidates for recommendation after excluding any item that has been already seen by the target user [46] [37]. Some researches have adopted this algorithm such as [38] and [46]. However, item-based algorithms might lead to suboptimal recommendation when a group of items distribution is different from the distribution of its items individually [37] [46].

In model-based approach, the recommender system learns user model and complex patterns based on ratings training data of for that user. Then, it uses the model to predict user's ratings for new items via a probabilistic approach. Different machine learning algorithms can be used to learn a user's model such as clustering and Baysian based algorithms. These algorithms are also known as the probabilistic models.We have covered the naive baysian and the weighted naive baysian model earlier in this chapter. Clustering algorithms consider the recommendation problem as a classification problem; the similar items and the similar users are clustered into classes. Each cluster, which contains a number of users and items, is treated independently by the recommendation process. That reduces the number of items and users required for processing and therefore the time required for processing. Then, the recommending of an item for a certain user is to determine the probability of that item being in that user's cluster given the cluster users' and items information [48]. The similarities between items or users is calculated by of the similarities methods we mentioned under the memory-based algorithms. Clustering has

the advantage of the faster recommendations generating, however, the recommendation quality is lower than the other algorithms [37] Clustering has been used at different levels It might be either used for model based as in [49], [50], [51] and [52] or for a partitioning algorithm for a memory-based algorithm such as in [48] and [53] Also, clustering can be used along with baysian network as in [5] However, in this work, we are not focusing on the clustering algorithms since we have adopted the weighted naive bayesian approach in our user modeling process

Both collaborative and content-based filtering have their advantages and disadvantages Therefore, the need of taking both technique into one recommender system has raised That is called hybrid technique Many researches have covered several ways of combining collaborative and content-based filtering We have listed the different possible types of hybrid recommender systems earlier in this chapter In addition, R Burke [54] [6], Su et al [37] and Adomavicius et al [11] have surveyed the different possibilities of combining different techniques to make a hybrid recommender system They have many differences in the way they classify and name the different hybrid recommender systems, however, they all agreed for the basic concept of the hybrid recommender system which is using more than one filtering technique in order to deliver the recommendations [6] [37] [11]

However, the narrowed related work in our case is the hybrid recommender system that is adopting the content-based filtering technique while involving some collaborative filtering features The content in our case is the social tags since they are efficient for content description as we covered earlier in this chapter Among the research on hybrid recommender systems that has been going on for decades, most of the recent research is about tags recommendation instead of using tags for recommendation Recommending tags to a user to annotate an item is different from and cannot be used to recommend an item directly[22] In addition, there is no much research about using tags information as items content Zhen et al [55] introduced *TagiCoFi* that stands for tags informed collaborative filtering, a recommender system that integrates tags information into a collaborative filtering model However, they do not use tags as a content description for items, instead they consider each user's contributed tags That means, the only tags considered when recommending a new item to a user are those created by that user for his previous items All the tags given by other user to the new items are not considered In addition, they have focused only on items rating prediction instead of items recommendation On the other hand, Sutter al [1] introduced user-item-tag relationship and used it in a memory based collaborative filtering However, due to the

poor quality of the used tag, the recommendations were not much accurate. In their earlier work, Liang et al. (2008) used the relationship user-item-tag [42] and used tag information to recommend items in a collaborative filtering model. Later, H.Liang et tl. (2010) [24] used tags for items descriptions and at the same time, they take into consideration who tags what. In fact, they consider all the possible relationships between users, items and tags (i.e. user-item, tag-item and user-item similarities). They add more tags to an item based on tags of similar items. Also, they add more tags to a user based on tags of similar users. However, they considered tags as an implicit rating; they did not use explicit users' ratings or any user's activity. A similar work that does not consider users activities is by Li et al. [10] which uses discover users' interest based on patterns of their co-occurring tags. Ghazanfar et al. [45] proposed a movies collaborative filtering based recommender system that uses movies content descriptions (including tags, plots and all the movies information provided by IMDB), users' ratings and movies' demographic information to model the users' interests and make recommendations. However, they focus on ratings prediction and they consider the processing of huge amount of textual content that describe each movie. Jomsri et al. [56] proposed a tag-based research papers recommender system in which they used tags as a content description for papers (items). They analyze each user's tagged\uploaded papers in order to extract user's patterns and use it to describe user's preferences. Then the recommendation is done based on the similarity of user's preferences and the new items. In fact, the proposed framework is pure content-based; they only user activities they considered are papers uploading and tagging. However, they did not consider the explicit user rating nor the implicit one. Wnag et al. [57] proposed an approach to integrate tags to the collaborative filtering approach. They used both user's tags and ratings to capture a user's topic of interest and to build user's neighborhood. However, the tags considered for a user are those tags used by that user. There is no tags sharing among the users. In addition, the approach is trying to predict user's ratings instead of finding the top N recommendations.

Chapter 3

Tag-based Recommender System Using Enriched User Model

3.1 The Proposed System

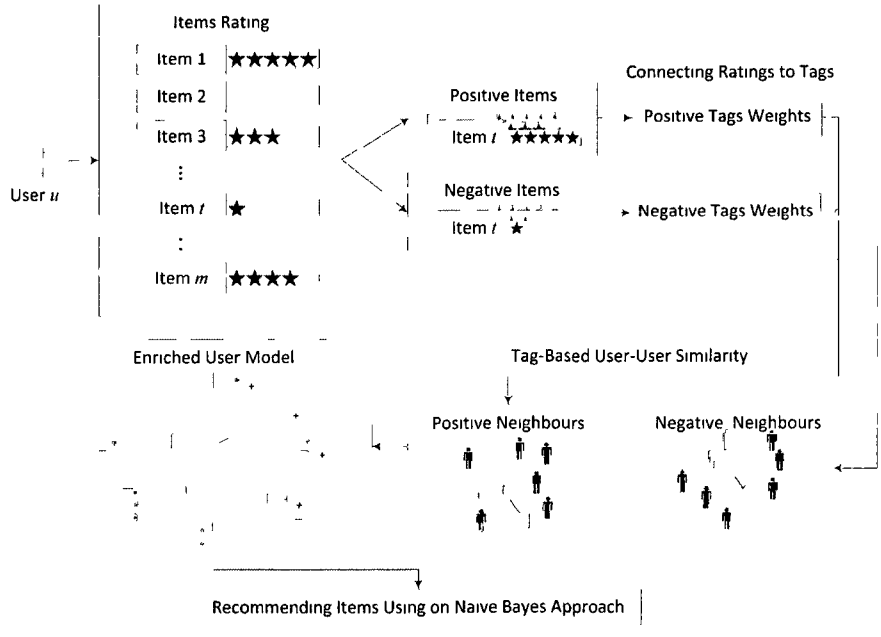

Figure 3.1: The proposed system at a glance

We propose a tag-based recommender system using collaborative user model. Figure

3.1 shows the system at a glance. We collect users' ratings of tagged items. After rating items, each user ends up with positive (i.e. relevant) and negative (i.e. irrelevant) items. The rated items' tags are classified into positive and negative tags according to their items. Mining frequent patterns of both positive and negative tags constitute the initial users models. Users similarities is calculated then based on users initial models similarities. Based on similar users' models, each user initial model is enriched. Finally, we make new items recommendation using naive bayes approach based on the enriched user's model. In this chapter, we go over every element of figure 3.1 in details.

3.2 Explicit User Rating

In our recommender system, we have several items that are provided to users. However, each user has a different preferences and therefore different impression for different items. An item preferred by user 1 might not be preferred by user 2 for example. In order to gather user feedback about an item, we facilitate the items rating process. Let I be a set of m items where $I = \{i_1, i_2, \ldots, i_t, \ldots, i_m\}$ and let U be a set of n users where $U = \{u_1, u_2, \ldots, u_j, \ldots, u_n\}$. $R_{u,i}$ is the rating of user u on item i. A user u might rate more than one item. Similarly, an item i might get several rates from several users. This is resulted in User-Item matrix. As shown in figure 3.2, in rating matrix R, each row represents a user rating history and each column represents an item rating history. For example, the user u_2 has rated the item i_2 with a value of 4. On the other hand, the item i_2 is relatively popular and has been rated the most.

A rate is a numerical value given by a user as a judgment on a certain item. Our rating scale is from 1 to 5. The value 1 of rating means the item is poor and the value 5 means the item is excellent and so on as in figure 3.3.

3.2.1 Positive\Negative Items

Although the rate scale we use is 1–5, each user would have their own independent rating behavior. A positive or a negative rate is dependent on the user. Each user have their own average rate. If a rating is above that average, then the item is positive for that user and vise versa. More formally, as in equation 3.1 we define the set $Pos(u)$ as the set of positive items for user u and the set $Neg(u)$ as the set of negative items for the user u. An item i rated by a user u with a rate $R_{u,i}$ is considered a positive item for that user if it is greater or equal to the average rates $\bar{R}_u$ of user u. On the other hand, an item i

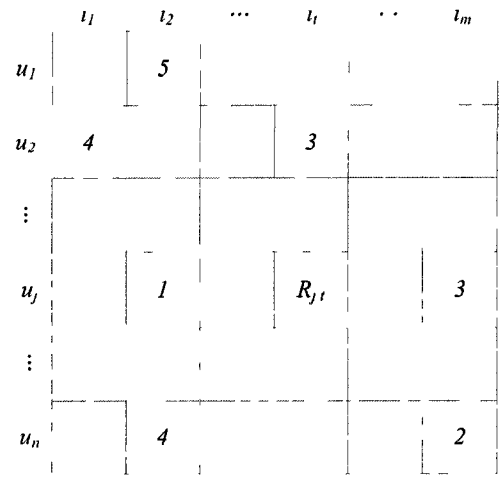

Figure 3.2: User-Item matrix

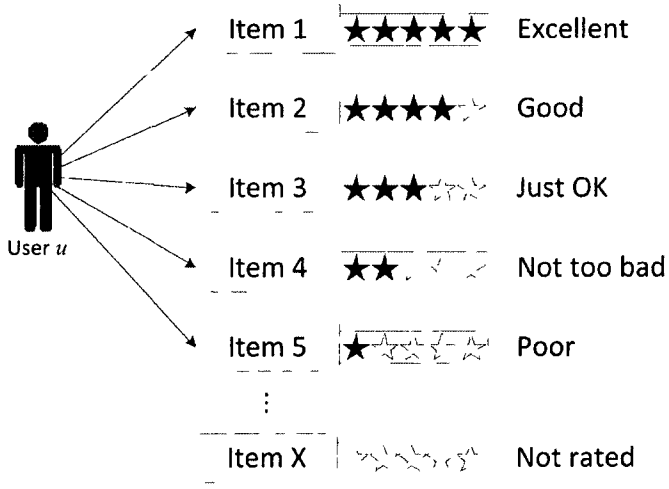

Figure 3 3: Rating scale and meaning

rated by a user u with a rate $R_{u,i}$ is considered a negative item for that user if it is less than the average rates $\bar{R}_u$ of user u.

$$Pos(u) = \{i \in I \mid R_{u,i} \geq \bar{R}_u\} \tag{3.1}$$

$$Neg(u) = \{i \in I \mid R_{u,i} < \bar{R}_u\} \tag{3.2}$$

$$Pos(u) \cap Neg(u) = \phi \tag{3.3}$$

For example, the user in figure 3.4 has rated 9 out of 11 items. The average rating of user1 is $\bar{R}_{User1} \approx 3$. Therefore, items 1, 3, 5, 6 and 10 are positive for user1 and items 4, 7, 8 and 9 are negative for user1. More formally, $Pos(\text{user1}) = \{i_1, i_3, i_5, i_6, i_7\}$ and $Neg(\text{user1}) = \{i_4, i_7, i_8, i_9\}$. The items 2 and 11 are mutual to user1 since they have not been rated by him. If user1 at any time change an existing rating or rate a new item the $\bar{R}_{user1}$ will be changed and eventually the sets $Pos(\text{user1})$ and $Neg(\text{user1})$.

