A COMPARISON OF DIFFERENT RECOMMENDATION TECHNIQUES FOR A HYBRID MOBILE GAME RECOMMENDER SYSTEM

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

HASSANE NATÚ HASSANE CABIR

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

NOVEMBER 2012

Approval of the thesis:

A COMPARISON OF DIFFERENT RECOMMENDATION TECHNIQUES FOR A HYBRID MOBILE GAME RECOMMENDER SYSTEM

Submitted by **HASSANE NATÚ HASSANE CABIR** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen _____
Dean, Graduate School of Natural and Applied Sciences

Prof. Dr. Adnan Yazıcı _____
Head of Department, Computer Engineering

Prof. Dr. Ferda Nur Alpaslan _____
Supervisor, Computer Engineering Dept., METU

Dr. Ruket Cakici _____
Co-Supervisor, Computer Engineering Dept., METU

Examining Committee Members:

Prof. Dr. Nihan Kesim Çiçekli _____
Computer Engineering Dept., METU

Prof. Dr. Ferda Nur Alpaslan _____
Computer Engineering Dept., METU

Assoc. Prof. Dr. Pınar Şenkul _____
Computer Engineering Dept., METU

Assoc. Prof. Dr. İlkay Ulusoy _____
Electrical and Electronics Engineering Dept., METU

Dr. Ruket Çakici _____
Computer Engineering Dept., METU

Date: 12.11.2012

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: Hassane Natú Hassane, Cabir

Signature :

ABSTRACT

A COMPARISON OF DIFFERENT RECOMMENDATION TECHNIQUES FOR A HYBRID MOBILE GAME RECOMMENDER SYSTEM

CABIR, Hassane Natú Hassane

M.S., Department of Computer Engineering

Supervisor: Prof. Dr. Ferda Nur ALPASLAN

Co-Supervisor: Dr. Ruket Cakici

November 2012, 89 pages

As information continues to grow at a very fast pace, our ability to access this information effectively does not, and we are often realize how harder is getting to locate an object quickly and easily. The so-called personalization technology is one of the best solutions to this information overload problem: by automatically learning the user profile, personalized information services have the potential to offer users a more proactive and intelligent form of information access that is designed to assist us in finding interesting objects. Recommender systems, which have emerged as a solution to minimize the problem of information overload, provide us with recommendations of content suited to our needs. In order to provide recommendations as close as possible to a user's taste, personalized recommender systems require accurate user models of characteristics, preferences and needs. Collaborative filtering is a widely accepted technique to provide recommendations based on ratings of similar users, But it suffers from several issues like data sparsity and cold start. In one-class collaborative filtering, a special type of collaborative filtering methods that aims to deal with datasets that lack counter-examples, the challenge is even greater, since these datasets are even sparser. In this thesis, we

present a series of experiments conducted on a real-life customer purchase database from a major Turkish E-Commerce site. The sparsity problem is handled by the use of content-based technique combined with TFIDF weights, memory based collaborative filtering combined with different similarity measures and also hybrids approaches, and also model based collaborative filtering with the use of Singular Value Decomposition (SVD). Our study showed that the binary similarity measure and SVD outperform conventional measures in this OCCF dataset.

Keywords: Recommender Systems, Personalization, User Modeling, Collaborative Filtering, Content Based Filtering, Information Extraction, Singular Value Decomposition

ÖZ

MELEZ MOBİL OYUN TAVSİYE SİSTEMİ İÇİN FARKLI ÖNERİ TEKNİKLERİNİN KARŞILAŞTIRILMASI

CABIR, Hassane Natú Hassane
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Ferda Nur ALPASLAN
Ortak Tez Yöneticisi: Dr. Ruket CAKICI

Kasım 2012, 89 sayfa

Bilgiler hızlı bir tempoda artmaya devam ederken bu bilgilere etkin şekilde erişmek her zaman mümkün olmamaktadır.Kişiselleştirme teknolojisi denilen teknoloji bu bilgi bombardımanı sorununa en iyi çözümlerden biridir: otomatik olarak kullanıcı profilini öğrenerek bilgi servislerini daha etkin kullanıma sunma ve böylece bizim için ilginç olan nesneleri bulmamıza yardımcı olmak için tasarlanmışlardır.. Bilgi yükleme problemini en aza düşürmede çözüm olarak ortaya çıkan tavsiye sistemleri ihtiyaçlarımıza uygun içerik önerisi sunar. Kullanıcı beğenisine mümkün olan en yakın öneriyi sağlamak amacıyla kişiselleştirilmiş tavsiye sistemleri için kesin karakteristik kullanıcı modeli, tercihleri ve ihtiyaçları gereklidir. Kolaboratif filtreleme benzer kullanıcıların değerlendirmelerine dayalı öneri sunmada yaygın olarak kabul edilen bir tekniktir; fakat bu teknik veri seyrekliği ve "cold start" gibi çeşitli problemlerden muzdariptir. Kolaboratif filtreleme metodlarının özel bir tipi olan karşıt örneklerden yoksun verikümeleri ile başa çıkmayı amaçlayan tek sınıflı kolaboratif filtreleme, bu verikümeleri seyrek olduğu için zorluğu daha büyüktür. Bu tez kapsamında büyük bir Türk e-ticaret sitesinin veritabanında gerçek müşteriler üzerinde yapılan bir dizi deney sunuyoruz.

TFIDF ağırlıklarıyla karıştırılmış içerik tabanlı tekniğin kullanılması, farklı benzerlik ölçümleri ile karıştırılmış bellek tabanlı kolaboratif filtrelemesi, melez yaklaşımlar ve Tekil Değer Ayrışımı'nın (TDA) model bazlı kolaboratif filtreleme kullanılmasıyla seyreklik problemi ele alınıyor. Bu çalışma OCCF verikümesinde ikili benzerlik ölçümünün ve TDA'nın geleneksel ölçümlerden üstün olduğunu göstermiştir

Keywords: Tavsiye Sistemleri, Kişiselleştirme, Kullanıcı Modelleme, Kolaboratif Filtreleme, İçerik Bazlı Filtreleme, Bilgi Çıkarımı, Tekil Değer Ayrışımı

To my family

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude and appreciation to Prof. Dr. Ferda Nur Alpaslan and Dr. Ruket Çakici for their encouragement and support throughout this study.

I am deeply grateful to my family for their love and support. Without them, this work could never have been completed.

I am deeply indebted to ISLAMIC DEVELOPMENT BANK (IDB) for providing me with this scholarship and to AUTORIDADE TRIBUTÁRIA DE MOÇAMBIQUE (MOZAMBIAN REVENUE AUTORITY) for authorizing my studies.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

RS	Recommender Systems
IR	Information Retrieval
IF	Information Filtering
CB	Content Based
CBR	Content Based Recommender
CF	Collaborative Filtering
PCC	Pearson Correlation Coefficient
VS	Vector Similarity
CSM	Cosine Similarity Measure
SG	Social Graph
PIP	Proximity – Impact – Popularity
PIPM	Proximity – Impact – Popularity Similarity Measure
SVD	Singular Value Decomposition

CHAPTER 1

INTRODUCTION

1.1 Background

The internet today is a very big place, is easier than ever to produce and publish any kind of information, by anyone, the only requirement is to be connected. Another factor influencing this dramatic grow of the internet, is the automatic generation of information, done implicitly by web application. This scenario has created so much information, making it harder to process all this information by ourselves [1].

This feeling of being overwhelmed by so much information reaches everyone, sooner or later. Information is only valuable if you can access it and use it efficiently. In order to accomplish such a remarkable goal, Recommender systems (RS) were born [2]. This technology assists us to navigate through all these information to find what is more interesting to us.

Developers and vendors use recommender systems to find out what items the user would be interested in without spending too much time searching. Recommender systems are being used in many domains, such as commercial, where better recommendations leads to better profit [5]. Recommender system is a multi-disciplinary field, making use of data mining, machine learning, artificial intelligent, and some others according to the domain.

Recommender systems are dynamic systems; they learn a user profile and keeps on updating it according to the user feedback [4]. They have been exploited for recommending a diverse range of items, for example travel destinations, electronic products, books, and even as friends [6, 7, 8]. When the user interacts

with the system, it learns his tastes and builds a profile, and later it uses this profile to produce the recommendations; the items that best match the user profile are included in the recommendation list [4].

The system learns the user's profile with information given explicitly by the user or implicitly but monitoring his behavior during the interactions. These new acquired information are then consolidated in the respective profile and used for future recommendations. For many systems, it is required that it has learned the user profile upfront, before the recommendations can be computed [6]. A recommender system usually uses the ratings given to items by users to measure the degree to which an item is preferred by a user [10]. It is also required to have item profiles, so that the user's preferences can be expressed by the item's features [9].

Recommender systems has improved since its creation, are using more advanced tools and techniques; this improvement is also noted in the internet and related technologies. These are the developments that motivate the research community to explore better information retrieval tools and techniques [11], presenting to the users satisfaction in using the system.

This thesis addresses a particularly hard case that emerges from a wide range of real-world prediction tasks, it is called one-class collaborative filtering (OCCF), and in this case we try to learn from implicit feedback only, under the constraint that each observation is a positive example [16]. There are many occurrences OCCF, for example, news, bookmarks and some e-commerce systems [17]. In this type of dataset, the training data consists simply of binary data capturing a user's action or inaction, for instance item bought in an online e-commerce website. This genre of dataset is extremely sparse, we only have positive examples and the negative and unlabeled examples are mixed together, making it impossible to make a distinction between them. For example, consider a user who did not buy a game called "Diablo III", there is no way to tell if this game is not interesting to the user or the user was not aware of it.

Some examples of OCCF include:

- Tracking the items a user bought in the past, to predict what items should be recommended next, or
- Using the user's bookmarks to predict which other unknown sites he might like. In both these cases, the observed transactions are positive examples only.

1.2 Recommender Systems As a Research Area

Recommender systems are known as a research area since mid-1990s, when the first studies became publicly available [8]. Starting from there, this technology prospered, new techniques and tools were developed. We still see more improvements year after year, due to the fact that the information is changing very rapidly, more customers join the system, more products are offered to customers, so does the recommender systems technology need to evolve its techniques and tools, to continue to give good results to the users in dealing with the information overload [14]. Examples of such applications include recommending books, CDs and other products at Amazon.com [12], movies by MovieLens [13].

1.3 Problem Definition

This thesis focuses on the comparison of the performance of different recommendation techniques when applied to OCCF games recommendation system. A variety of features are integrated by the system in order to minimize the so-called "data sparsity" problem in recommendation systems. Being One-Class Collaborative Filtering a special case of Collaborative filtering, in which the training data used consists only of positive examples, makes it extremely sparse [17] and confusing to interpret the missing values, if the user did not see the content or he simply did not find it interesting. To treat the prediction task in this OCCF dataset, a wide range of techniques are implemented, such as content based, memory based collaborative

filtering, model based collaborative filtering, and hybrid approaches combining different techniques aforementioned together.

1.4 Structure of This Thesis

The structure of this thesis is as follows:

In chapter 2, recommender systems are discussed in a more detailed manner, giving emphasis to the formal definition of the recommendation process. Additionally, current approaches and theories used with the existing recommendation systems are discussed.

Chapter 3 covers the related work about recommender systems; this includes the discussion of pure content based approach, pure collaborative filtering approach, and hybrid recommendation systems. And also, a wide range of recommendations systems applied to a diverse range of domains are presented along their advantages and disadvantages.

In chapter 4, the architecture and the system components of the proposed recommender system are presented, with a detailed description of the prediction algorithm of the system.

In chapter 5, the evaluation scheme used to test the performance of the proposed recommender system is explained. Additionally, the results of the experiments that were carried out during the evaluation process are discussed.

In Chapter 6 conclusions and some possible future work as the extension of this thesis is stated.

CHAPTER 2

RECOMMENDER SYSTEMS

2.1 Definition of a Recommender System (RS)

Recommender System (RS) is a technology which by the use of specific software tools and techniques produces recommendations of items considered interesting to a specific user [4, 6,18]. “Item” is the generic term denoting the object of the recommendation process.

Recommender systems are particularly helpful for anyone lacking the expertise to evaluate a wide range of options to select one that satisfies him [18]. As an example, consider the amazon.com e-commerce web site, the use of a recommender system has facilitated the task of buying a product, in some categories, the options reach thousands of alternatives [19]. Because each user has different tastes, each user profile in modeled according to the interests of the user, so each and every user get different recommendations.

Recommender systems started by observing that people often rely on recommendations given by others to make their own decisions [4, 20]. For example, friends ask friends which games they liked, which movies they liked, which restaurants they liked.

In a formal way, we can define the recommendation problem as follows [8]: Being *C* the set of all users and *S* the set of all items; let *u* be a utility function (rating most of the times) that measures usefulness of item *s* to user *c*, i.e., u :C × S → R, where R is a totally ordered set. We want to find an item s’ with the higher utility value, for all users. More formally:

$$\forall c \in C, S'c = \arg\max u(c, s), s \in S \quad (1)$$

In RSs, the utility function is the rating, which is given by a user to an item, representing how the user liked the item. Consider as an example, a user-item rating matrix for a movie recommendation presented in Table 1, the rating range is [1 – 5] and the symbol "-" is for movies the user did not rate yet. Therefore, RS should calculate a prediction for unrated movies and give recommendations using those predictions.

Table 1 – A sample of a rating matrix

	Avatar	Mission to mars	John Carter	Contact
João	4	-	5	2
Fabião	2	3	1	4
José	1	5	-	3
Maria	-	2	4	-

In Figure 1, is represented the top view of a recommendation process. Before the recommendation engine can produce recommendations, it needs the representation of user profiles along with item profiles. The recommendations can be represented in many ways, such as ranked lists, item thumbnails [25].

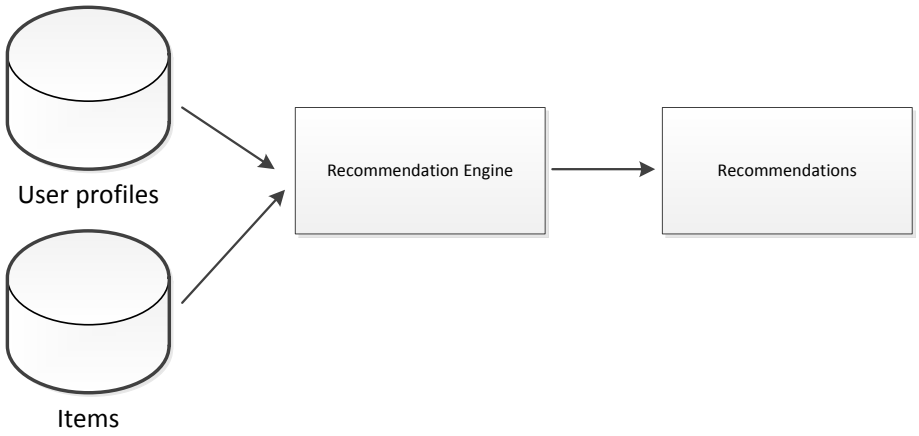


Figure 1 – Top view of a recommendation process

2.1.1 Terms and Concepts Common to Recommender Systems

In the Recommender Systems field, there are specific terms and concepts in use, before going any further is better to clarify the "vocabulary" [18, 26, 27].

