PROBABILISTIC MATRIX FACTORIZATION BASED COLLABORATIVE FILTERING WITH IMPLICIT TRUST DERIVED FROM REVIEW RATINGS INFORMATION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

EDA ERCAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF INFORMATION SYSTEMS

SEPTEMBER 2010

Approval of the Graduate School of Informatics:

Prof. Dr. Nazife Bolat Baykal
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science:

Assist. Prof. Dr. Tuğba Taşkaya Temizel
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science:

Assist. Prof. Dr. Tuğba Taşkaya Temizel
Supervisor

Examining Committee Members:

Prof. Dr. Yasemin Yardımcı Çetin	METU, II	
Assist. Prof. Dr. Tuğba Taşkaya Temizel	METU, II	
Assist. Prof. Dr. Erhan Eren	METU, II	
Assist. Prof. Dr. Sevgi Özkan	METU, II	
Assist. Prof. Dr. Pınar Şenkul	METU, CENG	

ABSTRACT

PROBABILISTIC MATRIX FACTORIZATION BASED COLLABORATIVE FILTERING WITH IMPLICIT TRUST DERIVED FROM REVIEW RATINGS INFORMATION

Ercan, Eda

M.S., Department of Information Systems

Supervisor : Assist. Prof. Dr. Tuğba Taşkaya Temizel

September 2010, 93 pages

Recommender systems aim to suggest relevant items that are likely to be of interest to the users using a variety of information resources such as user profiles, trust information and users past predictions. However, typical recommender systems suffer from poor scalability, generating incomprehensible and not useful recommendations and data sparsity problem.

In this work, we have proposed a probabilistic matrix factorization based local trust boosted recommendation system which handles data sparsity, scalability and understandability problems. The method utilizes the implicit trust in the review ratings of users. The experiments conducted on Epinions.com dataset showed that our method compares favorably with the methods in the literature.

In the scope of this work, we have analyzed the effect of latent vector initialization in matrix factorization models; different techniques are compared with the selected evaluation criteria.

Keywords: Probabilistic Matrix Factorization, Social Networks, Recommender Systems, Latent Vectors

ÖZ

OLASILIKSAL DİZEY ÇARPANLARINA AYIRMAYA DAYALI YORUM PUANLARI BİLGİSİNDEN ÇIKARILAN GİZLİ GÜVEN İLE ORTAKLAŞA FİLTRELEME

Ercan, Eda

Yüksek Lisans, Bilişim Sistemleri Bölümü

Tez Yöneticisi : Yard. Doç. Dr. Tuğba Taşkaya Temizel

Eylül 2010, 93 sayfa

Öneri sistemleri kullanıcı profilleri, güven bilgisi ve kullanıcının geçmiş tercihleri gibi çeşitli bilgileri kullanarak kullanıcının beğenisine en çok hitap etme olasılığı olan nesneleri önermeyi hedefler. Ancak, geleneksel öneri sistemlerinde ölçeklenebilir olmama, kapsamlı ve faydalı öneriler üretememe ve veri eksikliği gibi sorunlar bulunmaktadır.

Bu tez kapsamında önerilen olasılıksal dizey çarpanlarına ayırmaya dayanan yerel güven bilgisini kullanan sistemle, veri eksikliği, ölçeklenebilirlik ve anlaşılabilirlik problemleri ele alınmıştır. Bu metot kullanıcılar tarafından verilen yorum puanlarındaki gizli güven bilgisini kullanmaktadır. Epinions.com veri kümesi üzerinde yapılan deneyler, önerilen metodun literatürdeki metodlarla kıyaslanabilir sonuçlar ürettiğini göstermiştir.

Bu tez kapsamında, gizli yöney başlatmanın dizey çarpanlarına ayırma modellerine etkisi analiz edilip, seçilen değerlendirme ölçütleri ile farklı başlatma teknikler karşılaştırılmıştır.

Anahtar Kelimeler: Olasılıksal Dizey Çarpanlarına Ayırma, Sosyal Ağlar, Öneri Sistemleri, Gizli Yöneyler

To my family

ACKNOWLEDGMENTS

This study would not have been possible without the guidance and the help of several individuals who supported me in any respect in the preparation and completion of this study. Now, it is a pleasure to thank all of them for their kindful support.

Foremost, I would like to express my sincere gratitude to my advisor Asst. Prof. Dr. Tuğba Taşkaya Temizel, for her guidance and extended help during this thesis study. By the way, I wish her family a wonderful life with the upcoming baby.

My special thanks go to my parents Emin and Emel, and my dear brother Emre, for their endless love and support not just throughout this study but since the day I was born; without them I would fall apart.

At home, my lovely housemates Duygu Atılgan and Büşra Atamer accompanied me in the long days and nights with their cheerful presence.

At work, everything was easier with the encouraging words of my supervisor Levend Sayar.

Last but definitely not least, I owe my deepest gratitude to Ferhat Şahinkaya. He has made his support available in a number of ways, he supported me with his brilliant ideas as a colleague but more importantly, made me know he was with me every time I need.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

RS	Recommender System
CF	Collaborative Filtering
CBF	Content Based Filtering
MF	Matrix Factorization
PMF	Probabilistic Matrix Factorization

LIST OF NOTATIONS

S	Users set
T	Items set
R	Ratings matrix
C	Social network matrix
$\|R\|$	Number of elements in ratings matrix R
N	Number of users
M	Number of items
$u(i,j)$	Utility function, usefulness of j^{th} item to i^{th} user
i	Active user
j	Active item
k	Number of keywords in content based recommender systems
w	Weight in content based methods
r_{ij}	rating value given to j^{th} item by i^{th} user
r'_{ij}	Normalized rating value given to j^{th} item by i^{th} user
$\hat{r}_{ij}$	Prediction of the rating given to j^{th} item by i^{th} user
$\bar{r}_i$	Average of the ratings given by i^{th} user
c_{ik}	Trust value given to k^{th} user by i^{th} user
T_i	Set of items rated by i^{th} user
N_{ij}	Neighborhood set of query (i, j)
U	User latent feature matrix
V	Item latent feature matrix
Z	Factor feature matrix
d	Latent vector dimension
e_{ij}	Training error

CHAPTER 1

INTRODUCTION

1.1 Motivation

Nowadays, Internet has become crucial for our lives and its importance is growing day by day. This popularity causes World Wide Web to have huge amount of information. Day by day, it is more difficult for the users to find the information they are looking for due to the growing size and complexity of many web sites [69]. A web site can be personalized or pages that are related to the user's interest may be selected to help users find the information they are looking for more easily.

As the web sites continue to grow, recommender systems (RS) have become valuable resources for users who are trying to find an intelligent way to get what they are interested in from the enormous volume of information available to them [50]. Basically, recommender systems use the opinions of users of a system to help individuals identify the information or products most likely to be of interest to them or relevant to their needs [29].

An RS is an agent-based system which is a combination of an information filtering and intelligent agent system. It uses the stored preferences to locate and suggest items of interest to the users being served [9]. Recommender systems can also be considered as services which recommend users (optionally a ranked list of) new items such as movies, music, books, articles and news they would like, based on their stored choices. They are used in e-shopping, auction sites, or in recommender engines that recommend the new items mentioned above.

Recommender systems are usually classified based on how recommendations are made into the following categories [2, 55]:

- Content-based recommendations: Items similar to the ones the user has preferred in the past are recommended.
- Collaborative recommendations: Items that people with similar tastes and preferences preferred in the past are recommended.
- Hybrid approaches: These methods combine collaborative and content-based methods.

State-of-the-art methods show that recommender systems deal with the information load and as a result of this, increase the efficiency and improve the overall performance for both the user and the merchant [70].

1.2 Problem Definition

In today's world, many of the commercial applications deploy a recommender system to help users find what they look for or desire and also to increase the system's efficiency and productivity. Unfortunately, most of the state-of-the-art solutions are not capable of making ”quality” and ”useful” predictions or recommendations [61].

To overcome the uselessness and approve the quality of the systems, current approaches introduce the concept of ”social recommendation” and propose making the recommender systems more personalized with the help of social networks. Social recommender systems are based on the idea of preferences of the friends are mostly alike. Social recommenders exploit interests of the users whom are in the active user's social network. Most of the time, users of the system do not prefer spending time in defining their social network; they look for immediate and quality results without stating detailed information about themselves. When we do not have a lot of information available, it is crucial to interpret every single hidden data in the hope of discovering a valuable information. Therefore, it is important to analyze not only explicitly issued ratings and trusts, but also all information available to us; to find out more about user's characteristics and preferences.

Social network based recommender systems not only suffer from data sparsity, but also high computational complexity and training time are the impediments. Neighborhood based models do not scale linearly with increase in the number of users and items of the system; the system complexity grows up exponentially as the system expands. In the literature, matrix factorization models are proposed to decrease the complexity while keeping the quality at least the same.

1.3 Organization of the Thesis

This thesis will be structured in the following way:

In Chapter 2, detailed explanation about recommender systems, both a formal and more general definition is provided. Types of recommender systems and common techniques in recommender systems are presented. In addition, evaluation metrics and the most common datasets used in recommender systems are described.

In Chapter 3, trust based recommender systems are explained in a more elaborated way in the related work section. The definition and characteristics of trust are explained and types of trust information are described. The state-of-the-art solutions proposed in trust based recommender systems are given.

Chapter 4 includes the proposed method. The motivation behind the proposed method, description of our method and its details are discussed. The dataset characteristics are explained thoroughly to provide a better understanding of the dataset. Finally, details of the experimental setup are described mentioning selection of parameters in the experiments.

In Chapter 5 starts with the experimentation done in the scope of this work and continues with the evaluation of the experiments. The results of the proposed method are presented and comparison to the based methods are explained. Conclusion of the proposed method is also discussed in this chapter.

Chapter 6 draws the conclusions of this thesis work. In addition, some possible future work is stated.

1.4 Contribution of the Thesis

Trust based recommender systems emerged from making the recommender systems more likely to the real world. The proposed method is a trust based recommender system deploying a more personalized trust information with the help of local trust information.

This thesis is based on the idea of more specific trust information helps us to find the ”real” friends we are looking for. In order to reduce the computational complexity, trust information is exploited only for the users who are in need of this and are not happy with the predictions that they receive. Proposed method is based on Probabilistic Matrix Factorization providing time efficient results.

In the scope of this work, we have conducted several experiments to show that more personalized social recommender systems yield more quality and precise predictions.

CHAPTER 2

RECOMMENDER SYSTEMS

2.1 Why do we need recommender systems?

Recommender systems were proposed to solve two main problems that could not be addressed by current information filtering systems based on keywords. The first challenge caused by keyword-based information filtering systems is that when filtering with a single keyword, enormous number of on-topic documents are matched - discarding the quality of those documents. The second problem is not being able to deal with non-text documents that are based on human taste. Using human judgement to understand the quality of the documents and filtering non-text documents based on human taste is introduced by the recommender systems. For example, the Ringo system [60] is one of the first recommender systems deploying a collaborative filtering algorithm to recommend music. After Ringo system, in academia and commercial systems same techniques are applied to other art forms [29].

Typically, a recommender system compares the user's profile to some reference characteristics, and seeks to predict the "rating" that a user would give to an item. These characteristics may be from the informative characterics of the user or the user's social environment. There are two approaches to building recommender systems: Collaborative Filtering (CF) and Content-based (CB) recommending [43]. CF systems systems collect ratings given to items in a specific domain in order to have user feedback information to exploit similarities and differences among profiles of several users in determining how to recommend an item. On the other hand, CB methods provide recommendations by comparing content based representations of items to content based representations of user's liking. Hybrid recommendation systems combine CF and CB recommendation methods to make a prediction.

Two basic entities appear in a recommender system [70]: users and items. Items are the products entered to the system. Items may be any kind of product such as movies, music, books, articles, news, etc. Users are the people who utilize the recommender system; they provide their opinions about the item and receive recommendations from the system.

Recommender systems operate in e-business domains benefiting both the merchant and the customer. They benefit the customer by suggesting items that s/he is probably going to like and benefit the business people by increasing the sales after presenting customers the items that are likely to be interest of them [70].

2.1.1 Inputs of a Recommender System

There are various utility functions in the current recommender systems for estimating the recommendations for the users. Depending on the type of the algorithm in utility functions, different types of data can be provided as an input to the system [70]:

- *Ratings* are the opinions of users on items which are stated to the items by the users explicitly or implicitly. Explicit ratings may be binary (0 for dislike and 1 for like) or in a range such as from 1(bad) to 5(excellent). Implicit ratings can be gathered from browsing habits, user's purchases, search or history logs, etc.

- *Demographic data* represents the characteristics like age, gender, country, education of the users. Demographic data is provided explicitly by the user.

- *Content Data* refers to the information about the items that are rated by the user. Content data includes the extracted features of the item that are captured during a textual analysis of the item's description.

- *User Relationships* consist of like, dislike, trust, distrust relationships among users. Reputation can also be a type of user relationship.

2.1.1.1 Data Collection Techniques

Ratings can be gathered by explicit or implicit data collection. Explicit data collection is also known as active filtering, whereas implicit data collection is called as passive filtering.

Active filtering is used in systems where people with similar interests rate products. Explicit data collection of user preferences requires the evaluator to indicate a value for an item on a rating scale. Rating scale can be binary (i.e. likes/dislikes) or in a range such as 1-5, 1-10, etc. Ratings based on a binary scale are more limited compared to larger rating scales.

Active filtering has an advantage of making recommendations based on reliable sources, because it uses explicitly stated ratings given to similar products by users. Additionally, expressing a rating value to an item in an already defined scale is more accurate compared to implicit data extraction techniques such as clicking a link or time elapsed on a page visit.

On the other hand, the ratings could be biased. Some users tend to give extremely high/low ratings to the items. Another problem is that, since providing feedback requires explicit action by the user, less data may be available compared to a passive approach. In fact, active filtering may cause prediction for an item to be overestimated; to overcome this, the system should scale ratings in an intelligent way. For instance, an item which has an average rating of 5 and rated by 3 distinct users, will appear as a more liked item than an item with an average rating of 4 and rated by 100 distinct users.

Passive filtering deals with indirect indications of the users. In this approach, it is assumed that user actions ”imply” their opinions. However, in implicit data collection, extraction of user opinions cannot be performed as accurate as in explicit data collection.

Implicit data collection includes following and measuring user actions such as purchasing history, repeatedly viewing an item information, clicking related links to an item etc. Time spent while user is dealing with an item is one of the most important features in passive filtering. For instance, a movie trailer may be watched by a user as a whole or partially or may be closed after just a few seconds. These actions indicate liking of the user for this trailer. Whole watching of the trailer may be treated as a high rating, whereas closing the trailer in a short time may indicate a dislike rating.

2.1.2 Outputs of a Recommender System

State-of-the-art methods illustrate recommender systems with two kinds of outputs [55, 70]; output of a recommender system can be either prediction or recommendation.

- *Prediction* is a value representing the degree of rating for the active user-item pair that is estimated by the system. Predicted value should be in the range of the ratings that a user may give to an item such as binary, 1-5, 1-10 etc.

- *Recommendation* is a list of items that are estimated to be most liked items by the active user. Items specified in recommendation list must be not already consumed by the active user. Listing N items as the output is also known as top-N recommendation.

2.2 Formal Definition

In a commercial recommender system, users and items - aforementioned two basic entities can be defined as sets. The set of users contains all the users in the system, ranging from hundreds to millions of users depending on the dataset. The set of items consists of all the items in the system, including movies, music, books, etc.

A formal definition of recommendation can be given as follows[2]: Let S be the set of users and T be the set of items in the system. Let u be the utility function that measures the usefulness of an item j to a user i, i.e. $u : S \times T \rightarrow R$, where R is a totally ordered set (e.g., nonnegative integers or real numbers within a certain range). Then, for each user $i \in S$, we want to choose such item $j' \in T$ that maximizes the user's utility. More formally;

$$\forall i \in S, j'_i = arg \max_{j \in T} u(i, j) \tag{2.1}$$

2.3 Recommender Systems Approaches

In academia, researchers have done a lot to solve the problem of recommending items, and different approaches are proposed to offer a solution. As stated before, classification of recommender systems is based on how recommendations are made. In content based recommendation, the system tries to recommend items that are similar to the user's preference history, whereas in collaborative recommendation the system tries to find users with similar likings to the active user and the items that they preferred are recommended. In hybrid methods, content based and collaborative recommendation are combined [9].

In this section, the different types of recommender systems - content based, collaborative and hybrid recommender systems will be described.

2.3.1 Content Based Recommender Systems

A content-based filtering system suggests items to the active user depending on the content correlation between item in consideration and past preferences of the active user [69]. For each item in the dataset, items are compared with preferences of the active user. Items that are mostly correlated with user's preference history are added into recommendation list.

In content-based recommender systems, the utility $u(i, j)$ of item j for user i is estimated based on the utilities $u(i, j')$ assigned by user i to items $j' \in S$ that are "similar" to item j [2].

As an illustration, for a text-based dataset, words in a document are extracted as a set of terms [3]. Prefixes and suffixes of words are omitted and words are grouped according to their stems. For instance, the words *computing*, *computers* and *computer* could all be reduced to the stem *comput* [69]. Profile of a user is represented with the same terms. Content of the documents that user has found interesting are analyzed and the user profile is formed. Feedback stated either explicitly or implicitly helps in determining the documents that are likely to be interest of the user. Evaluating examined documents on a scale is a way of providing explicit feedback whereas observing and exploiting user actions to understand user's interest is implicit feedback. Implicit feedback is more convenient for the user but more difficult to deploy in a recommender system.

Content-based recommendation has its roots in information filtering. After the significant and successful implementations in text based domains, recommender systems built on context based approaches emerged. The advancements in traditional approaches promote building user profiles that contain the information about users' tastes, previous likings and past behaviors.

Consider a movie recommender system application: the system profiles user i, and then tries to understand his behavior by analyzing the highly rated items by himself. Highly rated items may belong to a specific genre, movies of the same director, and may share the same actors, actresses or may be even in the same period of production dates. For example, a user who has liked ”Reservoir Dogs” and ”Pulp Fiction”, may like another movie which is also directed by Tarantino; so that ”Kill Bill” may be recommended to that user.

Profiling items is a process of defining items as a set of characterizing attributes. It is usually computed by extracting a set of features from the item. Item profiles are used to determine the appropriateness of that item for recommendation and prediction purposes. The importance, i.e. weight, of the features in the scope of the context can be defined in several ways. For text-based items, one of the most common ways to compute keyword weights is the *TF-IDF* measure, *term frequency - inverse document frequency* [57]. If there are *k* number of keywords in consideration; regardless of the weight computation method, an item profile may be defined more formally as in Equation 2.2 [2]:

$$Content(j) = \left(w_{1j}, w_{2j}, ..., w_{kj}\right) \tag{2.2}$$

where w_{kj} denotes the weight for keyword k in item j.

On the other hand, content based recommender systems recommend items to the users that are similar to the items previously liked by that user, therefore it is important to build up profiles not only for items, but also for users. More formally, let $ContentBasedProfile\,(i)$ be the profile of user i containing preferences and tastes of this user [2]. These profiles are constructed by using keyword analysis techniques in information retrieval after analyzing the content of the items previously seen and rated by the user. Let $ContentBasedProfile\,(i)$ be defined as a vector of weights $(w_{i1}, w_{i2}, ..., w_{ik})$, where each weight w_{ik} denotes the importance of keyword k to user i. Each weight can be computed from individually rated content vectors using a variety of techniques such as TF-IDF (Term Frequency- Inverse Document Frequency) scores, cosine similarities or Pearson Correlation Coefficients [2].

Reviewing the utility function, we can say that a content-based recommender system is defined as:

$$u(i, j) = score\left(ContentBasedProfile\,(i)\,, Content\,(j)\right) \tag{2.3}$$

In most of the commercial content-based systems, the utility function is computed using the cosine similarity measure [9] which is discussed in Section 2.3.2.

2.3.2 Collaborative Recommender Systems

Collaborative filtering is a classical method in information retrieval and the mostly used approach in recommender systems to deal with information overload [34]. While recommending a new iotem or predicting the utility of an item for a user, collaborative filtering algorithm makes use of user's previous preferences and the opinions of other like-minded users [55]. Collaborative filtering is based only on past user behavior and does not create explicit user profiles to suggest an item, user's previous likings or product ratings are taken into account [30].

In collaborative recommender systems, the utility $u(i, j)$ of item j for user i is estimated based on the utilities $u(i', j)$ assigned to item j by those users $i' \in S$ who are "similar" to user i where S represents the set of users[2].

For instance, if a user i has given similar ratings to any item with users i'_1 and i'_2, then the items that are rated by i'_1 and i'_2 but not by user i may also be recommended to user i. In other words, collaborative filtering tries to find like-minded users for user i.

There have been many collaborative filtering systems proposed in the industry and academia. In academia, the first collaborative recommender system developed is argued to be Grundy system [52], which proposed a system using stereotypes for constructing user models based on a little amount of information about each user. In the industy, a well-known application of collaborative recommender system is Amazon.com, an e-commerce web site [33]. When sufficient number of ratings are available, CF becomes the preferred and accurate technique according to the previous researches and successful stories of commercial websites such as Amazon, TiVo, and Netflix [33]. There are several notable reasons for making collaborative filtering the most preferred recommender system approach. Collaborative filtering requires no domain knowledge [30, 32] and there is no need for huge number of data collection. Moreover, being based solely on user ratings allows finding and exploiting unexpected and complicated patterns that would be hard or impossible to profile using known features of data.