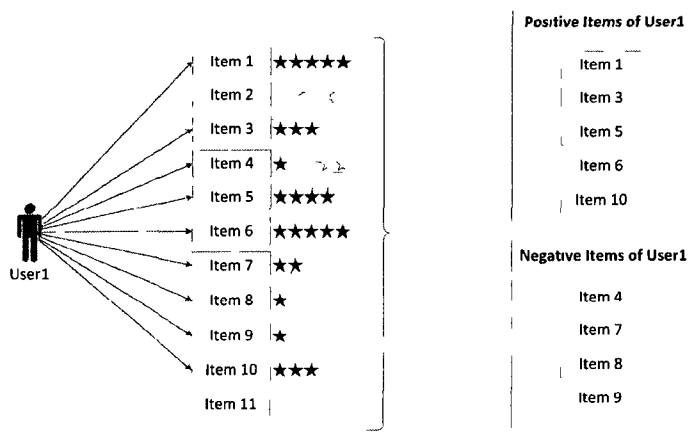

Figure 3.4: classifying items into positive and negative

3.3 Connecting Rating to Tags

Tags are text values that describe an item partially or completely. An Item can be tagged by several different tags. The tags can be free text entered by the end users, a predefined list of tags to be selected by the end users or assigned only by the media administrator. Items can be tagged either explicitly or implicitly or even both depends on the system design. For example, if the items are movies, the movie "Casino 1995" is tagged by

"Drama" and also by "Hit With A Baseball Bat" as well several other tags. Two or more items can share some tags. For example, if the items are movies, many of them might have the tag"action". In addition, in social tagging, an item can be tagged by the same tag more than one time. However, in our system, we have designed the tagging process as the following:

- Every item should have at least one tag
- Tags are pre-defined.
- Users can make user of the tags without modifying them.
- There is no maximum number of tags for an item.
- An item can be tagged by a tag only once.

As in figure 3.5, the relationship between items and tags are binary. An item i is either tagged by a tag t or not as in case of item i_1 tagged by tag t_1 and not tagged by tag t_2. Several items can be tagged by the same tag as shown in case of items i_1 and i_2 that are both tagged by the tag t_1.

	t_1	t_2	⋯	t_k	⋯	t_l
i_1	1	0	0	1	0	1
i_2	1	1	0	0	1	0
⋮	0	0	1	1	0	0
i_t	1	1	1	0	1	1
⋮	0	1	0	1	1	0
i_m	0	0	1	1	1	1

Figure 3.5: Item-Tag matrix

3.3.1 Computing a weight of tags

Rating an item implies rating all the tags of that item. Although rating scale is the same for all users, not all users are the same when it comes to giving a rate. In other words, some users tends to give high ratings and some other tend to give low ratings. Therefore, a weighting method is necessary to make the rating value more reasonable. We have used the weighting method in 3.4 in order to calculate the normalized rate $R'_{u,i}$ of a user u for an item i .

$$R'_{u,i} = \frac{R_{u,i}}{\sqrt{\sum_{j=1}^{m} R_{u,j}^2}} \tag{3.4}$$

Since rating an item implies rating all of its tags, we can use the equation 3.4 to calculate the normalized rate of a tag or in other words, the weight of a tag t that is assigned to item i and rated by user u and we call it $w_{u,i}(t)$. Equation 3.5 shows the definition of $w_{u,i}(t)$.

$$w_{u,i}(t) = \frac{R_{u,i}}{\sqrt{\sum_{j=1}^{m} R_{u,j}^2}} \tag{3.5}$$

A positive rate for an item implies that all the tags of that item are positive. As in figure 3.6, items 3, 5, 6 and 10 are positive and, at the same time, they are sharing some tags among them (i.e. positive tags). Therefore, the mean weight of each positive tag is considered. The mean weight of positive tag t for user u is calculated by the equation 3.6. The equation simply divides the sum of each positive tag's weight by the number of positive items contains that tag I_u^{pos}

$$\mu_{u,t}^{pos} = \frac{1}{|I_u^{pos}(t)|} \times \sum_{j \in I_u^{pos}(t)} w_{u,j}(t) \tag{3.6}$$

Similar to the positive case, a negative rate for an item implies that all the tags of that item are negative. The mean weight of negative tag t for user u is calculated by the equation 3.7. Also, as shown in figure 3.6, negative tags mean weights are considered. For those tags who do not belong to the negative class, the value of the mean weight is null.

$$\mu_{u,t}^{neg} = \frac{1}{|I_u^{neg}(t)|} \times \sum_{j \in I_u^{neg}(t)} w_{u,j}(t) \tag{3.7}$$

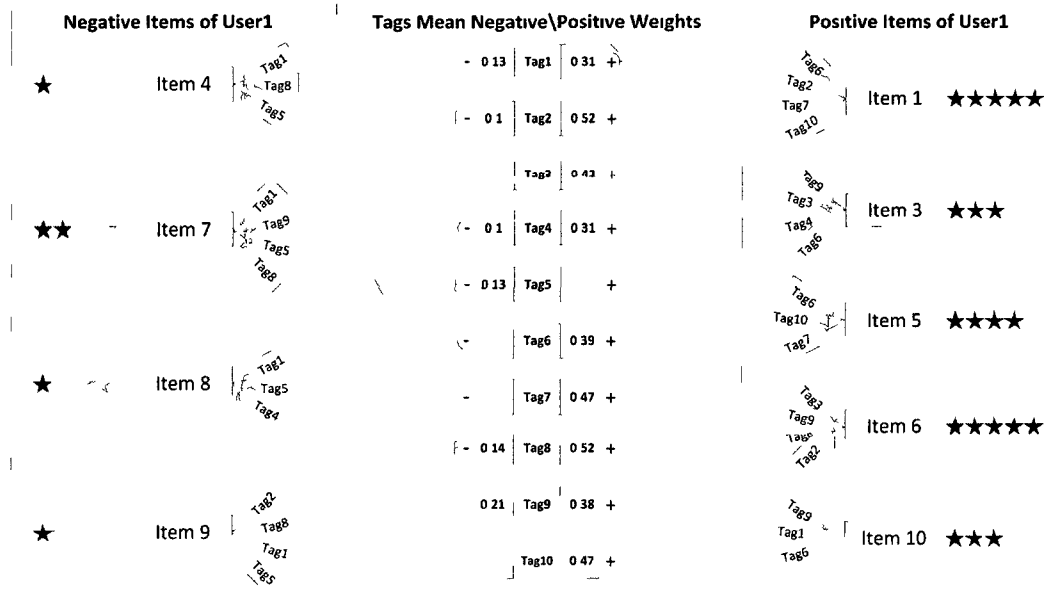

Figure 3 6 Tags mean positive and negative weights of user1 The empty weights means the tag does not belong to that class

The overall weight (i.e. global weight) of tag t for user u is calculated as in the equation 3.8. Note that some tags are either positive or negative. Therefore, in that case, the mean weight remains as is. Only tags that are shared among negative and positive items need their mean weights to be divided by 2 in order to get their overall weight.

$$\mu_{u,t} = \begin{cases} \frac{\mu^{pos}_{u,t} + \mu^{neg}_{u,t}}{2} & \text{if } t \in I^{pos}_u(t), t \in I^{neg}_u(t) \\ \mu^{pos}_{u,t}, & \text{if } t \in I^{pos}_u(t) \\ \mu^{neg}_{u,t}, & \text{if } t \in I^{neg}_u(t) \end{cases} \tag{3.8}$$

3.4 Tag-Based User-User Similarity

In order to fin the neighbors of a user, we calculate the similarity between that user and the rest of users. Let us have the user u as a target user, the k most similar users to the user u constitute the neighborhood of u. To be more specific, we consider two types of neighborhoods as follows. First, the users who are similar to the user u in terms of preferred topics. We call these users the positive neighbors. Second, the users who are similar to the user u in terms of the irrelevant topics. We call these users the negative users. In order to build user u neighborhoods, we need to define the positive and the negative similarities between user u and other users (denoted by user v) as follows:

- Positive Similarity: in which the two users are similar in terms of liking certain topics (i.e. similar positive patterns). Let IT^{pos}_u be the set of positive tags for the user u or in other words, the set of tags in the positive patterns of user u. As defined in the previous section, $\mu_{u,t}$ is the mean weight of tag for user u . Positive similarity between users u and v is calculated using equation 3.9.

$$sim(u,v) = \frac{\sum_{t \in IT^{pos}_u \cap IT^{pos}_v} \mu^{pos}_{u,t} \times \mu^{pos}_{v,t}}{\sqrt{\sum_{t \in IT^{pos}_u} \mu^{pos^2}_{u,t}} \times \sqrt{\sum_{t \in IT^{pos}_v} \mu^{pos^2}_{v,t}}} \tag{3.9}$$

- Negative Similarity: in which two users are similar in terms of disliking certain topics (i.e. similar negative patterns). Similar to the positive similarity, negative similarity between users u and v is calculated by equation. In this case, IT^{neg}_u is

the set of tags of the negative patterns of user u. Equation 3.9 shows how the negative similarity is calculated.

$$sim(u,v) = \frac{\sum_{t \in IT_u^{neg} \cap IT_v^{neg}} \mu_{u,t}^{neg} \times \mu_{v,t}^{neg}}{\sqrt{\sum_{t \in IT_u^{neg}} \mu_{u,t}^{neg^2}} \times \sqrt{\sum_{t \in IT_v^{neg}} \mu_{v,t}^{neg^2}}} \tag{3.10}$$

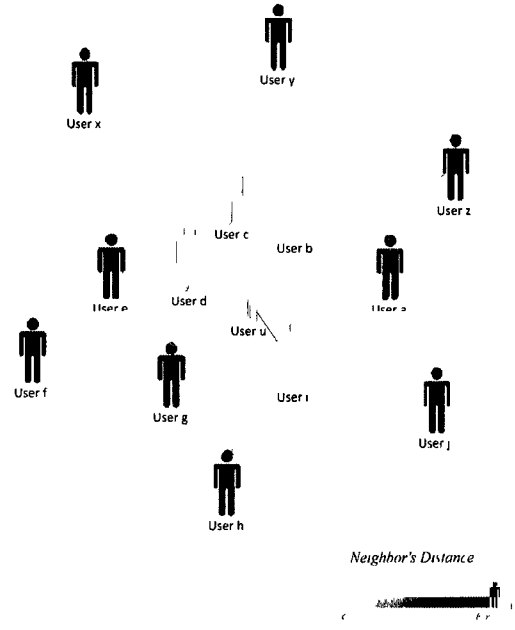

Figure 3.7: Similar users to user u constitute user u's neighborhood

Although similar users to user u have different degrees of similarity, they are in general called neighbors and therefore constitute user u's neighborhood. The closer neighbor to the user u is the more similar user to u. That is applied to positive and negative similarities separately. Figure 3.7 shows the general case of user u neighborhood. Different neighbors of user u have different distances to user u based on how similar they are to user u. The neighborhood size can be increased or decreased by increasing or decreasing a threshold value on users' similarity to user u. We will discuss the neighborhood size effects in Chapter 4.