- **Resources:** The targets of the recommendation process;
- **Recommenders:** Entities that produces customized recommendations to its users;
- **Descriptions:** Resources designed to express opinions or keep opinions about resources;
- **Preferences:** The tastes of a RS user;
- **Algorithms for Computing Recommendations:** A set of step-by-step procedures making use descriptions and preferences to evaluate resources;
- **Recommendations:** The outcome of the evaluation process for the user;

- **User's Interest:** Represents a degree of how much a user "likes" a specific item;
- **Prediction:** The predicted importance of an item to a user;
- **Rating:** A measure that represents a user's interest in a specific item;
- **Actual Rating:** A measure representing the real interest of the user in a specific item. Given explicitly;
- **Prediction Accuracy:** A measure that shows the degree the predicted rating conforms with the user's actual rating;
- **Prediction Technique:** The algorithm used by the system to calculate the predicted rating of an item;
- **Certainty**: The degree of belief that a recommender has in the accuracy of a prediction;
- **Feedback:** The user's response to the recommendations and predictions made by the recommender;

2.2 Personalization

Recommender systems and personalization walk together. Because of the vast amounts of information available to us, it is harder to locate the desired item. To minimize this situation, researchers came out with personalized information services [31], services designed to learn the profile of the user, his preferences, so that later recommendations can be made to this user based on the his profile. Personalization technology combines ideas from many fields, such as profiling, information retrieval, artificial intelligence and user interface design, to design information services suitable to us [31]. The authors of [28] consider Personalization as the

future of the internet; it has achieved great success in industrial applications such as Amazon and Netflix.

In [29], the authors state that the fundamental processes included in the personalization is to extract user data, generate a model and design mechanisms to update the models. A similar understanding is shared by the authors of [30], in which they state that the core objectives of personalization technology can be compiled as follows:

- The user profile must be taken into account in order to produce recommendations for the user in the system;
- The user profile should be generated with the minimum involvement from users;
- The recommendations should be generated almost in real time.

2.3 User Modeling

In RSs, the very first step is to learn the user profile; it can be manual or automatic [33]. In the case of manually generated, some researchers has shown that the profile is not very accurate [32]. In the automatic generation and update of the user profile, machine learning is used, is a field rich in tools and algorithms to manage profiles, but it requires a large set of training examples. These both approaches can be combined, and when it does, a more complete and accurate profile is generated [33].

Considering the previous studies about the topic, the most common approach to build a user profile mixes three different techniques [34].

In the first one, the user has to fill an initial form, but because users might not be able or willing neither to fill large forms nor to provide personal details and preferences explicitly. Explicit feedback asks the user about his interests, by showing him items of different categories, so that he can rate them. Users are generally not motivated to provide their feedback if they do not receive immediate

benefits, even when they would profit in the long-term [94]. The approach usually followed is to present the user with a limited number of fields and to let him decide which fields he is willing to fill. The second technique exploits demographic profiles, based on age groups for example, that give available information on the different categories of users, and can also be used to predict user's interests. The third technique dynamically updates the user profile by considering his previous interactions with the system.

In fact, these three techniques above discussed, complement each other, generating a more complete user profile. Moreover, when combined, it leads to a less intrusive system. Explicit feedback requires a user to evaluate content and indicate how relevant or interesting specific content is to him/her using like/dislike (a binary scale) or numerical ratings. Even though explicit feedback helps us to capture user preferences accurately, there is a serious drawback in that users do not tend to provide enough feedback. Users are generally not motivated to provide their feedback if they do not receive immediate benefits, even when they would profit in the long-term [94].

2.4 Too much Information

The creation and extension of online services had an impact on our lives. In one hand, it gave access to a variety of information, and on the other hand made it harder to find products suitable to our needs, the user is exposed to more information than he needs and, more importantly, is able to process.

The successful management of information depends on finding “smart” ways of reducing this information overload problem. One of the main solutions proposed and accepted for this problem are recommender systems, which provide automated and personalized suggestions of products to consumers.

2.5 Data and Knowledge Sources

Data is the first input to a recommender system. This data consists of information about items, users and user's transactions [36]. The items are the objects to be recommended. Items are characterized by their characteristics and by the rating given by the user. Users of a RS, as aforementioned, may have different profiles. In order to personalize the recommendations, RSs make use of these profiles. Transaction is a recorded interaction between a user and the RS and is used by the recommendation generation algorithm of the system. These transactions can be captured explicitly, given by the user, or implicitly by monitoring his behavior when using the system. Below we present the formal definition.

2.5.1 Formal Domain Definition

A detailed domain definition is useful to depict existing data in an organized way. The dataset can be mapped to an environment as follows:

- A set *P* of n uniquely identifiable peers.

  $P = \{ p_1, p_2, p_3, \ldots , p_n \}$

Each peer corresponds to a single user in the dataset. Additionally, the term "peer" stands for independent entities performing some actions in a system, such as nodes in peer-to-peer systems, software agents or intelligent web servers in other domains.

- A set *I* of m uniquely identifiable items.

  $I = \{ i_1, i_2, i_3, \ldots, i_n\}$

  Each item is identified separately and in our case, each item stands for a game.

- A set R of k uniquely identifiable purchases.

$$R = \{r_1, r_2, r_3, \ldots, r_n\}$$

Each user is allowed to purchase any item.

2.6 Recommendation Techniques

In order to produce a list of interesting items to a user, RSs should first predict that a specific item is worth recommending for a specific user. This prediction can be obtained by following multiples strategies, which classifies a RS. To provide a classification of the different types of RSs, we will make use of the taxonomy found in [6] and also [37], which distinguishes between six different classes of recommendation approaches: Content-based, Collaborative filtering, Demographic, Knowledge-based, Community-based and Hybrid. Each of this classes will be briefly summarized. To have more details about this taxonomy please consult [6] and [37].

2.6.1 Content-based RS

Recommender systems using this recommendation strategy, analyze a set of documents (features) of the items rated by the user, and then create the user profile based on the features of the objects rated by that user [6, 38]. The recommendation mechanism basically tries to match up the features of the item against the user profile. The best matches are included in the recommendation list.

The high level architecture of a CBR is depicted in Figure 2. The recommendation process is performed in three steps, each of which is handled by a separate component: content analyzer (pre-processes the data to be used in subsequent steps), profile learner (constructs the user profile) and the filtering component (produces the recommendations)

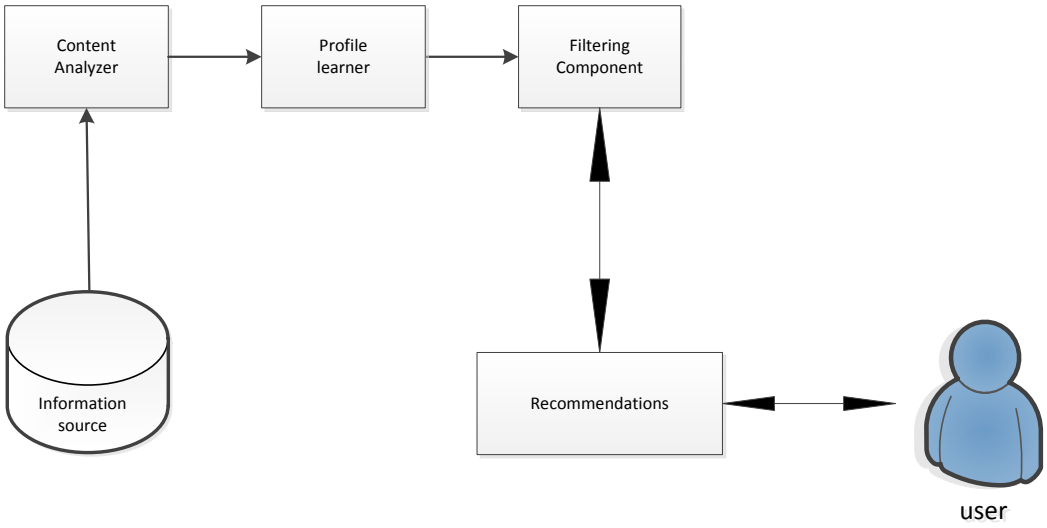


Figure 2 - A Top Level Architecture of Content-based Systems

2.6.1.1 Item profile

The item profile is the set of features that identifies an item. It can be anything. As an example consider an item in an electronics shop, it will have brand, model, color, country of production, year of production, description, price, serial killer, and so on. Thus all of these attributes define the item profile.

2.6.1.2 User profile

One important aspect of RSs is its capability to learn user's preferences. In order to provide proper recommendations to its users, personalized recommender systems require user profiles [91]. On top of that, as every user can have different preferences, each user profile should be represented with a individual set of features [92].

2.6.1.3 Pre-processing

High precision in information retrieval (IR) is commonly hard to accomplish for a variety of reasons; one for sure is the large number of variants for any given term. To deal with some of the issues, researchers proposed to use stemming algorithms to reduce terms to its root [88].

Another pre-processing step commonly applied with stemming, is removing stopwords, which consists on removing the words which do not contain important significance and are extremely common, such as bu, sen, burada, etc.

2.6.1.4 TFIDF (Term Frequency / Inverse Document Frequency) Weighting

When working with free text in IR, it is common to use a weighting scheme when searching documents. TFIDF is a popular weight scheme. TFIDF is a combination of term frequency and inverse document frequency and is calculated using the Equation (2) below [89]:

$$\text{tf-idf}_{t,d} = \text{tf}_{t,d} \times \text{idf}_t \qquad (2)$$

- Term frequency $\text{tf}_{t,d}$ - number of occurrences of term *t* in document *d*.
- Inverse document frequency idf_t – letting the total number of documents in a collection be *N*, the inverse document frequency (idf) of a term *t* is defined as follows (Equation 3):

$$idf_t = log \frac{N}{df_t} \qquad (3)$$

Where df_t is defined as the number of documents in the collection that contains the term t.

After applying the weight scheme, each document is viewed as a vector with one component corresponding to each term in the dictionary, together with a weight for each component that is given by Equation (2). For terms that do not occur in the document, this weight is zero.

After having a vector containing all the terms and their respective TF*IDF weights for each document in the collection, we can measure the similarity between documents by making use of any similarity measure.

Consider a document containing 100 words wherein the word "*canavar"* appears 3 times, and the most frequent word appears 10 times. Following the previously defined formulas, the normalized term frequency for "*canavar"* is then (3 / 10) = 0.3. Now, assume we have 10 million documents and "*canavar"* appears in one thousands of these. Then, the inverse document frequency is calculated as log(10 000 000 / 1 000) = 4. The tf*idf score is the product of these quantities: 0.3 × 4 = 1.2.

2.6.1.5 Pros and Cons

Content-based filtering is a good strategy in some situations; in this section we will learn some of its strong and weak attributes. Let us start with the advantages, comparing to collaborative filtering [36]:

- **User independence** – CBRs only exploit previous interactions of the active user to build his user profile, it does not depend on the opinions of other peers, as CF;

- **Transparency** – CBRs are transparence in the sense that the explanations on the recommendation list can be provided by explicitly showing the content features that caused the recommendation;

- **New item** – CBRs can recommend not yet rated items; as a consequence, they do not suffer from the first-rater problem, as CF.

Nonetheless, content-based systems have several shortcomings:

- **Limited content analysis** - No CBR can provide suitable recommendations if the analyzed content does not contain enough information to distinguish items the user likes from items the user does not like. Some representations capture only certain aspects of the content, but there are many others that can cause some impact;

- **Over-specialization** - CBRs have no inherent method for finding something unexpected. It only recommends items with higher degree of similarity to items previously rated. This is called serendipity problem.

- **New user** - When few ratings are available in the system, as for a new user, the CBR will not be able to provide reliable recommendations.

2.6.2 Collaborative Filtering RS

The authors of [42] give the following definition to collaborative filtering:

"Collaborative filtering (CF) is the process of filtering or evaluating items through the opinions of other people"

Differently from CBR, collaborative filtering approaches [43, 44] are based on the ratings of the active user as well as those of other users in the system. Collaborative filtering methods can be classified in two general classes: Memory based and model-based [8, 45, 46].

The general idea of model-based approach is to model the user-item matrix with factors representing hidden characteristics of the users and items in the system, this model is then trained using the available data, and later used to predict ratings of items to users. Model-based approaches are numerous and include Bayesian Clustering [46], Latent Semantic Analysis [49], Latent Dirichlet Allocation [50], Maximum Entropy [51], Boltzmann Machines [52], Support Vector Machines [53], and Singular Value Decomposition [54].

In memory based approach, the user-item ratings stored in the system are directly used to make predictions. Many memory based implementations uses nearest neighbor algorithm, which is the *de facto* algorithm for memory based approach. K-Nearest Neighbor (KNN from now on) is one of those algorithms that are very simple to understand but works incredibly well in practice. KNN can be implemented in two ways: user based or item-based recommendation. In the following sections we will summarize these two methods.

2.6.2.1 User-Based Nearest Neighbor Algorithms

User-based algorithms generate predictions for users based on ratings from similar users. We call these similar users neighbors. If a user *n* is similar to a user *u*, we say that *n* is a neighbor of *u*. User-based algorithms generate a prediction for an item *i* by analyzing ratings for *i* from users in *u*'s neighborhood. The prediction can be calculated using Equation 2. Where r_{ni} is neighbor *n*'s rating for item *i*.

$$pred\ u,i\ = \frac{n \in neighbors(u)\ r_{ni}}{number\ of\ neighbors} \quad (4)$$

2.6.2.2 Item-Based Nearest Neighbor Algorithms

While user-based algorithms base the predictions on similar users, item-based algorithms give predictions using similarities between items [63].

A prediction for a user *u* and item *i* is composed of a weighted sum of the user *u*'s ratings for items most similar to *i*, and can be calculated using equation 3.

$$pred\ u,i\ = \frac{\sum_{j \in ratedItems(u)} itemSim\ i,j\ * r_{ui}}{\sum_{j \in ratedItems(u)} itemSim\ i,j} \quad (5)$$

Note that in Equation 5, itemSim() is a measure of item similarity, not user similarity.

2.6.2.3 Advantages of Neighborhood Approaches

The main advantages of memory-based methods are:

- **Simplicity:** memory-based methods are intuitive and quite simple to implement.;

- **Justifiability:** these methods provide a reliable justification for the computed predictions. In item-based recommendation, for example, the list of items rated by the neighbors, along with the ratings given by the user to these items, may be showed to the user as a justification for the recommendation. This can help the user to better understand the recommendation [54];

- **Efficiency:** Efficiency is perhaps its strongest points. Although the calculation of one's neighbors is a very costly intensive task, it can be done offline, so that the recommendations can happen almost in real-time;

2.6.2.4 One-class collaborative filtering (OCCF)

In earlier sections, we have discussed the use of CF in RSs. We divided collaborative filtering algorithms into two sets based on the approach they use:

memory based and model based algorithms. However, any distinction was made taking into consideration the dataset.

Taking into consideration the dataset, we can classify as multi-class collaborative filtering and one-class collaborative filtering [16]. In a multi-class dataset, we have positive examples as well as negative examples, but in contrast, one-class dataset gives us only positive examples, the negative and unknown examples are mixed together and we cannot tell which is what.

The authors of [16] work on an OCCF dataset, containing only positive examples, classes are highly imbalanced, and the vast majority of data points are missing. In this study, three different collaborative filtering frameworks are studied: Low-rank matrix approximation, probabilistic latent semantic analysis, and maximum-margin matrix factorization. The authors proposed two novel algorithms for large-scale OCCF that allow to give weight to the unknowns. The experimental results demonstrated their effectiveness and efficiency on different problems, including the Netflix Prize data [16].

Now, let us consider an OCCF example, an e-commerce shop that lets its customers purchase their content on it. According to bought content, the service makes recommendations to the user. In this case, the dataset available to the recommendation system is all the purchases history of all customers. A possible portion of the actual dataset is given in Table 4. If a user buys content, we can conclude that the user liked that content, and we put a "1". However, if the user did not buy content, we cannot make any conclusions, either the user did not find it interesting or the user has not viewed the content yet. The user even may not be aware of the existence of such content. In other words, there is no way for the recommendation system to make distinction between the negative and missing positive entries in the dataset. After all, as can be seen in Table 2, the user-item matrix in such a system will consist of only positive (1's) and missing (dashes) entries. A "1" in the dataset means that the user has bought the given content. On the other hand, a dash (-) indicates that the user has not bought the content, which

means that either the user did not like the content (negative example) or the user was not aware of that content (actual missing data).

Table 2 - A sample dataset for one-class collaborative filtering.