Consider the movie recommender systems domain: to make a recommendation for user i, the system tries to find other users who have also rated the same items with user i, those users can be renamed as *like-minded users* for user i or *peers* of user i. Then, the movies that are rated highly by those *peers* are recommended to user i. For example, if a user liked "Reservoir

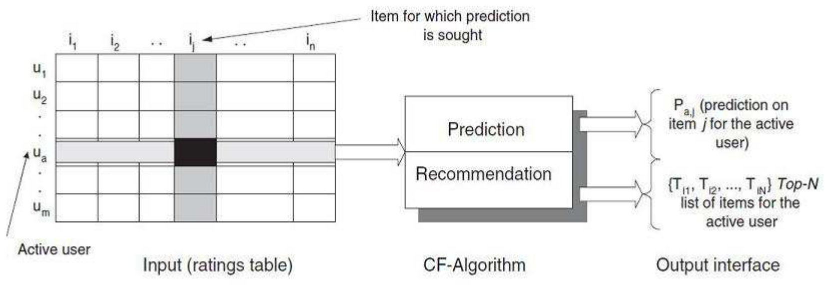
Item for which prediction is sought
i_1 i_2 . . i_j . . i_n
u_1 u_2 . . u_a . . u_m
Active user
Input (ratings table)
Prediction
Recommendation
CF-Algorithm
$P_{a,j}$ (prediction on item j for the active user)
$\{T_{i1}, T_{i2}, ..., T_{iN}\}$ Top-N list of items for the active user
Output interface

Figure 2.1: Collaborative Filtering process [55]

Dogs" and "Kill Bill", the system tries to find users that also have similar tastes with that user, i.e., rated the same movies. If users who have rated these two movies and also liked another movie "Illusionist" is found, then the movie "Illusionist" may be recommended to that user.

In collaborative filtering algorithm, the system can be represented as a user-item ratings matrix consisting of the rating $r(i, j)$ representing the rating user i has given a rating r on item j. This matrix is formed as a list of N users and list of M items, $S = \{u_1, u_2, ..., u_N\}$ and $T = \{i_1, i_2, ..., i_M\}$, respectively. Since there are M number of items in the system, each user can be modeled by an M dimensional vector. The vector consists of the ratings that user has given to each item. Since the system does not depend on data attributes, the items are not modeled with their features.

Figure 2.1 presents an overview for the recommendation and prediction processes in a collaborative filtering based recommender system [55]. Initially, the system extracts user vectors, as discussed in the previous paragraph. Similarity calculations between each user are carried out by a similarity function which gets two of any user vectors in the system and outputs a similarity score. Similarity scores are used to find out the most similar users of the active user. Only those found users are used in the process in order to produce either prediction or recommendation.

A simple example input to a collaborative filtering algorithm is shown in Figure 2.1; a user-item ratings matrix representing user's rating of that item. In the Table 2.1, 5 users and 8 items can be seen. In this table, the user-item pairs without a rating represent the items which the user did not rate.

The matrix shown in Table 2.1 is also an example of a sparse ratings matrix. A fully dense

Table 2.1: User-Item Ratings Matrix

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
u_1	3		4	4		2	5	
u_2				5		3	5	
u_3	4	5	4				4	3
u_4	2		5	5	4			3
u_5		5		3				

table would have $5 \times 8 = 40$ rating values; in this example we have 20 ratings, the ratings density is $20 \div 40 = 0.5$. However, in real world recommender systems the datasets are not so dense, even the datasets with enormous number of ratings have less than 0.1 densities. The data sparsity problem is examined thoroughly in drawbacks of collaborative filtering systems.

Various measures are used to compute the "similarity" of users u and u', namely $sim(u, u')$. Similarity measurement is based on the commonly rated items by both users. Most popular approaches are cosine similarity and correlation.

Cosine based similarity defines two users as two vectors in M dimensional item-space. Similarity between these two vectors is measured by computing the angle between them [13]. Cosine similarity is defined in Equation 2.4, where $\vec{u} \cdot \vec{u'}$ denotes dot product between the vectors $\vec{u}$ and $\vec{u'}$.

$$sim(u, u') = \cos(\vec{u}, \vec{u'}) = \frac{\vec{u} \cdot \vec{u'}}{\left\|\vec{u}\right\| \times \left\|\vec{u'}\right\|} \tag{2.4}$$

Correlation based similarity defines the similarity by computing the Pearson-r correlation between users u and u'. Let $S_{uu'}$ be the set of items that are co-rated by the users u, u'. Correlation based similarity is defined in Equation 2.5, where r_{ui} denotes the rating given by user u on item i and $\overline{r_u}$ denotes the average rating given by user u [2, 55].

$$sim(u, u') = corr(u, u') = \frac{\sum_{i \in S_{uu'}} (r_{ui} - \overline{r_u})(r_{u'i} - \overline{r_{u'}})}{\sqrt{\sum_{i \in S_{uu'}} (r_{ui} - \overline{r_u})^2 \sum_{i \in S_{uu'}} (r_{u'i} - \overline{r_{u'}})^2}} \tag{2.5}$$

It is crucial to note that, the same cosine similarity measure given in Equation 2.4 is used both in content-based and collaborative recommender systems. The difference between the two is in computing the vectors for users; in content-based approach, the vectors are computed with the help of TF-IDF scores, whereas in collaborative approach, each item in the system represents a dimension for the user vector.

2.3.3 Hybrid Recommender Systems

A hybrid recommender system recommends items to users via a combined method of both content-based and collaborative filtering approaches. The reason behind building hybrid recommender systems is to eliminate the deficiencies of both content-based and collaborative approaches which will be detailed in Section 2.4. Commonly, there are four ways to construct a hybrid recommender system which will now be discussed in detail [2, 63, 66]:

- *Combining separate collaborative and content-based recommenders* is a way to build hybrid recommender system. In this approach, after implementing separate recommenders, the results of these recommenders are combined in several different ways. One way is to combine outputs of these recommenders linearly or using a weighted scheme [14]. The other way is using an evaluation metric and dynamically choosing the best recommender using that evaluation metric at any time [18].

- *Content boosted collaborative filtering recommenders* are based on collaborative filtering techniques, but in addition to this content based profiles of users are created. The traditional Fab system [9] and [48] are examples of collaborative recommender systems that also store content-based user profiles. The use of content based user profiles handles cold start user problem of collaborative filtering by deploying the user information in the prediction process. Moreover, overspecialization problem of content based filtering is eliminated by this technique, since recommendations are performed not only based on user content profiles but also commonly rated items.

- *Collaborative content-based approach* uses a collaborative view of content-based user profiles. Recommender systems using dimensionality reduction techniques [62] are examples of this approach. Compared to traditional content-based recommender systems, this approach outputs better performance.

- *Unified content based and collaborative filtering* recommender systems are proposed in most recent systems. [11] is a rule-based classifier defining its rules using both content-based and collaborative approaches. Based on probabilistic latent semantic analysis, [58, 49] propose a unified model combining content-based and collaborative recommendations. Another Bayesian networks technique based on Markov Chain Monte Carlo methods for parameter estimation is proposed in [5, 19]. This technique uses user attributes to construct user profiles, item attributes to construct item profiles and the interactions to predict ratings. Another technique unifying content-based and collaborative approaches is a case-based reasoning method addressing new user and new item problems of recommender systems [17]. In [17], a knowledge-based restaurant recommender system suggesting food for people is proposed. To make better recommendations in this system, background knowledge about food should be available, such as ”Seafood is not vegetarian”. In a knowledge based recommender system, recommendations are performed depending on existing knowledge in a given domain and user-item matrix; however, knowledge acquisition for each item still remains as a problem. Domains in which information can be gathered in a machine readable format may be the target of knowledge based recommender systems. After making information available in such a system, the effect of cold start user or new item problems will be degraded.

2.4 Drawbacks of Recommender System Approaches

2.4.1 Content Based Recommender Systems

Content based methods are the preliminary proposed solutions to the recommendation problem. In most commercial systems, the content data of items cannot be extracted easily. First of all, tagging an item with a set of identifiers is performed manually for complex data types such as movies, videos, music, etc. Even if the system has most of the content information available, user’s liking of an item cannot be understood completely. A system, which is solely based on content analysis, is not capable of finding the reason why a user likes a certain item. For instance, a user may like a movie because of its director, genre, artists or just the soundtrack. However, soundtrack information may not be available in the system and the system may be searching for a similarity in unrelated fields. On the other hand, a generic content based recommender system makes recommendations only similar to the past user preferences. Therefore, the user cannot see a varied set of items in his recommendation list or the system cannot predict the user's opinion about a distinct item.

A pure content-based recommender system may suffer from a few shortcomings. Now, let us examine them closely [2, 9, 60].

Limited content analysis is the most important problem. In some domains, the items cannot be extracted into their features automatically, they should be analyzed manually or should be in a machine parsable form, such as movies, music, restaurants. Unfortunately, it is not easy to handle all the items in a system manually, and usually due to the limitation in resources, it is not possible to identify the items. Even if some techniques are defined to handle them, side effects may occur; such as ignoring the usefulness, quality and loading times of web pages while interpreting their texts.

Over-specialization is the second problem. When the system profiles each user based on his past likings and behavior, the user is limited to only being recommended of similar items. There is not a diversity in the range of recommended items; usually they turn out to be too similar so that the recommendations become less useful as the time passes and the system learns the user. In practice, randomness is injected to introduce some element of serendipity.

New user problem is a result of the backbone of content-based systems; it is caused by profiling users. Since the new coming user has not stated any preference about the items, the system cannot understand the user. Therefore, a cold start user may not receive accurate recommendations or predictions.

2.4.2 Collaborative Recommender Systems

Cold-Start (New User) problem of the content-based recommender systems still exists in collaborative recommender systems [2, 55]. When a new user enters to the system, the system knows nothing about him/her. This problem is also named as the new user problem in recommender systems, and is caused by new users inserted in the system who have not submitted any ratings. Without any information about the user preferences, the system is not capable of making a guess about the user's preferences and generating recommendations until a few items have been rated by that user. In order to make more personalized predictions, the system should collect some information about the new user. The most simple solution to gather enough information about the user is done by directly asking for ratings by presenting items to the user and this approach is still used by currently available recommender systems [42, 45].

First-Rater problem is brought by the new items in the system that have not yet received any ratings from any users. Recommender systems are always updated and so that the new items are added into the system. Since the item has not received any rating yet, the system cannot define semantic interconnections of this item to the other items in the system; and as a result, it is never recommended to any user [10]. Therefore, an item cannot be recommended unless it is rated by at least one user before. This problem is valid for items that are newly added to the system and also for the users with selective tastes [43].

Sparse ratings - Sparsity is caused by fewer number of ratings. Ratings matrix sparsity is the percentage of empty cells in user-item ratings matrix. In commercial systems, since there are enormous number of items, users are not capable of rating most of the items and so that a typical user-item ratings matrix is very sparse. In a sparse ratings matrix, mostly, it is not possible to find a number of users with significantly similar ratings. This usually occurs when systems have excessive number of items and users. For instance, in Amazon [4, 57], even very active users have purchased significantly less than 1% (i.e. about 20.000 items) of the items, since the system contains about 2 million books. According to this ratio of sparseness, generic nearest neighbor algorithms may not be able to make item recommendations for particular users. As a result of sparseness, the prediction accuracy of recommender systems becomes low.

Scalability As discussed in sparsity problem of recommender systems, most of the recommender systems have excessive number of items and users. In order to keep a recommender system up, it is expected to get a response from the system in a reasonable time, even though it is not so easy to handle huge amount of data. Furthermore, storage complexity of the system also increases along with the number of users and items. Both time and storage complexities make the system hard to scale and suffer serious scalability problems.

In an industry report published by Amazon.com [33], Amazon is declared to have an enormous dataset, consisting of 29 million customers and several million catalog items. However, current recommendation system methods are based on more scalable datasets- such as MovieLens with 35,000 customers and 3,000 items. For large datasets, it is more convenient to perform expensive calculations offline.

To continue with the scalability problem, the computation complexity should not grow exponentially with the number of customers and items. Therefore, using dimension reduction, sampling or partitioning could make the system more scalable. Cluster models may be considered in an offline method but in general their quality is less than online methods [33].

2.5 Common Techniques in Recommender Systems

2.5.1 Neighborhood Based Models

The most common approach to collaborative filtering based recommender systems is neighborhood models. Neighborhood models were mentioned in Section 2.3.2. There are basically two types of neighborhood models; one is the user oriented approach and the other is the item oriented approach.

In user oriented approach of neighborhood based models, for each prediction a set of similar users is selected who have rated the active item. Or, in item oriented approach, a set of similar items that are rated by the active user is selected. The prediction is computed from the ratings of the similar users set or analogously, from the set of similar items [65].

In real-world applications, the number of users is extensively larger compared to the number of items in the system. Therefore, usually it is expensive to store user to user similarities in memory and item oriented neighborhood models are preferred.

Now, let us define item oriented neighborhood models more formally. The set of items rated by user i are denoted by T_i and neighborhood set of query (i, j) is denoted by N_{ij}. Note that, $N_{ij} \subseteq T_i$. The prediction of the rating given by user i to the item j is [65]:

$$\hat{r}_{ij} = \sum_{k \in N_{ij}} w(i, j, k) \, . p(i, j, k) \tag{2.6}$$

where $w(i, j, k)$ is the k-th interpolation weight and $p(i, j, k)$ is the k-th subprediction at making prediction for (i, j)-th rating. There are several proposals to define N_{ij}, $w(i, j, k)$ and $p(i, j, k)$.

A simplified illustration of user oriented neighborhood models on movie recommenders do-

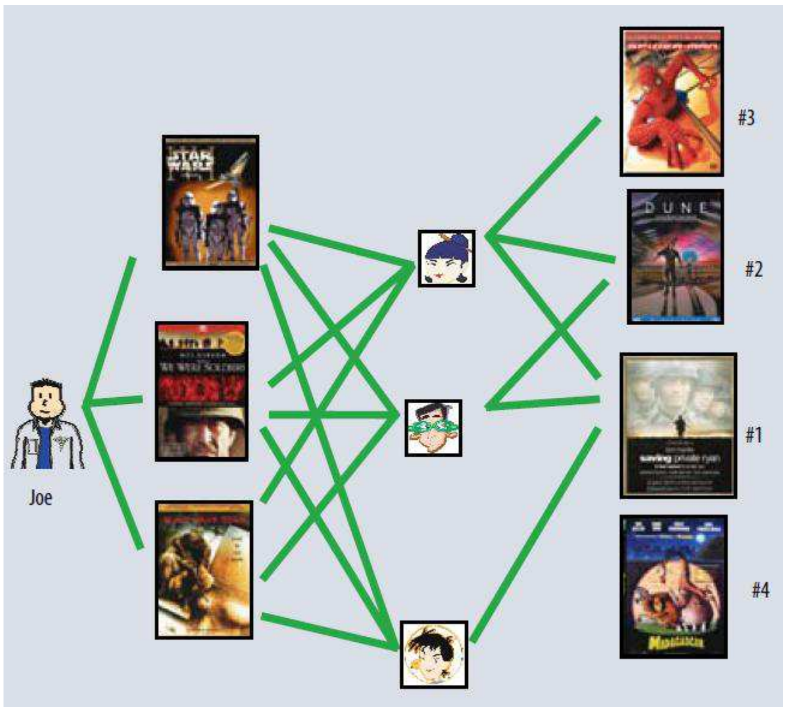

Figure 2.2: User Neighborhood Model Illustration [31]

main is given in Figure 2.2. In this figure, user *Joe* and three other users have liked the same three movies. The query is to recommend *Joe* a new movie. *Saving Private Ryan* is liked by all three users; so that the most liked movie among those three users is *Saving Private Ryan*, the second is *Dune* and the third *Spider man*. Therefore, *Saving Private Ryan* will be the first movie to recommend *Joe* [31].

2.5.2 Matrix Factorization Models

Matrix factorization (MF) is one of the well-known collaborative filtering approaches in the literature. MF is based on the idea of simulating users' and items' characteristics by presenting them with a small number of features. MF does not deal with the content information of users or items.

A linear factorization model, N number of users are modeled with a $N \times d$ matrix U, where d is the dimension size for features. In the same way, M number of items can be presented with a $M \times d$ matrix V. The preferences of users are modeled by the product of transpose of user matrix with the item matrix.

Matrix factorization models map users and items to a latent factor space in d dimensions, where d represents the dimensionality of the space. In the space, the ratings are modeled as inner products of user and item vectors. More formally, each item is presented with a vector $q_j \in \Re^d$ and each user is represented with a vector $p_i \in \Re^d$; and each rating is predicted by the product of vectors given in 2.7 and the training error measured for (i, j) th rating which is given to item j by user i is defined in Equation 2.8 [30, 65]:

$$\hat{r}_{ij} = {q_j}^T p_i \tag{2.7}$$

$$e_{ij} = r_{ij} - \hat{r}_{ij} \tag{2.8}$$

where $\hat{r}_{ij}$ denotes the prediction of the rating user i would give to the item j and e_{ij} denotes the training error for that user-item pair.

To learn the factor vectors p_i and q_j, the model should minimize the regularized error on the set of known ratings R as given in Equation 2.9 [31]:

$$\min_{q^*,p^*} \sum_{(i,j)\in R} \left(r_{ij} - {q_j}^T p_i\right)^2 + \lambda\left(\left\|q_j\right\|^2 + \|p_i\|^2\right) \tag{2.9}$$

In Equation 2.9, R denotes the known ratings set. The system learns the model by fitting previously given ratings. However, while learning from previously observed ratings, it is important to be able to make predictions for unknown ratings; so that the system should avoid overfitting by regularizing the learned parameters. The constant λ in this equation controls the extent of regularization. Several researches in academia propose different techniques for regularization.

A simplified illustration of latent factor approach on movie recommenders domain is given in Figure 2.3. In this figure, both users and items, i.e. movies, are characterized in two axes [31]. The users tend to like items that fall near themselves in the latent vector space.

Latent factor models have become widely used in recent applications due to their flexibility to fit real-world problems. These methods output more accurate predictions and scale better compared to the other approaches [30, 31].

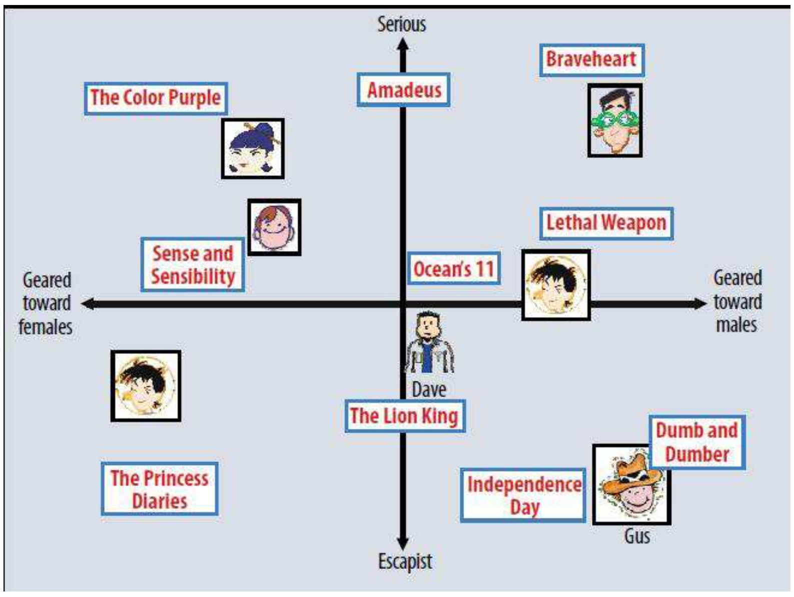

Figure 2.3: Matrix Factorization Illustration [31]

2.6 Evaluation Metrics for Recommender Systems

In order to compare the performances of the algorithms, we need to pick evaluation techniques used for recommender systems.

Leave-one-out technique is one of the most commonly used techniques in recommender systems. In this technique, one rating value is predicted and then the predicted value is compared with the original rating value. After applying the same principle for all the rating values in the dataset, the differences between the predicted values and the real values are used for calculating the error using a method such as Mean Absolute Error or Root Mean Squared Error.

Root mean squared error (RMSE) is defined as [64]:

$$RMSE = \sqrt{\frac{1}{|R|} \sum_{(i,j)\in R} \left(\hat{r}_{ij} - r_{ij}\right)^2} \tag{2.10}$$

where R is the finite set of tested rating values defined for user-item (i, j) pairs and $|R|$ is the number of elements in R. r_{ij} is the actual rating value given by user i to item j and $\hat{r}_{ij}$ denotes prediction of the rating that user i would give to item j.

Mean Absolute Error (MAE) is defined as:

$$MAE = \frac{1}{|R|} \sum_{(i,j)\in R} \left|\hat{r}_{ij} - r_{ij}\right| \tag{2.11}$$

where R is the finite set of tested rating values defined for user-item *(i, j)* pairs and $|R|$ is the number of elements in R. In Equation 2.11, r_{ij} and $\hat{r}_{ij}$ denote the same representations as in Equation 2.10.