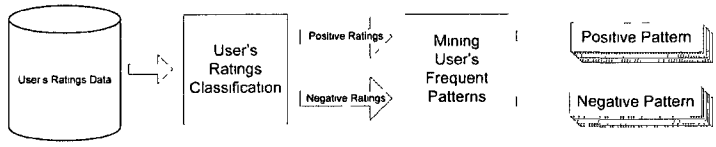

Figure 3.8: Mining users' positive and negative patterns

3.5 Enriched User Model

Each user have both positive and negative patterns. We have extracted users patterns for each user separately by mining items' tags that are rated by users as in figure 3.8. More formally, user u have both IT_u^{pos} and IT_u^{neg} that are defined in the previous section. The initial (i.e. original) user model IM_u as defined in equation 3.11 consist of the initial positive and negative model of user u.

$$IM_u =< IT_u^{pos}, IT_u^{neg} > \tag{3.11}$$

At this point, we take the initial user model and enrich it using similar users' models hoping that those users have richer patterns than the target user has. In general, similar users have some patterns as well as patterns that are available at one side only and at the same time include the existing common patterns. For example, user u has the model $IM_u =< IT_u^{pos}, IT_u^{neg} >$ while user v has the model $IM_v =< IT_v^{pos}, IT_v^{neg} >$. Assume that v has more ratings than user u and therefore has more positive and negative patterns. We want to enrich IM_u using IM_v. Therefore, we call user u *target user* and we call user v *enrich user*. As in figure 3.9, target user and enrich user have a pattern in common as well as a pattern in the enrich user side that includes the common pattern. In that case, we expand user model IM_u of user u to include the enriched pattern ET_u and become the enriched user model EM_u of user u. The enriched user model is defined in equation 3.12.

$$EM_u =< ET_u^{pos}, ET_u^{neg} > \tag{3.12}$$

Where ET_u^{pos} is the set of enriched positive tags of user u and ET_u^{neg} is the set of enrich negative tags of user u.

Finally, we define the Collaborative User Model of user u CM_u as a combination of both the initial model and the enriched model of user u as in equation 3.13.

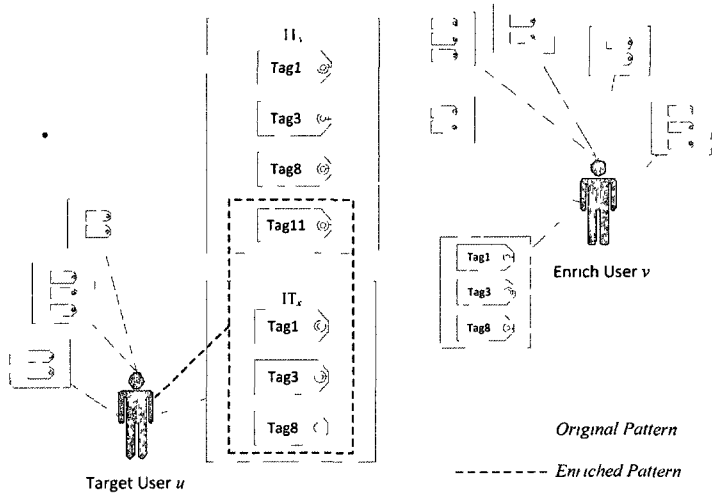

Figure 3.9: Enriching user's patterns

$$CM_u =< IM_u, EM_u > \tag{3.13}$$

3.6 Recommending Items Using Naive Bayes Approach

In order to recommend new items to a user, we use locally weighted Naive Bayes approach. Let the user u be the target user. Let CM_u be the collaborative model of the target user u. Let I_j be a an item with a set of tags $I_j = t_1, t_2, \ldots, t_m$. We want to determine the likelihood of item I_j for being of class C which is either positive or negative. The general form of naive bays approach is shown in equation 3.14

$$P(C|t_1, t_2, \ldots, t_m) = P(C|I_j) = P(C)\prod_{j=1}^{m} P(t_j|C) \tag{3.14}$$

3.6.1 The probability of Positive Class Items

Now, we use the same principle in equation 3.14 in order to determine the probability (i.e. the posterior probability) of item I_j being of class $C = Positive$ as in equation 3.15.

$$P(Pos|I_j) = P(Pos)\prod_{j=1}^{m} P(t_j|Pos) \tag{3.15}$$

The priori probability of of an item being positive is calculated in equation 3.16

$$P(Pos) = \frac{|I_u^{pos}|}{|I_u|} \tag{3.16}$$

Finally, the conditional probability $P(t_j|Pos)$ is calculated using equation 3.17.

$$P(t_j|Pos) = \frac{1 + \omega_j \cdot f_{u,j}^{pos}}{V_u + \sum_{t \in (IT_u^{pos} \cup ET_u^{pos})} \omega_t \cdot f_{u,t}^{pos}} \tag{3.17}$$

where $f_{u,j}^{pos}$ is the frequency of positive tag j in user collaborative model CM_u while V_u is the total number of tags in CM_u (positive and negative)as defined in equation 3.18.

$$V_u = |IT_u^{pos}| + |ET_u^{pos}| + |IT_u^{neg}| + |ET_u^{neg}| \tag{3.18}$$

In addition, ω_j is the weight of tag j for user u as defined in equation 3.19

$$\omega_j = \begin{cases} \mu_{u,j} & \text{if } t_j \in IM_u \\ \\ \mu_{v,j} & \text{if } t_j \in EM_u, t_j \in IM_v \end{cases} \tag{3.19}$$

where $\mu_{u,j}$ is the mean weight of tag j for user v and similarly, $\mu_{v,j}$ is the mean weight of tag j for user v.

3.6.2 The Probability of Negative Class Items

Similar to 3.15, we calculate the probability (i.e. the posterior probability) of item I_j being of class $C = Negative$ using the equation 3.20

$$P(Neg|I_j) = P(Neg)\prod_{j=1}^{m} P(t_j|Neg) \tag{3.20}$$

Similar to 3.16, the priori probability of of an item being positive is calculated in equation 3.21

$$P(Neg) = \frac{|I_j neg|}{|I_j|} \tag{3.21}$$

Finally, the conditional probability $P(t_j|Neg)$ is calculated using equation 3.22.

$$P(t_j|Neg) = \frac{1 + \omega_j \cdot f_{u,j}^{neg}}{V + \sum_{t \in (IT_u^{neg} \cup ET_u^{neg})} \omega_t \cdot f_{u,t}^{neg}} \tag{3.22}$$

where $f_{u,j}^{neg}$ is the frequency of negative tag j in user collaborative model, CM_u while V is the total number of tags in CM_u as previously defined in equation 3.18. ω_j is the weight of tag j for user u as defined previously in equation 3.19

3.6.3 Top N Recommendations

In order to give a recommendation, log-likelihood ratio is calculated for each candidate item in respect of the target user u. The higher the log-likelihood ratio, the more recommended the item is. Basically, we sort the items in descending order according to their log-likelihood ratio. Then we take the top N items and recommend them. The lo-g-likelihood ratio (denoted by score) is calculated by combining positive and negative conditional probabilities from the previous two sections. As in equation 3.23, we take the logarithm after dividing the two probabilities.

$$\begin{aligned} S(u, I_j) &= \ln \tfrac{P(Pos|I_j)}{P(Neg|I_j)} \\ &= ((\ln P(Pos|I_j)) - (P(Neg|I_j))) + \textstyle\sum_{j=1}^{m}((\ln P(t_j|Pos)) - (P(t_j|Neg))) \end{aligned} \tag{3.23}$$

We denote the log-likelihood ratio (above) by item's I_j score for user u.

Chapter 4

System Implementation

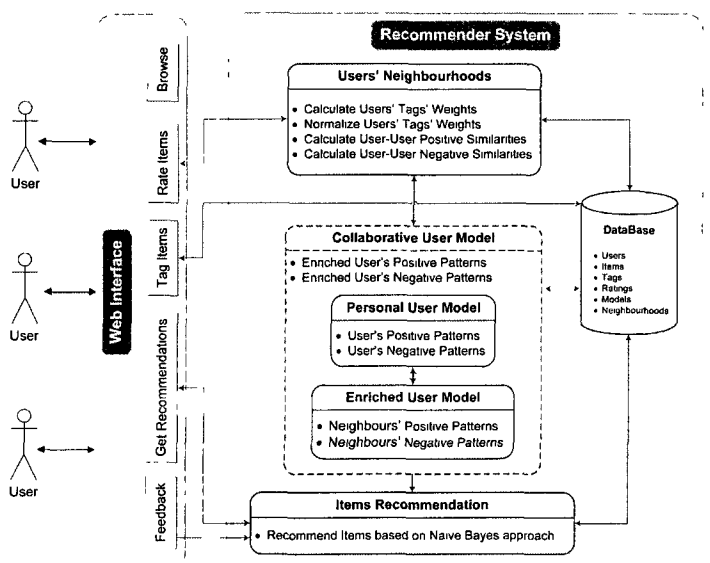

Figure 4.1: Recommender System Architecture

Our recommender system is a domain independent. As a proof on concept, we implemented it as shown in figure 4.1. The implemented system consists of the following main modules:

- Webinterface
- Recommender System

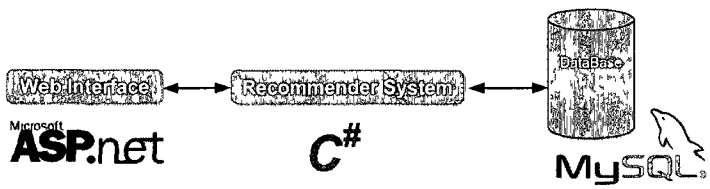

Figure 4.2: Recommender System Architecture Implementation

  - Users' Neighborhoods Processing
  - Users' Models Processing
    * Initial User Model
    * Enriched User Model
    * Collaborative User Model
  - Items Recommendation Process

- Database

We implemented the system using .net platform and mySql database. The web interface is implemented using asp.net. We have also used Asp Ajax Tool Kit controls for the web interface. The background code of the system is implemented using c# programming language. The database engine used is mySql. Figure 4.2 Shows the system architecture's implementation tools. In this chapter, we well first cover the dataset preparation then go over the implementation of the system's main modules.

4.1 Recommender System Database

As we mentioned in Chapter 3, we have selected IMDb dataset. The original format of the dataset is text. Figure 4.3 shows a sample of the original format of the IMDB dataset.

4.1.1 Dataset Prepration

Since our system design requires storing data in a rational database, we have converted the text-based dataset into a rational database format. Using an open source tool called emphJMDb[1], we were able to convert the IMDb text files into emphmysql database table.

[1]http://http://www.jmdb.de .

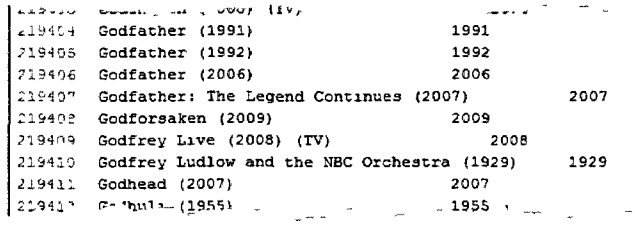

Figure 4.3: A sample of the text-based IMDB dataset

In order to reduce the huge content of the IMDb for performance reasons, we have filtered the dataset to include only the IMDB top movies of us box office (400 movies) and the IMDB top 250 movies of all the time. The total number of movies after removing the redundant items has became: 592 movies. Figure 4.4 shows the general steps of dataset preparation. However, after we created the database, we have found some issues that

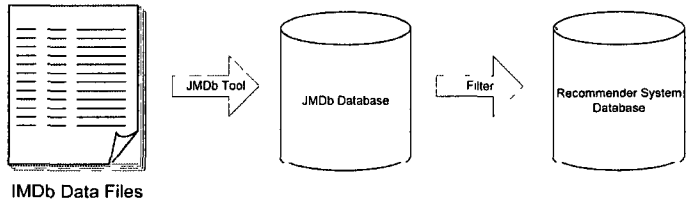

Figure 4.4: Dataset Prepration Steps

need to be solved as the following:

1. Delete any duplicate item. There are intersection between the top 250 movies and the top US box office movies.

2. Modify items titles to match the current IMDb titles. Some movies titles in the dataset does not match the movie title that is in IMDb website. That is resulted from several reasons one of them is just simple typing mistake in the dataset that is corrected on line. Another common reason is applied for the non-English titled movies where they are titled in the dataset with their original title while in IMDb website they are titled in English.

3. Discard any subject that has rated below than 10 items.

4. Add the IMDb URIs for each item. The IMDb dataset does not contains the movies URIs while in our survey we found it necessary to link each item with its IMDb web page in order to give the user the chance to get more information about any item. Therefore, we have added the URIs manually to the dataset.