	Diablo 3	Angry birds	Walking dead	Fruits ninja	Drive2survive
User 1	1	-	-	-	1
User 2	-	1	-	1	-
User 3	-	1	1	-	1
User 4	1	1	-	1	-
User 5	-	-	1	1	1

It is obvious that such a service will face a one-class classification problem during the recommendation step.

The key difference between traditional collaborative filtering and one-class collaborative filtering methods is that later one has only positive examples in training set [17]. However, traditional collaborative filtering methods can be used to attack one-class collaborative filtering problems. By interpreting missing data as negative examples one can obtain a dataset in which instances belongs to two classes. Of course this approach will be biased as it will mark some positive examples as negative. In [17], the authors discuss several strategies to distinguish negative examples out of missing data. In their work, the authors experimentally compared several approaches including all missing data as negative, no missing data as negative and some weighting schemes to tag an instance as negative. However in this study, for the sake of simplicity, we will consider a “dash” as negative example, so our rating matrix will have values 1 and 0.

2.6.3 Demographic RS

Demographic RS explores the demographic profile of the user. The basic idea is to generate recommendations for different demographic niches. This approach is

followed by many web sites, by using the user's language or country to redirect him to a particular web site. There is little research about this RSs [65].

2.6.4 Knowledge-based RS

In knowledge-based approaches, the RS makes use of extra information, which is frequently provided, information about both the active user and the items inventory. As an example, consider a system in automobile shop, where users don't buy cars often. This means that the purchase history is not is not enough to make a good recommendation, a more detailed and structured content may be available, including technical and quality features. Using only the purchase history would result in recommending only top-selling cars.

2.6.5 Community-based RS

This type of RSs recommends items based on the preferences of the user's friends. The research in this area still in its early phase, the authors of [37, 69] report that, social-network based recommendations are not more accurate than those produced from traditional CF approaches. Others have shown that in some cases social-network data yields better recommendations [71] and that adding social network data to traditional CF improves recommendation results [70].

2.6.6 Hybrid RS

These systems are based on the combination of the above mentioned techniques. A hybrid system combining techniques A and B tries to use the advantages of A to fix the disadvantages of B. For instance, CF methods suffer from new-item problems, i.e., they cannot recommend items that have no ratings. This does not limit content-based approaches since the prediction for new items is based on their description (features) that are typically easily available. Given two (or more) basic RSs techniques, several ways have been proposed for combining them

to create a new hybrid system, the interested reader can consult [6] for more detailed descriptions.

In following sections we will examine the hybridization designs.

2.6.6.1 Monolithic hybridization design

In monolithic hybrids we find a single recommender component that implements multiple techniques by preprocessing and combining several information sources. Hybridization is accomplished by a slight change of the algorithm behavior to explore different types of input data. Following Burke's taxonomy [72], both feature combination and feature augmentation methods fall in this category.

2.6.6.1.1 Feature combination hybrids

A feature combination hybrid is a monolithic recommendation component that uses features from different data sources as input data. For instance, Basu et al. [74] uses this technique to combine collaborative features, user's likes and dislikes, along with content features of items. The curious reader can consult reference [74] for more details.

2.6.6.1.2 Feature augmentation hybrids

Feature augmentation is another monolithic hybridization design that can be used to integrate multiple recommendation algorithms. Output from one recommendation technique is given to another as input. Content-boosted collaborative filtering is an actual example of this variant [75]. It predicts a user's assumed rating based on a collaborative mechanism that includes content-based predictions.

2.6.6.2 Parallelized hybridization design

Parallelized hybridization designs use many recommenders at the same level. Burke [72] classifies this category as mixed, weighted, and switching strategies.

2.6.6.2.1 Mixed hybrids

Using this strategy, the predictions from each of the recommendation techniques are presented together to the active user. Therefore the recommendation result for user *u* and item *i* is the resulting recommendations of each and every recommender. The items with the highest score for each recommender are presented to the active user, as in Burke et al. [76].

2.6.6.2.2 Weighted hybrids

In a weighted hybridization strategy, each of the participating recommenders contributes to the overall score by some pre-defined weight. The implementation is quite straightforward and it is a popular strategy when hybridizing.

2.6.6.2.3 Switching hybrids

Switching hybrids require a mechanism to decide in which situation favor one recommender and not the other, depending on the user profile and/or the results of the recommendation. In the NewsDude system [77], the authors implemented two content-based variants and a collaborative method to recommend news articles. Initially, a CB recommender is used. If it does not find similar articles, a CF system is called; and lastly, a naive Bayes classifier locates articles matching the long-term preferences of the active user.

2.6.6.2.4 Pipelined hybridization design

Pipelined hybrids make use of several recommendation techniques, in which the output of one serve as input for the next one; a preceding component may either preprocess input data to create a model that is used by the subsequent steps or produces a list for further processing by the next technique.

2.7 Similarity measures

CF is one of the most popular methods in recommender systems. The critical component of CF technique is the creation of the neighborhood. In memory based approach, KNN is champion, as already stated in previous sections. The success of this kind of classifiers depends on the application of a suitable similarity measure to compute the neighborhood.

In literature there is a wide range of proposed methods for computing the similarity between entities. Minkowski Distance, Pearson Correlation and Cosine Similarity are the most popular ones.

2.7.1 Euclidean distance and Minkowski distance

The most famous example of a distance measure is the Euclidean distance, defined by the Equation 6:

$$d(x, y) = \sqrt{\sum_{k=1}^{n} (x_k - y_k)^2} \qquad (6)$$

where n is the number of dimensions (attributes) and x_k and y_k are the k_{th} attributes (components) of data objects x and y, respectively.

The Minkowski Distance is a generalization of Euclidean Distance (Equation 7):

$$d(x,y) = \sum_{k=1}^{n} |x_k - y_k|^r \quad \frac{1}{r} \tag{7}$$

where *r* is the degree of the distance. Depending on the value of *r*, the generic Minkowski distance is known with specific names:

- For r = 1, the city block, (Manhattan, taxicab or L1 norm) distance;
- For r = 2, the Euclidean distance;
- For r→∞, the supremum (Lmax norm or L∞ norm) distance, which corresponds to computing the maximum difference between any dimension of the data objects.

2.7.2 Cosine Similarity

The similarity can also be defined by the angle or cosine of the angle between two vectors. It is widely used in text classification because two documents with equal word composition but different lengths can be considered identical. The cosine measure is given in Equation 8.

$$similarity = \cos\theta = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{8}$$

where • indicates vector dot product and ||x|| is the norm of vector x. This similarity is known as the cosine similarity or the L2 Norm .

2.7.3 Pearson Correlation

The similarity between objects (items or users for example) can also be given by their correlation, which measures the strength of the association between objects. Given the covariance of data points Σ(x,y) , and their standard deviation σ , we compute the Pearson correlation using Equation 9:

$$Pearson\ x,y\ = \frac{(x,y)}{\sigma_x \times \sigma_y} \qquad (9)$$

RS have traditionally used either the cosine similarity (Equation 6) or the Pearson correlation (Equation 7) – or one of their many variations through, for instance.

2.7.4 Jaccard Coefficient

The Jaccard coefficient measures the ratio of the number of commonly active features of x_1 or x_2 [100]. Equation 10 gives a definition of this measure which is often used in retail market applications.

$$J(x_1,x_2) = a / a + b + c \qquad (10)$$

Where:

- ***a*** represents the total number of attributes where x_1 and x_2 both have a value of 1.
- ***b*** represents the total number of attributes where the attribute of x_1 is 0 and the attribute of x_2 is 1.
- ***c*** represents the total number of attributes where the attribute of x_1 is 1 and the attribute of x_2 is 0.

$J(x_1,x_2)$ results in a number in the interval [0,1] inclusive, measuring the degree of similarity between x_1 and x_2. $J(x_1,x_2) = 1$ corresponds to objects x_1,x_2 that are identical while $J(x_1,x_2) = 0$ corresponds to objects that are very different.

2.8 Singular Value Decomposition - SVD

In this section, Singular Value Decomposition is presented. SVD is a very efficient factorization technique used in linear algebra. In literature there are diverse applications areas of SVD, but we are just interest in its contribution to RSs. Interested reader can consult [102,103] for more clarification and examples.

2.8.1 Definition

SVD comes from linear algebra, from a theorem that says that a rectangular matrix A_{mxn} can be represented by the product of three matrices - an orthogonal matrix U_{mxr}, a diagonal matrix S_{rxr} having all singular values of matrix A as its diagonal entries, and the transpose of an orthogonal matrix V_{nxr} [102] . This theorem is commonly displayed as:

$$A_{mn} = U_{mm} S_{mn} V_{nn}^{T} \qquad (11)$$

where $U^TU = I$; $V^TV = I$; the columns of U are orthonormal eigenvectors of AA^T , the columns of V are orthonormal eigenvectors of A^TA, and S is a diagonal matrix containing the square roots of eigenvalues from U or V in descending order; r is rank of the matrix A.

2.8.2 SVD and RS

What makes SVD interesting to recommender systems, is that the matrices generated by performing SVD provides with the best lower rank approximations of the original matrix A. It is possible to reduce the r x r matrix S to have only k largest diagonal values to obtain a matrix S_k, k < r. If the matrices U and V are reduced

accordingly, then the reconstructed matrix $A_k = U_k.S_k.V_k^T$ is the closest rank-k matrix to A [103]. Figure 3 show this process.

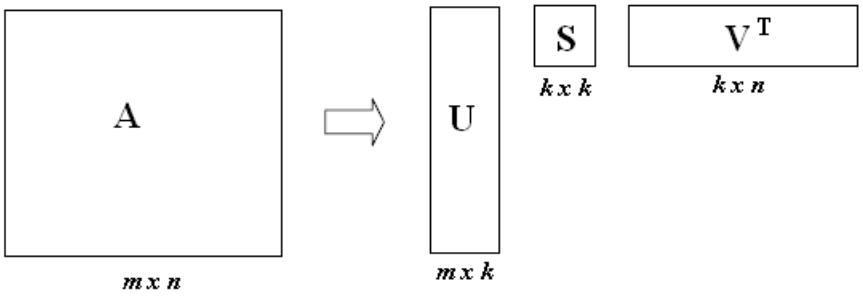


Figure 3 - Dimensionality Reduction Process using SVD

SVD is used in RS to first reveal the latent relationships between users and items to predict the usefulness of an item to a user, and then to produce a low-dimensional representation of the original user-item matrix for calculating the neighborhood in this reduced space, to be used later for produce recommendations. The U_k matrix contains information about users and matrix V_k^T contains information about items. Using this two matrices users and items similarities can be computed.

SVD-based recommendation algorithms produce high quality recommendations, but have to undergo computationally very expensive matrix factorization steps.

CHAPTER 3

RELATED WORK

In this chapter, the related work in the area of recommender systems will be given. To this goal, a wide range of recommender systems from different domains making use of pure content based, pure collaborative filtering, social graphs and mixed techniques will be presented. In addition to carefully analyze how the recommendation problem is being handled, the pros and cons of these techniques will be discussed.

3.1 Pure collaborative filtering recommender systems

One of the personalization technologies driving the adaptive web is collaborative filtering. As mentioned earlier in section 2.6.2, collaborative filtering (CF) is the process of filtering or evaluating items through the opinions of other people. Collaborative filtering techniques have been successful in predicting the rating for an unseen item to the user in recommendation systems.

The study conducted by the authors of [15] claims that although COR (Pearson correlation) and COS (cosine similarity) have won big space among the trusted similarity measures, they are not very successful when the number of ratings is very small. The similarity calculations in scenarios like this are not reliable.

In order to address this problem, the authors presented an alternative measure based on the following:

- The measure should explore other perspectives of the data, to make it more effective in very sparse datasets;

- It should allow the implementation in existing CF systems by simply replacing the similarity measure of the systems;

- It should show good results not only in very sparse datasets but also in dense datasets;

The measure is composed of three factors of similarity, Proximity, Impact, and Popularity, and hence, was named PIP. Using the PIP measure, the similarity between two users u_i and u_j is calculated using Equation 12:

$$SIM\ u_i, u_j = \sum_{k \in C_{i,j}} PIP(r_{ik}, r_{jk}) \qquad (12)$$

where r_{ik} and r_{jk} are the ratings of item k by user i and j, respectively, $C_{i,j}$ is the set of co-rated items by user u_i and u_j, and PIP(r_{ik}, r_{jk}) is the PIP score for the two ratings r_{ik} and r_{jk}, calculated using Equation 13,

$$PIP(r_1, r_2) = Proximity(r_1, r_2) \times Impact(r_1, r_2) \times Popularity(r_1, r_2) \qquad (13)$$

The Figure 4 below illustrates the basic ideas behind the three factors, and the Table 3 presents the formal description of the formulas:

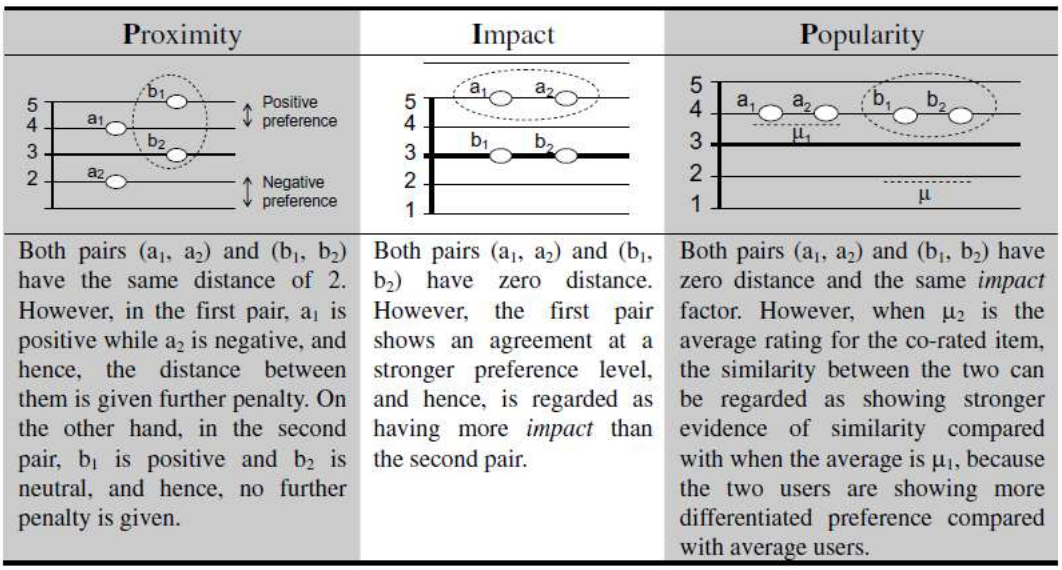


Figure 4 - Description of the three factors of PIP using example ratings

Table 3 presents the formal description of the formulas.

Table 3 - Formal description of the PIP formulas

Agreement	For any two ratings r_1 and r_2, let R_{max} be the maximum rating and R_{min} the minimum in the rating scale, and let $R_{med} = (R_{max} + R_{min}) / 2$ A Boolean function Agreement(r_1, r_2) is defined as follows: Agreement(r_1, r_2) = false if ($r_1 > R_{med}$ and $r_2 < R_{med}$) or ($r_1 < R_{med}$ and $r_2 > R_{med}$), and Agreement(r_1, r_2) = true otherwise
Proximity	A simple absolute distance between the two ratings is defined as: $D(r_1, r_2) = \|r_1 - r_2\|$ if Agreement(r_1, r_2) is true, and $D(r_1, r_2) = 2 \times \|r_1 - r_2\|$ if Agreement(r_1, r_2) is false Then the Proximity(r1, r2) is defined as: $Proximity(r_1,r_2) = \{\{2x(R_{max} - R_{min}) + 1\} - D(r_1,r_2)\}^2$
Impact	Impact Impact(r_1, r_2) is defined as:

	Impact(r_1, r_2) = ($\|r_1 - R_{med}\| + 1$)($\|r_2 - R_{med}\| + 1$) if Agreement(r_1, r_2) is true, and Impact(r_1, r_2) = 1/($\|r_1 - R_{med}\| + 1$)($\|r_2 - R_{med}\| + 1$) if Agreement(r_1, r_2) is false
Popularity	Let μ_k be the average rating of item k by all users, Then Popularity(r_1, r_2) is defined as: Popularity(r_1, r_2) = 1 +(((r1+ r2)/2)- μ_k)^2 if ($r_1 > \mu_k$ and $r_2 > \mu_k$) or ($r_1 < \mu_k$ and r2 < μ_k), and Popularity(r_1, r_2) = 1 otherwise

The authors of [15] after extensive evaluation concluded that the presented PIP measure showed superior performance for very sparse datasets.