In the scope of this work, instead of using leave-one-out technique for evaluation, the dataset is divided into three parts- training, validation and test data and the predictions for the test data is done as a whole. All the error calculations are done for the three data parts- data is treated as a whole without breaking the integrity. Training data is used for the algorithm to learn the characteristics of the dataset, whereas validation and test data are completely unseen for the algorithm. Then validation dataset is used to make improvements and used as a buffer- here test data is still unseen. Afterwards, evaluation of the algorithm is done on the test dataset using four evaluation metrics - Mean Absolute Error, Root Mean Squared Error, Mean Absolute User Error and Mean Absolute Item Error - which will be now explained in detail.

Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) are used to measure the accuracy of the implemented algorithms. However, those measures are not always informative about the quality of the predictions in a recommender system, especially when there are both heavy raters and cold-start users. For instance, let us suppose there is a small system of 101 users. In this system, each of the 100 users rated 1 item and the remaining 1 user rated 100 items. In this case, the predictions for the 100 users will not be realistic and they all will be unsatisfied. On the other hand, 1 user who has rated 100 items will receive almost perfect predictions. As a result, 1 out of 101 users will be happy and the other 100 will be unhappy. Consequently, to handle this issue, other measures called Mean Absolute User Error [37, 39] and also Mean Absolute Item Error are used in evaluation. To sum up, there are two conventional ways to compute MAE. One way is the micro-averaged MAE, which is given in the Equation 2.11, and is computed by pooling all user instances from the test set. The other way is macro-averaged MAE and based on computing MAE on test instances of each user and then taking average of the per-user MAE values [71]. Mean Absolute User Error and Mean Absolute Item Error are the macro-averaged Mean Absolute Errors.

Mean Absolute User Error (MAUE) [37, 39] indicates the mean absolute error rates for each specific user. MAUE measures the users' satisfaction from the system in a more accurate manner. More formally, MAUE is defined as:

$$MAUE = \frac{1}{N}\sum_{i=1}^{N} MAE_i \tag{2.12}$$

$$MAE_i = \frac{1}{M}\sum_{j=1}^{M} \left|\hat{r}_{ij} - r_{ij}\right| \tag{2.13}$$

where N represents the number of users in the test set, and i represents the active user and Equation 2.13 is computed for all users. In Equation 2.13, M is the number of items that the active user rated and $\hat{r}_{ij}$ is the predicted rating given to the item j by active user i and r_{ij} is the actual rating value given to the item j by active user i.

Mean Absolute Item Error (MAIE) is defined in the scope of this thesis work, as a complementary method to MAUE. MAIE is used to measure the accuracy of predictions for each specific item. MAIE is based on the idea of the sparsity of information related to an item affecting the prediction accuracy. For instance, for an item just added into the system, prediction accuracy would be low, because of the lack of rating information attached to the item. In the same way, formal definition of MAIE can be stated as:

$$MAIE = \frac{1}{M}\sum_{j=1}^{M} MAE_j \tag{2.14}$$

$$MAE_j = \frac{1}{N}\sum_{i=1}^{N} \left|\hat{r}_{ij} - r_{ij}\right| \tag{2.15}$$

In Equation 2.14 and 2.15, M represents the number of items in the test set, and j represents the active item and Equation 2.15 is computed for all items. In Equation 2.15, N is the number of users who have rated the active item and $\hat{r}_{ij}$ is the predicted rating given to the item j by active user i and r_{ij} is the actual rating value given to the item j by active user i.

2.7 Typical Datasets in Recommender Systems

2.7.1 Epinions

Epinions is a consumer reviews platform on the Web where users can review any kind of item, ranging from movies, cars, books to even home and garden, sports items, etc. In Epinions.com [22], the author of the review has to give a rating to the product that she is reviewing; the rating is in the range of 1 to 5. Users can give ratings to reviews: review ratings are *Not Helpful*, *Somewhat Helpful*, *Helpful*, *Very Helpful* and *Most Helpful*. There is also an ”Off Topic” rating which shows the review is posted under wrong subject. Users can express reviews with rating *Don’t Show* and *Show*; reviews are labeled with the one they select. If they think that the review violates site rules, ”Don’t Show” should be selected.

Users can have a ”Web of Trust”. *Web of Trust* is a list of reviewers whose ratings and reviews are found valuable by the active user. The Web of Trust mimics ”the way people share word-of-mouth advice every day” [22] - if a friend gives you good advice whenever you ask, you’re likely to take that person’s suggestions into consideration in the upcoming advices. If you and your friend both like the same types of films, you’re more likely to trust your friend’s recommendations on what to see.

Users can also have ”Block List”. *Block List* is a list of authors whose reviews are not found valuable by the active user. If the user encounters a member whose reviews are consistently irrelevant, inaccurate or somewise not reflecting her/his own ideas, user can add that member to her/his Block List.

Since each user has a ”Web of trust” and ”Block list”, we can say that users have binary trust values in Epinions.com. Each user may either trust a user or block that user; a user cannot express trust to another user with a value of e.g. 0.3. If a user wants to express trust or distrust to another user, adds her either to ”Web of Trust” or ”Block List.”

However, the block list is kept private in Epinions.com, so apart from the datasets that are given by the Epinions.com staff, none of the crawled datasets include distrust information.

Paolo Massa and Paolo Avesani used Epinions dataset derived from Epinions.com web site. The authors coded a crawler that recorded the ratings and trust statements issued by a user starting from a random seed user and then continued with the users trusted by those users and recursively did this [39].

Epinions dataset used in papers of Massa and Avesani contains 49,290 users who has given 664,824 ratings to 139,738 items with a rating matrix sparsity of 99.99135%. Each user in the dataset has rated at least one item. 49,290 users issued 487,181 trust statements to each other. The average number of the reviews is 13.49 with a standard deviation of 34.16. Large majority of users are the ones who are called cold-start users; 52.82% of the users, namely 26,037 users expressed less than 5 reviews. The average number of users in the Web of Trust (friends) is 9.88 with a standard deviation of 32.85. In the Epinions.com dataset, the average rating is 3.99 and almost half of the ratings are 5 which is the maximum possible rating value in Epinions.com web site [37, 40].

Epinions dataset used in [35] is crawled from the web site on January 2009. To give a brief outline of user-item ratings matrix and social trust networks of Epinions dataset, we can have a look at Table 2.2 and Table 2.3 [35]:

Table 2.2: Epinions.com User-Item Ratings Statistics

Statistics	User	Item
Max number of ratings	1960	7082
Avg number of ratings	12.21	7.56

Table 2.3: Epinions.com Social Trust Network Statistics

Statistics	Trust per User	Be Trusted per User
Max number of users	1763	2443
Avg number of users	9.91	9.91

The final version of Epinions dataset is the extended Epinions dataset [21]. Extended version is provided by the Epinions.com staff, so that the dataset includes distrust information.

To sum up all data information, characteristics of different Epinions datasets are provided in Table 2.4:

Table 2.4: Epinions.com Datasets

Statistics	Users	Items	Ratings	Trusts	Distrusts	Density
Trust Ens [35]	51,670	83,509	631,064	511,799	0	0.02%
Trust Aware [39]	49,290	139,738	664,824	487,181	0	0.01%
SoRec [36]	40,163	139,529	664,824	487,183	0	0.01%
Ext. Epinions [21]	132,000	-	1,560,144	717,667	123,705	-

In [35] and [36], the dataset consists of users who have rated at least one item.

2.7.2 IMDB - Internet Movie Database

IMDB allows the use of their data for non-commercial purposes and provides offline datasets. However, there are only global trust values- no local, for instance ”2 out of 4 people found the following comment useful”. The IMDB Terms of Service disallows crawling their web pages except with explicit permission.

2.7.3 Netflix

Netflix.com [45] provides Netflix dataset for the well-known million dollar programming prize - Netflix Prize [12]. In Netflix dataset a rating record is a quadruple $(i, j, r_{ij}, date_{ij})$ where ith user has rated jth item with the rating r_{ij} on date $date_{ij}$. Dataset consists of 4.2 million ratings given to 17770 movies by 480189 users. Ratings are on a five star (integral) scale from 1 to 5. Netflix dataset does not have any trust information. Netflix owns data about the movies such as actors, directors but they do not include it to the datasets.

2.7.4 Twitcritics

Twitcritics web site finds tweets about the new movies from Twitter.com [68] and automatically determines if they are positive or negative reviews. The main page shows a list of the movies that have been tracked so far, as well as their ’rating’. The rating is the percentage of tweets that are positive [67].

At first, we were impressed about the idea of collecting such information about the movies; however, when examining thoroughly to find out whether the information is useful or not, we encountered some missing classifications. To make things clear, let us look at an example: our focus will be on movie ”A Beautiful Life” and following a few tweets about it:

- *alixcart* - A beautiful life does not just happen, it is built daily by Prayer, Humility, Sacrifice and Love. May that beautiful life be urs always - Negative
- *kness16* - really love ”the beautiful life”..too bad it got canceled after 2 episodes - Negative
- *paulomontterus* - A beautiful life only live who have a beautiful attitude. - Negative

All of the above examples are classified as negative, but if we try to exploit and label them manually, the first and the third would be irrelevant and the second would be positive.

2.7.5 MovieLens

MovieLens is a movie recommendation website; MovieLens gathers information of movies that the users love or hate and by the help of this, tries to help users find the movies they will like. MovieLens datasets are available in three different formats:

- 100,000 ratings for 1682 movies by 943 users
- 1 million ratings for 3900 movies by 6040 users
- 10 million ratings and 100,000 tags for 10681 movies by 71567 users

MovieLens system does not have any trust information. In [59], GroupLens research group explores Tagommenders, which is composed of recommender algorithms that predict users' likings for items based on their inferred preferences for tags. In [59], in addition to the publicly available dataset, movie ratings, movie clicks, tag applications and tag searches are also used.

2.7.6 FilmTrust

The social networking in FilmTrust website requires each user to give a trust rating to the user they add as a friend. The trust rating represents how much they trust the user's like about movies. Relationships are only one-way, not symmetric. Users of the system can see the people they have trusted or the people who trusted themselves. However, to keep the system work well with honestly stated ratings, not all trust ratings in the system are visible to everyone. Trust ratings are kept confidential to prevent users from discouraging to give accurate ratings [23, 24]. Movie ratings are from half star to four stars.

In FilmTrust, a prediction of a rating is computed the trusted user's ratings using a weighted scheme. Trust values represent how much the active user trusts in the other user. The rating calculated by weighting the trusted friends' ratings of the movie is the user's opinion about that movie [25]. If the movie in consideration has received more than one rating, ratings are sorted according to the trust value of those trusted users.

Unfortunately, the dataset is not provided publicly and we do not have a FilmTrust dataset that we can analyze in detail.

CHAPTER 3

TRUST BASED RECOMMENDER SYSTEMS

3.1 Overview of Trust based Recommender Systems

Trust based recommender systems propose a new approach to select or weight users whose advices are asked and ratings are considered during the recommendation [15, 37, 46]. The idea behind trust based recommender systems is the observation of ”asking friends” while making a decision [16].

Social recommender systems propose that considering trustworthiness of a partner should be the standard basis for partner selection. In recent research, it is proved that trust relationships among users help the system to increase the coverage of recommender systems while preserving the quality of recommendations [29].

A system combining social networks and collaborative filtering is first proposed as a ReferralWeb [28] system aiming to expand users’ awareness of their existing communities by instantiating a larger community using their network information and make the user discover connections to people and information that would otherwise be hidden. This work uses the term ”social network” to include groups of people linked by professional activities. The network is modeled by a connected graph- nodes representing the users of the systems and edges representing the connections of the individuals. The network is then used as a guide to make queries searching for people or documentation. ReferralWeb is based on the idea of providing referrals via a chain of named individuals instead of unnamed, anonymous recommendations. ReferralWeb system is critical because it introduces the term trustworthiness. The reason to use named referrals is declared as ”... because not all sources of information is desirable”. For instance, a user may prefer considering referrals to people who are closely related to a trusted expert.

A research [61] comparing the quality and usability of recommendations made by online systems and friends showed that user's friends consistently provided better results than recommender systems. In this work, an empirical study comparing recommendations of a user's friends and recommendations of online recommender systems was conducted. In the context of [61] a good recommendation is a recommendation that interests the user, and a useful recommendation is the one that the user has not experienced before. The results of the experiment show us that users tend to prefer recommendations made by friends but since they like useful recommendations, they find recommender systems more effective. A recent study [20] on Amazon.com focused on the importance of understanding the factors affecting the helpfulness of the reviews, the relationship of the reviews and purchases.

Trust has many definitions in all disciplines and contexts. However, more specifically in recommender systems trust fall into two categories:

- *Context-specific/ interpersonal trust* describes a system where a user trusts another user with respect to one specific situation, but not necessarily to another. Context-specific/ interpersonal trust may also be referred as domain-expert [54]. The friend you ask animation movies may not be the same friend as from whom you get independent movies advice.

- *System/ impersonal trust* describes a user's trust in a system as a whole. System/ impersonal trust is also known as friendship-trust [54]. This represents the belief of a person who credits a friend's opinions irrespective of the context.

In trust-based recommender systems, trust information is gathered mainly in two ways: either by automatically inferring trust information from item ratings based data [46] or by using explicit trust statements given by the users [35, 36]. Also, there are methods using trust propagation which is a method to extract not specified trust relationships among users with the help of already existing trust statements. Trust propagation is a method based on implicit trusts [26].

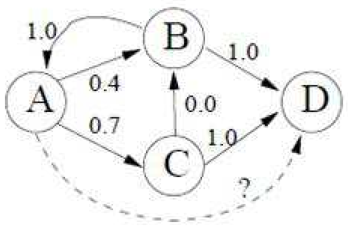

Figure 3.1: Trust based network representation

3.2 Characteristics of Trust

In trust based networks, the web of trust can be represented as a network - nodes are the users of the system and edges are trust statements.

In the Figure 3.1, we can see that user A has issued a trust statement in user B (with value 0.4), and also issued a trust in user C (with value 0.7), therefore user B and user C are in the web of trust of user A [37]. User C has issued a trust with value 0 in user B; corresponding to the block list of user C in *Epinions.com*.

Analyzing the characteristics of trust, it is important to note that:

- *Trust is not symmetric:* In the Figure 3.1, the relationship between user A and user B is an example of this fact. User A issued a trust in user B with a value of 0.4, whereas user B issued a trust with a value of 1.0 in user B.

- *Trust is subjective:* The values of trust issues are relative per user. In the figure, we can see that user A stated trusts with an average of 0.55, values are 0.4 for user B and 0.7 for user C. On the other hand, user B stated trusts with an average of 1.00, values are 1.0 for both user A and user D. This is a tiny example of differences in mean and standard deviations of trust values per users; the deviations differ in a small range, however mean values are based directly on the user opinions. Some users state higher values of trust, while some of the other users prefer trusts with small values.

The problem of trying to predict how much user A trusts user D may be inferred by propagating trust. If we already know that user A trusts user C, and user C trusts user D, we might be able to guess that user A trusts user D. This operation is referred as atomic propagation, since the conclusion is reached based on a single argument rather than a chain of arguments [26]. However, since trust is a subjective issue, a user who is trusted by the active user may be distrusted by another user. Trust propagation helps us to define a relevance measure among users besides ratings information, which can be used as an alternative or additional user similarity measure [39].

3.3 Local and Global Trust Information

In social networks, basically two types of trust information are used as stated in [39]: global and local trust information. Both local and global trust metrics try to predict the trustworthiness of a user.

Global trust information considers only the community's opinion about a certain user and does not take into account a single user's subjective thinking of that user [39]. Therefore, in global trust, the system can be personalized to some extent and this makes it hard to make predictions for certain users. While making a recommendation for a user, the system should apply the selectivity criteria defined by that user via previous ratings, trusted users, reviewed items, etc.

To illustrate, let us consider a system which only stores global trust information. User A has a high reputation among this system, but we know that user B distrusts user A. While making a prediction for user B, we have to consider user A's rating values, since user A has high reputation in the system.

Local trust metric tries to predict the trustworthiness of a user in a more personalized way. Local trust metric provides a trust score which depends on the personal opinion of the judging user [38].

Local trust metric is the complementary of global trust metric, it takes into account the active user's subjective thinking while calculating her trust score to another user. It allows people personalizing user trust metrics independent of other users based on their web of trust.

3.3.1 Comparison of Local and Global Trust

The local trust metric is defined over $S \times S$, where S is a vector representing the user set; therefore it is computationally expensive compared to the global trust, which is defined over S. In local trust metric, there are N number of trust scores for each user where N represents the user count in the system; whereas in global trust, there is only one trust score for each user, which is also known as the reputation of that user.

Therefore, in a recommender system global trust metric is computed for each user after a single run, whereas the local trust metric is computed for each user-user pair independently. This fact makes it harder to integrate local trust metric into the real world applications because of its computational time and storage complexity.

On the other hand, local trust information has a less coverage compared to the global trust metric since local trust metric needs more information about the user to predict a trust score. In a recent work, an idea of using global trust when no local trust metric is available is proposed [38].

Global trust information is more prone to malicious user attacks. The easiest way to make a malicious attack is to get a number of fake user accounts, and make all fake users trusting to each other. In this way, global trust metric based recommender system will assign a high trust value (reputation) to those fake users. On the other hand, local trust metric will exclude those users from the trust propagation process and will not affect the personalization of other users who don't trust the fake users explicitly [6]. Therefore, local trust metric is more attack-resistant.

3.4 Examples of Trust-based Recommender Systems

3.4.1 Trust-aware Recommender System [37, 39]

Architecture

Trust aware Recommender System (abbreviated by the authors as TaRS) is a trust-aware collaborative filtering for recommender systems proposed in [37, 39]. The architecture of TaRS is given in Figure 3.2.

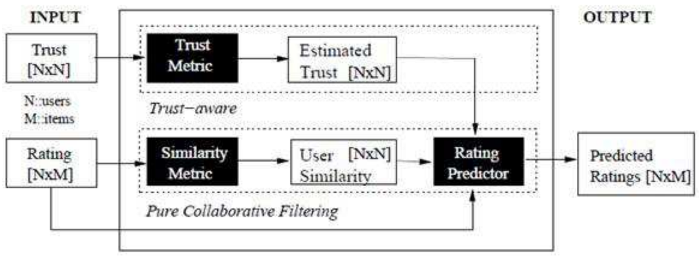

Figure 3.2: Trust-aware Recommender System (TaRS) Architecture [37]

The input of the recommender system is not only the user item ratings matrix, but also user to user trusts matrix representing the explicit trust statements issued by the users to each other. The output is predicted ratings matrix containing the predictions of ratings that the users would give to the items.

Trust Metric Module represents the trust based component of TaRS. This module takes user to user trusts matrix as an input and outputs an estimated trust matrix, which is hopefully a dense matrix compared to its input. The aim of this module is to construct a new social network based on estimation of how much each user would trust every other user in the system. The conversion algorithm depends on a predetermined maximum propagation distance. According to this approach, after selecting the maximum propagation distance, d, predicted trust will be $(d - n + 1)/d$, where n represents the user distance from source user. Users that are not reachable from the source user within distance d are not neighbors of the source user and they are not taken into consideration in the evaluation.

On the other hand, Similarity Metric Module implements a generic CF approach algorithm based on Pearson Correlation Coefficient similarity calculation. Input of this module is the standard user-item ratings matrix and it outputs user to user similarity matrix. User to user similarity is calculated using the correlation between users via already given ratings to the items in the system. Pearson Correlation Coefficient is the mostly used technique for computing user similarities [27] and it is implemented as in Equation 3.1:

$$w_{a,u} = \frac{\sum_{i=1}^{m} (r_{a,i} - \overline{r}_a)(r_{u,i} - \overline{r}_u)}{\sqrt{\sum_{i=1}^{m} (r_{a,i} - \overline{r}_a)^2 \sum_{i=1}^{m} (r_{u,i} - \overline{r}_u)^2}} \tag{3.1}$$

where a is the active user, u is the user in consideration, i is the item in consideration, $r_{a,i}$ is the rating provided by user a to item i, $\overline{r}_a$ is the average rating provided by user a and m is the number of items commonly rated by user a and user u. Result of this calculation, $w_{a,u}$, represents the similarity weight between the active user and the user in consideration.

Rating Predictor Module combines results of Trust Metric and Similarity Metric Modules. Matrices, estimated trust matrix and user similarity matrix are used to find out neighbors of the active user in the system. Built-in algorithm used in Rating Predictor Module can be varied in different systems. As a result of extracted neighbors of the active user, predicted ratings for user-item pairs are calculated as the following formula in Equation 3.2:

$$p_{a,i} = \overline{r}_a + \frac{\sum_{u=1}^{k} w_{a,u}(r_{u,i} - \overline{r}_u)}{\sum_{u=1}^{k} w_{a,u}} \tag{3.2}$$

where a is the active user, u is the user in consideration, i is the item in consideration, $r_{u,i}$ is the rating provided by user u to item i, $\overline{r}_u$ is the average rating provided by user u and k is the number of selected neighbors. $w_{a,u}$ is already calculated similarity weight as a result of Equation 3.2. $p_{a,i}$ is the predicted rating provided by user a to item i.