4.1.2 Database Desgin

After the preparation was completed. We decided to create our own database rather than using JMDb database. We have normalized the database architecture and build it based on our implementation needs. Figure 4.5 shows the entity relationship diagram of our system database.

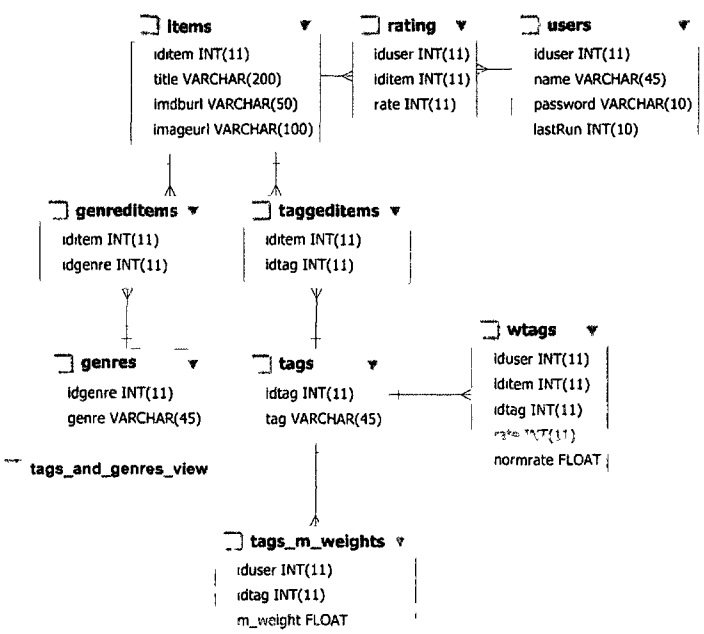

Figure 4.5: Recommender System database entity relationship diagram

Data Tables

In this section we will list a short description of each database entity.

Database Table: Items

Items database table holds items information. In our dataset, each item represent a movie from IMDb. Table 4.1 explains the Items database table attributes.

Items		
Column	Type	Description
IdItem	Integer	A unique identifier of each item. In this case, it is the IMDb movie's id.
Title	Varchar	The title of the item. In this case, it is the IMDb movie's title.
Url	Varchar	The item description page URL. In this case, it is the URL of IMDb movie's web page.
imageURL	Varchar	Item's image URL.

Table 4.1: Items database table

Database Table: Users

Users database table holds users' authentication information as well as their models creation time if there is any. Table 4.2 explains the User database table attributes.

Users		
Column	Type	Description
IdUser	Integer	A unique identifier of each user.
Name	Varchar	The username for login.
Password	Varchar	the password for login
LastRun	Integer	Indicates the time of the last time the user model has been built for this user. A null value means the user model has not been built yet.

Table 4.2: Users database table

Database Table: Rating

Rating database table holds the users' rating information Each item rating made by a user has a record in this table The record is created once an user rated a certain item Therefore, a user who has not yet rated any item will have zero record in this table Similarly, if an item has not been yet rated by any user, it will have zero record in this table This table eliminate the many-to-many relationship between the User and Items tables Table 4 3 explains the Rating database table attributes

Rating		
Column	Type	Description
IdUser	Integer	A unique identifier of a user who rated an item
IdItem	Integer	A unique identifier of the rated item
Rate	Integer	The rating value given by a user The value is ranged from 1-5 5 means excellent and 1 means poor A validation is done at the web interface level to insure that the rating value is within the range

Table 4 3 Rating database table

Database Table: Genres

Genres database table holds the movies genres information Each record of this table represent one genre Table 4 4 explains the Genres database table attributes

Genres		
Column	Type	Description
IdGenre	Integer	A unique identifier of a movie genre
Genre	Varchar	The genre's text

Table 4 4 Genres database table

Database Table: GenredItems

GenredItems database table holds the actual genred movies information Each record of this table represent one genred movie Each movie should have at least one record in

this table or, in other words, should have at least one genre. A movie might have more than one genre therefore more than one record in this table. This table eliminates the many-to-many relationship between the tables Items and Genres. Table 4.5 explains the GenredItems database table attributes.

GenredItems		
Column	Type	Description
IdItem	Integer	A unique identifier of the genred item.
IdGenre	Integer	A unique identifier of a movie genre.

Table 4.5: GenredItems database table

Database Table: Tags

Tags database table holds the tags information. Each record of this table represent one tag. Each tag has only one record in this table. Table 4.6 explains the Tags database table attributes.

Tags		
Column	Type	Description
IdTag	Integer	A unique identifier of a Tag.
Tag	Varchar	The tag's text.

Table 4.6: Tags database table

Database Table: TaggedItems

TaggedItems database table holds the actual tagged items information. Each record of this table represent one tagged item. Each item should have at least one record in this table or, in other words, should have at least one tag. An item might have more than one tags and therefore more than one record in this table. This table eliminates the many-to-many relationship between the tables Items and Tags. Table 4.7 explains the TaggedItems database table attributes.

TaggedItems		
Column	Type	Description
IdItem	Integer	A unique identifier of the tagged item.
IdTag	Integer	A unique identifier of an item's tag.

Table 4.7: TaggedItems database table

Database View: Tags_And_Genres_View

In order to increase the accuracy of the movie tagging. We have considered the movies' genres as tags. Therefore, we have created this view in order to look at the Tags and Genres data tables as a whole. In this view every genre is treated like a tag and each tag is remaining as a tag.

Database Table: WTags

WTags database table holds the tags weight information for each user. Each record in this table represent a rated item with one of its tags as well as the tag weight for a certain user for a certain item. Each rated item will have a number of records equals to its number of tags. Each user will have a unique tag weight for every rated item. If a user rated some items that share the same tags, the weight of hte tags might be different depends on the rate of each item. In addition, Each record in the table Rating (i.e. each user's rating transaction) has a matching record in WTags table with extra tag weight information. Table 4.3 explains the WTags database table attributes.

Database Table: Tags_M_Weights

Tags_M_Weights database table holds the tags mean weight information for each user. Each record in this table represents a tag weight with respect to a user. The mean weight is calculated from several normalized weights for each tag with respect to each user in table WTags Table 4.9 explains the Tags_M_Weights database table attributes.

4.2 Web Interface

As in figure 4.6, the system web interface consists of the following three parts:

- User Login page

Wtags		
Column	Type	Description
IdUser	Integer	A unique identifier of a user who rated an item
IdItem	Integer	A unique identifier of the rated item
IdTag	Integer	A unique identifier of an item's tag
Rate	Integer	The rating value given by a user The value is ranged from 1-5 5 means excellent and 1 means poor
Normrate	Float	The normalized tag weight for each tag of an item rated by a certain user

Table 4 8 WTags database table

Tags_M_Weights		
Column	Type	Description
IdUser	Integer	A unique identifier of a user who rated an item
IdTag	Integer	A unique identifier of an item's tag
M_Weight	Float	The mean of tag weight for each tag of an item rated by a certain user

Table 4 9 Tags_M_Weights database table

- Survey (rating) page.
- User Model page.
- Recommendations page.

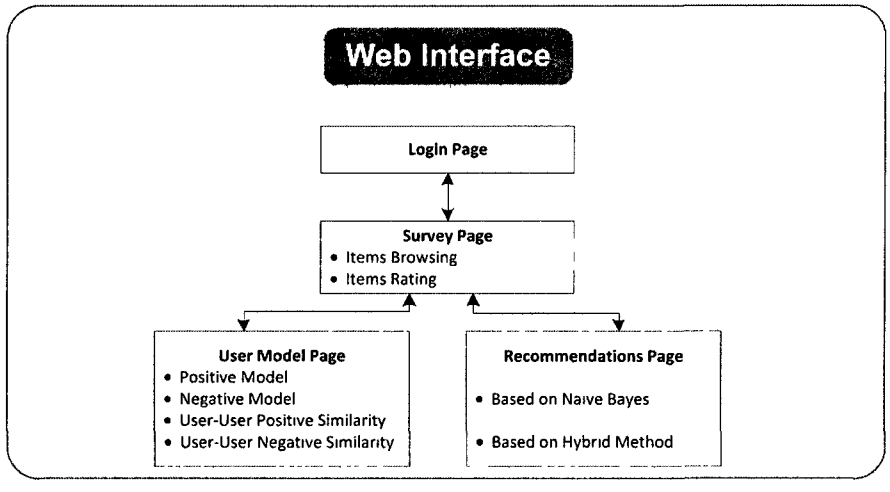

Figure 4.6: Web Interface Architecture

4.2.1 Login Page

In order to keep track of each user, we require each user to login to do the survey. Instead of asking each user to register, have created login information for each survey subject and sent them by email. Figure 4.7 shows a snapshot of the survey user login page.

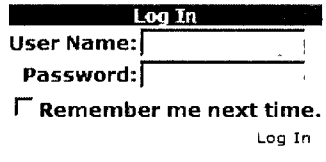

Figure 4.7: Login page

4.2.2 Survey Page

Figure 4.8: Survey page snapshot

Since the IMDb dataset does not contain users ratings information, we created a survey page where users can browse items and rate them. Survey page is the main page of the web interface. As in figure 4.8, the survey page consists of the four parts:

- The left bar 4.8.A: Where user can select a movie genre in order to narrow the displayed movies in the page.
- The movies display 4.8.B area: Where the user can browse the movies and rate them. In addition, the user can click on a movie poster/name in order to open that movie's profile page in IMDb website. If the number of movies is more that 20, the first 20 movies will be displayed in the first page and so on.The movies are sorted alphabetically. As in figure 4.9 User can rate a movie by just moving the mouse

over a set of 5 stars below each movie's poster. 5 stars means the best and 1 star means poor. Empty stars means that the user does not know the movie. For user convenience, the rating process does not refresh the web page. It is done through asynchronous request.

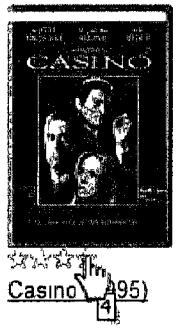

Figure 4.9: Rating a movie

- The user model link 4.8.C: At any time, the user can checkout her model. By clicking on this link, the user will be redirected to the user model page. The user can select either 20 or 30 minimum supports.
- The suggestion link 4.8.D: At any time, the user can see the system recommendations. By clicking on this link, the user will be redirected to the recommedndations page

4.2.3 User Model Page

In user model page, user can see the following:

- Positive and negative patterns: As in figure 4.10, user can see their most frequent patterns sorted from high to low. Positive patterns are the ones that the user like or in other words, they got above average rating from the user. Negative patterns are the ones that the user does not like or in other words, they got below average rating from the user. User can browse the whole patterns by clicking on the page numbers at the bottom. In addition, if the user made new ratings, she can click on the update button to see the new patterns she just developed.

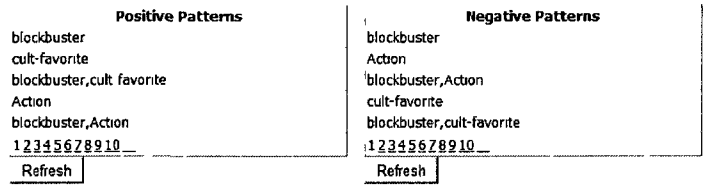

Figure 4.10: User positive and negative patterns

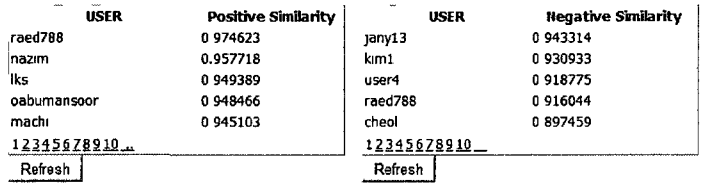

Figure 4.11: User positive and negative similarity

- User-User similarity: As in figure 4.11, user can see the similarities between him and other users in both positive and negative ways.