In the study presented by the authors of [10], they developed a web-based movie RS called Movies2Go that reasons with user preferences to recommend movies. It combines voting based ranking procedure along with properties that use syntactic features like actor/actress of movies together with a learning based approach that processes semantic features of movies like its synopsis.

Their RS give three major functionalities to its users:

- Stores the user profile with weights of different attributes in many dimensions like favorite actors, favorite actresses, favorite directors, preferred time period (in terms of years), relative likings of different movie genres like comedy, drama, action etc. It also keeps the importance of each dimension as specified by the users (e.g., whether the user rates the genre dimension above the director dimension etc.);

- The system makes recommendations to the user by a combination of voting and nearest neighbor algorithms based on the user profile. The user can filter

the results (e.g, a user may be looking only for ‘comedy’ movies directed by a certain director);

- It employs a simple learning mechanism to learn keyword occurrences in the synopsis of movies that a user rates. This makes the system more suitable to individual needs, which provides better recommendation.

The authors claim that MOVIES2GO provides satisfactory recommendations by using this voting system.

Another interesting work is one done by the authors of [45]; they have enhanced the neighborhood-based approach leading to noticeable improvement of prediction accuracy, without increasing running time. First, they pre-processed the data by removing the so-called “global effects” to make the ratings more comparable. Second, they showed how to simultaneously derive weights for all neighbors, unlike previous approaches where each weight is computed separately.

Their method does not require training many parameters or a long pre-processing, making it very practical for large applications. They evaluated these methods on the Netflix dataset, where it delivered significantly better results than the commercial Netflix Cinematch recommender system [45].

One of the best examples of CF is without a doubt Amazon.com [12], the authors used recommendation algorithms to personalize the online store for each customer. The store radically changes based on customer profile. Clicking on the “Your Recommendations” link leads customers to an area where they can filter their recommendations by product line and subject area, rate the recommended products, rate their previous purchases, and see the justifications.

Because existing recommendation algorithms cannot scale well Amazon.com’s tens of millions of customers and products, they have developed their own, an item based algorithm, rather than matching the user to similar

customers, item based CF matches each of the user's purchased and rated items to similar items, then combines those similar items into a recommendation list. To determine the most-similar match for a given item, the algorithm builds a similar-items table by finding items that customers tend to purchase together [12].

As previously discussed, the critical aspect of CF is the calculation of similarities; in this system, it creates the expensive similar-items table offline. Unlike traditional collaborative filtering, the algorithm also performs well with limited user data, producing high-quality recommendations based on as few as two or three items [12].

3.2 Pure content based recommender systems

Content based approach, as previously discussed, requires only two pieces of information: a description of the item characteristics and a user profile that describes the (past) interests of a user. The recommendation task then consists of determining the items that match the user's preferences best, as discussed in section 2.6.1.

As an example, consider the RS presented by the authors of [84], PTV. PTV is a recommender system designed to make TV program suggestions to users based on their individual profile. PTV profiles created in this system contain lists of positively and negatively rated TV programs. In this RS, users register themselves in the PTV interface (Web site) and then they can access personalized programming guides presented as HTML or WML pages. The system incorporates user profiles, content-based reasoning and collaborative methods to make recommendations. When registering a new user, the system creates a profile which stores preferences about programs, channels, genres, timetables, etc. Later on, a component of the system collects the feedback information of the users to enrich the user profile. Additionally, this information can be continuously improved by taking into account all decisions adopted by the user any time he interacts with the system. This allows a great reduction in the information that must be explicitly given by the user. So, in

this case, the first time the user turns on the system, he must inform only about a few characteristics to build a preliminary profile [84].

The study presented by the authors of [82], a CB RS. In the system, the attributes used for content-based recommendation are weighted, according to their level of importance for the users of the system. In order to define the weight of each attribute, is used a set of linear regression equations obtained from a social network graph. The main idea of this approach is to use existing recommendations by users to construct a social network graph with items as nodes [82]. After testing this approach extensively, the authors conclude that the presented approach consent with IMDB recommendation, and it even present good results and also shows the effectiveness of feature weighting [82].

3.3 Hybrid recommender systems

As discussed in section 2.6.6, this kind of RSs are based on the combination of the above mentioned techniques, exploring the advantages of one to fix the disadvantages of the other.

In movies/TV domain, there is the AVATAR system [83]; AVATAR is a TV RS that recommends contents for a user, semantically similar to those he watched before. AVATAR includes user profiles, content-based reasoning and collaborative methods, to produce the recommendations. It also makes use of a technique which provides an initial user profile from the information acquired during the registration process and after that, the feedback information (watched programs, changes in preferences, etc) needed for the explicit interaction and collaboration of the user through specific Web pages. The system implements an ontology using the OWL (Web Ontology Language) language, where resources and relations typical in the TV domain are identified by classes, properties and specific instances. In order to model the preferences, a dynamic subset of TV ontology is used, built incrementally by adding new classes, properties and instances. Specifically, when AVATAR knows a new content, the system adds to the profile this instance, the class referred

to it, the hierarchy of super classes defined in knowledge base related to this class and some properties defining the main characteristics of this TV content [83]. After evaluated this approach, the authors were happy with the results.

Another study in this category is the Machbook system [85], which is a web based recommendation system that has been developed by using personalization techniques and combining machine learning and automated reasoning. In Matchbook, the users are able to browse the content in order to find the right matches for themselves in terms of both character and appearance.

For the machine learning part, by observing just the implicit behaviour of the user (i.e. visiting other user's profiles) the interests of the user are learnt. "Add to Favourite List" means a positive example and "Skip to the Other User" means a negative example. According to these positive and negative examples, some hypotheses about the user's taste are maintained by using Candidate Elimination Algorithm. In this manner, the users in the system are personalized so that the system is able to provide them with the appropriate matches [85]. As another machine learning algorithm, k-Nearest Neighbour Algorithm is used in order to find the people sharing the most common interests specified during the registration.

To make recommendation to the user, the system finds the relation defined for that user and tries to match the related properties with the users in the system. After the matches are found by the reasoner, Naïve Bayes Classifier is applied and the matches are ordered according to the results calculated from the classification method [85].

3.4 Dimensionality reduction approach

In [103], the authors explore the potential of SVD for making recommendations. Their study was driven to find alternatives to the weakness of Pearson nearest neighbor in large, sparse datasets. They started by applying SVD to produce matrices U, S and V, then they reduced the matrix S to dimension *k*, by choosing top

k Eigenvalues. After that they computed the square root of the reduced matrix S_k, giving matrix $S_k^{1/2}$. And at the end they computed the two resultant matrices $U_k S_k^{1/2}$ and $S_k^{1/2} V_k^T$.

With the resultant matrix $U_k S_k^{1/2}$, they computed the neighbourhood by applying cosine similarity to each and every row vector. After computing the neighborhood for a given customer C, they scanned through the purchase record of each of the *k* neighbours and performed a frequency count on the products they purchased. The product list is then sorted and most frequently purchased N items are returned as recommendations for the target customer.

After extensive evaluation, they concluded that the SVD-based technique was consistently producing worse results than traditional collaborative filtering when using an extremely sparse e-commerce dataset. However, the SVD-based approach produced better results than a traditional collaborative filtering algorithm when using a denser dataset as MovieLens dataset.

3.5 One Class Collaborative Filtering (OCCF) Dataset

As referred previously, OCCF datasets contains only positive examples. All of the negative examples and missing positive examples are mixed together and there is no way of distinguishing between them. There are many strategies that tries to solve this problem.

One of the strategies is to label negative examples, but this approach is very expensive or even close to impossible to implement, as in a e-commerce system such as amazon.com with millions of items, the user is only interested in a tiny small fraction of it, so to ask users to label negative examples among all these millions of choices is a non-sense. Another strategy used is to label all missing values as negative examples; empirical studies show that this approach works well most of the time, but it biases the results, because some unknown positive examples may be considered as negative examples. On the other hand, if we ignore all missing values

and use only the positive examples to feed the system, it will model only not missing data, all predictions processed by the system will be positive examples. Therefore, all missing as negative (AMN) and all missing as unknown (AMU) are two extremes.

Taking as an example the work done by the authors of [17], they proposed two frameworks to tackle OCCF. They balance between AMN and AMU. The first one method is based on weighted low rank approximation [104, 105], giving different weights to the error terms of positive examples and negative examples; and the other consists in randomly sampling some missing values as negative examples based on some sampling strategies.

For the first framework, the authors of [17] propose Weighted Alternating Least Squares (wALS). wALS is based on the Weighted Low-Rank Approximation (wLRA) method presented in [105]. Both wALS and wLRA methods use a matrix W which keeps weights in its cells. Weight of a rating represents our confidence level about that rating, which means that if a rating has a high weight, then we are highly confident that this rating is correct. Although, accommodating a weight matrix is the common idea of wALS and wLRA, the weight matrices used is the main distinguishing point between two algorithms.

Similarly to SVD, wALS also uses a matrix decomposition. The goal is to find a matrix $R \in R_{+}^{mxn}$ such that $R = UV^{T}$, where $U \in R^{mxr}$ and $V \in R^{nxr}$. R is expected to minimize the error in Frobenius form as given by Equation (14).

$$\in = \quad \mathrm{R} - \mathrm{UV}^{\mathrm{T}} \ _{\mathrm{F}}^{2} \qquad (14)$$

wALS is an improved version of wLRA. Both methods try to solve the optimization problem $argmin\, L(R)$, where $L(R)$ is defined using Equation (15).

$$L\ R \ = \ \underset{ij}{} \ W_{ij} \ R_{ij} - R_{ij} \ ^{2} \qquad (15)$$

$$L\ U,V\ = \sum_{ij} W_{ij}\ R_{ij} - U_i V_j^T\ ^2$$

To overcome the overfitting problem of wLRA, the authors of [17] proposed a regularization term to Equation (15) which is shown in Equation (16).

$$L\ U,V\ = \sum_{ij} W_{ij}\ R_{ij} - U_i V_j^T\ ^2 + \lambda\ U_i\ _F^2 + V_j\ _F^2 \quad (16)$$

The interested user can consult [17] for more details.

For the weight scheme, the authors of [17] used Uniform, User Oriented, and Item Oriented. The weight matrix W is very important to the performance of OCCF. With W=1, it is equivalent to the case AMN with the bias mentioned earlier. The main idea behind the correction of the bias is to let W_{ij} represent the confidence level of the training data (R) used to build the collaborative filtering model. For positive examples, $W_{ij} = 1$, whenever $R_{ij} = 1$. For the case of missing data, most of the cases are probably negative examples, and the confidence level of missing values being negative is not as high as the positive example case, so they gave lower weights.

The uniform weighting scheme considers missing values as being negative examples with the equal strength to all users or items, i.e., it uniformly give a weight $\delta\epsilon\ 0,1$ for all negative examples. The second weighting scheme, User Oriented, takes into consideration the number of positive examples, if it has more positive examples; it infers that she does not like the other items, meaning that the missing values are negative with higher probability. The third weighting scheme, Item Oriented, also takes into consideration the number of positive examples; if an item has fewer positive examples, the missing values for that item is negative with higher probability as well. All the three weighting schemes are summarized in the Table 4 below. The parameter for the three schemes is the ration between the sum of the positive examples weights and the sum of the negative example weights.

Table 4 - Weighting schemes

	Positive examples	Negative examples
Uniform	$W_{ij} = 1$	$W_{ij} = \delta$
User oriented	$W_{ij} = 1$	$W_{ij} = \underset{j}{} R_{ij}$
Item oriented	$W_{ij} = 1$	$W_{ij} = m - \underset{i}{} R_{ij}$

For the sake of simplicity, we will not discuss any further the second framework, negative example sampling.

The authors (Rong Pan, et al. 2008) conducted their experiments using two separate dataset: yahoo news consisting of user_id – news_url pairs which consists of 3158 unique users and 1536 news stories; the second dataset is from a social bookmarking site http://del.icio.us, it contains 246,436 posts with 3000 users and 2000 tags. These datasets were split in 80/20 ratio, in a cross-validation setup.

To evaluate the performance of the system, the authors make use of MAP (Mean Average Precision) and half-life utility metrics. MAP is mostly applied in IR. The authors of [17] used it to get the overall performance based on precision values at various recall levels on a test set. It is used to calculate the mean of average precisions (AP) of all users in test set, it is calculated using the Equation 17 below:

$$AP_u = \frac{\sum_{i=1}^{N} prec(i) \times pref(i)}{\#of\ prefered\ items} \qquad (17)$$

Where:

i is the position in the ranked list

N is the number of retrieved items

Prec(i) is the precision values

Pref(i) is a binary flag giving 1 if the i-th item is selected or 0 otherwise

Half-life utility is used to estimate the likeliness of a user choosing an item in a ranked list, with the assumption that the user will choose each item in the list consecutively with an exponential decay of possibilities, and it is calculated using the Equation 18 below:

$$R = \frac{{}_u R_u}{{}_u R_u^{max}} \qquad (18)$$

Where:

R_u is the expected utility of the ranked list for user u and R_u^{max} is the maximum achievable utility. R_u is defined using Equation 19 as follows:

$$R_u = {}_j \frac{\delta(j)}{2^{(j-1)(\beta-1)}} \qquad (19)$$

Where:
$\delta(j)$ equals 1 if the item at position j is preferred by the user and 0 otherwise, and β is the half-life parameter, set to 5 by the authors.

To evaluate the performance of the proposed approach, the authors compared their results with two baselines: AMN and AMU. For the AMN strategy, several collaborative filtering algorithms were applied, including alternating least squares with the missing as negative assumption (ALS-AMAN), singular value decomposition (SVD), and a neighborhood-based approach including user-user similarity and item-item similarity algorithms. For the AMU strategy, the authors used a simple approach which consisted in ranking the items by their overall popularity; another approach tried was to convert OCCF into a one-class classification, such as one-class SVM.

Figure 5 to 8 below evaluates the impact of the number of features (parameter d) on SVD and wALS. Looking at the figure, we see that for SVD, the performance increases and then decreases as the number of features increases. On the other hand, for wALS, the performance is more stable and keeps on increasing. The performance of wALS usually converged at around 50 features. In their experiments, the authors used the optimal feature count for SVD (10 for Yahoo news data and 16 for user-tag data) and wALS (50).