Evaluation

Experiments of the system TaRS are done on Epinions dataset. The dataset is crawled by themselves. Dataset contains trust statements, users, items and ratings given to the items by those users. Description of the dataset is provided in Table 3.1:

Table 3.1: TaRS - Epinions dataset description

Users	Items	Reviews(Ratings)	Trusts
49,290	139,738	664,824	487,181

Rating matrix sparsity is 99.99135%. Majority of the users are cold start users; 26,307 users expressed less than 5 reviews and they represent 52.82% of the users in the dataset. In Epinions web site, each review is associated with a rating value and in the dataset, the ratings are in the range from 1 to 5, where 1 is the worst and 5 is the best. It is important to note that, the distribution of the ratings is not uniform; 45% of the ratings are 5, 29% are 4, 11% are 3, 8% are 2 and 7% are 1. Results of the experiments with the described dataset represent that simple algorithms seem very effective. Since 45% of the ratings in the dataset are 5, an algorithm which gives 5 for all rating predictions is measured. According to the results, the MAE of the algorithm over all the actual rating values is 1.008.

RS algorithms that use trust information are compared with standard CF algorithm. Only the users explicitly trusted by the active user are used in the algorithm and as a result, trust based algorithm is able to predict fewer ratings than CF, however the predictions are spread more equally over all the users (meaning that they are at least partially satisfied) and for MAUE, CF performs worse than trust based algorithm. To sum up, CF works well for heavy raters in scope of both coverage and error rates, whereas it performs very poorly for cold start users.

For global and local trust metrics, in this work, it is stated that global trust is not suitable for finding good neighbors. Hence, using global trust metric which provides unpersonalized information in a recommender system is not a good idea, since recommender systems try to make personalized recommendations.

3.4.2 Moleskiing [6, 7, 8]

Architecture

Moleskiing.it is a community web site trying to make ski mountaineering safer by using information and communication technologies [6, 7]. It is a recommender system making use of trust propagation. In the paper, the effect of local trust metric is discussed.

In ski mountaineering, since snow conditions and avalanches may cause the tour to be very risky, knowing the conditions of the ski route is a key to make the ski trip safer. The information on ski trips is valuable if they are useful and reliable. Recently written reviews are more useful because they reflect the up-to-date conditions. Also, the quality of the information is important - since the know-how and experience of the reviewer will directly affect the quality of the review [8]. Therefore, the reliability of the ski mountaineers is the key point.

Users form their ”web of trust” by rating other users in the system. Then, the system shows user the information provided by her/his web of trust. This method requires each user to state her/his web of trust; however, in Moleskiing each user has issued a trust statement to only a few users. Trust propagation is used to make use of the information provided by the users that the active user did not trust.

A preliminary trust metric is implemented in Moleskiing application. The goal in making a preliminary trust metric is to limit the computation of trust scores to a certain time. Trust metric is named MoleTrust and it works in two steps. The first step is destroying cycles and the next step is a graph walk over the modified social network, from the active user to the target user. Trust is not propagated over a threshold distance where the default is 3 and trust edges coming from users with a predicted trust score of less than 0.6 are not considered, since they are predicted to be not trustworthy.

Evaluation

Since there were no available comparable proposed trust metrics at the time of this work, the evaluation and experimentation do not provide exact results of computation. To conclude, in a trust-enhanced recommender system, selecting reliable and useful information is beneficial to make use of better quality local trust metrics [7].

3.5 Trust-based Recommender Methods

3.5.1 Probabilistic Matrix Factorization

In many collaborative filtering approaches, the algorithm cannot handle very large ”realistic” databases and also cannot produce quality predictions for users who have very few ratings [53]. In Probabilistic Matrix Factorization (PMF) model described in [53], the algorithm scales linearly with the number of observations and outputs well predictions for users with sparse ratings.

In Section 2.5.2, matrix factorization models are described briefly. The idea behind factor models is that preferences of a user are determined by a small number of unobserved factors.

In Figure 3.3 [53], the graphical model of PMF is illustrated, where V_j is the movie j's latent vector and U_i is the user latent vector for user i and σ^2 represents the variance of Gaussian distribution. Training such a model is trying to find to best rank-d approximation to the observed $N \times M$ target matrix R, where d is the number of latent vector dimension, N and M are respectively the number of users and items in the system. In other words, the aim is to

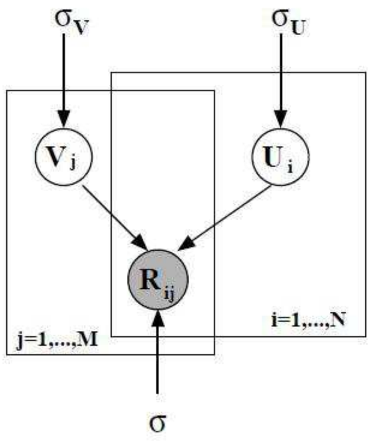

Figure 3.3: Probabilistic Matrix Factorization Graphical Model [53]

find the matrix $\hat{R} = U^T V$ which minimizes the sum-squared distance to the target matrix R.

Instead of using a simple linear-Gaussian model, which can make predictions outside the boundaries of valid ratings, a logistic function $g(x) = 1/\left(1 + exp^{(-x)}\right)$ is applied to the dot product of user and movie feature vectors. Therefore, the prediction can be stated more formally as in Equation 3.3:

$$p\left(R|U, V, \sigma^2\right) = \prod_{i=1}^{N} \prod_{j=1}^{M} \left[\mathcal{N}\left(R_{ij}|g\left(U_i^T V_j\right), \sigma^2\right)\right]^{I_{ij}} \tag{3.3}$$

where $\mathcal{N}\left(x|\mu, \sigma^2\right)$ is the probability density function of the Gaussian distribution with mean μ and variance σ^2, and I_{ij} is the indicator function which is equal to 1 if user i has rated item j and if not equal to 0.

Evaluation of Probabilistic Matrix Factorization on Netflix Dataset

PMF evaluation is done on the Netflix Prize dataset [45]. The dataset contains 480,189 users, 17,770 movies and more than 100 million ratings.

The ratings from 1 to R_{max} in the dataset are mapped into the interval $[0, 1]$ using the function $f(x) = (x - 1)/(R_{max} - 1)$, where x represents the rating value and $f(x)$ represents the mapping function.

To speed up the training, mini-batches of size 100,000 are used and feature vectors are updated after each mini-batch. The model is trained using 30 factors and the training took about an hour time. The parameters in the experiments are a learning rate of 0.005 and a momentum of 0.9.

More efficient results are obtained in PMF compared to the other MF techniques. Efficieny is a result of finding only point estimates of model parameters and hyperparameters, instead of inferring the full posterior distribution over them [53].

3.5.2 Sorec: Social Recommendation Using Probabilistic Matrix Factorization

SoRec [36] is a factor analysis method based on the probabilistic graphical model fusing the user-item ratings matrix with the users' trusts matrix by sharing a common latent low-dimensional user feature matrix. This approach assumes observed data is a linear combination of several latent factors. Experiments in SoRec are done with the extended Epinions.com dataset [21].

In this work, basically, the PMF method proposed in [53], and described in Section 3.5.1 is applied to the Epinions.com dataset. Probabilistic matrix factorization is composed with social networks and the idea of social network matrix factorization is proposed. Latent factor analysis based on probabilistic matrix factorization is applied on the dataset and as a result of this, user latent feature space and movie latent feature spaces are learnt.

PMF method provides some advantages compared to the state-of-the-art solutions. With the help of defining user and movie latent feature spaces, more quality and useful predictions can be computed even for the users who have rated very few items. Computational complexity is reduced and the method is applicable for the large datasets.

It is important to note that, in the scope of SoRec [36] study, it is assumed that there are m number of users and n number of items in the system; and k represents any user in the system. To preserve the originality of this work, the notations are not modified in line with this thesis work, all the notations are in the same presentation of the SoRec method proposed in [36].

Graphical model of social recommendation is given in Figure 3.4. In this figure, $C = \{c_{ik}\}$ denotes the $m \times m$ social network matrix. c_{ik} takes a value in the interval $[0, 1]$ where 0 represents a complete distrust of user i in user k and the other weights can be interpreted as

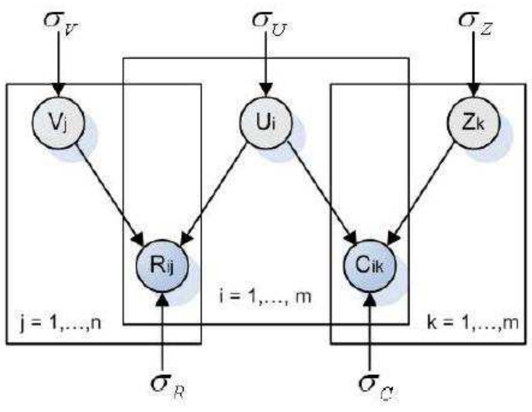

Figure 3.4: Social Recommendation Graphical Model [36]

how much user i is interested in user k. And due to being a social network matrix, C is an asymmetric matrix. While having only 0 or 1 as a trust value in Epinions.com, the adjustment of trust values to the interval $[0, 1]$ will be described in the following parapragh.

In most of the online social networks, users are capable of issuing explicit trust ratings to each other. Recall that, in Epinions.com trust values are issued as complete trust(1) or distrust(0). In the scope of this work, to normalize those trust values, the confidence of binary (0 or 1) trust values are adjusted with a simple, graphic based decision. Let us consider adjusted trust value c_{ik} representing trust of user i in user k; if user i trusts lots of users, then the confidence of his trust scores are decreased and if user k is trusted by lots of users, confidence value is increased. More formal explanation is in the Equation 3.4:

$$c^*_{ik} = \sqrt{\frac{d^-(v_k)}{d^+(v_i) + d^-(v_k)}} \times c_{ik} \tag{3.4}$$

The confidence of trust values are adjusted after modeling the social network as a social graph; the users are the nodes in the graph. For instance user i is thought as node v_i. In Equation 3.4, c^*_{ik} is the adjusted trust value, whereas c_{ik} is the issued trust value (in this case either 0 or 1); $d^+(v_i)$ is the outdegree of node v_i and $d^-(v_k)$ is the indegree of node v_k. After computing c^*_{ik} values for each (i, k) combination, the c_{ik} values in the matrix C are substituted.

Rating values from 1 to R_{max} are mapped to the interval $[0, 1]$ using the same function in Probabilistic Matrix Factorization [53]; $f(x) = (x - 1)/(R_{max} - 1)$, where x represents the rating value and $f(x)$ represents the mapping function.

User latent feature, factor feature and item latent feature matrices are respectively represented as U, Z and V. In this work, Z, the factor feature matrix is introduced. The idea of social network matrix factorization is to derive a d-dimensional feature presentation U of the social network graph. It is crucial to revisit the definitions and notations of feature matrices; $U \in \Re^{d\times m}$, $Z \in \Re^{d\times m}$ and $V \in \Re^{d\times n}$ are respectively latent user, factor and item feature matrices and column vectors U_i, Z_k and V_j represent the user-specific, factor-specific and item-specific latent feature vectors respectively. In SoRec system, in addition to finding $\hat{R} = U^T V$ that minimizes the distance from the target R ratings matrix, another goal is set; with the social network information introduced, finding $\hat{C} = U^T Z$ which minimizes the distance from the target C trusts matrix.

Based on Figure 3.4, the log of the posterior distribution for social recommendation is given by the Equation 3.5:

$$
\begin{aligned}
&\ln p\left(U, V, Z|C, R, \sigma_C^2, \sigma_R^2, \sigma_U^2, \sigma_V^2, \sigma_Z^2\right) = \\
&-\tfrac{1}{2\sigma_R^2} \textstyle\sum_{i=1}^m \sum_{j=1}^n I_{ij}^R \left(r_{ij} - g\left(U_i^T V_j\right)\right)^2 \\
&-\tfrac{1}{2\sigma_C^2} \textstyle\sum_{i=1}^m \sum_{k=1}^m I_{ik}^C \left(c_{ik}^* - g\left(U_i^T Z_k\right)\right)^2 \\
&-\tfrac{1}{2\sigma_U^2} \textstyle\sum_{i=1}^m U_i^T U_i - \tfrac{1}{2\sigma_V^2} \sum_{j=1}^n V_j^T V_j - \tfrac{1}{2\sigma_Z^2} \sum_{k=1}^m Z_k^T Z_k \\
&-\tfrac{1}{2} \left(\left(\textstyle\sum_{i=1}^m \sum_{j=1}^n I_{ij}^R\right) \ln \sigma_{ij}^2 + \left(\sum_{i=1}^m \sum_{k=1}^m I_{ik}^C\right) \ln \sigma_C^2\right) \\
&-\tfrac{1}{2} \left(ml \ln \sigma_U^2 + nl \ln \sigma_V^2 + ml \ln \sigma_U^2\right) + C
\end{aligned}
\tag{3.5}
$$

where C denotes a constant that does not depend on the parameters. $sigma^2$ represents the variance of the Gaussian distribution. The function $g(x)$ in Equation 3.5 is the logistic function $g(x) = 1/(1 + \exp(-x))$, which makes it possible to bound the range of products $U_i^T V_j$ and $U_i^T Z_k$ within the range $[0, 1]$. I_{ij}^R and I_{ik}^C are the indicator functions. I_{ij}^R is equal to 1 if user i has rated item j and 0 otherwise. In the same way, I_{ik}^C is equal to 1 if user i has trusted user k and 0 otherwise.

Maximizing log posterior over three latent features minimizes the sum-of-squared-errors objective functions which is shown in Equation 3.6:

$$\begin{aligned}\mathcal{L}(R, C, U, V, Z) = \\ \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}^{R}\Big(r_{ij} - g\left(U_i^T V_j\right)\Big)^2 \\ +\frac{\lambda_C}{2}\sum_{i=1}^{m}\sum_{k=1}^{m} I_{ik}^{C}\Big(c_{ik}^{*} - g\left(U_i^T Z_k\right)\Big)^2 \\ +\frac{\lambda_U}{2}\|U\|_F^2 + \frac{\lambda_V}{2}\|V\|_F^2 + \frac{\lambda_Z}{2}\|Z\|_F^2\end{aligned} \tag{3.6}$$

where $\lambda_C = \sigma_R^2/\sigma_C^2$, $\lambda_U = \sigma_R^2/\sigma_U^2$, $\lambda_V = \sigma_R^2/\sigma_V^2$, $\lambda_Z = \sigma_R^2/\sigma_Z^2$, and $\|.\|_F^2$ denotes the Frobenius norm. And a local minimum of objective function given by Equation 3.6 can be found by performing gradient descent in U_i, V_j and Z_k. The gradient descents are represented in the following Equations 3.7, 3.8 and 3.9:

$$\begin{aligned}\frac{\partial L}{\partial U_i} = \sum_{j=1}^{n} I_{ij}^{R} g'\left(U_i^T V_j\right)\left(g\left(U_i^T V_j\right) - r_{ij}\right)V_j \\ +\lambda_C \sum_{k=1}^{m} I_{ik}^{C} g'\left(U_i^T Z_k\right)\left(g\left(U_i^T Z_k\right) - c_{ik}^{*}\right)Z_k + \lambda_U U_i\end{aligned} \tag{3.7}$$

$$\frac{\partial L}{\partial V_j} = \sum_{i=1}^{m} I_{ij}^{R} g'\left(U_i^T V_j\right)\left(g\left(U_i^T V_j\right) - r_{ij}\right)U_i + \lambda_V V_j \tag{3.8}$$

$$\frac{\partial L}{\partial Z_k} = \lambda_C \sum_{i=1}^{m} I_{ik}^{C} g'\left(U_i^T Z_k\right)\left(g\left(U_i^T Z_k\right) - c_{ik}^{*}\right)U_i + \lambda_Z Z_k \tag{3.9}$$

where $g'(x)$ represents the derivative of logistic function $g(x)$ and is equal to
$g'(x) = \exp(x)/(1 + \exp(x))^2$.

Evaluation

The Epinions dataset used in the experiments of this study consists of 40,163 users who have rated at least one item and there are 139,529 items in total. The users have expressed 664,824 ratings attached to the reviews. Therefore, the user-item density of the matrix is 0.01186%. Mean Absolute Error (MAE) metric is used to measure the prediction quality. The algorithm proposed in their study is compared to the Maximum Margin Matrix Factorization (MMMF) [51], Probabilistic Matrix Factorization (PMF) [53], and Constrained Probabilistic Matrix Factorization (CPMF) [53]. In the experiments, different dataset sizes are used for training - 99%, 80%, 50%, 20% -, also different number of dimensions, - 5, 10 -,is selected for the number of latent features. MAE comparison shows that the algorithm proposed in Sorec system outperforms all the methods at all times.

As a future work, it is stated that using not only local trust information but also using distrust information can output more quality predictions. In addition to that, while making a social network based just only on the explicitly stated trust information, propagation of trust or information diffusion is ignored; therefore, to decrease the data sparsity, trust propagation can be added.

CHAPTER 4

THE PROPOSED METHOD

4.1 Motivation

Recent studies have showed that most of the people prefer to ask friends while picking a new item rather than using a recommender system [61].

In commercial systems, implementing such social networks, people may not spend that much time to find the users whom they can trust. This results in sparse user to user trust matrix which causes less accurate predictions.

On the other hand, same users may prefer giving ratings to the other users' reviews with or without issuing explicit trust statements to those reviewers. For example, rating a specific reviewer's most of the reviews with high scores indicates the active user's liking and agreement in these reviews. When the user does not have any trust information and ratings, this information will be valuable for prediction.

Our proposed method uses a hybrid model based on user ratings. If the user has sufficient number of past ratings on variety of items, these user ratings can be used for prediction. However, if no past ratings are available, then the user's ratings on other people's reviews can be utilized for prediction.

Social interactions and connections are not only limited to user trust network. Users may read other people's opinions, agree with their choices but may not classify them as trusted explicitly. Such information may contain important clues about their likes and dislikes. To demonstrate, we have analyzed Epinions.com [22] where we crawled and obtained a dataset comprising 92,205 users and 27,903 products with a total of 169,252 ratings. 75.78% of the users did not rate any product, and 17.73% of the users rated 1-4 items. Therefore, only 6.49% of the users have given more than a few ratings and stated sufficient information about their liking. While there are only 577,950 trust values, we have obtained 3,680,950 review ratings in total. 23.27% of the users have not stated any trust value. On the other hand, only 865 out of 169,252 reviews did not receive any review rating value, which means that 99.49% of the reviews are rated by the users. As a consequence, review ratings may be helpful for a more robust user-item prediction.

4.2 Description

The proposed model has the following steps:

1. Normalization of the rating values [53] is done according to the Equation 4.1:

   Let r_{ij} is the rating of i^{th} user to the j^{th} item, where $1 \leq i \leq N$ and $1 \leq j \leq M$. The normalized form of the rating r_{ij} is equal to r'_{ij} and computed as follows:

   $$r'_{ij} = \left(r_{ij} - minRateValue\right)/(maxRateValue - minRateValue) \tag{4.1}$$

   $maxRateValue$ and $minRateValue$ indicate the maximum and minimum rating values in a system respectively. We have selected maxRateValue as 5 and minRateValue as 1 based on Epinions.com ratings. Therefore, before starting experimentation, all ratings values from 1 to 5 are normalized to the interval $[0, 1]$.

2. Initialization of user-item matrices:

   As stated in Section 2.5.2, in a linear factorization model, user matrix U and item matrix V represent users and items features in latent vectors respectively.

   For a given user, latent vector elements measure the extent of interest in items, so that the elements may be positive or negative. The latent vector for a given item also tries to model the item's characteristics as to have positive or negative values.

   In this model, user and item latent vectors are initialized randomly in uniform distribution in the range $[0, 1]$. In the experiments held in the scope of this work, we have seen that latent vector initialization is one of the crucial steps in a matrix factorization model. The results of the latent vector initialization experiments are given in detail in Section 5.1.1.

3. Training phase:

   The training phase consists of training the model with the PMF algorithm.

   In the experiments, to limit the predictions inside range of valid ratings, the dot product of user and item latent vectors, x, is passed to the logistic function $g(x) = 1/(1 + \exp(-x))$ [53].

   Here, the conditional distribution over the observed ratings can be defined as in Equation 4.2 [36, 53]:

$$p(R|U, V, \sigma_R^2) = \prod_{i=1}^{N} \prod_{j=1}^{M} \left[\mathcal{N}\left(r_{ij}|g\left(U_i^T V_j\right), \sigma^2\right)\right]^{I_{ij}^R} \tag{4.2}$$

   where $\mathcal{N}\left(x|\mu, \sigma^2\right)$ is the probability density function of the Gaussian distribution with mean μ and variance σ^2, r_{ij} is the rating given to the item j by user i, U is the user latent vectors matrix, V is the item latent vectors matrix and I_{ij} is the indicator function which gives an output of 1 if user i has rated item j and otherwise 0.