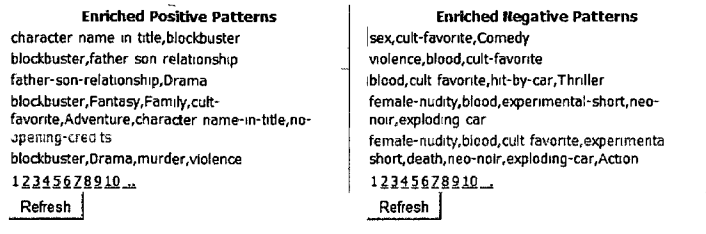

Figure 4.12: User's enriched positive and negative patterns

- Enriched positive \negative patterns: As in figure 4.12, user can see their enriched positive and negative patterns. Those patterns are extracted from the similar users. used to recommend new items as we will see in the following section.

4.2.4 Recommendation Page

As in figure 4.13, user can see the new items recommendations. The items are sorted from highest to lowest. At anytime, user can go back and rate more items or change his

Figure 4.13: Items recommendations

rate for rated items. This will lead to new items recommendations.

4.3 Recommender System

As in figure 4.1, the recommender system consists of the following modules and submodules:

- Users' Models.
  - Personal user model.
  - Neighbors modeling.
  - Enriched user model
- Users' Neighborhoods.
  - Users' tags' weights
  - User-User positive/negative similarities.
- Items Recommendation Process

In this section we will go over each module and its submodules.

User's Models

In this module, the system performs the following steps:

1. Build the personal user model.
   (a) Collect all the tags of the items rated with above-average by each user.

(b) Mine the frequent patterns of that user. We used FP-growth open source code [58] in order to mine the frequent patterns.

(c) Store positive patterns information in the database.

(d) Collect all the tags of the items rated with below-average by each user.
Mine the frequent patterns of that user.

(e) Store negative patterns information in the database.

2. Build the neighbors model

(a) Determine the neighborhood's size.

(b) Collect each neighbor's positive and negative patterns.

3. Enrich user model.

(a) For each user (target user), the system collect the user's neighbor' positive patterns (patterns of the enrich users).

(b) Compare the target user positive patterns with the enrich users' positive ones.

(c) Enrich the target user positive patterns using the enrich users' positive ones.

(d) Store the enriched target user positive patterns in the database.

(e) For each user (target user), the system collect the user's neighbor' negative patterns (patterns of the enrich users).

(f) Compare the target user negative patterns with the enrich users' negative ones.

(g) Enrich the target user negative patterns using the enrich users' negative ones.

(h) Store the enriched target user negative patterns in the database.

Users' Neighbourhoods

In this module, for each user (target user), the system performs the following steps:

1. Collect all the tags of all the items rated by that user as well as their rates.

2. Calculate the normalized rate for each tag and store the normalized rate information in the WTags database table.

3. For each tag from the previous step, the system calculate the mean tag weight and store that information in the tags_mean_weights database table.
4. Calculate the target user positive similarity with the rest of the users.
5. Calculate the target user negative similarity with the rest of the users.
6. Store the similarity information in the database.

Items Recommendation

In this module, for each user (target user) the system performs the following steps:

1. Collect all the items that has not yet been rated by the user.
2. Using naive bayes method, the system calculate the user-item likelihood.
3. Order the items based on their scores from the previous step.
4. Store user-item scores in the database
5. Recommend the top n items to the target user.

Chapter 5

System Evaluation and Experimental Results

In this chapter, we present the evaluation metrics we used to evaluate our system as well as compare it's results against benchmark algorithms' results. In addition, we will go over the experimental setup and the data set.

5.1 Experimental Setup

We have selected a famous dataset of IMDB [1] (Internet Movies Database). IMDB is a famous on line database for movies, TV shows and their crews. It also contains rating and tagging for the database items. We have taken the dataset and made a survey for rating the items. We have 100 subject responded to the survey. IMDB provides a copy of its database for non-profit use. We have selected IMDB dataset for several reasons:

1. The dataset is well organized.
2. The items of the dataset are tagged.
3. The dataset content (i.e. movies) are popular. Therefore, it helps to collect rating information from survey subjects.

We have faced some technical challenges that required data preparation. We have explained that in Chapter 4. We have made an on-line survey as explained in Chapter 4. The proper responses were 99. The rest are either not responded or have rated below

[1] http://www.imdb.com .

10 items. The latter were used as cold start user as we will see later in this chapter. We have randomly divided each user's ratings data into two sets: training data and test data as seen in figure 5.1. For each user, the test data set contains ten items' ratings. These items are randomly selected among each users' positive items. In other words, the test data set contains items that are rated above average by each user. The rest of the ratings are in training data set. The total rating data is 10603 records. The test data set has 990 records and the training data set has 9613 records. We have implemented the experiments using .net framework 3.5 and mysql 5 database. We run it on intel Core i7 workstation running at 2.80 G Hz with 8 GB memory based on Windows 7 operating system.

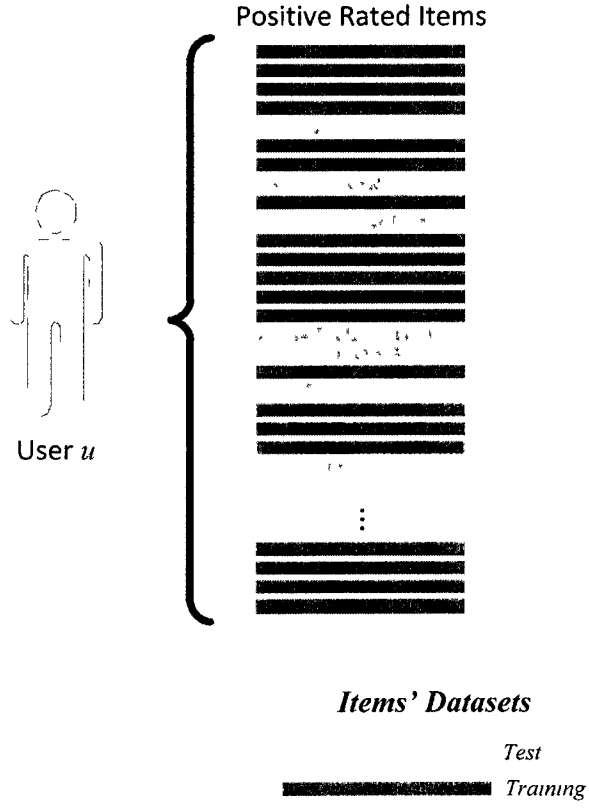

Figure 5.1: Dividing each user's ratings data into training and test data

5.2 Evaluation Metrics

In order to measure our system quality, we adopted two evaluation measures as the following

1 Precision of user u at top N items *Precision* is considered the most popular measure for evaluating recommender systems [17] Originally, it has been introduced along with *recall* measure by [59] for evaluating the quality of indexing systems In addition, it has been used widely for evaluating information retrieval systems as well as evaluating the quality of recommender systems For example, it has been used to evaluate the recommender systems efficiency by [60], [30] and [61] *Precision* determines how accurate are a set of recommended items recommended by a system by comparing them to the top N list of items that should have been recommended In recommender systems, precision is always underestimating the true users' precision, however, it is an efficient measure for such systems when approximated [62] A proper way to approximate precision is to split rated items into test and training sets [17] which we have followed [62] Since we are measuring the quality of a recommender system (i e the top N recommended items) and not an information retrieval system, we will not use the *recall* measure as it impractical most of times to evaluate a recommender system[17] *Recall* measure is used to measure how the system is able to retrieve *all* the relevant items [62] which is out of our interest since in that case we should ask each user to watch all the available movies in our data set and rate it in order to compare their ratings with the recommendation results In *precision* measure, we divide the rating data of each user into two sets First, set of items that have been rated by the user Second, the set of items that have not been rated by the user We assume that the user might be interested in the items that has not been rated yet In addition, items should be classified in a binary fashion into two classes appropriate or non appropriate The appropriate item is the item that has been classified by the user as appropriate (i e from the first set) or classified by the recommender system as appropriate (i e from the second set) The non appropriate item is the item that has been classified by the user as non appropriate (i e from the first set) or classified by the recommender system as non appropriate (i e from the first set) The measure is then performed by computing the ratio of the number of appropriate items recommended to the number of appropriate items In case of a system that has a non-binary rating scale such as in our system, the rating of each item should be converted into binary

format [17] Therefore, in our system we have converted the five stars rating scale into a binary one (positive or negative) as we have covered in chapter 3 We take the top N recommended items for user u as an appropriate items set We then take the ration of that set to the N items from the test data set The higher the precision value, the better the system is The precision of user u at N number of items is calculated using equation 5 1

$$P(u)@N = \frac{|Test(u) \cap TopN(u)|}{|TopN(u)|} \tag{5 1}$$

where $Test(u)$ is the set of items rated by user u in his test data while $TopN(u)$ is the set of top N items recommended to user u The average precision at top N for all users in the test data can be taken by equation 5 2

$$P@N = \frac{\sum_{u=1}^{m} P(u)@N}{m} \tag{5 2}$$

2 Ranking accuracy of user u at top N items Although *precision* is an efficient measure, it does not consider an item's rank (i e position) within a top N items It only tells if certain item is classified successfully as "good" or "bad" or not However, among a set of "good" items, it cannot tell which item is the "best" and so on For example, if a system A successfully recommended 3 out of top 10 items and system B successfully recommended 3 out of 10 items as well, *precision* measure is not able to tell which system was better in means of items' position within the top N items In other words, the recommended three items might fall at the top of the N list or might fall at the bottom We need to know how two recommender systems that seem equal under the *precision* measure are are actually different Therefore, we use *ranking score* measure (RK) which is introduced by [63] Equation 5 3 shows how to calculate the ranking score

$$RK(u)@N = \sum_{i \in Test(u) \cap TopN(u)} \frac{1}{rank(i)} \tag{5 3}$$

where $rank(i)$ refers to the recommended rank of item i within $TopN(u)$ (i e $1 \leq i \leq N$) The average ranking score at top N for all users in the test data is shown in equation 5 4

$$RK@N = \frac{\sum_{u=1}^{m} RK(u)@N}{m} \tag{5 4}$$

The more the recommended items are in a higher at the top N list, the more the RK value is. Therefore, the recommended items by a certain system can be tested by matching their items' rank against another systems items' rank.

5.2.1 Benchmark Algorithms

In order to test our system quality, we implemented three benchmark algorithms. First, item based collaborative filtering ICF which uses cosine similarity (denoted as *ICF*) [63]. Second, the user based collaborative filtering (denoted by *UCF*) which also employs cosine similarity [64]. Third, the vector space model (denoted by *Vector*) which is introduced by [10]. Using our dataset, We found that *ICF* performs the best at neighborhood size = 50 therefore we will call it *ICF50.* In addition, the *UCF* performs the best at neighborhood size = 80 therefore we will call it *UCF80.* On the other hand, *Vector* does not require a neighborhood size. Finally, we found that our system performs the best at neighborhood size = 40 therefore we call it *EM40* as we will see later in this chapter. We evaluated our system by comparing it to the benchmark algorithms as we will cover in the following sections.

5.3 Experimental Results

In this section, we cover the experiments done on our system and the benchmark algorithms. This section is organized as follow. First, we determine the most appropriate neighborhood size as well as the feasibility of building neighborhoods based on positive/negative similarities system using the precision and the ranking scores measures. Second, we compare the IM and EM models of our system. Third, we compare the top N items accuracy of our algorithm against the benchmark algorithms' ones. Fourth, we compare the our system performance for cold start users against the benchmark algorithms ones. In all experiments we set N size to 10. In the third one, we tested N values from 1 to 10 as we will explain in that section.