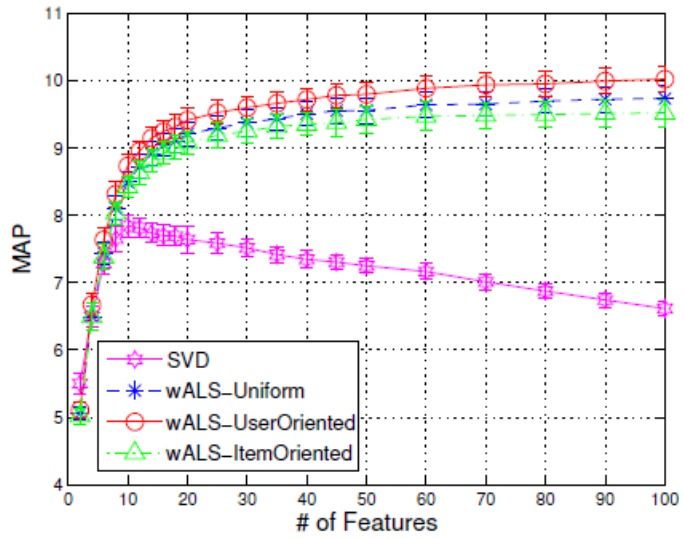


Figure 5 - MAP values for yahoo news data

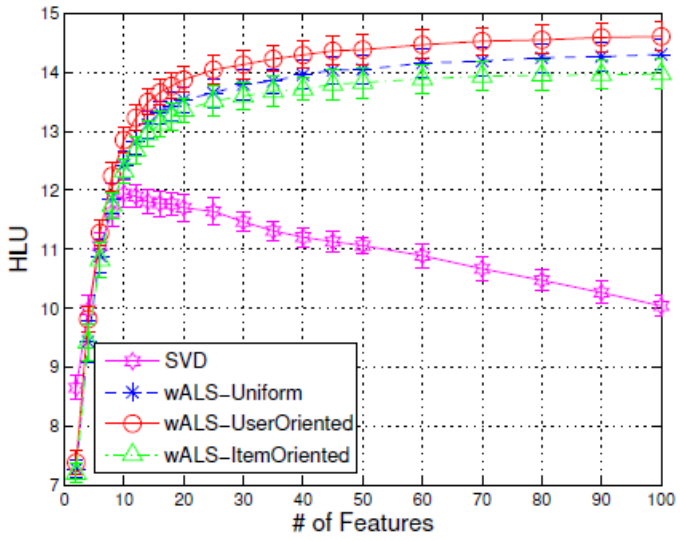


Figure 6 - HLU values for yahoo news data

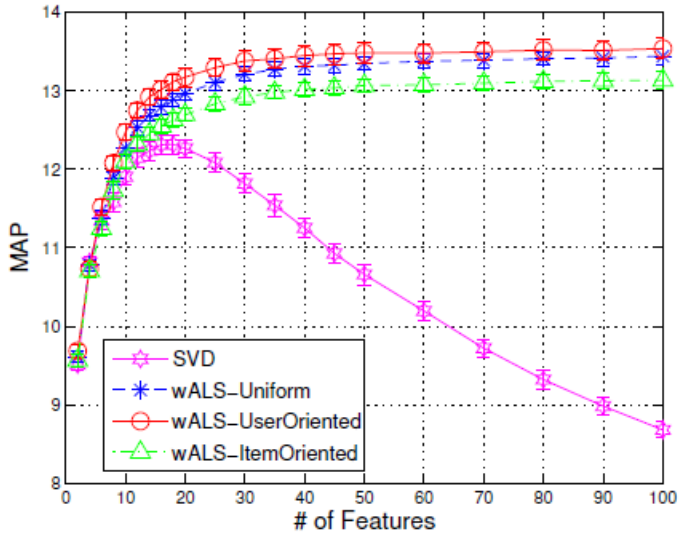


Figure 7 - MAP values for delicious data

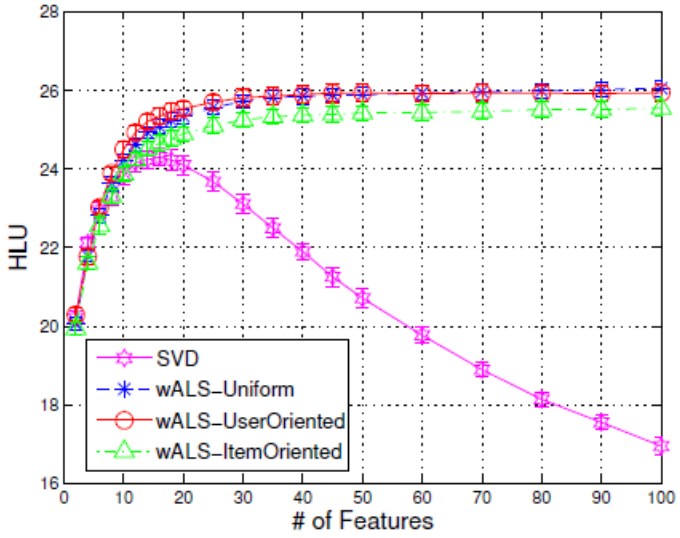


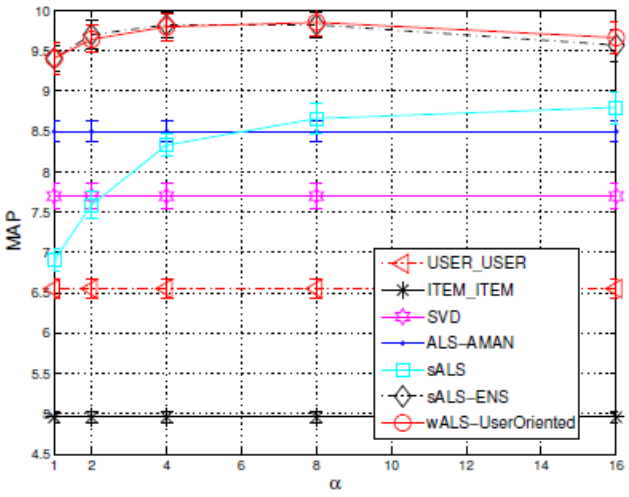
Figure 8 - HLU values for delicious data

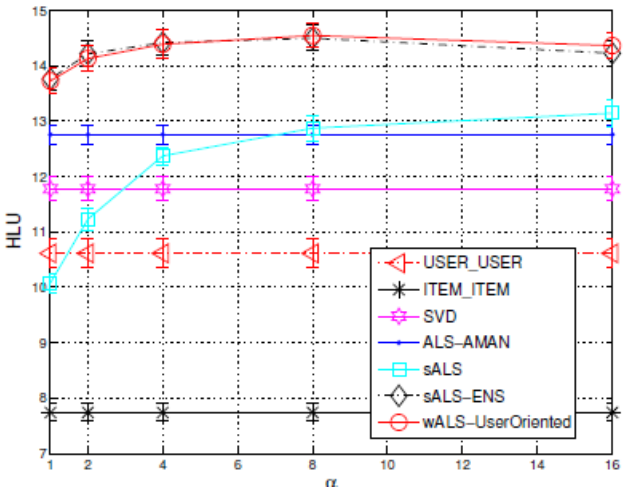
Figure 9 below compares the weighting schemes proposed by the authors (Rong Pan, et al. 2008): Uniform, User oriented, Item oriented.

	Yahoo News				User-Tag			
	sALS-ENS		wALS		sALS-ENS		wALS	
	MAP	HLU	MAP	HLU	MAP	HLU	MAP	HLU
Uniform	9.55	13.98	9.56	14.05	13.33	25.81	13.35	25.89
UserOriented	**9.80**	**14.42**	**9.82**	**14.39**	**13.44**	**25.90**	**13.48**	**25.94**
ItemOriented	9.42	13.76	9.41	13.83	13.03	25.32	13.07	25.43

Figure 9 - Comparisons of different weighting and sampling schemes

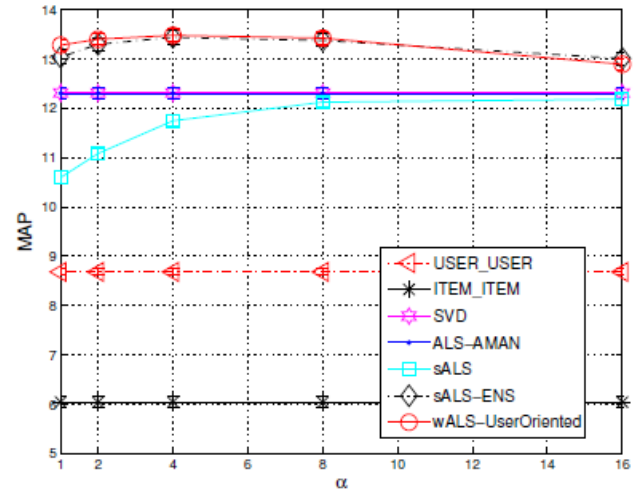
Figures 10 and 11 show the performance comparisons of different methods based on the missing as unknown strategy (Popularity and SVM), methods based on the missing as negative strategy (SVD and ALS-AMAN) and their proposed methods (wALS, sALS).

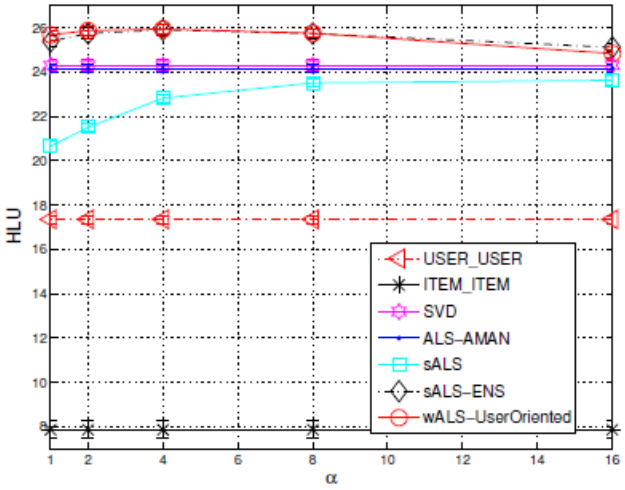




	POP	OCSVM		POP	OCSVM
MAP	3.4528	2.2545	HLU	5.3461	2.5958

Figure 10 - Impact of the ratio of negative examples and positive examples, for yahoo news





	POP	OCSVM		POP	OCSVM
MAP	7.315	3.2500	HLU	15.7703	3.6260

Figure 11 - Impact of the ratio of negative examples and positive examples, for delicious data

Experimental results show that the proposed methods outperform state of the art algorithms on real life data sets including social bookmarking data from delicious and a Yahoo news dataset.

CHAPTER 4

THE PROPOSED SYSTEM

The proposed system consists of a set of configurations exploring different recommendation algorithms. More specifically, it deals with the data sparsity problem with the help of content based, collaborative filtering, and hybrid concepts.

The collaborative filtering approach is implemented in two setups: memory based and model based. The memory based setup is strengthened by using the user based technique. As previously stated, the core component of CF is the neighborhood creation mechanism, which integrates the similarity measures. In this experiment we will be using PIP (Proximity-Impact-Popularity), Cosine, Jaccard similarity measures. The model based setup is based on SVD.

The content based part, which solely focuses on item features, is implemented in two steps: Pre-processing and the prediction process. The pre-process step is required because we are dealing with item descriptions, which is free text, to build the user profile and item profile so that the prediction phase can use it.

4.1 Dataset Overview: Başarı Mobile

Başarı Mobile is the first company who had moved the first step towards Mobile Entertainment Sector in Turkey by 1998. It started to give service under merger departments of Information Technologies and Mobile Internet Services in the structure of Başarı Holding. Since August 2003, **BAŞARI MOBILE IT PRODUCTS AND SERVICES CO.,** keeps up its scope of activities in Value Added Services of Mobile Environments and Information Technologies.

Başarı Mobile, with its dynamic working team and adaptable technological infrastructure, is one of the distinguished mobile content and application provider, reseller and carrier company in Turkey. Especially, experienced at developing

operator side application programming interfaces, user pattern recognition and mobile device adaption for the value added services and always keeps close on customer-satisfaction by giving aftersales support and providing rich and innovative content and developing personalized ways of retrieving and presenting mobile contents. The Company is managing, storing and serving over 15.000 multimedia content products in the distributed network of mobile environment. The Company's WAP (Wireless Application Protocol) and web-based mobile content download platforms have approximately over 28.000 daily visitors and receiving about 9.000 download requests [90].

The ratings in this dataset are expressed implicitly by monitoring the purchase behaviour of the users. This dataset is what is considered to be OCCF, each "1" represent a positive example and a "-" symbol represent missing data, but for the sake of simplicity, in our experiments we will consider it as a negative example, taking a "0". The user-item matrix looks like in the Table 4 below.

Table 5 - User-item matrix for OCCF dataset

	Diablo 3	Angry birds	Walking dead	Fruits ninja	Drive2survive
Sitóe	1	0	0	0	1
Fabião	0	1	0	1	0
Teresinha	0	1	1	0	1
Carlota	1	1	0	1	0
Xiquisso	0	0	1	1	1

4.2 The Methods

Although we have chosen widely used algorithms, our results are mostly not comparable to previous research since we were unable to use a standard dataset. However, the behavior of the algorithms (convergence rates, running times, memory consumption, etc.) was as expected in our test experiments.

This section covers the details of the experiments we have conducted. We start our discussion with the *de facto* standard memory-based algorithm: k-Nearest Neighbor and later in this chapter we will be talking about content based, and the model based approach exploiting using SVD properties.

4.2.1 Collaborative filtering

There are several algorithms proposed in academia to solve the collaborative filtering problem. We have chosen a popular one, KNN (K-Nearest Neighbor), to focus our research on the value of one-class collaborative filtering problem. Choosing well studied algorithms also makes us feel confident with the robustness of the algorithms and the results they produce. In fact, several algorithms that have been proposed perform well on a specific dataset while hardly tolerable to any dataset changes.

With no doubt, k-Nearest Neighbor algorithm is the most known algorithm in machine learning field. This algorithm is simple to implement and produce acceptable results for different datasets. In its essence, k-NN searches for the *k* most similar users of a user, called neighbors. After finding the neighbors, the algorithm makes predictions for a user based on that user's neighbors' ratings. The process of selecting neighbors is shown in Algorithm 4.1.

Require: Neighbor number k, user list **U**, user rating vector set **V**
Ensure: User neighbor vector set N
For user u_i in **U do**

Initialize priority queue Q_i

$v_i \leftarrow V[u_i]$

for user u_j in U do

if $u_i \neq u_j$ then

$v_i \leftarrow V[u_j]$

s←similarity(v_i, v_j)

enqueue (Q_i,u_j,s)

end if

end for

$N[u_i] \leftarrow$ dequeue (Q_i,k)

end for

Algorithm 4.1- Neighbor selection process of k-NN algorithm.

The similarity metric selection is the critical part of Algorithm 4.1. We already discussed several popular similarity metrics in Section 2.7. As we stated, Pearson Correlation and Cosine Similarity have been preferred by most of the previous researchers. However, they were dealing with multi-class datasets. Neighbor selection can be seen as the training phase of k-NN algorithm. Training phase can be extended to cover prediction calculations if the algorithm is expected to find predictions for all possible user-item pairs. However, generally prediction step is done in testing phase in which the algorithm is expected to calculate predictions only for test cases. The basic approach to make predictions is to equally value the ideas of neighbors. In other words, the final rating is the average rating of all neighbors. The rating can be calculated using Equation 4, presented below for practicality.

$$pred\ u,i\ = \frac{n \in neighbors(u)\ r_{ni}}{number\ of\ neighbors}$$

In the next section we are going to talk about neighborhood creation and the similarity measures applied.

4.2.2 Finding Friends

4.2.2.1 User Item Matrix

In a RS consisting of M users and N items, there is a $M \times N$ user-item matrix R. Each entry $r_{m,n} = x$ represents the rating that user m gives to item n, where $x \in \{0,1\}$. The default $r_{m,n}$ value, meaning that the rating is unknown, is 0.

The user-item matrix can be decomposed into row vectors:

$$R = [u_1, \ldots, u_M]^T, u_m = [r_{m,1}, \ldots, r_{m,N}]^T, m = 1, \ldots, M.$$

The row vector um represents the ratings of user m for all of N items.

Alternatively, the matrix can also be represented by its column vectors:

$$R = [i_1, \ldots, i_N]^T, i_n = [r_{1,n}, \ldots, r_{M,n}]^T, n = 1, \ldots, N.$$

The column vector i_n represents the ratings of item m by all of M users.

An example user-item matrix R is presented in Table 5:

Table 6 - User item Matrix

	i_1	i_2	i_3	i_4	…	i_N
u_1	1	0	0	0	…	1
u_2	1	0	0	1	…	0
u_3	0	1	0	0	…	0
…	…	…	…	…	…	…
u_5	0	0	0	0	…	1

4.2.2.2 PIP Similarity Measure

In sections 3.1 we have talked about the PIP (Proximity – Impact - Popularity) Similarity Measure, which was originally designed for multiclass collaborative filtering recommender systems. The choice of this similarity measure is because it showed good results in multi-class CF when using a very sparse dataset. However the problem we have in hands, is an application of OCCF. Below, in Table 5, follows the formal description of the formulas for OCCF.