   We also place zero mean spherical Gaussian priors on user and item feature vectors in Equations 4.3 and 4.4:

$$p(U|\sigma_U^2) = \prod_{i=1}^{N} \mathcal{N}\left(U_i|0, \sigma_U^2 I\right) \tag{4.3}$$

$$p(V|\sigma_V^2) = \prod_{j=1}^{M} \mathcal{N}\left(V_j|0, \sigma_V^2 I\right) \tag{4.4}$$

   Hence, through a Bayesian inference, we have the Equation 4.5 [53]:

$$\begin{gathered} p(U, V|R, \sigma_R^2 R, \sigma_U^2 R, \sigma_V^2) \propto p(R|U, V, \sigma_R^2) p(U|\sigma_U^2) p(V|\sigma_V^2) \\ = \prod_{i=1}^{N} \prod_{j=1}^{M} \left[\mathcal{N}\left(r_{ij}|g\left(U_i^T V_j\right), \sigma^2\right)\right]^{I_{ij}} \\ \times \prod_{i=1}^{N} \mathcal{N}\left(U_i|0, \sigma_U^2 I\right) \times \prod_{j=1}^{M} \mathcal{N}\left(V_j|0, \sigma_V^2 I\right) \end{gathered} \tag{4.5}$$

4. Evaluation on validation data:

   In the method we propose, we have reviewed the latent vectors of the users having a mean absolute user error (MAUE) greater than a certain threshold in the training set.

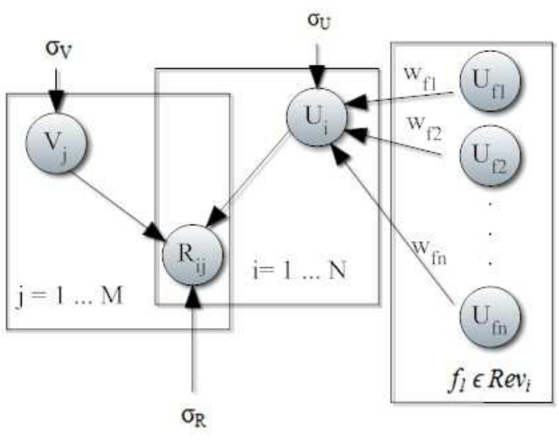

Figure 4.1: Graphical model of the Proposed Method

The threshold is selected as 3 after executing the algorithm for thresholds 2.0, 2.5, 3.0, 3.5, 4.0 which showed that best results are achieved with this threshold value. The experiments determining the threshold are described in detail in Section 5.1.4.

The graphical model of the proposed method is given in Figure 4.1.

Algorithm 1 Reviewing user latent vectors

```
for $i = 1$ to $N$ do
  if $MAUE(U, V, R, i) > MAUE_{threshold}$ then
    $orderedList \leftarrow sort(Rev_i)$ {Order reviewers that user has rated according to the number of ratings} {Select the most rated $n$ reviewers from the list, and name as $f_1$, $f_2$ ... $f_n$.}
    for $k = 1$ to $n$ do
      $f_k \leftarrow orderedList(k)$
      $w_{f_k} = \frac{rating\ count\ given\ to\ friend_k\ by\ the\ user}{total\ rating\ count\ given\ to\ top\ n\ friends\ by\ the\ user}$ {Calculate normalized weights}
    end for
    $U_i = w_{f_1}U_{f_1} + w_{f_2}U_{f_2} + ... + w_{f_n}U_{f_n}$ {Update latent vectors of the users with the new ones.}
  end if
end for
```

In Algorithm 1, U and V are respectively user and item latent vector matrices and R is the ratings matrix. Rev is an $N \times N$ review ratings matrix, where N is the user count, and

Rev_i is a vector containing the review rating counts given to each user by user i. In other words, Rev_{ik} is the ratings count given to the user k's reviews by user i. *orderedList* is a list of users whom are rated by user i and the list is a descending list according to the number of review ratings given to those users; therefore $orderedList(k)$ gives us the k^{th} user that has received review ratings from user i. U_i is the latent vector of i^{th} user. Here, the idea is to replace the error prone user latent vector with more tolerant latent vectors in a linearly weighted scheme.

Update procedure is done as a replacement as seen in Algorithm 1. MAUE evaluation is performed on the training data after training; predictions are compared with actual rating values and errors for each user are calculated. Users with MAUE greater than the threshold are eliminated for the review phase. Friends of those users, namely the users whom the active user has given most of his/her review ratings are selected. Latent vectors of the users with high error rates are replaced with the weighted top three friends' latent vectors. We have selected three friends; because in the experiments, it is observed that incorporating more than three friends into the calculation does not affect the accuracy significantly. Latent vectors of the three friends are weighted linearly in line with the review rating count that the active user has given to them.

4.3 Dataset Characteristics

Epinions.com is a consumer reviews platform which is explained in detail in Section 2.7.1. In this section, characteristics and detailed information about crawled Epinions.com dataset will be described. To reduce the computational complexity, preprocessing is applied on the dataset which will also be elaborated.

We extracted the dataset from Epinions.com website in January 2010. The crawler is based on the idea stated by Massa [46]. Initially, we choose a seed user, who had significant amount of trust information. Then starting from that user, recursively all trusted users and users who trust the selected user and products that are reviewed by the selected user are visited. As a consequence, all users in our dataset satisfy at least one of the following conditions: have at least one rating, one stated trust, one trusted by, or one review rating information. During crawling, the domain is restricted to movies only. The ratings that are given by the same users to the items besides movies are not considered because we tried to move from system-level trust to context-specific trust.

Dataset consists of 92,205 users who have rated a total of 27,903 movies with 169,252 ratings. 577,950 trust values are stated explicitly by users to each other. In Epinions, every product rating is associated with a review and users other than the reviewer may rate that review. Dataset contains 3,680,950 review ratings. Density of the user item matrix is 0.02716%. User-item rating matrix is very sparse compared to the well-known datasets used in most of the collaborative filtering methods, MovieLens has a density of 4.25% and Eachmovie has 2.29%. MovieLens dataset has 6,040 users, 3,900 movies and 1,000,209 ratings and in Eachmovie dataset there are 74,424 users, 1,648 movies and 2,811,983 ratings. Epinions dataset is notably sparse compared to the other datasets.

169,252 ratings are issued to 27,903 movies by 92,205 users. However, the distribution of the number of ratings stated by the users are not uniformly distributed, the number of ratings are binned to make things clear which is shown in Table 4.1. 69,872 of the users did not give any ratings to any movie in the system.

Table 4.1: Epinions.com - Number of ratings issued by users

Rating count	0	1-4	5-9	10-49	50-...	Total
Number of users	69,872	16,349	3,119	2,394	471	92,205

Table 4.2 presents that the users mostly prefer giving ratings to the items that they like, 31.00% and 33.80% of the ratings are 4 and 5, respectively.

Table 4.2: Epinions.com - Ratings distribution

Rating value	1	2	3	4	5	Total
Number of ratings	12,477	17,689	29,402	52,475	57,209	169,252
Percentage	7.37	10.45	17.37	31.00	33.80	100%

Besides from the ratings, the second valuable information in the system is the trust information. Table 4.3 displays the number of trusts issued to the other users in the system. As we can see from this table, 21,456 (23.27%) of the users did not state any trust information.

Table 4.3: Epinions.com - Number of trusts issued by users

Trust count	0	1-4	5-9	10-49	50-...	Total
Number of users	21,456	53,968	6,958	7,312	2,511	92,205

Table 4.3 analyzes the dataset from the ”Web of trust” point. On the other hand, we can look at the number of trusts that each user has received in Table 4.4. As we have seen in Section 3.5.2, in literature, being trusted by lots of people presents a higher ”reputation” and it affects the trust value adjustments.

Table 4.4: Epinions.com - Number of trusts given to a user

Trust count	0	1-4	5-9	10-49	50-...	Total
Number of users	34,101	44,310	5,835	5,760	2,199	92,205

Finally, the most productive information is review ratings - dataset contains 3,680,950 review ratings. As described in the beginning of this section, in Epinions.com, there are eight different review ratings. However, ”Show”, ”Don’t Show” and ”Off Topic” are given to less than 0.55% of the reviews totally; this is the reason of omitting those ratings from Table 4.7.

Looking at the review ratings analysis in Table 4.5, it is seen that only 0.51% of the reviews did not receive a review rating.

Table 4.5: Epinions.com - Number of review ratings given to the reviews

Review rating count	0	1-4	5-9	10-49	50-...	Total
Number of reviews	865	29,824	38,478	80,251	19,384	169,252

From the users side, we know that 69,872 of the users did not give a rating to any item in the system, Table 4.6 displays that 63,235 of the users did not give ratings to a review; which makes a 7.19% difference on the number of users side.

Table 4.7 presents similar results to Table 4.2; users also prefer giving ratings to the reviews that they like - 88.33% of the ratings are ”Very Helpful”. In Table 4.7, the columns 1, 2, 3,

Table 4.6: Epinions.com - Number of review ratings issued by users

Review rating count	0	1-4	5-9	10-49	50-...	Total
Number of users	63,235	15,237	3,970	5,350	4,413	92,205

4, 5 represent respectively Not Helpful, SomewhatHelpful, Helpful, Very Helpful and Most Helpful.

Table 4.7: Epinions.com - Review ratings distribution

Review rating	1	2	3	4	5
Count	9,307	80,652	253,536	2,839,176	14,303
Percentage	0.29	2.51	7.89	88.33	0.44

4.3.1 Preprocessing of Epinions.com dataset

As mentioned in the above tables, the dataset contains 69,872 users with 0 ratings and 22,333 users with 1 or more ratings. In the preprocessing step, the users with 0 rating are removed from the dataset since we cannot make any prediction for those users. The dataset is divided into three parts as train, validation and test with different percentages which will be described thoroughly in the following chapters. The system cannot make a prediction for a user without any rating since the user will not have the chance to be involved in the test dataset; this is the reason why we eliminated those users. Since those users - the users without a rating - are not involved in the testing phase, they do not provide any information to the system, so that they are also removed from the trusts and review ratings data to reduce computational complexity.

The break down of the data after preprocessing step is as follows:

- 22,333 users
- 27,903 products
- 169,252 ratings
- 270,652 trust values

- 3,214,401 review ratings

Density of user-item matrix is $\frac{169{,}252}{22{,}333 * 27{,}903} = 0.027\%$. Revisiting the analysis of number of ratings given by the users, which was previously presented in Table 4.1, moving out the users with 0 ratings results in the following Table 4.8:

Table 4.8: Epinions.com - Number of ratings issued by users on preprocessed dataset

Rating count	1-4	5-9	10-49	50-...	Total
Number of users	16,349	3,119	2,394	471	22,333

If we look at the trusts that a user has stated (web of trust), we can see that maximum number of trusts that a user has given is 1,318. The binned data is as follows in Table 4.9:

Table 4.9: Epinions.com - Number of trusts issued by users on preprocessed data

Trust count	0	1-4	5-9	10-49	50-...	Total
Number of users	5,742	9,079	2,439	3,668	1,405	22,333

We should also analyze the data from the side of trusts that are given to a user: maximum number of trusts that is given to a single user is 1,692- perhaps we can say that this user is the most trusted user in the whole system.

Table 4.10: Epinions.com - Number of trusts given to a user on preprocessed data

Trust count	0	1-4	5-9	10-49	50-...	Total
Number of users	6,060	10,274	2,097	2,669	1,233	22,333

In the dataset information, we have described the notion of the review ratings (4.3). In the review ratings data, we can see that, 37.48% of the users (namely 8,370 users), have not stated any review ratings. And we can say that we do not have the same amount of information about all users. Whereas 8,370 users have not stated any ratings to a review, the user who has stated maximum number of review ratings has given 51,322 review ratings. These are the main impediments blocking the methods from producing excelling predictions. However, in the preprocessing step, where we eliminated the users without ratings, a notable difference is observed in the number of users without any review ratings, the numbers are 63,235 in the crawled data and 8,370 in the preprocessed data. Detailed analysis is shown in Table 4.11:

Table 4.11: Epinions.com - Number of review ratings issued by users

Review rating count	0	1-4	5-9	10-49	50-...	Total
Number of users	8,370	5,638	1,906	3,109	3,310	22,333

From Table 4.8, it is seen that we do not have much information about most of the users- 19,758 users, 88.469% of the users have given only 1-10 ratings to the products. Enriching the information we know about users with sparse ratings may increase success in the predictions. The users have stated ratings to the products, personal trust to the other users and also ratings to the reviews written by other users. In most of the current approaches, while making predictions about ratings, trusts and review ratings are discarded. However, we have huge trusts and review ratings data compared to the ratings data.

Analyzing trust information of the users with poor number of ratings may introduce us a new perspective of the problem. It is crucial to break down the analysis of the dataset deeply in order to understand the causes of the poor quality predictions. Initially, we have filtered users who have given 1-10 ratings, generated a file with detailed web of trust information about those users. Each user is examined independently. To distinguish between the active user and his web of trust, the users in the web of trust will be renamed as reviewers. Any user in the active user's web of trust is a reviewer. Initially, the reviewers in active user's web of trust are sorted in ascending order, with the number of ratings they have given to the products. 27.23% of the users with sparse ratings, namely 5,380 users, cannot be enriched with the information in their web of trust; they have either no people in their web of trust, in order words they did not issue trust to any user in the system or they have trusted people without any rating. On the other hand, 14,378 users have reviewers who have rated items in their web of trust. This finding is an important step in our work, after this, our study is focused on providing benefit from the trust information in the most efficient way. The related experiments are described in Section 5.1.2.

4.4 Experimental Settings

In the experimental analysis, 80% of the ratings data is used as the training dataset, 10% is used for validation and the remaining 10% is used for testing. The percentages are selected in line with literature - most of the PMF experiments in the literature used 80% of the dataset as the training data [35, 36]. The dataset division is carried out after randomizing ratings data. In the original dataset, the rating tuples are sorted in ascending user ids. Therefore, to prevent using a subset of users for training, another subset for validation and one another for testing, the dataset is randomized and training, validation, and testing datasets are formed afterwards.

During all the experiments, training, validation and test data are all kept the same- they are generated at the beginning of the experiments and then the experiments are performed. Four evaluation metrics defined in Section 2.6 are used to measure the performance of the system with the changes in each parameter. We have repeated each experiment 5 times independently with the same parameters using the same training, validation and test datasets and comparable evaluation metrics represented in the following tables and figures are gathered from the averages of these runs. Mini-batch size, number of batches, epoch number and latent vector dimension are selected according accuracy of the results using the evaluation metrics for the validation set.

To sum up, all of the experiments in the scope of this work are carried out with latent vectors dimension equal to 10, mini-batches of size 1000, 50 batches and 60 epochs.

The experiments described in the this section are done on a personal computer having Intel Core 2 Duo CPU 2.00 GHz and 1 GB memory. The implementations have been executed on MATLAB 7.6.0.

Instead of performing batch learning, we subdivided the data into mini-batches, and updated the latent vectors after each mini-batch.

4.4.1 Latent Vector Dimension

The major challenge in PMF is computing the mapping of user and items to latent vectors. After having successful assignments for user and item latent vectors, making a prediction is just calculating the dot product. Therefore, the choice of initial values is of great importance.

We carried out experiments in order to select the best values for the parameters. The related experiments will be described in Section 5.1.1.

In the experiments to select the optimal latent vector dimension, which are carried out in the scope of this thesis work, MAE, RMSE, MAUE and MAIE measures are used to evaluate the performance of the system. The PMF experiments are carried out with a dimension size equal to 5 and 10 in [36]; however, in order to make a valid comparison, we focused on detailed experimentation. The experiments are done with latent dimensions of size equal to 5, 10, 20 and 30. The results are presented in Table 4.12 and Table 4.13, *d* in both tables represents the latent vector dimension size. The results of the experiment show that latent dimension equal to 10 reduces error in the system over the validation and test sets. The increase in the training times along with the increase of the dimension size, which is presented in Table 4.12, is caused by making more more computationally complex operations where the dimension size is higher. As mentioned in Section 2.5.2, in matrix factorization models, the predictions are computed as a product of user latent vectors and item latent vectors and this makes the computation more complex if the dimensions of the latent vectors increase. As a result of this experiment, we have selected 10 as the dimension of the latent vectors in our system.

Table 4.12: Latent Vector Dimension Selection - Training

d	Training				Time
	RMSE	MAE	MAUE	MAIE	(sec)
5	1.05	0.80	0.71	0.65	92.45
10	1.07	0.82	0.75	0.74	162.76
20	1.16	0.87	0.84	0.95	309.43
30	1.25	0.92	0.90	1.11	431.59

4.4.2 Mini-batch Size

Selecting mini-batches from the training data is performed by randomly selecting rating tuples from the training part of the dataset. Since the selection process is done randomly, some of the rating tuples are used more than once, even usage time may be equal to the number of batches and some of the tuples are never used. Let *minibatches* represent the mini-batch

Table 4.13: Latent Vector Dimension Selection - Test and Validation

d	Test				Validation			
	RMSE	MAE	MAUE	MAIE	RMSE	MAE	MAUE	MAIE
5	1.25	0.98	1.05	0.99	1.26	0.99	1.07	0.99
10	1.18	0.90	0.92	0.88	1.18	0.91	0.93	0.89
20	1.31	0.96	0.92	1.00	1.30	0.96	0.92	0.99
30	1.36	0.99	0.94	1.04	1.35	0.98	0.94	1.03

size. Since $numberofbatches \times minibatches$ number of tuples are selected from the dataset, their multiplication size cannot be greater than the training data size. As a result, the experiments are carried out with mini-batches of different sizes as 1000, 2000 and 5000. The experiments showed that increasing only mini-batch size when the number of batches kept constant produces similar results with increasing the number of batches when mini-batch size is kept constant. Therefore, mini-batches of 1000 are used in the comparison of algorithms in order to consume less memory.

4.4.3 Number of Epochs

The epoch number ranges from 20 to 200 in steps of 20. During all the experiments, data is kept the same - data is separated into three parts at the beginning. Mini-batch size is 1000, 50 batches and 10 dimensional latent vectors are used.

Results for the training data are indicated with red lines, validation data with blue lines and finally test data results are colored in black in the following figures. RMSE, MAE, MAUE and MAIE error metrics are shown respectively in Figure 4.2, Figure 4.3, Figure 4.4 and Figure 4.5.

The detailed evaluation tables with complete results of all metrics are given in Appendix B. Overfitting of model occurs after a threshold - whereas the error decreases for training data constantly, the error rates for test and validation data increases. As can be seen from all below 4 diagrams, the best error rates are obtained at epoch count equal to 60. Therefore, in the following sections, 60 epochs are used.