5.3.1 Determining the neighborhood size

In this experiment, we want to determine the right size of a user's neighborhood as well as testing the feasibility of building neighborhoods based on positive/negative similarities. At $N = 10$, we want to maximize the precision $P@10$ and the ranking score

RK@10 while minimize the neighborhood size. The bigger the neighborhood is, the better recommendation is expected. However, neighborhood size is critical when we take the complexity and the performance time into consideration. In addition, we would like to compare the two neighborhoods building methods. We call the typical way of building neighborhood based on user-user similarity *ratingSimi*. We call our method of building neighborhood based on positive/negative user-user similarity *pos/neg*. We test each neighborhood building method with different sizes using *user precision* and *ranking scores* as the following:

1. User precision at $N = 10$ (P@10): After applying the users precision measure for all users, we had the results for each neighborhood building method with different sizes as shown in table 5.1. In general, the *pos/neg* method is higher than the basic method *ratingSimi*. As in figure 5.2, the *pos/neg* curve experiences slight changes and peaks at neighborhood size = 60. However, that size is relatively huge for a user neighborhood besides the change of precision value is relatively low. We found that the size 40 is better as we will see in the next measure.

Table 5.1: Users precisions at $N = 10$ with different neighborhood sizes and building methods

	neighborhood size					
neighborhood method	10	20	30	40	50	60
pos/neg	0.22349	0.22222	0.22096	0.22096	0.224747	0.22727
ratingSimi	0.19091	0.20101	0.19091	0.19091	0.19697	0.19192

2. User ranking score at $N = 10$ (RK@10): When we applied the *ranking score* measure for all users for each neighborhood building method with different sizes, we got the results in table 5.2. With a small neighborhood size such as 10, the *ratingSimi* method outbalances the *pos/neg* method and peaks at neighborhood size = 50 outbalancing the *pos/neg* method slightly. However, the latter outbalances in general as in figure 5.3. The ranking score for both methods is not as stable as users precisions. However, it peaks at neighborhood size = 40 for *pos/neg* method.

   Therefore, for our algorithm, the *pos/neg* method is selected for its dominant results. In addition, the neighborhood size = 40 is selected for our system since it seems reasonable for both measures especially the *ranking score*. It gives higher score in *ranking score* and a reasonable score in users precision. Furthermore, having the lowest neighborhood size among the benchmark algorithm, as shown in

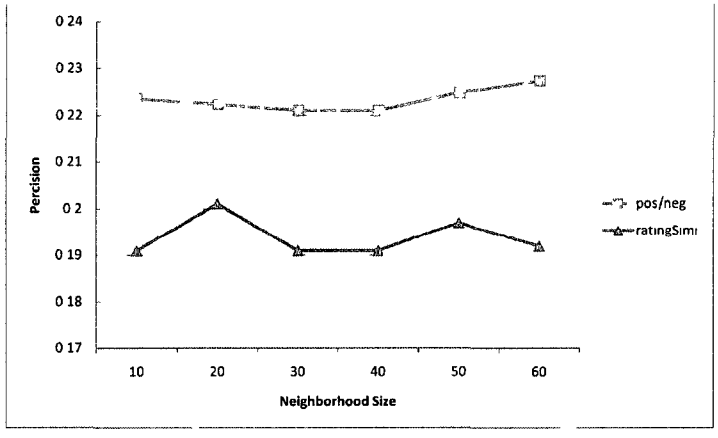

Figure 5.2. Users precision at $N = 10$ for both neighborhoods methods

Table 5.2. Users ranking scores at $N = 10$ with different neighborhood sizes and building methods

	neighborhood size					
neighborhood method	10	20	30	40	50	60
pos/neg	0 8249	0.84911	0.84838	0.85965	0.84467	0.85624
ratingSimi	0 83716	0.8377	0.8372	0.83248	0.84657	0 83221

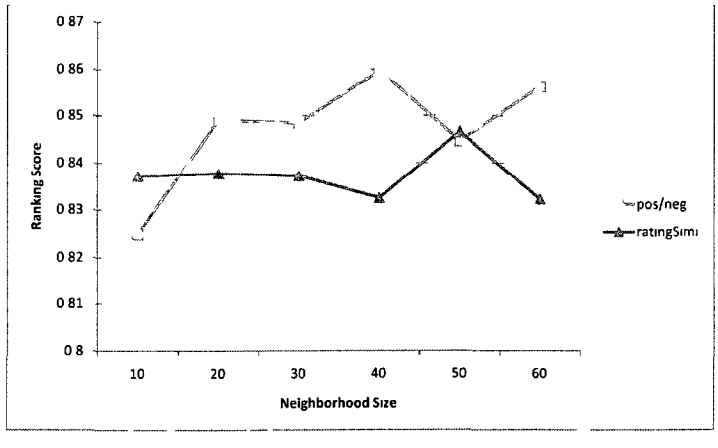

Figure 5 3: Users ranking scores at $N = 10$ for both neighborhoods methods

figure5.4 is an advantage. The small the neighborhood size, the less complex the recommendation process is. Figure 5.4 shows the optimum neighborhood sizes of the benchmark algorithms.

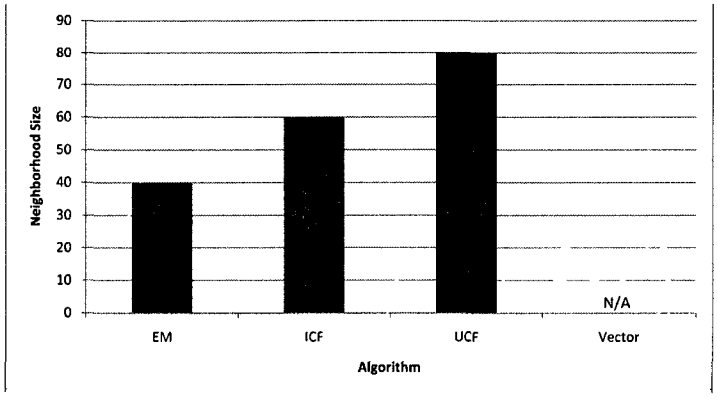

Figure 5.4: The optimum neighborhoods size of the benchmark algorithms, note that neighborhood size is not applicable for the *vector space* algorithm

5.3.2 Comparing the initial model IM with the enriched model EM

Table 5.3: Precision and ranking score for IM and EM at top 10 items

	IM	EM
Precision @10	0.18485	0.22096
Ranking Score @10	0.69988	0.85965

The initial user model IM as defined in equation 3.11 is the set of positive and negative patterns of a user. The enriched user model EM is the set of enriched positive and negative patterns of a user. The enrichment source is the user's most similar neighbors. Similar neighbors number constitute the neighborhood size. In previous section, we have chosen neighborhood size = 40. That means each user's enriched model has been enriched by looking at 40 neighbor's initial models. We measure the EM quality using *precision* and *ranking score* at $N = 10$ and compare the results to the IM ones. The results are summarized in table 5.3.

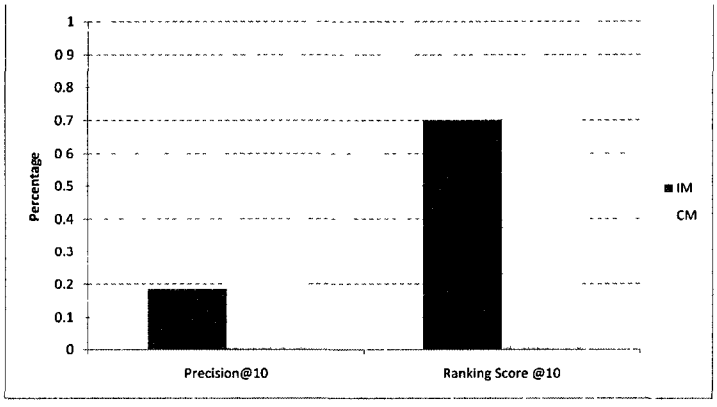

Figure 5.5: EM exceeds IM in both measures

Apparently, the enriched user model EM gives higher precision and ranking scores than IM as in figure 5.5 which is at top 10 items.

5.3.3 Performance Comparison With Benchmark Algorithms

We compared our approach with benchmark algorithms using both measures in tow ways as follow. First, we measure the algorithms performance for all proper responded users (i.e. the 99 users mentioned in this chapter preface). Second, we measure their performance for the cold start users (i.e. 10 users) and compare it with the performance for the top 10 active users. We will cover the experimental results for both measures (Precision and Ranking scores) using different size of top N from 1 to 10.

Performance comparison for all users

In this section, we compare the performance of the algorithms for all users except the cold start ones. We compare the precision of each algorithm at different N sizes from 1–10. We also calculate and compare items ranking accuracy scores of each algorithm using different N sizes from 1–10.

1. **Users Precision at** $1 \leq N \leq 10$**:** Precision value is variant depends on the top N list size. If a system does not require a fixed N size, a vector of different N sizes should be used when evaluating the recommender system's performance [17]. Therefore, we have tested our algorithm against the benchmark ones using

values for N size from 1 to 10. As in table 5.4, the precision value decreased while the N value increased which is a an expected typical behavior for most of the recommendation algorithms [17] which is shown in figure 5.6.

Table 5.4: Precision values at different top N list sizes

	icf50	ucf80	vector	EM40
P@1	0.373737	0.414141	0.151515	0.444444
P@2	0.308081	0.348485	0.141414	0.388889
P@3	0.289562	0.292929	0.151515	0.353535
P@4	0.257576	0.270202	0.141414	0.337503
P@5	0.238384	0.238384	0.131313	0.310606
P@6	0.228956	0.213805	0.117845	0.276768
P@7	0.206349	0.200577	0.105339	0.252525
P@8	0.191919	0.195707	0.107323	0.245339
P@9	0.180696	0.199776	0.104377	0.235209
P@10	0.166667	0.189899	0.10202	0.22096

The *Vector* algorithm is the slowest in decreasing of precision value; however, it has a low value at N=1 already. While the algorithms *UCF80* and *ICF50* shows a very similar performance, the *EM40* were superior all the way.

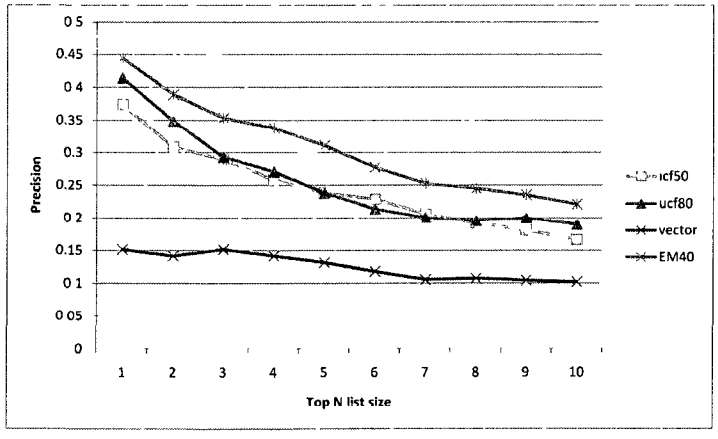

Figure 5.6: Precision values decreases while top N list size increases

2. **Ranking accuracy score at $1 \leq N \leq 10$:** Ranking accuracy scores are increasing

when N increased. That is because of the higher chance of putting the right item at the right rank when there are more choices. Even though, *vector* has a relative very low increase comparing with the result of the algorithm. As in figure 5.7, the three algorithms *ICF50*, *UCF80* and *EM40* had a very close score at $N = 1$ the gap increased when the value of N increased. Practically, the bigger the size of N, the more recommendation the user get. Therefore, the performance is more critical with higher N values.

Although the neighborhood size of *EM40* is lowest among the benchmark algorithms, its ranking scores were the highest. Therefore, it exceeds the performance of the other benchmark algorithms not only in matter of scores but also with the lowest neighborhood size.