Table 7 - Formal description of the PIP formulas for OCCF

Agreement	For any two ratings r_1 and r_2, let R_{max} be the maximum rating and R_{min} the minimum in the rating scale, and let $R_{med} = (R_{max} + R_{min}) / 2 = (1+0)/2 = 0.5$ A Boolean function Agreement(r_1, r_2) is defined as follows: Agreement(r_1, r_2) = false if (r_1 = 1 and r_2 =0) or (r_1= 0 and r_2 = 1), and Agreement(r_1, r_2) = true otherwise
Proximity	A simple absolute distance between the two ratings is defined as: $D(r_1, r_2) = 0$ if Agreement(r_1, r_2) is true, and $D(r_1, r_2) = 2$ if Agreement(r_1, r_2) is false Then the Proximity(r1, r2) is defined as: $Proximity(r_1,r_2) = \{3 – D(r_1,r_2)\}^2$
Impact	Impact Impact(r_1, r_2) is defined as: $Impact(r_1, r_2) = (\|r_1 - R_{med}\| + 1) \times (\|r_2 - R_{med}\| + 1)$ if Agreement(r_1, r_2) is true, and $Impact(r_1, r_2) = 1/(\|r_1 - R_{med}\| + 1)(\|r_2 - R_{med}\| + 1)$if Agreement($r_1$, r_2) is false

Popularity	Let μ_k be the average rating of item k by all users, Then Popularity(r_1, r_2) is defined as: Popularity(r_1, r_2) = 1 +(((r1+ r2)/2)- μ_k)^2 if ($r_1 > \mu_k$ and $r_2 > \mu_k$) or ($r_1 < \mu_k$ and r2 < μ_k), and Popularity(r_1, r_2) = 1 otherwise

Consider as an example, two user vectors v_1 and v_2, rating six items i (i={1,2,3,4,5,6}) given as follows:

v_1 = {1,0,1,0,0,1}

v_2 = {0,1,1,0,1,1}

The PIP similarity value for this two users is calculated as described in the pseudo-code below:

Pipsimilarity(v1,v2,ratingMatrix)

pipMeasure = 0.0;

agreement=false;

distance = 0;

averageRating = (maxRating + minRating) / 2 = (1+0) / 2 = 0.5

proximity=0;

impact = 0;

popularity = 0;

for each item i Є itemList

Integer ratingV1 = getRating(v_1,i)

Integer ratingV2 = getRating(v_2,i)

If (ratingV1 == 1 && ratingV2 == 1)

agreement = true;

else agreement = false

if(agreement)

distance = 0;

else distance = 2;

proximity = (3 - distance)^2;

if(agreement)

impact = |(r1 - rmed)+1||(r2 - rmed)+1|;*

else impact = 1/ |(r1 - rmed)+1||(r2 - rmed)+1|;*

let averageRatingItem = average rating of item i by all users

if ((ratingV1 > averageRatingItem && ratingV2 > averageRatingItem) || (ratingV1 < averageRatingItem && ratingV2 < averageRatingItem))

popularity = 1+(((ratingV1+ratingV2)/2)- averageRatingItem)^2;

else popularity = 1;

*pipMeasure = pipMeasure + (proximity * impact * popularity);*

return pipMeasure;

Consider for example a sample dataset containing three user vectors and six items, as follows:

v_1 = {1,0,1,0,0,1}

v_2 = {0,1,1,0,1,1}

v_3 = {0,0,1,1,0,0}

***i* = 1**

ratingV1 = 1;

ratingV2 = 0;

averageRatingItem = 0.33;

agreement = false;

distance = 2;

proximity = (3 - 2)^2 = 1

Impact = 1 / (| 1 - 0.5 | + 1) * (| 0 - 0.5 | + 1) = 0.44

Popularity = 1

PIP = 0.44

***i* = 2**

ratingV1 = 0;

ratingV2 = 1;

averageRatingItem = 0.33;

agreement = false;

distance = 2;

proximity = $(3 - 2)^2 = 1$

Impact = 1 / (| 1 - 0.5 | + 1) * (| 0 - 0.5 | + 1) = 0.44

Popularity = 1

PIP = 0.44

***i* = 3**

ratingV1 = 1;

ratingV2 = 1;

averageRatingItem = 1;

agreement = true;

distance = 0;

proximity = $(3 - 0)^2 = 9$

Impact = (| 1 - 0.5 | + 1) * (| 1 - 0.5 | + 1) = 2.25

Popularity = 1

PIP = 20.25

***i* = 4**

ratingV1 = 0;

ratingV2 = 0;

averageRatingItem = 0.33;

agreement = false;

distance = 2;

proximity = $(3 - 2)^2 = 1$

Impact = 1 / (| 1 - 0.5 | + 1) * (| 0 - 0.5 | + 1) = 0.44

Popularity = 1

PIP = 0.44

***i* = 5**

ratingV1 = 0;

ratingV2 = 1;

averageRatingItem = 0.33;

agreement = false;

distance = 2;

proximity = $(3 - 2)^2 = 1$

Impact = 1 / (| 1 - 0.5 | + 1) * (| 0 - 0.5 | + 1) = 0.44

Popularity = 1

PIP = 0.44

***i* = 6**

ratingV1 = 1;

ratingV2 = 1;

averageRatingItem = 0.67;

agreement = true;

distance = 0;

proximity = $(3 - 0)^2 = 9$

Impact = (| 1 - 0.5 | + 1) * (| 0 - 0.5 | + 1) = 2.25

Popularity = $1 + (((1+1)/2) – 0.67)^2 = 1.11$

PIP = 9 * 2.25 * 1.11 = 22.48

PIPMeasure = 0.44 + 0.44 + 20.25 + 0.44 + 0.44 + 22.48 = 44.49

4.2.2.3 Jaccard Similarity measure

In section 2.7.4, we presented the Jaccard coefficient. The choice of this similarity measure is that we believe that Jaccard Similarity is more suitable to one-class collaborative filtering applications since we are dealing with binary rating vectors. Thus, we had used Jaccard Similarity as defined in Equation (8), presented below for practicality.

$$J(x_1,x_2) = a / a + b + c$$

Consider for example, two user vectors v_1 and v_2, rating six items i (i={1,2,3,4,5,6}) given as follows:

v_1 = {1,0,1,0,0,1}
v_2 = {0,1,1,0,1,1}

The Jaccard similarity value for these two users is calculated as described in the pseudo-code below:

```
Jsimilarity(v1,v2, ratingMatrix)
p = 0;
q = 0;
r = 0;
for each item i Є itemList
    if (rating (v1,i) = 1 and rating (v2,i) = 1) then p = p +1;
    if (rating (v1,i) = 1 and rating (v2,i) = 0) then q = q +1;
    if (rating (v1,i) = 0 and rating (v2,i) = 1) then r = r +1;
return p/(p+q+r)
```

Running the algorithm for the vectors v_1 and v_2 , we have $p = 2$, $q = 1$, $r = 2$. Applying these values to equation 12 we have:

$J(v_1,v_2) = 2 / 2 + 1 + 2 = 2 / 5 = 0.4$. Meaning the users v_1 and v_2 are 40% similar with each other.

4.2.2.4 Cosine Similarity Measure

The cosine similarity measure is defined by the cosine of the angle between two vectors, section 2.7.2, according to equation (6), presented below for practicality.

$$similarity = \cos\theta = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

The resulting similarity ranges from −1 meaning exactly opposite, to 1 meaning exactly the same, with 0 usually indicating independence, and in-between values indicating intermediate similarity or dissimilarity.

In the case of measuring the similarity of two user vectors, the cosine similarity will range from 0 to 1, since the given ratings are not negative. The angle between two user vectors cannot be greater than 90°. Consider for example the following user vectors:

v_1 = {1,0,1,0,0,1}

v_2 = {0,1,1,0,1,1}

Applying the equation we have:

$v_1 . v_2 = (1*0) + (0*1)+(1*1)+(0*0)+(0*1)+(1*1) = 2$

$\|v_1\| = (1^2 + 0^2 +1^2 + 0^2+0^2+1^2)^{1/2} = 3^{1/2} = 1.73$

$\|v_2\| = (0^2 + 1^2 +1^2 + 0^2+1^2+1^2)^{1/2} = 4^{1/2} = 2$

$COS(v_1,v_2) = 2 / (1.73 * 2) = 0.58$

4.2.3 Content based filtering

This part explains in detail, how the content information of the games is included in the prediction process. Content-based filtering method, which is another well-known technique in recommender systems, have been developed using learning procedures. These procedures require training data to identify personal preferences

(user profile) based on the previously purchased items and item profile of all items existing in the system.

4.2.3.1 Item profile

As the proposed system works in the game domain, its games are considered as its items.

The profile of a game in the system contains the following features:

- Game number – identification number of the game
- Name – the name of the game
- Description – the description of the game
- Category – the category of the game
- Cost – the cost of the game

However, in the proposed system we will only use the description field; we believe this field characterizes better the item.

4.2.3.2 User profile

Before going into further detail, the notation and definitions required for understanding our approach are introduced. Let $C = \{c_1, c_2, \ldots, c_n\}$ be the set of all content, $T = \{t_1, t_2, \ldots, t_m\}$ be the set of all terms, and $U = \{u_1, u_2, \ldots, u_l\}$ be the set of distinct users. The content c_j is a set of terms, each of which may appear in multiple contents with different weights that quantify the importance of the term for describing the content.

In our study, a weight $w_{i,j}$ associated with a pair (t_i, c_j) (i.e., a term t_i of a content c_j) is computed by making use of a TF-IDF weighting scheme [93]. To build a user profile, we extract the descriptions of all items rated by the user. The preference indicator of implicit feedback can be represented as a form of a pair (u_h,

c_j), where $u_h \in U$ is a user and $c_j \in C$ is a specific content. The pair implies that user u_h bought content c_j. In other words, the profile of a user is composed of the ratings given to the games along with the content information of these games.

4.2.3.3 User profile generation

Our approach to modeling user interests mainly consists of the extraction of the terms that constitute the description of the items and all the descriptions are concatenated. After this procedure, the result is pre-processed by stemming the words and removing stopwords [95]. After extracting terms, each interest content c_j is represented as a vector of attribute-value pairs as follows:

$$c_j = \{(t_{1,j}, w_{1,j}), (t_{2,j}, w_{2,j}), \ldots, (t_{m,j}, w_{m,j})\} \quad (20)$$

where $t_{i,j}$ is the extracted term in c_j and $w_{i,j}$ is the weight of t_i in c_j. $w_{i,j}$ is computed by the static TFIDF term-weighting scheme [93] and is defined in section 4.3.3.6.

4.2.3.4 Lucene

Apache Lucene(TM) is a high-performance, full-featured text search engine library written entirely in Java. It is a technology suitable for nearly any application that requires full-text search, especially cross-platform [87].

In our proposed system, Lucene is used only as a convenient and efficient means of converting documents into term vectors and for extracting statistics about the corpus for computing TFIDF. Lucene itself cannot be reliably used to compare indexed documents with the query vector, since there is no guarantee that Lucene's scores will hold stable across different sets of documents.

4.2.3.5 Preprocessing (removing stopwords and stemming the words)

As stated previously, we make use of a Turkish dataset. Because of the agglutinative nature of the language, stemming is an essential task for indexing and searching documents in Turkish [97]. Turkish, has a rich morphological structure. Words are usually composed of a stem and of at least two or three affixes appended to it. This is why it is usually harder to analyze a Turkish text. Stemming is therefore a more essential task for indexing and information retrieval purposes in agglutinative languages [97].

For this study, we made use of a stemmer called Turkish Analyzer that integrates with Lucene information retrieval system. This stemmer for turkish words makes use of Snowball language. Snowball is a language in which stemming algorithms can be easily represented. [96]. Snowball has been widely used in stemming tools.

At the end, this pre-processing step outputs two vectors for each item, one containing the processed terms and the corresponding weight of each term, so that similarity calculations can be performed.

4.2.3.6 Overall prediction value

In the cases where the object is a hybrid algorithm, Collaborative Filtering (CF) and Content based filtering (CB) for example, the overall prediction can be calculated using Equation (21) as follows:

$$Overall\_prediction = \lambda * CF + (1 - \lambda) * CB \qquad (21)$$

Where parameter λ is used to fuse information from both methods used, for example CF and CB, to predict the rating for the active users.

4.3 Singular Value Decomposition

As earlier mentioned, what makes SVD interesting to recommender systems, is that the matrices generated by performing SVD provides with the best lower rank approximations of the original matrix A. SVD-based recommendation algorithms produces good results, but its matrix factorization steps is very expensive to execute.

As an example to dimension reduction, consider the <user, game> rating matrix in Table 7 below:

Table 8 - Example user/game matrix

User/Game	Game 1	Game 2	Game 3	Game 4	Game 5
User 1	1	5	0	5	4
User 2	5	4	4	3	2
User 3	0	4	0	0	5
User 4	4	4	1	4	0
User 5	0	4	3	5	0
User 6	2	4	3	5	3

By applying SVD decomposition to this matrix we get:

U					S					V^T				
0.5	0.4	-0.3	-0.4	0.3	16.5	0	0	0	0	0.3	0.6	0.3	0.6	0.3
0.5	-0.3	0.6	0.3	0	0	6.2	0	0	0	-0	0.2	-0.4	-0.3	0.8
0.2	0.8	0.3	0.2	-0.5	0	0	4.4	0	0	0.7	0	0.1	-0.6	0.3
0.4	-0.3	0.1	-0.7	-0.3	0	0	0	2.9	0	-0	-0.1	0.9	-0.2	0.2
0.4	-0.2	-0.6	0.4	-0.5	0	0	0	0	1.6	0.2	-0.7	0	0.5	0.5
0.5	0	-0.1	0.3	0.6										

Matrices U, S, and V^T are now calculated. From here we can reduce the dimensions to 3, tridimensional. To do this, we simply take the first three diagonal values from S producing S_K, then reduce U and V^T accordingly to produce U_k and V_k^T. The end result is as follows:

U_K			S_K			V^T_K				
0.5	0.4	-0.3	16.5	0	0	0.3	0.6	0.3	0.6	0.3
0.5	-0.3	0.6	0	6.2	0	-0.4	0.2	-0	-0.3	0.8
0.2	0.8	0.3	0	0	4.4	0.7	0	0.1	-0.6	0.3
0.4	-0.3	0.1								
0.4	-0.2	-0.6								
0.5	0	-0.1								

Now we find the most similar users using the 3-Dimensional matrices above with one of the similarity calculation algorithms discussed previously.

CHAPTER 5

EVALUATION

This chapter presents the details of how the proposed system is evaluated in order to test its performance. First, the dataset and the metrics used for the experimental evaluation are introduced. Then, the results of the conducted experiments are stated and discussed.

In previous chapter, we gave details of our research in which we test different recommendation techniques using a one-class collaborative filtering problem. As previously stated, one-class collaborative filtering problems are harder to solve than multi-class since datasets contain only positive examples. With the absence of counter-examples it is hard to train algorithms.

5.1 Data Set

The experimental evaluation of the proposed system was conducted using the Başarı Mobile proprietary dataset, introduced in previous sections. The first thing we did with the dataset was to *clean* it, with that we mean remove content other than games, and removing users who did not buy any content. After this preprocessing step, we end up with 817,200 ratings, 2,635 games and 126,068 users.