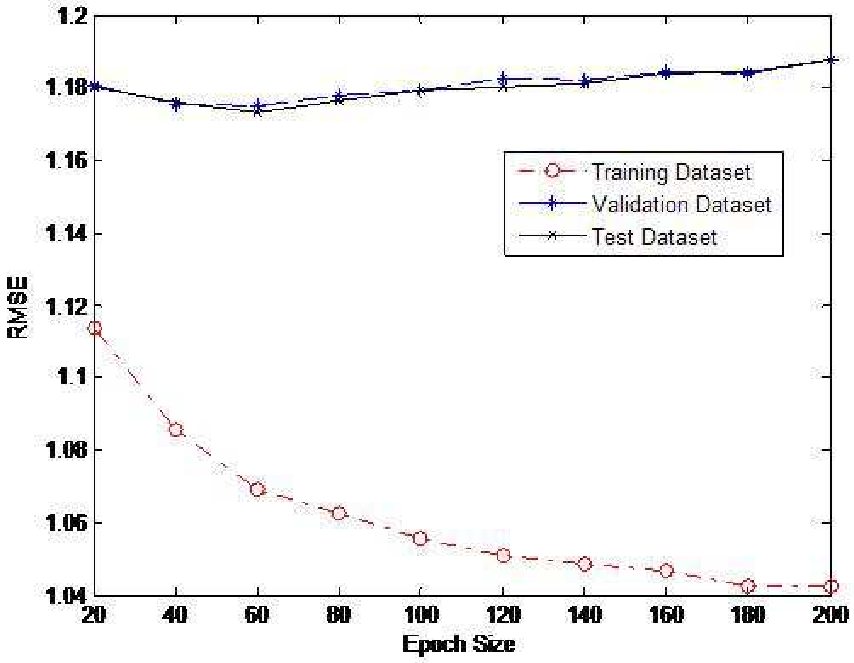

Figure 4.2: RMSE Comparison of Different Number of Epochs

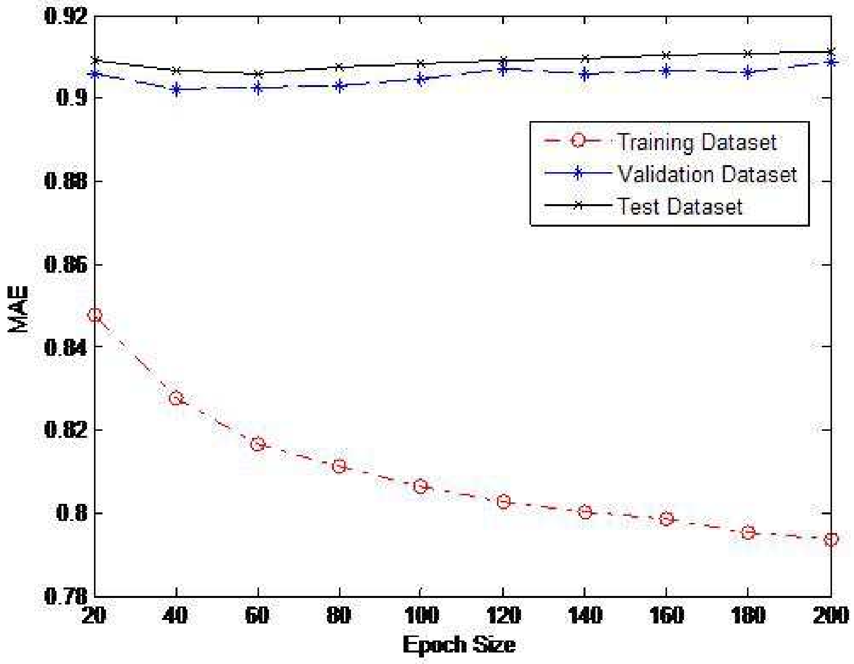

Figure 4.3: MAE Comparison of Different Number of Epochs

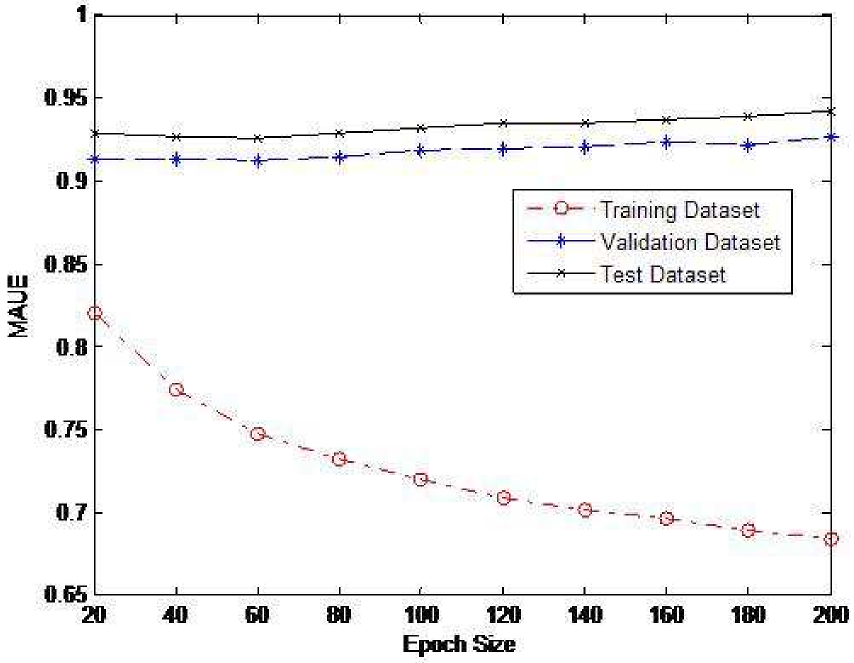

Figure 4.4: MAUE Comparison of Different Number of Epochs

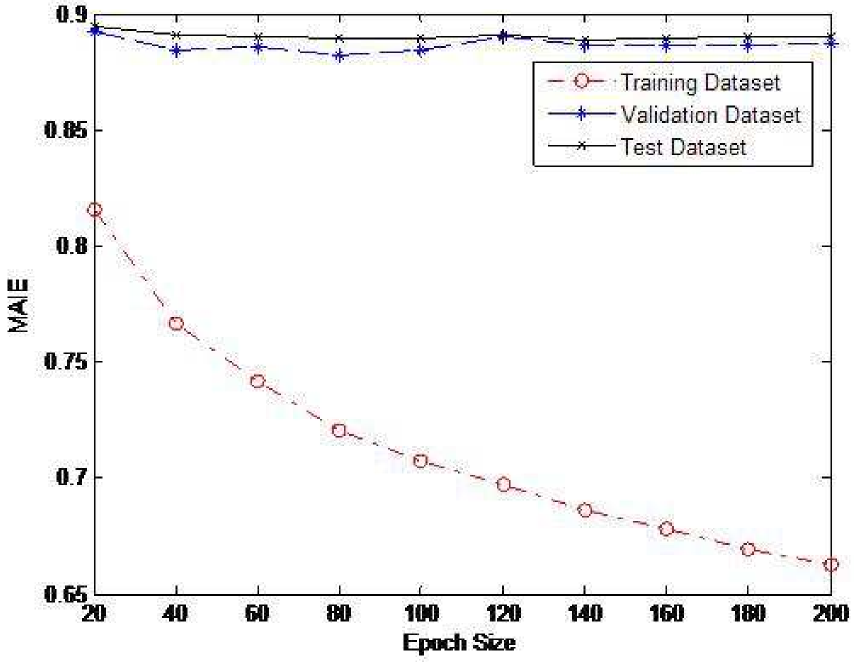

Figure 4.5: MAIE Comparison of Different Number of Epochs

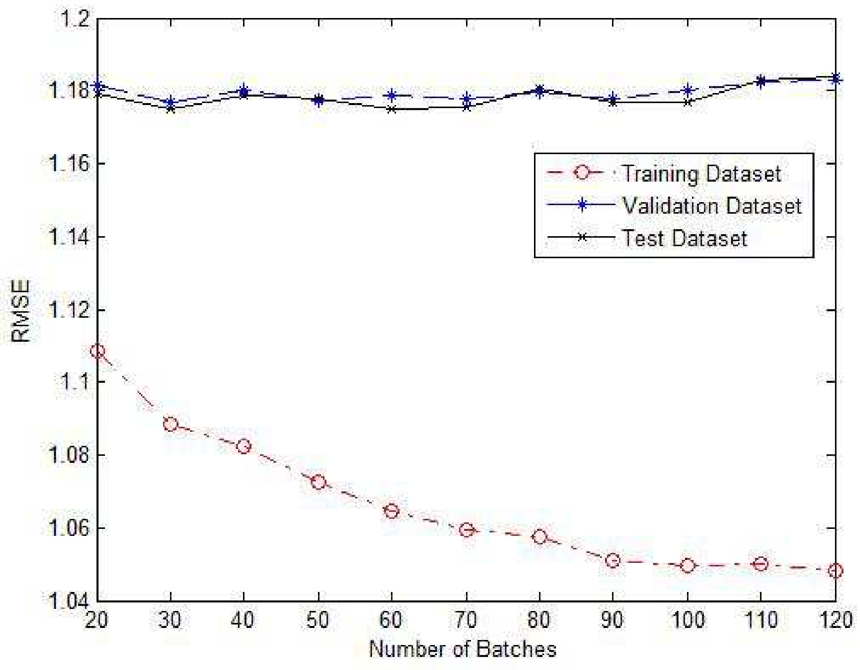

Figure 4.6: RMSE Comparison of Different Number of Batches

4.4.4 Number of Batches

Experiments with different number of batches are done in the range of number of batches equal to 20 to 120 in steps of 10. During all experiments, training, validation and test data are all kept the same - three parts of the dataset are generated before the experiments and then the results are evaluated according to the comparison of four mentioned evaluation techniques. Experiments with test data are exploited during the evaluation phase.

Results of the experiments are shown graphically; RMSE, MAE, MAUE and MAIE error metrics are displayed respectively in Figure 4.6, Figure 4.7, Figure 4.8 and Figure 4.9 for training, validation and test parts of the dataset.

As mentioned above, the evaluation results of validation dataset are examined. However, as seen from the graphics, all four error metrics output their best results at different number of batches. The detailed evaluation tables with complete results of all metrics are given in Appendix A. In the scope of this work, in order to make a valid comparison with the state-of-the-art methods, in cases where the evaluation metrics do not fully agree with each other,

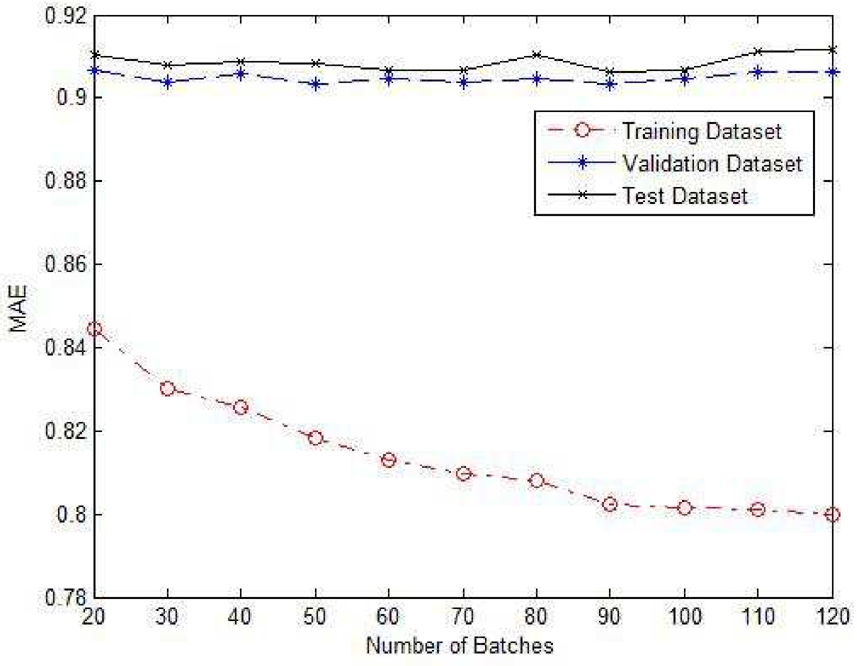

Figure 4.7: MAE Comparison of Different Number of Batches

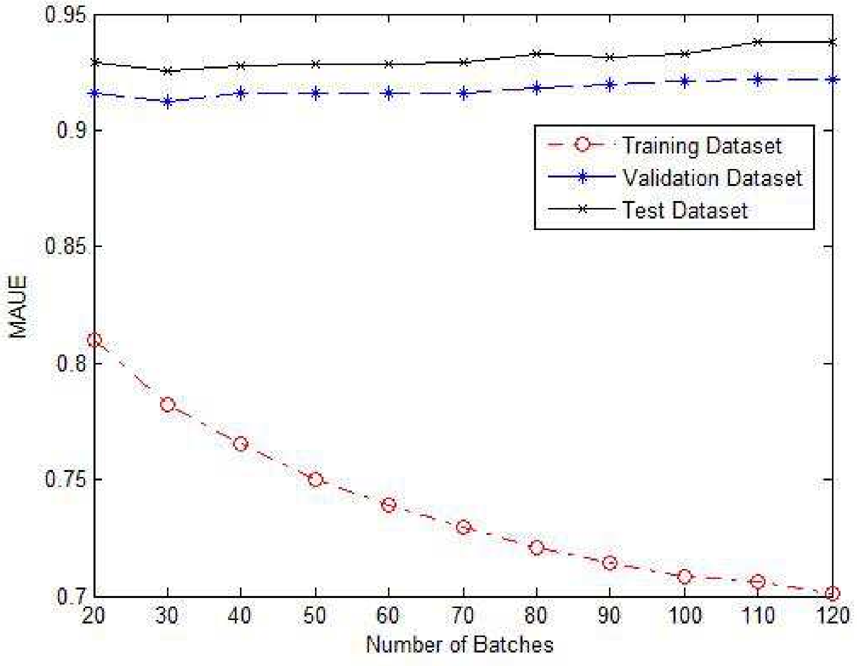

Figure 4.8: MAUE Comparison of Different Number of Batches

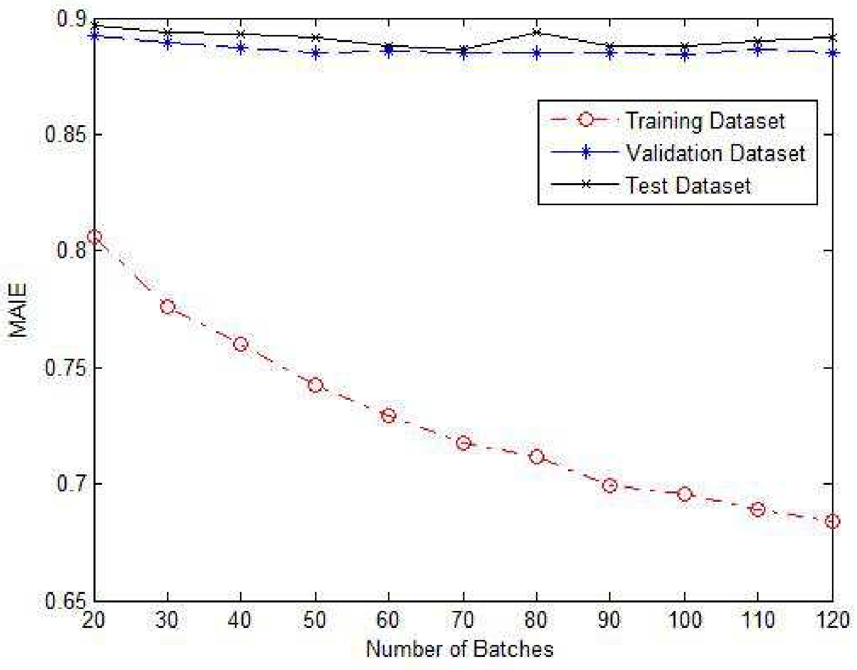

Figure 4.9: MAIE Comparison of Different Number of Batches

results of MAE is taken into account. Therefore, best results are achieved where number of batches is equal to 50 according to the MAE evaluation metric. As a result of this experiment, optimum number of batches are determined and so that the experiments will be held with number of batches equal to 50.

CHAPTER 5

EVALUATION

5.1 Experiments

In this section, experiments done throughout the thesis work will be described. The experiments represented in this section are categorized apart from the experimental settings described in Section 4.4 because the experiments which will be examined in this section make up the critical breakpoints of the algorithm.

The experiments in Section 5.1.1 and 5.1.4 are held with the experimental settings defined in Section 4.4. Each metric is calculated as an average of 5 independent executions of the algorithm and they are all evaluated results of the test part of the dataset.

5.1.1 Impact of Latent Vector Initialization

Latent vectors initialization is one of the key issues in latent vectors. In literature, the concept of initializing values in latent vectors was not analyzed thoroughly. We have only encountered with random initialization of the vectors [31, 53]. The idea behind matrix factorization models is assuming that the items and users are positioned in different coordinates, however since this experiment tries to analyze the reasons and consequences of latent vector initialization, four cases are evaluated to measure the differences between the initialization techniques.

Since the algorithm takes mini-batches randomly, if a user or an item is not selected, then the latent vector of the user or item remains as the original one- no changes occurs in the randomly assigned latent vector values, therefore initialization of the latent vectors are crucial.

1. Experiment 1:

   Latent vectors of all users and items are initialized to 0 for all dimensions. For instance, a given user in a system of latent dimension equal to 10, 10 elements of the user's latent vector are equal to 0.

   The results show that the idea of having neutral latent vectors for the trained users makes it harder to converge. The results of this experiment are shown in Table 5.1, Table 5.2, Table 5.3. This experiment is abbreviated as ”Zero User and Item” (Zero User and Item Latent Vectors).

2. Experiment 2:

   Latent vectors of all users are initialized to 0 for all dimensions, but latent vectors of items are generated randomly in normal distribution with mean 0 and standard deviation 1. In Experiment 2, to make the system realistic, items are assigned to different places in space by initializing the item latent vectors in Gaussian distribution.

   The results of this experiment are shown in Table 5.1, Table 5.2, Table 5.3. This experiment is abbreviated as ”Zero User and Random ND Item” (Zero User and Random Normally Distributed Item Latent Vectors).

3. Experiment 3:

   Latent vectors of all users and all items are initialized randomly in normal distribution with mean 0 and standard deviation 1. Experiment 3 aims to assign almost all of the users to a unique place in the latent vectors space to improve the predictions.

   The results of this experiment are shown in Table 5.1, Table 5.2, Table 5.3. This experiment is abbreviated as ”Random ND User and Item” (Random Normally Distributed User and Item Latent Vectors).

4. Experiment 4:

   Uniform distribution is used to generate user and item latent vectors randomly in the range [0, 1]. The results of this experiment are shown in Table 5.1, Table 5.2, Table 5.3. This experiment is abbreviated as ”Uniform Rand User and Item” (Random Uniformly Distributed User and Item Latent Vectors).

   Recall that, all experiments are run 5 times and all evaluation metrics are computed as an average of those 5 independent executions. The results of the experiment shows us that error rates are decreased compared to the previous experiments. Best results are achieved with initializing user and item latent vectors with uniform normal distribution. This is a result of the uniform normal distribution; uniform initialization of latent vectors makes the conversion easier so that the predictions become more accurate.

5. Experiment 5:

   Latent vectors of all users and items are initialized to 1 for all dimensions. For instance, if latent dimension of a system is equal to 10, then latent vector of an active user will be equal to 1 in all dimensions, this is the same case of the users in the system.

   This experiment outputs better evaluation results in all four evaluation metrics compared to the Experiment 1, which is also another constant user-item latent vector initialization technique, for all training, validation and test sets. The results of this experiment are shown in Table 5.1, Table 5.2, Table 5.3. This experiment is abbreviated as ”One User and Item” (One User and Item Latent Vectors).

Table 5.1: The effect of latent vector initialization - Training phase

Initialization Technique	Training				Time
	RMSE	MAE	MAUE	MAIE	(sec)
Zero User and Item	3.00	2.73	2.76	2.56	150.85
Zero User and Random ND Item	**1.03**	**0.78**	**0.69**	**0.58**	155.11
Random ND User and Item	**1.03**	**0.78**	**0.69**	**0.58**	155.32
Uniform Rand User and Item	1.06	0.80	0.72	0.69	154.65
One User and Item	1.24	0.91	0.90	1.09	155.74

At first glance, comparing validation results represented in Table 5.3, we can see that ”Uniform Rand User and Item” (Random Uniformly Distributed User and Item Latent Vectors) performs best among the four initialization techniques. Best results are marked with bold characters, therefore we can see that uniformly distributed random user and item latent vectors are more effective compared to the other initialization methods.

Although the accuracy changes considerably between different initialization techniques, the training times are very similar.

Table 5.2: The effect of latent vector initialization - Validation data

Initialization Technique	Validation			
	RMSE	MAE	MAUE	MAIE
Zero User and Item	2.99	2.72	2.77	2.62
Zero User and Random ND Item	1.66	1.24	1.38	1.41
Random ND User and Item	1.67	1.25	1.39	1.42
Uniform Rand User and Item	**1.22**	**0.95**	**0.99**	**0.95**
One User and Item	1.47	1.25	1.12	1.33

Table 5.3: The effect of latent vector initialization - Test phase

Initialization Technique	Test			
	RMSE	MAE	MAUE	MAIE
Zero User and Item	3.00	2.74	2.79	2.64
Zero User and Random ND Item	1.67	1.25	1.42	1.41
Random ND User and Item	1.68	1.26	1.43	1.42
Uniform Rand User and Item	**1.22**	**0.95**	**1.01**	**0.95**
One User and Item	1.46	1.27	1.10	1.32

5.1.2 User's Web of Trust Analysis Using Trusted Users

In social networks, user's web of trust is used to enrich the information about user. In this study, our aim is to prove the concept of modeling a user with the help of her/his web of trust is significant. However, using all the reviewers that are trusted by the active user in the computations with trust network can be a costly operation. In order to overcome the computational complexity which will be caused by this, a representative reduction in the number of trusted users is inquired.

In this section, only the users that are explicitly trusted or distrusted by the current user are taken into account. Initially, the user's web of trust (users that the active user trusted) is sorted according to the number of ratings they have given to the products. In this experiment, 5 users with the highest number of ratings are analyzed. From now on, 5 users with the highest number of ratings will be renamed as the top 5 users; 4 users with the highest number of ratings as the top 4 fours and so on. The sum of the rating counts that the top 5 users have expressed are compared to the sum of the rating numbers the top 4, 3 and 2 users have expressed.

This analysis is done by two-sample Kolmogorov-Smirnov test which is used to compare the distributions of the values in the two data vectors. The results of the analysis are given in Table 5.4 and Figure 5.1.

Table 5.4: User's web of trust distribution analysis

	The top 5 - top 4 users	The top 5 - top 3 users	The top 5 - top 2 users
h	0	0	1
p	0.81	0.07	0.00
k	0.01	0.01	0.03

In Table 5.4, the result h represents the null hypothesis that two samples are from the same continuous distribution and is equal to 1 if the test rejects the null hypothesis at the 5% significance level; 0 otherwise [41]. p represents asymptotic *p* value, and becomes very accurate for large datasets such as the dataset used here. Knowing the 5% significance level, we can say that the hypothesis with p value less than 0.01 are rejected. Test statistic k is the maximum

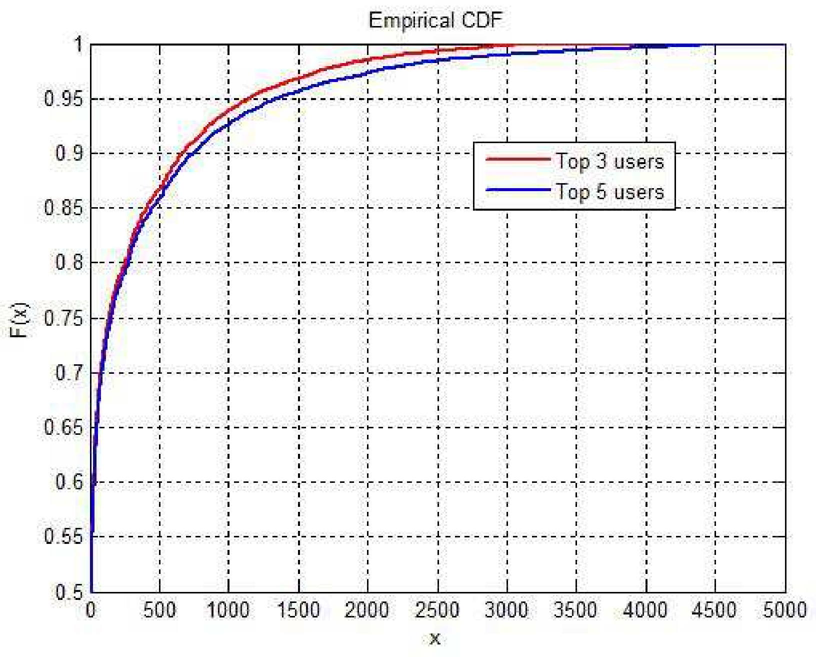

Figure 5.1: Distribution of user's web of trust

difference between the curves of two samples.