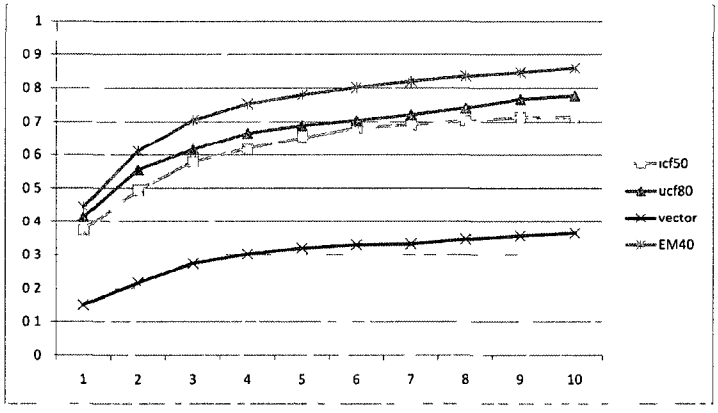

Figure 5.7: Ranking accuracy scores at different values of N

Performance comparison for cold start and top active users

Table 5.5: the average number of ratings for the cold start users and the top active ones

Users type	Average number of ratings
Cold start	5.3
Active	309.7

One important requirement for our system is to reduce the cold start problem. Colds start users as we have covered in chapter 2 are those users who have rated only few items

With only few item rated by a user, a recommender system faces a challenge knowing that user's topic(s) of interest and therefore a difficulty of successfully recommending new items to that user. On the other hand, the top active users are those users who have the highest number of ratings among the systems users. We had both types of users in our dataset as shown in table 5.5. In order to test the benchmark algorithms against the cold start problem, we collected the cold start users in our dataset and applied the two measures (i.e. precision and ranking score) on them. In addition, we compare the results with the top active users in our dataset. In that way, we created a clear view of the cold start users' results by showing the contrast between them and the top active users' results. The number of cold start users is 10, therefore we have selected the top 10 active users for comparison purpose. In this section, we will compare the benchmark algorithms performance with our algorithm at a glance for cold start users. Then, we will compare cold start users results in more detail for each benchmark algorithm using both measures. Finally we will compare the cold start users results with the top active users ones for better understanding.

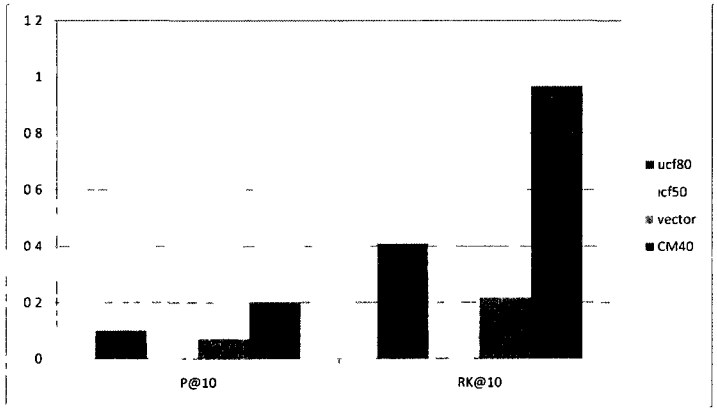

Figure 5.8: Precision and ranking score of cold start users at a glance

Figure 5.8 shows a comparison of the algorithms average results of cold start users using both measures (precision and ranking score). The diagram shows how hard it was for all the algorithms to give a good precision value. That is because the cold start user does not have enough items to determine her interest by an algorithm. The highest benchmark algorithm in precision was *UCF80*. However, our algorithm *EM40* precision is the double higher than *UCF*. Thanks to the enrichment process from similar users. When it comes to ranking score, figure 5.8 also shows that *UCF80* is the highest in

ranking scores among the benchmark algorithms. However, *EM40* gives an outstanding performance as seen in the that figure. The enrichment process of cold start users' models of *EM40* is able to improve the ranking score of them. Once again the *EM40* is able to perform that with a relatively low neighborhood size.

For precision of cold start users in details , table 5.6 shows the precision of each algorithm for each cold start user. It also shows the average precision of all cold start users for each algorithm. If an algorithm gives a user 0 value, it means the algorithm has failed to match any item the true top N of that user. Among the benchmark algorithms, *UCF80* were the highest in average precision with 3 failures with users 9, 14 and 26. However, *EM40* exceeded *UCF80* by having only two failures and a higher average precision. The failure of *EM40* at users 14 and 74 is due to two reasons. First, the precision is underestimated as we discussed earlier in this chapter. Second, these users in particular have non-popular patterns which made them impossible be to enriched and that is an exceptional case that occurs with *EM40*.

Table 5.6: Precision of different algorithms for the cold start users

User ID	**UCF80**	**ICF50**	**Vector**	**EM40**
9	0	0.1	0	0.1
14	0	0	0	0
23	0.1	0	0	0.4
26	0	0	0.1	0.3
40	0.2	0	0.4	0.2
55	0.1	0	0	0.2
74	0.3	0.1	0.1	0
96	0.1	0	0	0.1
97	0.1	0	0.1	0.1
110	0.1	0.3	0	0.6
P@10	0.1	0.05	0.07	0.2

In case of the top active users, table 5.7 shows the precision of each algorithm of the top active users. It also shows the average precision of all the top active users for each algorithm. As in the cold start users' case, the algorithm *UCF80* has the highest precision average among the benchmark algorithms. However, this time both *UCF80* and *ICF50* exceed *EM40*. That is because of two reasons. First the algorithm *EM40* is not designed to help the already active users. This type of users have rated 309.7 items

in average and therefore we believed we should not concern about recommending items for such users. In this type of data set (i.e. movies), a user who rated 309 movies is a "movie fan" that is not targeted by our algorithm. However, these active users are being taken advantage of by *EM40* for enriching the other users' models which is proven in the previous section. Therefore, *EM40* performs better in case of the cold start users. Second, the neighborhood size of *UCF80* and *ICF50* are 80 and 50 respectively while the neighborhood size of *EM40* is 40. If we increase a neighborhood size of *EM40* algorithm we will eventually get more enrichment sources and therefor better results. However, the neighborhood size is always a implementation concern in means of the time needed to run the algorithm. Since we are not targeting the already active users and we concern about the performance of the algorithm practically, we tolerate this result in table 5.7.

Table 5.7: Precision of different algorithms for the top active users

User ID	**UCF80**	**ICF50**	**Vector**	**EM40**
12	4	5	3	6
18	5	3	3	2
31	3	3	2	2
32	1	3	1	2
56	3	3	1	1
58	4	3	1	1
73	3	1	1	2
76	4	1	0	2
88	1	1	1	2
105	4	1	2	3
P@10	0.32	0.24	0.15	0.23

For ranking score of cold start users in details, table 5.8 shows the ranking scores of each algorithm for each cold start user. It also shows the average ranking score of all cold start users for each algorithm. If a user gets 0 in ranking score for an algorithm, it means that algorithm has failed for that user. In other words, the algorithm could not guess the right ranks for all the top N items of that user. Algorithm *UCF80* exceeded the rest of the benchmark algorithms. However, *EM40* is relatively much higher in ranking score than *UCF80*. As in cold start users precision in table 5.6 both algorithm have failed with exactly the same users. That is because of the same reasons mentioned before. In fact, Since the precision is underestimated, the ranking score is eventually underestimated

too.

Table 5.8: Ranking score of different algorithms for the cold start users

User ID	**UCF80**	**ICF50**	**Vector**	**EM40**
9	0	0.1667	0	0.0047
14	0	0	0	0
23	0.25	0	0	1.7111
26	0	0	0.3333	1.375
40	1.3333	0	0.629	1.5
55	0.5	0	0	1.3333
74	0.8611	0.25	1	0
96	0.5	0	0	1
97	0.5	0	0.2	0.3333
110	0.1429	0.625	0	2.4083
RK@10	0.40873	0.10417	0.21623	0.96657

In case of the top active users, the ranking scores of each algorithm for each top active user is shown in table 5.9. The average ranking score of all the top active users for each algorithm is shown in the same table at the bottom row. From the table, *UCF80* gives the highest ranking score and *ICF50* comes after that. As in the precision top active users, *UCF80* and *ICF50* exceed *EM40*. We tolerate this results for the same reasons explained in table 5.7 context.

Table 5.9: Ranking score of different algorithms for the top active users

User ID	**UCF80**	**ICF50**	**Vector**	**EM40**
12	1.5417	2.2833	0.5762	2.2944
18	2.0444	1.3	0.4861	0.4444
31	0.8667	1.6667	0.3	0.6429
32	1	1.4444	0.1111	0.225
56	0.6444	1.3429	0.125	0.5
58	1.8611	0.6012	0.125	0.0833
73	1.254	0.5	0.5	0.7
76	1.8	0.1667	0	1.5
88	0.1429	1	1	1.125
105	1.6111	0.1667	1.125	0.6944
RK@10	1.27663	1.04719	0.43484	0.82094

Chapter 6

Conclusion and Future Work

With the huge amount of the social media available online, the need for recommender systems has increased. In this thesis, we have proposed a novel way to model the user interest in social media recommender systems. We took advantage of both content-based and collaborative filtering techniques to build our hybrid recommending approach. We used social tags as well as explicit ratings to model users interest. We model both user's positive and negative topics of interest and then we enrich the user model using models of users who have similar taste. The recommendation is done based on the enriched user model by employing We use locally weighted Naive Bayes By enriching users models, we aimed to make better recommendations as well as to reduce the cold start users problem. To the best of our knowledge, the topic-driven enrichment of social tag based user model by the collaboration of other similar users is a novel contribution in social media recommender systems.

In order to evaluate our model's performance, we have tested it against three benchmark algorithms. First, a user based collaborative filtering model. Second, an item based collaborative filtering model. Third, the vector space model. We started the experiments by testing the feasibility of building model based on user's positive and negative topics of interest. Using user precision and items ranking accuracy scores measures, the user model based on user's positive and negative topics outperformed the one that is based on user-user similarity. In addition, we calculated the optimum user's neighborhood size for our model as well as the benchmark ones. In the second experiment, we have tested the effect of user's model enrichment on the recommendation quality. We compare the users precision and the items ranking accuracy scores of the enriched user model (i.e. collaborative user model) with the initial user model ones for all users. We looked at the

top 10 recommended items of both models. We have found that the enriched user model exceeded the initial user model in both measure. In the third experiment, we compare the user precision and the items ranking accuracy scores of the enriched user model with the three benchmark algorithms ones. We first determined optimum user's neighborhood sizes of the item based collaborative filtering and the user based collaborative filtering benchmark algorithms. The sizes were 60 and 80 respectively. The third benchmark algorithm is not eligible for neighborhood size test. After determining the neighborhood sizes, we applied the two measures (user precision and items ranking accuracy scores) on all the algorithms at different N sizes from 1–10. For both measures and for each N size, The enriched user model has outperformed the benchmark algorithms. Finally, in the fourth experiment, we tested the algorithms performance for the cold start users and the top active users. We found that the enriched user model outperformed the other algorithms in user precision as well as in items ranking accuracy scores for cold start users.

However, our study opens the door for some interesting future work. First, as we have mentioned earlier, when the tagging is in a free form, there exists a tag noise formed by both user's ambiguous and synonym tags. Therefore, some recent recommender systems considered tags semantic. We intend to involve user tags semantic into our approach since we believe it would help to build a better user model by reducing the tags noise. Second, we plan to consider larger datasets for further testing and evaluation to our approach and comparison to the state-of-the-art tag-based algorithms. Third, we plan to test the feasibility of adopting the proposed collaborative user model in the social search which is an emerging topic in social web.