The density of the user-item matrix created from this dataset is:

$$\frac{817,200}{2635 \text{ x } 126,068} = 0.25\ \%$$

In order to evaluate the prediction mechanism of the proposed system, cross validation method was used and among the various cross validation methods, the holdout method was preferred. Because the resource consumption is quite different for memory and model based techniques, we separated two subsets of the data. One

to be used with memory based algorithms including content-based and the other for model based using SVD decomposition. Following this method, the memory based dataset was separated into two sets, called the training set and the testing set. Thus, after a subset of 500 users containing at least 5 ratings was randomly extracted from the data set, 300, 200 and 100 of them were selected as the testing users respectively. And the rest 200, 300, 400 were selected as the training users. In the case of model-based, the subset contains 3000 users randomly extracted from the dataset, 500, 1000 and 1500 of them were selected as the testing users respectively. And the rest 2500, 2000, 1500 were selected as the training users.

5.2 Evaluation Metrics

In collaborative filtering applications, Precision is one of the most popular metrics for evaluating information retrieval systems, along with Recall [40], as introduced by Cleverdon in 1968 [98]. To calculate the Precision and Recall, The item set must be separated into two classes—relevant or not relevant. That is, if the rating scale is not already binary, which is exactly the case with the Başari Mobile dataset.

Precision is defined as the ratio of relevant items selected to number of items selected, shown in Equation (22):

$$Precision = \frac{\#tp}{\#tp + \#fp} \qquad (22)$$

Precision represents the probability that a selected item is relevant.

Recall on the other hand, shown in Equation (23), is defined as the ratio of relevant items selected to total number of relevant items available. Recall represents the probability that a relevant item will be selected.

$$Recall = \frac{\#tp}{\#tp + \#fn} \qquad (23)$$

Where:

#tp – true positives

#fp – false positives

#fn – false negatives

In one-class collaborative filtering problems, final values obtained would be either a "1" or a "0". During prediction phase, any value between the range (0, 1) is rounded. The effect of a predicted rating of 0.6 would be same as the effect of a predicted rating of 0.99.

Note that our dataset contains only positive test cases. So we could not find out the number of false negative (#fn) and true negative (#tn) examples after any experiment, that is why we use Precision as our evaluation metric.

5.3 Results of the Algorithms: CF (Memory based) and CB

In order to test the performance of the proposed system's prediction approach, the Precision values obtained for the three dataset, which were explained in detail in section 5.1, are summarized in the Table 8. The parameters *k* (number of "friends") used throughout the prediction process were set to 25. The tested algorithms include pure collaborative filtering, pure content-based filtering and hybrid filtering:

- Pure collaborative filtering (CF): PIP (Proximity – Impact – Popularity similarity measure), COS (Cosine similarity measure), Jaccard similarity measure;

- Pure content based filtering (CB) : COS using TFIDF (Term Frequency / Inverse Document Frequency) weights;

- Hybrid approach: PIP + CB, Jaccard + CB, COS + CB.

Table 9 - The results of memory based algorithms

Methods	Testing Users		
	100	200	300
PIP	0.6382	0.6245	0.5813
TFIDF	0.4075	0.483	0.4952
COS	0.8521	0.8624	0.8174
JACCARD	0.8525	0.8358	0.8107
PIP + CB (λ = 0.8)	0.6751	0.6481	0.6124
PIP + CB (λ = 0.4)	0.535	0.5108	0.5092
JACCARD + CB (λ = 0.8)	0.8742	0.8685	0.8215
JACCARD + CB (λ = 0.4)	0.635	0.5988	0.5525
COS + CB (λ = 0.8)	0.8669	0.8658	0.8199
COS + CB (λ = 0.4)	0.605	0.5988	0.5458

The values are put into a chart in order to make it easier to understand.

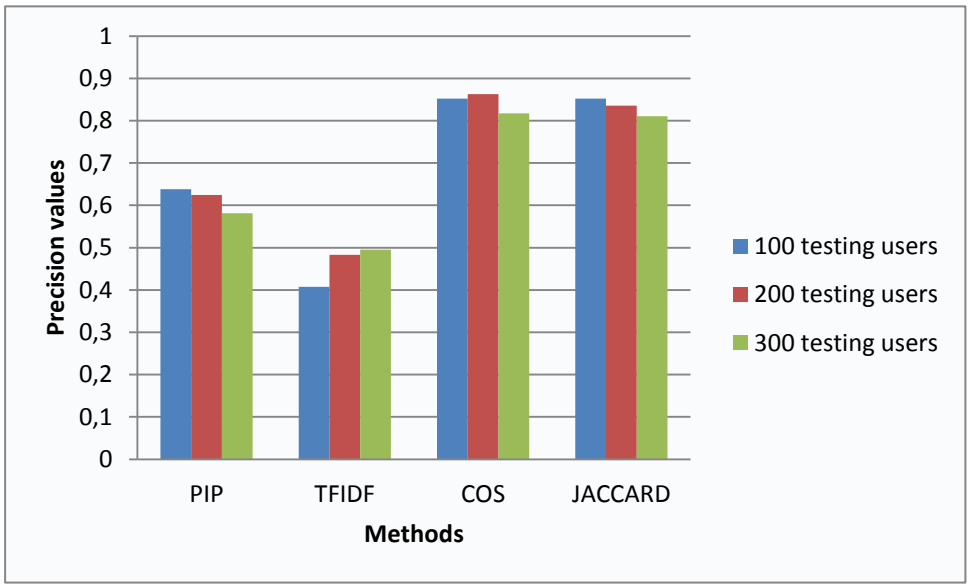


Figure 12 Precision values for different methods

As can be seen from Table 7 and the chart in Figure 5, the content based approach produces the worst results. Among the collaborative filtering algorithms, the PIP produces poor results when compared to COS and Jaccard; both COS and Jaccard result in very close Precision values. In the case of hybrid approaches, the Jaccard + CB gives best results while PIP + CB gives the worst results.

When analyzing the obtained results displayed in Table 7 and the chart in Figure 5, we see that the number of training and testing users influence differently each method. The CB approach, which does not take advantage of the neighbors, produce better results with a higher number of testing users and worst when a small number of testing users is used. The reason for this behavior, we believe, is because the users like diversity, they don't like many games of same kind. In collaborative filtering approach, Cosine similarity produced better results with medium size of testing users, but worst with a higher number of testing users. The small and medium size of the testing users produced very close results. On the other hand, when running the experiment with Jaccard, it shows us that the higher the number of training users, the better the accuracy results, this was the outcome we were expected to obtain in all collaborative filtering experiments, but due to the characteristics of the dataset, the results were different. The PIP similarity measure produced the worst results in all datasets.

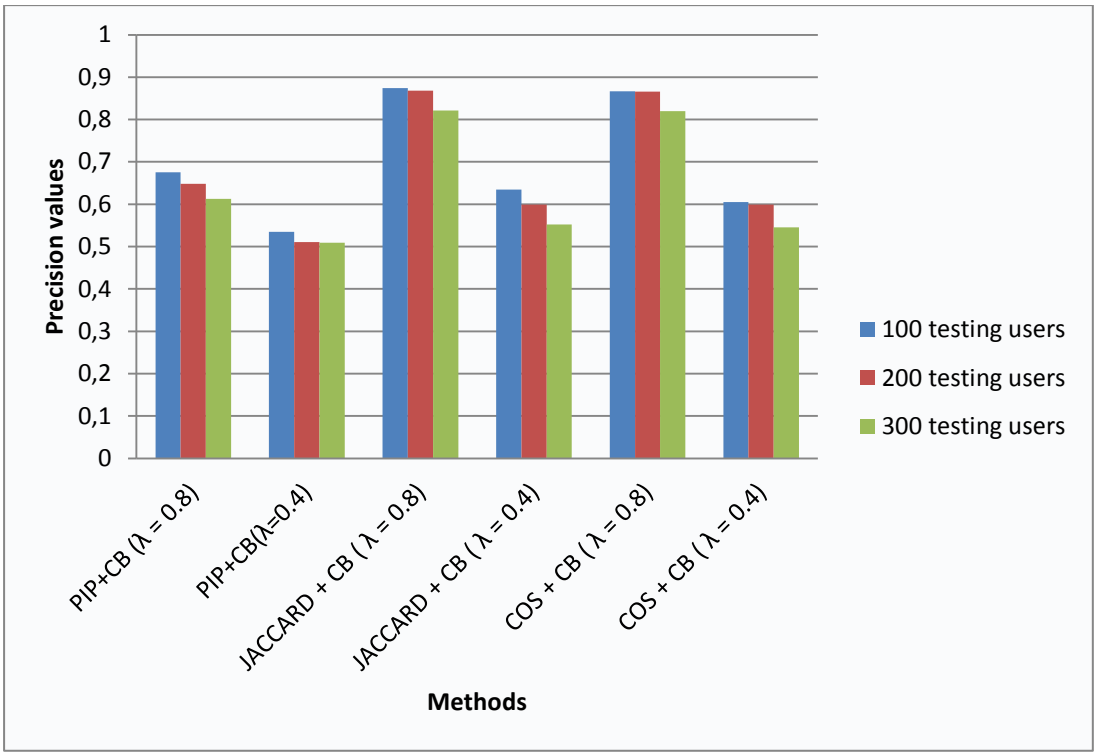


Figure 13 Hybrid approaches results

Turning our eyes to hybrid approaches (Figure 6), we can detect a pattern, the higher the training users, the higher the Precision values obtained. This behavior is consistent in all hybrid approaches tested. We conclude that when the number of training users is high, the system have a chance to train better.

The vast majority of previous work done in this field by researchers is found on multi-class collaborative filtering problems and they have proposed methods accordingly. During the course of this thesis, we realized that some of these methods fit very well on one-class collaborative filtering problems, with some delicate modifications.

On top of the limitations of CF, OCCF includes also the missing counter-examples and a higher sparsity rate on the datasets. In order to overcome these issues, we proposed the use of CB approach in combination with pure CF, making a hybrid approach. Without the counter-examples, the conventional CF methods would give very bad performance or even not at all. One way overcome these issues

in OCCF problems is to assume that all missing entries are counter-examples, but this approach involves a high bias.

Now let us consider the impact of the λ parameter; as stated earlier, this parameter is used to fuse information from both collaborative prediction and content based prediction to predict the missing rating for the active users. It determines the extent to which the item similarity relies on collaborative filtering methods or content similarity. With λ = 1, it indicates that the similarity depends completely on collaborative similarity, whereas it depends completely on content similarity when λ=0.

For the purpose of determining the sensitivity of λ, several experiments were carried out on all configurations in which the value of λ was varied from 0 to 1. The experimental results show that more accurate predictions can be obtained when the value of λ is around 0.8. Because in this way, the prediction can both exploit collaborative filtering and content based similarity in certain and sensible amounts, which shows that CF and CB approaches both have a very important and indispensable role for rating prediction. On the other hand, for small values of λ (λ<= 4), it produces poor results.

5.4 Results of the Algorithms: SVD Model-based

In this section, we are going to discuss the results of SVD algorithm on our datasets. We had tested the SVD algorithm on 3 datasets. Similar to memory based approach, we made the assumption that all the missing data is negative, which results in a rating matrix R such that $R_{ij} \in \{0, 1\}$.

The first step in this experiment was to construct the rating matrix, as in memory based setup, we then factor the matrix R, by using a linear algebra toolbox, and obtain a low-rank approximation after applying the following steps described in [103]:

- Apply the SVD decomposition to matrix R to obtain U, S and V;
- Collapse the matrix S to dimension k;
- Calculate the square-root of the matrix S_k, to obtain $S_k^{1/2}$;
- Calculate two resultant matrices: $U_k S_k^{1/2}$ and $S_k^{1/2} V_k^T$

We observe that the dimension of $U_k S_k^{1/2}$ is $m \times k$ and the dimension of $S_k^{1/2} V_k^T$ is $k \times n$. This $m \times k$ matrix is the k dimensional representation of m customers. We then performed vector similarity (cosine similarity) to form the neighborhood in that reduced space.

The important variable that may change the results of SVD is the number of features used in calculations. Table 9 shows the outcomes of SVD algorithm for different feature counts. The results are summarized in the Table 9 below:

Table 10 - Results of model SVD-based CF

Feature count	Testing users		
	500	1000	1500
2	0.28	0.29	0.27
5	0.55	0.56	0.53
7	0.60	0.61	0.60
9	0.65	0.65	0.64
11	0.67	0.67	0.66
13	0.70	0.71	0.70
20	0.72	0.74	0.72
50	0.80	0.80	0.79
70	0.81	0.80	0.78
100	0.84	0.85	0.83
150	0.87	0.85	0.85
190	0.87	0.87	0.87
200	0.87	0.87	0.87
220	0.87	0.86	0.86
250	0.86	0.86	0.85
500	0.83	0.82	0.82
600	0.83	0.82	0.82
700	0.83	0.81	0.80
1000	0.81	0.80	0.80

The values are put into a chart in order to make it easier to understand.

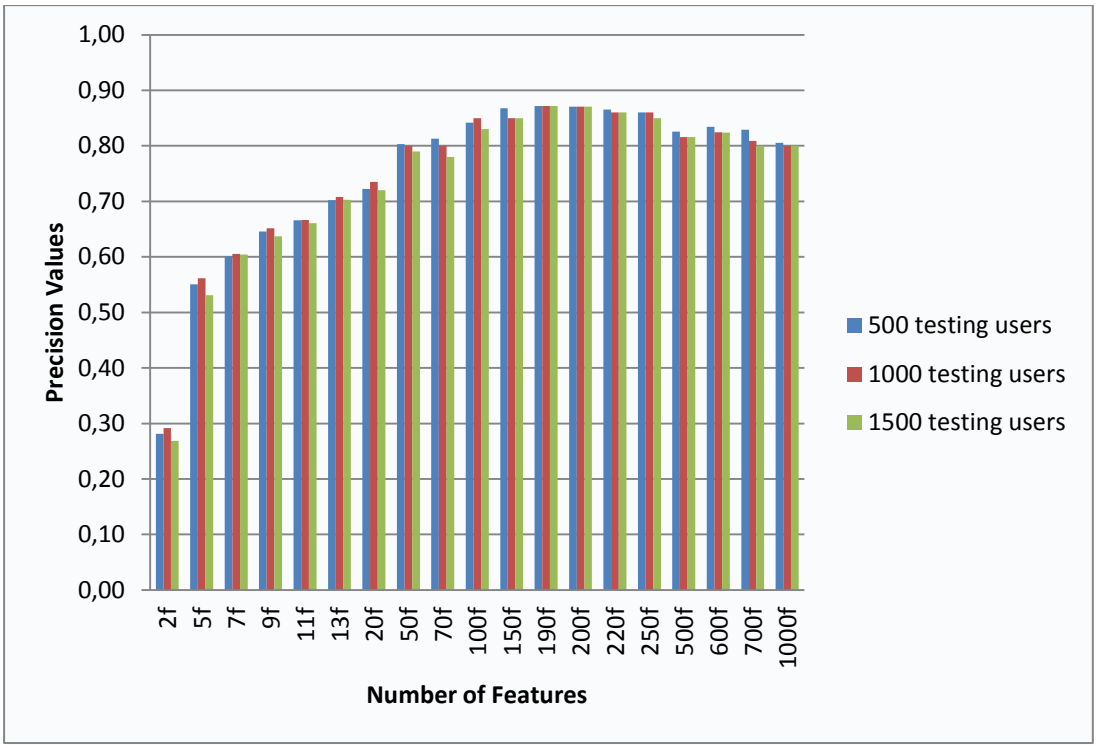


Figure 14 Feature count in SVD

As can be seen from Table 9 and the chart in Figure 7, the first conclusion is that for number of dimensions lower than 9 it produces worse precision values when compared to traditional collaborative filtering approach.

Taking into consideration the ration of testing/training datasets, x, we see that for $x < 0.5$ produces better results.

Precision values of our test runs first increased and then decreased as we increased the number of features. This behavior is consistent with the results obtained by authors of [103]. In [103], optimal feature count is said to be around 14. For our case, as can be seen from Table 9, 190 is the optimal feature count. For feature count greater than 190, the precision shows a slight decrease but still high.

With the optimal feature count, SVD seems to perform as good as the best k-NN based algorithms previously tested, CF with Jaccard. However, the results were inconsistent when compared to the results of [103], in which, for e-commerce dataset, the results were consistently worse than traditional collaborative filtering approach.