As the results show us, the top 2 users do not represent the top 5 users. Therefore, we should use at least the results of top 3 users in our social network experiments. To decrease computational complexity, the top 3 users are selected instead of the top 4 or 5 users.

Figure 5.1 plots the distribution of the top 3 users and top 5 users. In this figure, the axis labeled with x represents the dataset and the axis labeled with F(x) represents the cumulative distribution function. Examining the figure, parallellism of two datasets, namely distribution of top 3 users and distribution of top 5 users can be seen.

5.1.3 User's Web of Trust Analysis Using Review Ratings

The reviewers are the users who have written reviews to any of the products. Reviews may be rated by other users. Therefore, a user may rate more than one review of the same reviewer.

Another comparative analysis, similar to the web of trust analysis presented in Section 5.1.2, is done to the same users - users who have given 1-10 ratings - with the review ratings they have given. An analysis on dataset shows us that 41.01% of the users, namely 8,103 users, did not express ratings to any of the reviews and the remaining 11,655 users rated at least one review. The idea of this analysis is that a user may not need to express explicit trust information to the users s/he trusts. Giving remarkable number of ratings to the same user's reviews may indicate user's interest in the reviewer.

In the dataset, there are users who have parallel trust statements and review ratings; on the other hand, there are also users who do not have any trust statements but rich review ratings information. For instance, let us look at the Table 5.5. In this table, only a couple of example users who have rated reviews of a certain user without issuing a trust value are listed. In Table 5.5, the active user is identified with the corresponding $User\ ID$. The reviewer who has received the highest number of ratings from the active user is labeled as $Friend1$, and number of review ratings given to her/his reviews are referred as $Friend1$ *RR Count*. $Friend1$ $Trust\ Value$ column shows us if the active user has issued an explicit trust value to that reviewer, namely $Friend1$.

Table 5.5: User's web of trust and review rating examples

User ID	Friend1 ID	Friend1 RR Count	Friend1 Trust Value
57	5477	31	0
58	4648	27	0
58	882	10	0
63	328	5	0
66	1250	21	0
68	4061	27	0

Reviewers are sorted in descending order according to the number of review ratings given by the current user. Same as the previous analysis, 5 reviewers with the highest number of ratings are extracted for evaluation. After this point, 5 reviewers with the highest number of ratings will be referred as the top 5 reviewers, 4 reviewers with the highest number of ratings will be referred as the top 4 reviewers, and so on.

In this experiment, the aim is to check that whether fewer top reviewers reflect the characteristics of the top 5 reviewers or not. If number of ratings given by less number of reviewers is already correlated with the number of ratings given by the top 5 reviewers, we should eliminate the redundant ones. Therefore, top 5 reviewers are compared with top 4 reviewers. If results show us they are correlated, top 5 reviewers are compared with top 3 reviewers. This process continues till the minimum number of top reviewers reflecting top 5 reviewers are found out. As the previous experiment, this analysis is also done by two-sample Kolmogorov-Smirnov test. Results of this experiment are presented in Table 5.6. The abbreviations h, p and k represent the same means as stated in Section 5.1.2.

Table 5.6: User's review network distribution analysis

	Top 5 - top 4 reviewers	Top 5 - top 3 reviewers	Top 5 - top 2 reviewers
h	0	0	1
p	0.08	0.05	0.00
k	0.04	0.08	0.03

The results are nearly same with web of trust analysis. The top 2 reviewers do not represent the sample of top 5 reviewers; so that at least 3 reviewers' results should be used. To decrease computational complexity, top 3 reviewers are selected to be used in the experiments instead of using the top 5 or 4 reviewers with the highest number of ratings.

The top 3 reviewers with the highest number of ratings is used in the review phase of the proposed method as described in Section 4.2. Latent vector of a user with a higher MAUE is replaced with the latent vectors of those 3 reviewers in a linearly weighted approach.

5.1.4 MAUE Threshold Selection for the Review Phase

In the fourth step of the proposed method, described in Section 4.2, reviewing user latent vectors is mentioned. Reviewing process is added to the algorithm to improve the algorithm and output more accurate and high quality predictions. The users who are satisfied with the predictions they have received, are not involved in this reviewing user latent vectors phase. However, since we do not have the option to ask each user if they are satisfied or not about the predictions, we have to define an evaluation criteria. This criteria is the MAUE in the scope of this thesis, due to MAUE's ability to measure the error rates on user basis.

Recall that, the proposed method is a hybrid method; evaluation scores are computed with the tuples in validation part of the dataset. According to the results, the users who are satisfied with the predictions they receive, in other words users who have a MAUE below a certain threshold remain the same after the training. On the other hand, more training is needed for the latent vectors of unsatisfied users, users with high MAUE, and so that Algorithm 1 is applied to those users.

After selecting MAUE as our evaluation metric, choosing a threshold to set apart the happy users from the unhappy ones is our next step. Experiments with thresholds 2.0, 2.5, 3.0, 3.5, 4.0 are done to determine the most effective threshold value for the reviewing phase. MAUE is calculated for the validation dataset and latent vectors of the users who have scored more than 4.0 of MAUE are reviewed with the latent vectors of the three reviewers whom the active user have rated mostly. The same process is repeated for users with MAUE more than 3.5, 3.0, 2.5 and 2.0.

As mentioned in the Section 5.1, for each threshold value the algorithm is run for 5 times independently with the same training, test and validation parts. The results of the threshold selection executions are given in the following Table 5.7.

Table 5.7: Effect of MAUE threshold selection on reviewing users latent vectors

Threshold	RMSE	MAE	MAUE	MAIE
4.0	1.19	0.93	0.94	0.90
3.5	1.19	0.92	0.93	0.89
3.0	1.17	0.88	0.91	0.88
2.5	1.17	0.91	0.93	0.88
2.0	1.19	0.93	0.93	0.86

5.2 Results

In this section, evaluation results of the proposed method will be presented and then the reasons behind these results will be discussed. Table 5.8 shows the performance of the proposed method, described in Section 4, evaluation results are listed for training, validation and test sets.

Table 5.8: Proposed method evaluation results

Dataset part	RMSE	MAE	MAUE	MAIE
Training	1.07	0.81	0.75	0.73
Validation	1.17	0.89	0.91	0.86
Test	1.17	0.88	0.91	0.88

All four error metrics, RMSE, MAE, MAUE and MAIE, are smaller for the training dataset than for validation and test datasets, since the model may overfit for training data. Evaluation metrics produce nearly same results in test and validation sets due to preparing test and validation datasets in the same way and not making them visible to the method before the evaluation phase. As stated in Section 2.6, MAE computes the total error rate without focusing on how many users are satisfied with the predictions. Therefore, it produces inaccurate results when there are both heavy raters and cold start users. In our dataset, there are both heavy raters and cold start users. The small difference in MAE and MAUE values show that the proposed method works also for cold start users.

After comparing the results of training, validation and test sets, the test set is analyzed in detail. Before examining the results, the distribution of testing data is given in Table 5.9. The dataset is divided into 7 parts, according to the number of rating tuples of the corresponding users in the training set. Parts are respectively users with 0 ratings, 1-10, 11-20, 21-40, 41-80, 81-160 ratings and users who have stated more than 160 ratings in the training set. As seen from the Table 5.9, 8.88% namely 1,983 users are pure cold start users, who did not take place in the training phase. 18,277 users have only rated 1-10 items and therefore, 90.72% of the users did not provide enough information about themselves for the test phase. In the test set, 6.41%, namely 1,085 out of 16,925 tuples belong to cold start users - those who were not in the training set. The system is tested for the users with 1-10 rating with 4,929 tuples which makes 29.09% of the test set. Therefore, a total of 35.50%, the huge part of the test set is composed of cold start users. Another important point about the test set is, 23.79% of the tuples belong to heavy raters, the users with more than 160 rating values. In this case, as mentioned in Section 2.6, it is important to measure the accuracy of the system without dominating cold-start users with the heavy raters; therefore it is crucial to deploy and exploit

MAUE metric instead of MAE.

Table 5.9: Distribution of testing data according to number of ratings

Dataset part	0	1-10	11-20	21-40	41-80	81-160	>160
Number of users	1,983	18,277	1,050	561	242	120	100
Percentage of users	8.88	81.84	4.70	2.51	1.08	0.54	0.45
Number of test tuples	1,085	4,924	1,685	1,913	1,594	1,697	4,027
Percentage of test tuples	6.41	29.09	9.96	11.30	9.42	10.03	23.79

In the Table 5.10, test results using four evaluation criteria are presented for each dataset part. As we can see from the MAUE, the results are not only good for heavy raters but also for the cold start users.

Table 5.10: Results of the Proposed Method on different user rating scales

Dataset part	0	1-10	11-20	21-40	41-80	81-160	>160
RMSE	1.20	1.20	1.16	1.14	1.20	1.16	1.14
MAE	0.97	0.90	0.88	0.87	0.89	0.90	0.88
MAUE	0.96	0.91	0.88	0.86	0.90	0.89	0.89
MAIE	1.00	0.87	0.88	0.87	0.92	0.91	0.89

Another detailed analysis is done on the test set, by splitting the data into 7 parts using the number of trusts issued by each user. In Table 5.11, number of users represents the user count in the training set. Parts are respectively users with 0 trust values, 1-10, 11-20, 21-40, 41-80, 81-160 trust values and users who have stated trust values to more than 160 users in the training set. 5,742 users, corresponding 25.71% of the users did not state any trust value in any user. 52.82% of the users issued trust values to 1-10 users.

The proposed method is evaluated with the test set composed of tuples belonging mostly to users who have issued 1-10 trust values, which is the 30.19% of the test set. A huge part of the tuples, namely 2,477 tuples, 14.64% of the test set is ratings of users who have not stated trust value to any other user in the system. On the other hand, there are rating tuples of users with rich trust information, 11.85% of the tuples are the ratings of users who have trusted more than 160 users.

Table 5.11: Distribution of testing data according to number of trusts

Dataset part	0	1-10	11-20	21-40	41-80	81-160	>160
Number of users	5,742	11,797	1,666	1,382	1,025	525	196
Percentage of users	25.71	52.82	7.46	6.19	4.59	2.35	0.88
Number of test tuples	2,477	5,109	1,455	1,936	1,869	2,074	2,005
Percentage of test tuples	14.64	30.19	8.60	11.44	11.04	12.25	11.85

Table 5.12: Results of the Proposed Method on different user trust scales

Dataset part	0	1-10	11-20	21-40	41-80	81-160	>160
RMSE	1.18	1.19	1.20	1.16	1.14	1.12	1.11
MAE	0.93	0.92	0.91	0.90	0.88	0.86	0.85
MAUE	0.92	0.90	0.91	0.89	0.88	0.82	0.80
MAIE	0.94	0.90	0.90	0.88	0.88	0.85	0.85

5.3 Comparison

In this section, evaluation results of the proposed method are compared with Probabilistic Matrix Factorization (PMF) method [53] and SoRec method [36]. These three methods are compared using four evaluation criteria, RMSE, MAE, MAUE and MAIE, defined in Section 2.6 and their training times.

PMF method aims to factorize the items and users in n-dimensional space, and represents their characteristics without having content information. In PMF method, only user-item ratings matrix is considered.

SoRec [36], which is described in detail in Section 3.5.2, is a factor analysis method based on the probabilistic graphical model fusing the user-item ratings matrix with the users' trusts matrix by sharing a common latent low-dimensional user feature matrix. The experimental analysis done in the scope of SoRec shows that the method generates better results compared to the other collaborative filtering algorithms which do not consider social networks [35].

Comparative results of these three algorithms are shown in the Table 5.13. The error metrics are gathered from the results of test data computations. The methods are executed 5 times independently and the results in the table are the mean of these results.

Table 5.13: Comparison results

Method	RMSE	MAE	MAUE	MAIE	Training Time (sec)
PMF	1.64	1.28	1.45	1.41	197.03
SoRec	1.20	0.95	0.99	0.93	13063.34
Proposed Method	1.17	0.88	0.91	0.88	230.06

The results indicate better performance in the proposed method described in Section 4; using users' review ratings information instead of user to user trust information gives more accurate predictions. This situation is a result of basically having more dense review ratings information.

Training time for these three methods include the time spent to make a recommendation for the user. For instance, training time of the proposed method not only covers the learning phase of the algorithm, but also the review phase of latent vectors.

MAUE in PMF algorithm is significantly higher than the MAUE in the proposed method, since we care about making good predictions for not only a small part of users, but for all users existing in the system. The achievement of the updating phase in the proposed method is apparent with the improvement in MAUE and MAIE metrics.

The results show that although there is no significant difference between the proposed method and SoRec algorithm in terms of accuracy, the difference between the execution times are considerable high. In the proposed method, it is important to make all users happy, so that we do not focus on only learning the characteristics of the users with high number of ratings; also users with sparse ratings are examined in detail. The time efficiency in the algorithm is a result of not spending unnecessarily extra time on the users for whom the system already makes nearly perfect predictions. It is crucial to enhance the latent vectors of the users with sparse ratings, therefore there is no need to bloat the algorithm with needless trust information.

With the help of review ratings information, we are able to exploit trust information which is stated in an implicit manner. During experimentation, it was observed that users mostly prefer giving ratings to the reviewers that they follow implicitly. In other words, the review ratings that a user has given are not extended to a huge number of reviewers. As the experiments show, review rating information also produces quality results compared to explicit trust based algorithms. Therefore, review ratings information can be dealt as a trust statement in a trust-based implementation.

After comparing the results of the methods on test set as a whole, now, let us analyze the results on the parts of the test set which are introduced in Section 5.2. The first analysis will be on the results of different user rating scales of the test set. The related results of the proposed method is presented in Table 5.12. In Table 5.14, results of PMF method using four evaluation criteria on different user rating scales are presented. From Table 5.14, it is seen that, the algorithm is not capable of making useful predictions for cold start users since it outputs a MAE of 3.04 and MAUE of 3.03.

Table 5.14: Comparison of PMF on different user rating scales

Dataset part	0	1-10	11-20	21-40	41-80	81-160	>160
RMSE	3.28	1.50	1.39	1.47	1.44	1.60	1.55
MAE	3.04	1.15	1.05	1.09	1.08	1.20	1.17
MAUE	3.03	1.16	1.04	1.10	1.06	1.18	1.16
MAIE	3.03	1.28	1.12	1.16	1.12	1.25	1.24

In Table 5.15, results of the SoRec method evaluation on different user rating scales is presented using four evaluation metrics.

As seen from the comparison of results presented in Table 5.10 and Table 5.15, there is a significant improvement in all four error metrics for the cold start users.

The next analysis is done by evaluating both method on the parts of the test set introduced in Section 5.2 which categorizes users based on the number of trust statements. The related results of the proposed method is presented in Table 5.12.

Table 5.15: Comparison of SoRec on different user rating scales

Dataset part	0	1-10	11-20	21-40	41-80	81-160	>160
RMSE	1.46	1.22	1.17	1.15	1.21	1.17	1.17
MAE	1.32	0.95	0.91	0.89	0.93	0.92	0.91
MAUE	1.31	0.97	0.91	0.90	0.92	0.92	0.91
MAIE	1.37	0.92	0.91	0.89	0.93	0.93	0.92

Table 5.16: Comparison of PMF on different user trust scales

Dataset part	0	1-10	11-20	21-40	41-80	81-160	>160
RMSE	1.78	1.78	1.53	1.52	1.58	1.59	1.64
MAE	1.34	1.34	1.17	1.14	1.18	1.18	1.22
MAUE	1.56	1.48	1.28	1.22	1.26	1.28	1.12
MAIE	1.33	1.38	1.20	1.19	1.25	1.23	1.26

In Table 5.17, results of the SoRec method evaluation on different user trust scales is presented using four evaluation metrics.

Table 5.17: Comparison of SoRec on different user trust scales

Dataset part	0	1-10	11-20	21-40	41-80	81-160	>160
RMSE	1.26	1.25	1.22	1.18	1.17	1.13	1.17
MAE	0.99	0.99	0.94	0.93	0.91	0.88	0.90
MAUE	1.05	1.02	0.96	0.93	0.94	0.88	0.82
MAIE	0.99	0.96	0.94	0.93	0.91	0.88	0.90

The algorithms are also compared on basis of their computational complexities. In all three methods, the main computation of gradient methods is evaluating the object function L and its gradients against variables [36].

In PMF method [53], because of the sparse ratings matrix R, the computational complexity of the object function L is $O(\rho_R d)$, where ρ_R is the number of nonzero rating tuples in the matrix

R and d is the number of latent vector dimension of U. The computational complexities of the gradients $\frac{\partial L}{\partial U}$ and $\frac{\partial L}{\partial V}$ in PMF method are the same and equal to $O(\rho_R d)$. Therefore, in one iteration total computational complexity is $O(\rho_R d)$, linear with the number of nonzero entries in ratings.

In SoRec method [36], due to the sparsity of ratings and trusts matrices, respectively R and C, the computational complexity of the object function L is $O(\rho_R d + \rho_C d)$, where ρ_R and ρ_C are respectively the number of nonzero entries in the matrices R and C and d represents the dimension number of the feature representation. Recall that $U \in \Re^{d \times m}$ and $Z \in \Re^{d \times m}$. The computational complexities of the gradients $\frac{\partial L}{\partial U}$, $\frac{\partial L}{\partial V}$ and $\frac{\partial L}{\partial Z}$ in SoRec method are respectively $O(\rho_R d + \rho_C d)$, $O(\rho_R d)$ and $O(\rho_C d)$. Therefore, in one iteration total computational complexity is $O(\rho_R d + \rho_C d)$, linear with the number of nonzero entries in ratings and trusts matrices.

In the proposed method, Section 4, the computational complexity is not much than the PMF method. Firstly, note that the dimension of latent vector features are presented as d. The main computation of two algorithms are the same because they both use only the ratings matrix in the training phase and so that the computational complexity of the object function L is $O(\rho_R d)$, where ρ_R is the number of nonzero rating tuples in the matrix R which are used in the training. The computational complexities of the gradients $\frac{\partial L}{\partial U}$ and $\frac{\partial L}{\partial V}$ in proposed method are the same and equal to $O(\rho_R d)$. In addition to the training phase, there are computations in the review phase of the proposed method. However, in the review phase where the Algorithm 1 is applied, only the unsatisfied users are handled. The computational complexity of Algorithm 1 is $O(\bar{n} d)$ where $\bar{n}$ represents the number of not satisfied users. However, since the number of unhappy users are less than the total number of users, in one iteration total computational complexity of the proposed method is $O(\rho_R d)$, meaning that it is linear with the number of nonzero entries in ratings.

According to the computational complexities of the three algorithms discussed above, the proposed method outperforms the complexity of SoRec method, meanwhile producing better predictions.

5.4 Conclusion

The proposed method is mainly based on implicit trust information. In the scope of this thesis work, the experiments are done with the crawled Epinions.com [22] dataset. However, the idea proposed is not only limited to the datasets gathered from Epinions.com website; it is important to note that building social trust networks is not only making use of explicitly stated trust scores, but also interpreting implicit trust information may also be useful in the recommender systems.

Proposed method is capable of making predictions for the users without any ratings if and only if they rated one or more reviews in the system. However, if we do not know anything about the user; namely a user without any rating, and also without any trust or review rating information, it is hard to make a prediction for that user.

When we take a look back to the beginning of this work, we can say that the most time-consuming part was during the crawling stage of the dataset from the Epinions.com, dataset collection was completed in about 3 months. We studied the website for a few weeks before starting to crawl the data from the website to make a list of what to collect and do not miss any of the important features about the items and users that are presented in the site. However, if we misunderstood and could not catch a key feature, it would again consume a long period of time, which would be demotivating for us. Basically, this is a result of not being able to find a dataset both with ratings and trusts information including explicit and implicit data collected from the actions of the users. Dataset is the main limitation for the researchers studying social recommender systems.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

Trust based recommender systems mimic the real world in a way in which people usually ask for an advice about something to their friends. The experiments show that people tend to believe in information that was concretely given to them by the help of personal interactions better than the abstract information. However, finding quality and practical trust relationships is a challenging issue. Users do not prefer wasting their time to find the reviewers that they trust. They do not tend to keep the reviewers in mind whose reviews or suggestions they find consistently helpful by issuing explicit trust statements. It is important to make use of every action of the user to find out more information about her/him and understand the user's behavior by exploiting implicit data. Implicit data may be any action that is directly related to ratings information or other types of information. This work is based on the idea of making the trust based recommender systems more personalized with the help of implicit "trust" data. More specific trust information helps us to find the "real" friends we are looking for.