Bibliography

[1] K Tso-Sutter, L Marinho, and L Schmidt-Thieme Tag-aware recommender systems by fusion of collaborative filtering algorithms In *SAC '08 Proceedings of the 2008 ACM symposium on Applied computing*, pages 1995–1999, New York, NY, USA, 2008 ACM

[2] N Belkin and B Croft Information filtering and information retrieval two sides of the same coin? *Communication of the ACM*, 35(12) 29–38, 1992

[3] P Gill, M Arlitt, Z Li, and A Mahanti Youtube traffic characterization a view from the edge In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, IMC '07, pages 15–28, New York, NY, USA, 2007 ACM

[4] C Yang, Y Hsu, and S Tan Predicting the determinants of users' intentions for using youtube to share video Moderating gender effects *Cyberpsychology, Behavior, and Social Networking*, 2009

[5] J Breese, D Heckerman, and C Kadie Empirical Analysis of Predictive Algorithms for Collaborative Filtering In *Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence (UAI98)*, pages 43–52, 1998

[6] R Burke Hybrid recommender systems Survey and experiments *User Modeling and User-Adapted Interaction*, 12 331–370, 2002 10 1023/A 1021240730564

[7] M Balabanović and Y Shoham Fab content-based, collaborative recommendation *Communication of the ACM*, 40(3) 66–72, 1997

[8] K Bischoff, C Firan, W Nejdl, and R Paiu Can all tags be used for search? In *Proceeding of the 17th ACM conference on Information and knowledge management*, CIKM '08, pages 193–202, New York, NY, USA, 2008 ACM

[9] R. Wetzker, C. Zimmermann, C. Bauckhage, and S. Albayrak. I tag, you tag: translating tags for advanced user models. In *Proceedings of the third ACM international conference on Web search and data mining*, WSDM '10, pages 71–80, New York, NY, USA, 2010. ACM.

[10] X. Li, L. Guo, and Y. Zhao. Tag-based social interest discovery. In *WWW '08. Proceeding of the 17th international conference on World Wide Web*, pages 675–684, New York, NY, USA, 2008. ACM.

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

[12] P. Resnick and H. Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

[13] J. Herlocker, J. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237, New York, NY, USA, 1999. ACM.

[14] R. Meteren and M. Someren. *Using content-based filtering for recommendation*, volume 4203/2006, pages 312–321. Citeseer, 2000.

[15] M. Pazzani and D. Billsus. Content-based recommendation systems. In Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, editors, *The Adaptive Web*, volume 4321 of *Lecture Notes in Computer Science*, pages 325–341. Springer Berlin / Heidelberg, 2007. 10.1007/978-3-540-72079-9_10.

[16] B. Schafer, D. Frankowski, J. Herlocker, and S. Sen. Collaborative filtering recommender systems. *The adaptive web: methods and strategies of web personalization*, pages 291–324, 2007.

[17] J. Herlocker, J. Konstan, L. Terveen, and J. Riedl. Evaluating collaborative filtering recommender systems. *ACM transactions on information systems*, 22(1):5–53, 2004.

[18] J. Konstan, B. Miller, D. Maltz, J. Herlocker, L. Gordon, and J. Riedl. Grouplens: applying collaborative filtering to usenet news. *Communication of the ACM*, 40(3):77–87, 1997.

[19] B. Miller, I. Albert, S. Lam, J. Konstan, and J. Riedl. Movielens unplugged: experiences with an occasionally connected recommender system. In *IUI '03: Proceedings of the 8th international conference on Intelligent user interfaces*, pages 263–266, New York, NY, USA, 2003. ACM.

[20] U. Shardanand and P. Maes. Social information filtering: algorithms for automating "word of mouth". In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 210–217, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[21] M. Montaner, B. Lpez, and J. de la Rosa. A taxonomy of recommender agents on the internet. *Artificial Intelligence Review*, 19:285–330, 2003. 10.1023/A:1022850703159.

[22] S. Sen, J. Vig, and J. Riedl. Tagommenders: Connecting users to items through tags. In *18th International World Wide Web Conference*, pages 671–671, April 2009.

[23] S. Golder and B. Huberman. Usage patterns of collaborative tagging systems. *Journal of Information Science*, 32(2):198–208, 2006.

[24] H. Liang, Y. Xu, Y. Li, R. Nayak, and X. Tao. Connecting users and items with weighted tags for personalized item recommendations. In *Proceedings of the 21st ACM conference on Hypertext and hypermedia*, HT '10, pages 51–60, New York, NY, USA, 2010. ACM.

[25] A. Milicevic, A. Nanopoulos, and M. Ivanovic. Social tagging in recommender systems: a survey of the state-of-the-art and possible extensions. *Artificial Intelligence Review*, 33:187–209, 2010. 10.1007/s10462-009-9153-2.

[26] L. Jiang, H. Zhang, and J. Su. Learning k-nearest neighbor naive bayes for ranking. In Xue Li, Shuliang Wang, and Zhao Yang Dong, editors, *Advanced Data Mining and Applications*, volume 3584 of *Lecture Notes in Computer Science*, pages 175–185. Springer Berlin / Heidelberg, 2005. 10.1007/11527503_21.

[27] J. Han, M. Kamber, and M. Kaufmann. *Data Mining: Concepts and Techniques (2 nd edition.* Morgan Kaufmann, 2006.

[28] E. Frank, M. Hall, and B. Pfahringer. Locally weighted naive bayes. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 249–256. Morgan Kaufmann, 2003.

[29] S. Sahebi, C. Wongchokprasitti, and P. Brusilovsky. Recommending research colloquia: a study of several sources for user profiling. In *Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems*, HetRec '10, pages 32–38, New York, NY, USA, 2010. ACM.

[30] C. Basu, H. Hirsh, and W. Cohen. Recommendation as classification: using social and content-based information in recommendation. In *Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, AAAI '98, pages 714–720, Menlo Park, CA, USA, 1998. American Association for Artificial Intelligence.

[31] J. Allan, J. Carbonell, G. Doddington, J. Yamron, and Y. Yang. Topic detection and tracking pilot study final report. In *In Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, pages 194–218, 1998.

[32] Y. Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1:69–90, May 1999.

[33] J. Rocchio. Relevance feedback in information retrieval. In *The SMART Retrieval System*. Prentice Hall, 1971.

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

[35] R. Mooney and L. Roy. Content-based book recommending using learning for text categorization. In *Proceedings of the fifth ACM conference on Digital libraries*, DL '00, pages 195–204, New York, NY, USA, 2000. ACM.

[36] M. Ghazanfar and A Prugel-Bennett. An improved switching hybrid recommender system using naive bayes classifier and collaborative filtering. In *The 2010 IAENG International Conference on Data Mining and Applications*, April 2010.

[37] X. Su and T. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009(ID 421425), 2009.

[38] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, WWW '01, pages 285–295, New York, NY, USA, 2001. ACM.

[39] H. Kim, A Ji, I. Ha, and G. Jo. Collaborative filtering based on collaborative tagging for enhancing the quality of recommendation. *Electronic Commerce Research and Applications*, 9(1):73 – 83, 2010. Special Issue: Social Networks and Web 2.0.

[40] G. Salton and M. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, Inc., New York, NY, USA, 1986.

[41] H. Liang, Y. Xu, Y. Li, and R. Nayak. Collaborative filtering recommender systems based on popular tags. In *Proceedings of the Fourteenth Australasian Document Computing Symposium*, Sydney, Australia, 2009. University of New South Wales.

[42] H. Liang, Y. Xu, Y. Li, and R. Nayak. Collaborative filtering recommender systems using tag information. In *Web Intelligence and Intelligent Agent Technology, 2008 WI-IAT '08. IEEE/WIC/ACM International Conference on*, volume 3, pages 59 –62, December 2008.

[43] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, CSCW '94, pages 175–186, New York, NY, USA, 1994. ACM.

[44] M. Szomszor, C. Cattuto, H. Alani, K. OHara, A. Baldassarri, V. Loreto, and V. Servedio. Folksonomies, the semantic web, and movie recommendation. In *4th European Semantic Web Conference, Bridging the Gap between Semantic Web and Web 2.0*, 2007.

[45] M. Ghazanfar and A. Prugel-Bennett. A scalable, accurate hybrid recommender system. In *WKDD '10. Third International Conference on Knowledge Discovery and Data Mining 2010.*, pages 94 – 98, january 2010.

[46] G. Karypis. Evaluation of item-based top-n recommendation algorithms. In *Proceedings of the tenth international conference on Information and knowledge management*, CIKM '01, pages 247–254, New York, NY, USA, 2001. ACM.

[47] C. Shahabi, F. Banaei-Kashani, Y. Chen, and D. McLeod. Yoda: An accurate and scalable web-based recommendation system. In Carlo Batini, Fausto Giunchiglia, Paolo Giorgini, and Massimo Mecella, editors, *Cooperative Information Systems*, volume 2172 of *Lecture Notes in Computer Science*, pages 418–432. Springer Berlin / Heidelberg, 2001. 10.1007/3-540-44751-2_31.

[48] M. Connor and J. Herlocker. Clustering items for collaborative filtering. In *SIGIR-2001 Workshop on Recommender Systems*, New Orleans, Louisiana, USA, 2001.

[49] L. Ungar and D. Foster. Clustering methods for collaborative filtering. In *Proceedings of the Workshop on Recommendation Systems.* AAAI Press, Menlo Park California, 1998.

[50] S. Chee, J. Han, and K. Wang. Rectree: An efficient collaborative filtering method. In Yahiko Kambayashi, Werner Winiwarter, and Masatoshi Arikawa, editors, *Data Warehousing and Knowledge Discovery*, volume 2114 of *Lecture Notes in Computer Science*, pages 141–151. Springer Berlin / Heidelberg, 2001. 10.1007/3-540-44801-2_15.

[51] L. Si and R. Jin. Flexible mixture model for collaborative filtering. In *ICML*, pages 704–711, Washington DC, USA, 2003. AAAI Press.

[52] G. Xue, C. Lin, Q. Yang, W. Xi, H. Zeng, Y. Yu, and Z. Chen. Scalable collaborative filtering using cluster-based smoothing. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, pages 114–121, New York, NY, USA, 2005. ACM.

[53] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. In *Proceedings of the Fifth International Conference on Computer andInformation Technology*, 2002.

[54] R. Burke. Hybrid web recommender systems. In Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, editors, *The Adaptive Web*, volume 4321 of *Lecture Notes in Computer Science*, pages 377–408. Springer Berlin / Heidelberg, 2007. 10.1007/978-3-540-72079-9_12.

[55] Y. Zhen, W. Li, and D. Yeung. Tagicofi: tag informed collaborative filtering. In *Proceedings of the third ACM conference on Recommender systems*, RecSys '09, pages 69–76, New York, NY, USA, 2009. ACM.

[56] P. Jomsri, S. Sanguansintukul, and W. Choochaiwattana. A framework for tag-based research paper recommender system: An ir approach. In *Proceedings of the 2010 IEEE 24th International Conference on Advanced Information Networking and*

Applications Workshops, WAINA '10, pages 103–108, Washington, DC, USA, 2010. IEEE Computer Society.

[57] Z. Wang, Y. Wang, and H. Wu. Tags meet ratings: Improving collaborative filtering with tag-based neighborhood method. In *Proceedings of the Workshopon Social Recommender Systems (SRS10)*, Hong Kong, China, 2010. ACM.

[58] B. Goethals. http://adrem.ua.ac.be/ goethals/software/, june 2010.

[59] C. Cleverdon, J. Mills, and M. Keen. Factors determining the performance of indexing systems. *ASLIB Cranfield project, Cranfield*, 1966.

[60] D. Billsus and M. Pazzani. Learning collaborative information filters. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pages 46–54, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[61] N. Good, J. Schafer, J. Konstan, A. Borchers, B. Sarwar, J. Herlocker, and J. Riedl. Combining collaborative filtering with personal agents for better recommendations. In *Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence*, AAAI '99/IAAI '99, pages 439–446, Menlo Park, CA, USA, 1999. American Association for Artificial Intelligence.

[62] J. Wang, S. Robertson, A. de Vries, and M. Reinders. Probabilistic relevance ranking for collaborative filtering. *Information Retrieval*, 11:477–497, 2008. 10.1007/s10791-008-9060-1.

[63] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM transactions on information systems*, 22(1):143–177, 2004.

[64] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Analysis of recommendation algorithms for e-commerce. In *Proceedings of the 2nd ACM conference on Electronic commerce*, EC '00, pages 158–167, New York, NY, USA, 2000. ACM.