After analyzing the Epinions.com dataset, we have noticed that invaluable implicit trust information exists in review ratings data - most of the users express ratings to the reviews, rather than writing reviews. In addition to this, an active user's review ratings are generally ratings given to the same user's reviews. This fact resulted in an assumption of "users may prefer giving ratings to users that they find helpful, rather than trying to track reviewers and adding them to their web of trust". With the help of this assumption, the study focused on interpreting review ratings and deploying review ratings information in the system to make a more well defined social network.

Before starting the implementation, matrix factorization models in trust based recommender systems are examined in detail. While building up the recommender system, deploying a matrix factorization model instead of neighborhood based model is preferred due to their cost and time efficient results in the literature. As the dataset, Epinions.com web site is selected as the data source and is crawled on movies domain. Since our main goal is to define more specific trust information, a track, namely movies, is selected to deploy a interpersonal trust information instead of system level trust. The reason behind selecting movies domain is due to the relatively richer information compared to the other tracks on the web site. Beside the proposed method, crawling the dataset is the other major contribution of this study.

Probabilistic matrix factorization based model is applied to the crawled Epinions.com dataset. The proposed method is a hybrid model combining review ratings information as the trust information with the ratings while making a prediction for the user. Explicit trust information is not deployed in the system to reduce the computational complexity. Instead of trust statements, review ratings are incorporated with the ratings whenever it is needed. For the users with enough ratings information, the model uses only ratings while predicting the rating of an item. However, the users who are not happy with the predictions that they receive, the system looks for the advices of the users in their trust network which is assumed to be defined implicitly with review ratings. The proposed method scales linearly with the number of nonzero ratings in the user-item ratings matrix while making quality and useful predictions. Experimental results confirm that making an implicit trust derived from review ratings information boosted collaborative filtering algorithm produces both a scalable and favorable recommender system compared to state-of-the-art methods.

6.2 Future Work

Recommender systems, especially trust based recommender systems constitute hot research topics for many researchers. Different approaches and methods are being proposed in a very promising way.

Trust based recommender systems suffer from not being able to find a dataset with rich information. Therefore, the dataset itself may also be improved in a several aspects. In order to provide context specific trust information, while defining trust value, subset of trust value can be asked. Another option is to prompt users explicitly if they are happy or not with the prediction they have received.

A major consideration for the future, particularly to improve the quality of trust relationships, may be considering distrust information while building up trust network. However, incorporating distrust statements with trust statements is not a straightforward problem. Distrust relationships may not be propagated like trust issues since propagation of distrust may also include trust relationships, meaning that the probability to make an error increases. And also inferring distrust information from the implicit data available is a hard task - giving low ratings to a user's one or more reviews does not always mean that the active user distrusts that reviewer; the reviewer may be a strong reviewer who writes reviews to lots of items, so that there may be one or more reviews which do not catch your attention, and also there may be other reviews which you think as exactly the same. Another concern about distrusts is that, only Epinions.com provides an extended dataset including the distrust relationships among users [21]. Due to not having plentiful different datasets with distrust information, experimentation may be biased to Epinions.com dataset.

The dataset used is extracted from Epinions web site in movies domain based on the intuition of trusting a friend's taste in movies does not mean trusting the same friend's taste in any other area such as cars, books, music, sports, etc. In the future, it may be a good idea to experiment the proposed method on another domain, such as books, sports, etc. to make a more general and robust model. In addition to this, to deploy a more personalized trust, the same idea may extend as, we may like a friend's taste in action movies but we may not share the same opinion on science-fiction movies. In the future, we may experiment the system with more localized trust information. The trust information may be extracted from the system on more specific domains; genre information may be a starting point for building localized trust such as comedy, drama, action, science-fiction, animation, etc. In addition to that, localized trust may be built in different sub-domains and their results may be compared. For instance, the results of localized sub-domains based on release date, directors, genre, tags etc. may be discussed to find which one is the most effective way to build more localized and quality trust information.

Another improvement for the proposed method may be making an ensemble of trust scores and review ratings of the users. In a comparative analysis done in this work, we have seen that the trust network and reviewers that are commonly rated by the user are parallel to each other. Even if, we have seen that the users in our dataset have more diverse and quality review ratings information compared to the explicit web of trust of the same users; this may not be the same case in another dataset. After analyzing the dataset in detail, if explicit and implicit trusts present diverse information, incorporating both will enrich the social network and as a result, making predictions will be easier.

REFERENCES

[1] A. Abdul-Rahman and S. Hailes. A distributed trust model. In *Proceedings of the 1997 workshop on New security paradigms*, page 60. ACM, 1998.

[2] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, pages 734–749, 2005.

[3] S. Ahn and C.K. Shi. Exploring movie recommendation system using cultural metadata. *Transactions on Edutainment II*, pages 119–134, 2009.

[4] Amazon. Amazon.com: Online shopping for electronics, apparel, computers, books, dvds & more, August 2010. http://www.amazon.com.

[5] A. Ansari, S. Essegaier, and R. Kohli. Internet recommendation systems. *Journal of Marketing Research*, 37(3):363–375, 2000.

[6] P. Avesani, P. Massa, and R. Tiella. Moleskiing: a trust-aware decentralized recommender system. In *1st Workshop on Friend of a Friend, Social Networking and the Semantic Web. Galway, Ireland*, 2004.

[7] P. Avesani, P. Massa, and R. Tiella. A trust-enhanced recommender system application: Moleskiing. In *Proceedings of the 2005 ACM symposium on Applied computing*, page 1593. ACM, 2005.

[8] P. Avesani, P. Massa, and R. Tiella. Moleskiing. it: A trust-aware recommender system for ski mountaineering. *International Journal for Infonomics*, 2005.

[9] M. Balabanović and Y. Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):72, 1997.

[10] A.B. Barragáns-Martínez, E. Costa-Montenegro, J.C. Burguillo, M. Rey-López, F.A. Mikic-Fonte, and A. Peleteiro. A hybrid content-based and item-based collaborative filtering approach to recommend TV programs enhanced with singular value decomposition. *Information Sciences*, 2010.

[11] C. Basu, H. Hirsh, and W. Cohen. Recommendation as classification: Using social and content-based information in recommendation. In *Proceedings of the National Conference on Artificial Intelligence*, pages 714–720. JOHN WILEY & SONS LTD, 1998.

[12] R.M. Bell, J. Bennett, Y. Koren, and C. Volinsky. The million dollar programming prize. *IEEE Spectrum*, 46(5):28–33, 2009.

[13] E. Bigdeli and Z. Bahmani. Comparing accuracy of cosine-based similarity and correlation-based similarity algorithms in tourism recommender systems. In *Management of Innovation and Technology, 2008. ICMIT 2008. 4th IEEE International Conference on*, pages 469–474. IEEE, 2008.

[14] D. Billsus and M.J. Pazzani. User modeling for adaptive news access. *User Modeling and User-Adapted Interaction*, 10(2):147–180, 2000.

[15] P. Bonhard. Improving recommender systems with social networking. In *Proceedings Addendum of the 2004 ACM Conference on Computer-Supported Cooperative Work. Chicago, IL, USA*, 2004.

[16] P. Bonhard. Who do trust? Combining recommender systems and social networking for better advice. In *Proceedings of the Wokshop Beyond Personalization 2005, in conjunction with the International Conference on Intelligent User Interfaces IUI'05*, pages 89–90. Citeseer, 2005.

[17] R. Burke. Knowledge-based recommender systems. *Encyclopedia of Library and Information Systems*, 69(Supplement 32):175–186, 2000.

[18] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin. Combining content-based and collaborative filters in an online newspaper. In *Proceedings of ACM SIGIR Workshop on Recommender Systems*, pages 60–64. Citeseer, 1999.

[19] M.K. Condli, D.D. Lewis, D. Madigan, C. Posse, and I. Talaria. Bayesian mixed-effects models for recommender systems. In *Conference SIGIR-99 Workshop on Recommender Systems: Algorithms and Evaluation*. Citeseer, 1999.

[20] C. Danescu-Niculescu-Mizil, G. Kossinets, J. Kleinberg, and L. Lee. How opinions are received by online communities: a case study on amazon. com helpfulness votes. In *Proceedings of the 18th international conference on World wide web*, pages 141–150. ACM, 2009.

[21] Epinions.com. Extended epinions dataset - trustlet. http://www.trustlet.org/wiki/ Extended_Epinions_dataset, August 2010.

[22] Epinions.com. Reviews from epinions. http://www.epinions.com/, August 2010.

[23] J. Golbeck. Semantic web interaction through trust network recommender systems. In *Proceedings of the ISWC 2005 Workshop on End User Semantic Web Interaction, Galway, Ireland*. Citeseer, 2005.

[24] J. Golbeck and J. Hendler. Filmtrust: Movie recommendations using trust in web-based social networks. In *Proceedings of the IEEE Consumer communications and networking conference*, volume 42, pages 43–44. Citeseer, 2006.

[25] J. Golbeck and A. Mannes. Using trust and provenance for content filtering on the semantic web. In *Proceedings of the Models of Trust for the Web Workshop*. Citeseer, 2006.

[26] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins. Propagation of trust and distrust. In *Proceedings of the 13th international conference on World Wide Web*, pages 403–412. ACM, 2004.

[27] J.L. Herlocker, J.A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, page 237. ACM, 1999.

[28] H. Kautz, B. Selman, and M. Shah. Referral Web: Combining social networks and collaborative filtering. *Communications of the ACM*, 40(3):63–65, 1997.

[29] Joseph A. Konstan. Introduction to recommender systems: Algorithms and evaluation. *ACM Trans. Inf. Syst.*, 22(1):1–4, 2004.

[30] Y. Koren. Collaborative filtering with temporal dynamics. *Communications of the ACM*, 53(4):89–97, 2010.

[31] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37, 2009.

[32] J.S. Lee and S. Olafsson. Two-way cooperative prediction for collaborative filtering recommendations. *Expert Systems with Applications*, 36(3):5353–5361, 2009.

[33] G. Linden, B. Smith, and J. York. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, 7(1):76–80, 2003.

[34] H. Luo, C. Niu, R. Shen, and C. Ullrich. A collaborative filtering framework based on both local user similarity and global user similarity. *Machine Learning*, 72(3):231–245, 2008.

[35] H. Ma, I. King, and M.R. Lyu. Learning to recommend with social trust ensemble. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 203–210. ACM, 2009.

[36] H. Ma, H. Yang, M.R. Lyu, and I. King. Sorec: Social recommendation using probabilistic matrix factorization. In *Proceeding of the 17th ACM conference on Information and knowledge management*, pages 931–940. ACM, 2008.

[37] P. Massa and P. Avesani. Trust-aware collaborative filtering for recommender systems. *On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE*, pages 492–508, 2004.

[38] P. Massa and P. Avesani. Controversial users demand local trust metrics: An experimental study on epinions.com community. *Proceedings of the National Conference on Artificial Intelligence*, 20(1):121, 2005.

[39] P. Massa and P. Avesani. Trust-aware recommender systems. In *Proceedings of the 2007 ACM conference on Recommender systems*, page 24. ACM, 2007.

[40] P. Massa and P. Avesani. Trust metrics in recommender systems. *Computing with social trust*, pages 259–285, 2009.

[41] The Mathworks. Two-sample kolmogorov-smirnov test, August 2010. http://www.mathworks.com/access/helpdesk/help/toolbox/stats/kstest2.html.

[42] D.G. Medhi and J. Dakua. MovieReco: A recommendation system. In *WEC'05: The Second World Enformatika Conference*. Citeseer, 2005.

[43] P. Melville, R.J. Mooney, and R. Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *Proceedings of the National Conference on Artificial Intelligence*, pages 187–192. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2002.

[44] R. Mukherjee, G. Jonsdottir, S. Sen, and P. Sarathi. Movies2go: An online voting based movie recommender system. In *Proceedings of the fifth international conference on Autonomous agents*, page 115. ACM, 2001.

[45] Netflix. Netflix: Tv & movies instantly streamed online. http://www.netflix.com, August 2010.

[46] J. O'Donovan and B. Smyth. Trust in recommender systems. In *Proceedings of the 10th international conference on Intelligent user interfaces*, pages 167–174. ACM, 2005.

[47] G. Orr. Linear neural networks. http://www.willamette.edu/˜gorr/classes/cs449/ linear2.html, August 2010.

[48] M.J. Pazzani. A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, 13(5):393–408, 1999.

[49] A. Popescul, L.H. Ungar, D.M. Pennock, and S. Lawrence. Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. In *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, pages 437–444. Citeseer, 2001.

[50] A.M. Rashid, I. Albert, D. Cosley, S.K. Lam, S.M. McNee, J.A. Konstan, and J. Riedl. Getting to know you: learning new user preferences in recommender systems. In *Proceedings of the 7th international conference on Intelligent user interfaces*, pages 127–134. ACM, 2002.

[51] J.D.M. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22nd international conference on Machine learning*, page 719. ACM, 2005.

[52] E. Rich. User modeling via stereotypes. *Cognitive science*, 3(4):329–354, 1979.

[53] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. *Advances in neural information processing systems*, 20:1257–1264, 2008.

[54] K. Sarda, P. Gupta, D. Mukherjee, S. Padhy, and H. Saran. A distributed trust-based recommendation system on social networks, 2008.

[55] 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*, page 295. ACM, 2001.

[56] 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*, page 167. ACM, 2000.

[57] B.M. Sarwar, G. Karypis, J.A. Konstan, and J. Riedl. Application of dimensionality reduction in recommender system–a case study. In *ACM WebKDD 2000 Web Mining for E-Commerce Workshop*. Citeseer, 2000.

[58] A.I. Schein, A. Popescul, L.H. Ungar, and D.M. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260. ACM, 2002.

[59] S. Sen, J. Vig, and J. Riedl. Tagommenders: Connecting users to items through tags. In *Proceedings of the 18th international conference on World wide web*, pages 671–680. ACM, 2009.

[60] U. Shardanand and P. Maes. Social information filtering: Algorithms for automating ”word of mouth”. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 210–217. ACM Press/Addison-Wesley Publishing Co., 1995.

[61] R. Sinha and K. Swearingen. Comparing recommendations made by online systems and friends. In *Proceedings of the Delos-NSF workshop on personalization and recommender systems in digital libraries*. Citeseer, 2001.

[62] I.M. Soboro and C.K. Nicholas. Combining content and collaboration in text filtering. In *Proceedings of the IJCAI Workshop on Machine Learning in Information Filtering*, volume 86, page 91. Citeseer, 1999.

[63] X. Su, T.M. Khoshgoftaar, X. Zhu, and R. Greiner. Imputation-boosted collaborative filtering using machine learning classifiers. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 949–950. ACM, 2008.

[64] G. Takács, I. Pilászy, B. Németh, and D. Tikk. Investigation of various matrix factorization methods for large recommender systems. In *Proceedings of the 2nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition*, pages 1–8. ACM, 2008.

[65] G. Takács, I. Pilászy, B. Németh, and D. Tikk. Matrix factorization and neighbor based algorithms for the netflix prize problem. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 267–274. ACM, 2008.

[66] M. Tiemann, S. Pauws, and F. Vignoli. Ensemble learning for hybrid music recommendation. *Int. Cont. on Music Information Retrieval*, pages 179–181, 2007.

[67] Twitcritics. Real time movie reviews from twitter. http://www.twitcritics.com/, August 2010.

[68] Twitter.com. Twitter. http://www.twitter.com/, August 2010.

[69] R. Van Meteren and M. Van Someren. Using content-based filtering for recommendation. In *Proceedings of the Machine Learning in the New Information Age: MLnet/ECML2000 Workshop*. Citeseer, 2000.

[70] E. Vozalis and K.G. Margaritis. Analysis of recommender systems algorithms. In *Proceedings of the 6th Hellenic European Conference on Computer Mathematics and its Applications (HERCMA-2003), Athens, Greece*, 2003.

[71] S. Yoo, Y. Yang, F. Lin, and I.C. Moon. Mining social networks for personalized email prioritization. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 967–976. ACM, 2009.

APPENDICES

APPENDIX A NUMBER OF BATCHES EXPERIMENTS

In this appendix, results of the number of batches selection experiments are given. All experiments done to determine the best number of batches are carried out with latent vectors dimension equal to 10, mini-batches of size 1000 and 60 epochs.

Each experiment is executed 5 times with the same parameters using the same training, validation and test datasets and evaluation metrics are gathered from the averages of these runs.

The detailed results of the experiments are given in the following tables. The tables are organized according to the evaluation metrics RMSE, MAE, MAUE and MAIE, which are presented respectively in the tables A.1, A.2, A.3 and A.4.

Table A.1: Number of batches experiment - RMSE

Number of Batches	Training	Validation	Test
20	1.1083	1.1815	1.1791
30	1.0885	1.1771	1.1748
40	1.0825	1.1802	1.1788
50	1.0726	1.1771	1.1776
60	1.0646	1.1785	1.1751
70	1.0596	1.1779	1.1756
80	1.0577	1.1796	1.1807
90	1.0510	1.1780	1.1767
100	1.0495	1.1799	1.1770
110	1.0498	1.1824	1.1828
120	1.0481	1.1827	1.1837

Table A.2: Number of batches experiment - MAE

Number of Batches	Training	Validation	Test
20	0.8443	0.9067	0.9103
30	0.8302	0.9040	0.9078
40	0.8258	0.9057	0.9086
50	0.8182	0.9034	0.9085
60	0.8131	0.9045	0.9066
70	0.8097	0.9037	0.9066
80	0.8079	0.9047	0.9103
90	0.8024	0.9034	0.9063
100	0.8014	0.9045	0.9066
110	0.8010	0.9062	0.9112
120	0.8000	0.9062	0.9115

Table A.3: Number of batches experiment - MAUE

Number of Batches	Training	Validation	Test
20	0.8102	0.9163	0.9294
30	0.7820	0.9124	0.9255
40	0.7652	0.9163	0.9280
50	0.7502	0.9161	0.9286
60	0.7395	0.9162	0.9283
70	0.7295	0.9163	0.9294
80	0.7212	0.9182	0.9329
90	0.7142	0.9196	0.9313
100	0.7085	0.9208	0.9325
110	0.7060	0.9215	0.9377
120	0.7010	0.9217	0.9381

Table A.4: Number of batches experiment - MAIE

Number of Batches	Training	Validation	Test
20	0.8061	0.8922	0.8966
30	0.7759	0.8891	0.8940
40	0.7600	0.8870	0.8933
50	0.7422	0.8848	0.8916
60	0.7291	0.8859	0.8879
70	0.7175	0.8847	0.8862
80	0.7117	0.8850	0.8934
90	0.6990	0.8850	0.8876
100	0.6959	0.8845	0.8882
110	0.6888	0.8862	0.8903
120	0.6839	0.8852	0.8914

APPENDIX B NUMBER OF EPOCHS EXPERIMENTS

In this appendix, results of the number of epochs selection experiments are given. All experiments done to determine the best number of batches are carried out with latent vectors dimension equal to 10, mini-batches of size 1000 and number of batches equal to 50. Number of epochs are changed between 20 to 200 in steps of 20.

Each experiment is repeated 5 times with the same parameters using the same training, validation and test datasets and evaluation metrics are gathered from the averages of these runs. The detailed results of the experiments are given in the following tables. The tables are organized according to the evaluation metrics RMSE, MAE, MAUE and MAIE, which are presented respectively in the tables B.1, B.2, B.3 and B.4.

Table B.1: Number of epochs experiment - RMSE

Number of Epochs	Training	Validation	Test	Time (sec)
20	1.09	1.20	1.21	54
40	1.07	1.20	1.20	101
60	1.06	1.20	1.20	149
80	1.05	1.20	1.20	196
100	1.05	1.20	1.20	242
120	1.04	1.20	1.20	332
140	1.04	1.21	1.21	380
160	1.04	1.21	1.21	434
180	1.04	1.21	1.21	470
200	1.03	1.21	1.21	517

Table B.2: Number of epochs experiment - MAE

Number of Epochs	Training	Validation	Test
20	0.85	0.97	0.98
40	0.84	0.96	0.96
60	0.82	0.94	0.94
80	0.80	0.94	0.95
100	0.80	0.94	0.94
120	0.79	0.94	0.94
140	0.79	0.94	0.95
160	0.79	0.94	0.95
180	0.79	0.94	0.95
200	0.79	0.94	0.95

Table B.3: Number of epochs experiment - MAUE

Number of Epochs	Training	Validation	Test
20	0.85	0.99	1.03
40	0.77	0.98	1.02
60	0.72	0.97	1.00
80	0.71	0.98	1.00
100	0.66	0.92	0.92
120	0.69	0.99	1.00
140	0.69	0.99	1.00
160	0.69	0.99	1.00
180	0.68	0.99	1.00
200	0.68	0.99	1.00

Table B.4: Number of epochs experiment - MAIE

Number of Epochs	Training	Validation	Test
20	0.78	0.96	0.95
40	0.71	0.94	0.94
60	0.68	0.93	0.93
80	0.67	0.93	0.93
100	0.66	0.93	0.93
120	0.66	0.93	0.93
140	0.66	0.93	0.93
160	0.66	0.93	0.93
180	0.65	0.93	0.93
200	0.65	0.94	0.94