For the conducted experiments, we have used a laptop SONY VAIO F SERIES with 4GB of DDR3 memory, 500GB of 7200rpm hard disc capacity, i7 quad-core Intel CPU with 2.00GHz clock, 64-bit architecture and Windows 7 Home Premium 64-bit operating system. During the execution of the experiments, the CPU usage peaked at around 18% and memory consumption was about 82% (3.20 GB). The time complexity for CF methods was consistent with each other, with the exception of PIP similarity measure, which takes very much longer. While the CF methods took around 45 minutes to complete, when applied with PIP similarity measure, it took around 61 hours. The CB approach was the fastest, taking around 10 minutes. For all the methods, parallelized should be taken in consideration when applying in the real world system and powerful machines are a critical requirement, if the response time is a major concern, which it is in e-commerce services and also a special attention should be given to system optimization.

CHAPTER 6

CONCLUSION AND FUTURE WORK

Within this thesis work, an evaluation of different collaborative filtering, content based filtering, hybrid algorithms as well as model based approach has been presented.

In order for the recommendation systems to function correctly; they need to learn the user's tastes. Obviously, the most direct way to achieve this is to explicitly ask the user for what he/she likes/dislikes. However, users hardly cooperate in such a situation. Thus, the only option that remains is to look for implicit data hidden in the user's behavior. This implicit data can be anything like clicking on an item, time spent on an item on a web page, adding the item to the shopping basket, and so on. The problem with such data is that they are not that informative. In most cases, this data will only reveal that a user is interested in something. It is hardly possible to find cases where a recommender system is able to conclude that a user is not interested in something. Lack of counter-examples makes such cases a natural candidate for applications that should use one-class collaborative filtering methods. Being able to deal with cases that counter-examples do not exist, makes one-class collaborative filtering applications remarkable.

First, a brief introduction to recommender systems has been given by stating the current approaches and theories used in these systems. Then, the related work in the area has been covered by analyzing a variety of recommendation systems from different domains together with their advantages and disadvantages. After that, the architecture, and the prediction mechanism of the proposed system has been examined in detail. And lastly, the evaluation scheme used to test the prediction performance of the proposed system has been explained. In addition, the results of the conducted experiments have also been discussed.

Empirical analysis show that the binary similarity measure and singular value decomposition produce better results on this dataset.

Further research issues include the exploration of this dataset by applying other types of product recommendation. For example, model based Singular Vector Decomposition or clustering approaches.

REFERENCES

[1] Toby Segaran: Programming Collective Intelligence, O'Reilly Media, Inc.,2007

[2] J. Ben Schafer, Joseph Konstan, John Riedl: Recommender Systems in e-commerce, ACM Conference on Electronic Commerce (EC-99), 1999

[3] Sanjeev Kumar Sharma, Dr. Ugrasen Suman: Design and Implementation of Architectural Framework of Recommender System for e-Commerce, IRACST - International Journal of Computer Science and Information Technology & Security (IJCSITS), Vol. 1, No. 2, December 2011

[4] Tariq Mahmood, Francesco Ricci, Mahmood, T., Ricci, F.: Improving recommender systems with adaptive conversational strategies, ACM, 2009

[5] Anne Yun-An Chen and Dennis McLeod: Collaborative Filtering for Information Recommendation Systems, CiteSeer^{x}, 2009

[6] Burke, R.: Hybrid web recommender systems. In: The Adaptive Web, pp. 377–408. Springer Berlin / Heidelberg, 2007

[7] Anna Goy, Liliana Ardissono, and Giovanna Petrone: Personalization in E-Commerce Applications, In: The adaptive web, Pages 485-520, Springer-Verlag Berlin, Heidelberg, 2007

[8] Gediminas Adomavicius and Alexander Tuzhilin: Towards the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions, IEEE Transactions on Knowledge and Data Engineering, Volume 17 Issue 6, June 2005, Page 734-749

[9] Martin Szomszor, Ciro Cattuto, Harith Alani, Kieron O'Hara, Andrea Baldassarri, Vittorio Loreto, Vito D.P. Servedio: Folksonomies, the Semantic Web, and Movie Recommendation, In: *4th European Semantic Web Conference, Bridging the Gap between Semantic Web and Web 2.0,* 2007

[10] Rajatish Mukherjee, Partha Sarathi Dutta, and Sandip Sen: MOVIES2GO - A new approach to online movie recommendation, In Proceedings of IJCAI Workshop on Intelligent Techniques for Web Personalization, Seattle, WA, USA, August 2001

[11] Shinhyun Ahn and Chung-Kon Shi: Exploring Movie Recommendation System Using Cultural Metadata, In Cyberworlds, 2008 International Conference on, pp. 431 – 438, 2008

[12] Greg Linden, Brent Smith, and Jeremy York: Amazon.com Recommendations Item-to-Item Collaborative Filtering, In Internet Computing, IEEE, pp. 76 – 80, 2003

[13] Bradley N. Miller, Istvan Albert, Shyong K. Lam, Joseph A. Konstan, John Riedl: MovieLens Unplugged: Experiences with an Occasionally Connected Recommender System, In IUI '03 Proceedings of the 8th international conference on Intelligent user interfaces, Pages 263-266 , ACM New York, NY, USA, 2003

[14] Wade, W. A grocery cart that holds bread, butter and preferences. NY Times, Jan. 16, 2003. http://www.nytimes.com/2003/01/16/technology/circuits/16safe.html accessed June 19, 2012, 19:08

[15] Hyung Jun Ahn: A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem, In Information Sciences: an

International Journal, Volume 178 Issue 1, January, Pages 37-51 , Elsevier Science Inc. New York, NY, USA, 2008

[16] Rong Pan, Martin Scholz: Mind the Gaps: Weighting the Unknown in Large-Scale One-Class Collaborative Filtering, In KDD '09 Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, Pages 667-676, ACM New York, NY, USA, 2009

[17] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N. Liu, Rajan Lukose, Martin Scholz, Qiang Yang: One-Class Collaborative Filtering, In Data Mining, IEEE International Conference on, 0:502–511, 2008

[18] Resnick, P., Varian, H.R.: Recommender systems. In Communications of the ACM, Volume 40, Issue 3, pages 56–58, ACM New York, NY, USA, 1997

[19] Dietmar Jannach: Finding Preferred Query Relaxations in Content-based Recommenders. In: 3rd International IEEE Conference on Intelligent Systems, pp. 355–360 (2006)

[20] McSherry, F., Mironov, I.: Differentially private recommender systems: building privacy into the net. In: KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 627–636. ACM, New York, NY, USA (2009)

[21] Schwartz, B., Ward, A.: Doing Better but Feeling Worse: The Paradox of Choice. ECCO, New York (2004)

[22] Goldberg, D., Nichols, D., Oki, B.M., Terry, D.: Using collaborative filtering to weave an information tapestry. Commun. ACM 35(12), 61–70 (1992)

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

[24] Jannach, D., Zanker, M., Felfernig, A., Friedrich, G.: Recommender Systems An Introduction. Cambridge University Press, 2010

[25] Gozde Özbal: A content boosted collaborative filtering approach for movie recommendation based on local & global user similarity and missing data prediction, Master Thesis, METU, September 2009

[26] Supporting people in finding information. Hybrid recommender systems and goal-based structuring. Mark Van Setten. Telematica Instituut Fundamental Research Series, vol. 016. Enschede, the Netherlands: Telematica Instituut, 2005 ISBN 90-75176-89-9

[27] Terveen, L., Hill, W., Beyond Recommender Systems: Helping People Help Each Other. In Carroll, J., (Ed.), HCI in the New Millennium. Addison Wesley, 2001

[28] Zhang, Y., Koren, J.: Efficient Bayesian Hierarchical User Modeling for Recommendation Systems, In SIGIR '07 Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, Pages 47 – 54, July 2007

[29] Vázquez, E.G.: Contribuciones al Modelado del Usuario en Entornos aptativos de Aprendizaje y Colaboración a través de Internet mediante técnicas de Aprendizaje Automático, Phd Thesis, Universidad Complutense de Madrid, 2002

[30] ChoiceStream Technology Brief, Review of Personalization Technologies: Collaborative Filtering vs. ChoiceStream's Attributized Bayesian Choice Modeling, http://www.choicestream.com, accessed July 12, 2012, 02:41

[31] Uchyigit, G., Ma, M. Y.: Personalization Techniques and Recommender Systems, World Scientific Publishing, London, 2008

[32] Boger, Z., Kuflik, T., Shapira, B. and Shoval, P. (2001). Automatic keyword identification by artificial neural networks compared to manual identification by users of filtering systems, Information Processing & Management, Vol. 37 (2), pp. 187-198.

[33] Kuflik, T., Shoval, P.: Automatic Generation of Content-Based User Profiles Compared to Rule-Based Profiles for Information Filtering, In IUI '03 Proceedings of the 8th international conference on Intelligent user interfaces, Page 317, ACM New York, NY, USA, 2003

[34] Abbattista F., Degemmis M., Fanizzi N., Licchelli O., Lops P., Semeraro G., and Zambetta, F.: Learning User Profiles for Content-Based Filtering in e-Commerce, In CiteSeer^{x}, 2002

[35] Turetken, O., Sharda, R.: Development of a fisheye-based information search processing aid (FISPA) for managing information overload in the web environment, In Decision Support Systems, Volume 37 Issue 3, Pages 415 – 434, Elsevier Science Publishers B. V. Amsterdam, The Netherlands, The Netherlands, 2004

[36] Ricci, F., Rokach, L., Shapira, B.: Introduction to Recommender Systems Handbook, In Proceedings of Mathematik für Ingenieure, ACM, 2011

[37] Golbeck, J.: Generating Predictive Movie Recommendations from Trust in Social Networks, In iTrust'06 Proceedings of the 4th international conference on Trust Management, Pages 93-104 , Springer-Verlag Berlin, Heidelberg, 2006

[38] Mladenic, D., Stefan, J.: Text-Learning and Related Intelligent Agents: A Survey, In Intelligent Systems and their Applications, IEEE, **Volume:** 14 , Issue: 4 , **Page(s):** 44 - 54 , 1999

[39] Mitchell, T.: Machine Learning. McGraw-Hill, New York (1997)

[40] Herlocker, L., Konstan, J.A., Terveen, L.G., Riedl, J.T.: Evaluating Collaborative Filtering Recommender Systems. ACM Transactions on Information Systems 22(1), 5–53 (2004)

[41] Schafer, J.B., Frankowski, D., Herlocker, J., Sen, S.: Collaborative filtering recommender systems. In: The Adaptive Web, pp. 291–324. Springer Berlin / Heidelberg (2007)

[42] Schafer, J.B., Konstan, J.A., Riedl, J.: E-commerce recommendation applications. Data Mining and Knowledge Discovery 5(1/2), 115–153 (2001)

[43] Deshpande, M., Karypis, G.: Item-based top-N recommendation algorithms. ACM Transaction on Information Systems 22(1), 143–177 (2004)

[44] Last.fm: Music recommendation service (2012). http://www.last.fm. Accessed July 15 01:28, 2012

[45] Bell, R.M., Koren, Y.: Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In: ICDM '07: Proc. of the 2007 Seventh IEEE Int. Conf. on Data Mining, pp. 43–52. IEEE Computer Society, Washington, DC, USA (2007)

[46] Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: Proc. of the 14th Annual Conf. on Uncertainty in Artificial Intelligence, pp. 43–52. Morgan Kaufmann (1998)

[47] Konstan, J.A., Miller, B.N., Maltz, D., Herlocker, J.L., Gordon, L.R., Riedl, J.: GroupLens: applying collaborative filtering to usenet news. Communications of the ACM 40(3), 77–87 (1997)

[48] Hill, W., Stead, L., Rosenstein, M., Furnas, G.: Recommending and evaluating choices in a virtual community of use. In: CHI ’95: Proc. of the SIGCHI Conf. on Human Factors in Computing Systems, pp. 194–201. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA (1995)

[49] Hofmann, T.: Collaborative filtering via Gaussian probabilistic latent semantic analysis. In: SIGIR ’03: Proc. of the 26th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, pp. 259–266. ACM, New York, NY, USA (2003)

[50] Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. Journal of Machine Learning Research 3, 993–1022 (2003)

[51] Zitnick, C.L., Kanade, T.: Maximum entropy for collaborative filtering. In: AUAI ’04: Proc. Of the 20th Conf. on Uncertainty in Artificial Intelligence, pp. 636–643. AUAI Press, Arlington, Virginia, United States (2004)

[52] Salakhutdinov, R., Mnih, A., Hinton, G.: Restricted Boltzmann machines for collaborative filtering. In: ICML ’07: Proceedings of the 24th international conference on Machine learning, pp. 791–798. ACM, New York, NY, USA (2007)

[53] Grcar, M., Fortuna, B., Mladenic, D., Grobelnik, M.: k-NN versus SVM in the collaborative filtering framework, In *Data Science and Classification*, p. 251—260, 2006

[54] Bell, R., Koren, Y., Volinsky, C.: Modeling relationships at multiple scales to improve accuracy of large recommender systems. In: KDD ’07: Proc. of the 13th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, pp. 95–104. ACM, New York, NY, USA (2007)

[55] Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: KDD'08: Proceeding of the 14th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, pp. 426–434. ACM, New York, NY, USA (2008)

[56] Takács, G., Pilászy, I., Németh, B., Tikk, D.: Major components of the gravity recommendation system. SIGKDD Exploration Newsletter 9(2), 80–83 (2007)

[57] Good, N., Schafer, J.B., Konstan, J.A., Borchers, A., Sarwar, B., Herlocker, J., Riedl, J.: Combining collaborative filtering with personal agents for better recommendations. In: AAAI '99/IAAI '99: Proc. of the 16th National Conf. on Artificial Intelligence, pp. 439–446. American Association for Artificial Intelligence, Menlo Park, CA, USA (1999)

[58] Ramakrishnan, N., Keller, B.K., Mirza, B.J.: Privacy Risks in Recommender Systems. IEEE Internet Computing. p. 54-62, 2001

[59] Herlocker, J.L., Konstan, J.A., Borchers, A., Riedl, J.: An Algorithmic Framework For Performing Collaborative Filtering. In Proceedings of the 22nd International Conference on Research and Development in Information Retrieval (SIGIR '99). (1999) Berkeley,California. ACM Press p. 230-237

[60] MacQueen, J.: Some Methods for Classification and Analysis of Multivariate Observations. In Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability. (1967) p. 281-297

[61] Johnson, S.C.: Hierarchical Clustering Schemes. Psychometrika, (1967) 32(3): p. 241-254

[62] Lam, S.K. Riedl, J.: Shilling Recommender Systems For Fun And Profit. Proceedings of the 13th international conference on World Wide Web. (2004) ACM Press: New York, NY, USA. p. 393-402

[63] Sarwar, B., Karypis, G., Konstan, J.A., Riedl, J.: Item-Based Collaborative Filtering Recommendation Algorithms. Proceedings of the 10th international conference on World Wide Web. (2001) Hong Kong. ACM Press p. 285-295

[64] Fouss, F., Renders, J.M., Pirotte, A., Saerens, M.: Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. IEEE Transactions on Knowledge and Data Engineering 19(3), 355–369 (2007).

[65] Mahmood, T., Ricci, F.: Towards learning user-adaptive state models in a conversational recommender system. In: A. Hinneburg (ed.) LWA 2007: Lernen - Wissen - Adaption, Halle, September 2007, Workshop Proceedings, pp. 373–378. Martin-Luther-University Halle-Wittenberg (2007)

[66] Bridge, D., G¨oker, M., McGinty, L., Smyth, B.: Case-based recommender systems. The Knowledge Engineering review 20(3), 315–320 (2006)

[67] Arazy, O., Kumar, N., Shapira, B.: Improving social recommender systems. IT Professional 11(4), 38–44 (2009)

[68] Sinha, R.R., Swearingen, K.: Comparing recommendations made by online systems and friends. In: DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries (2001)

[69] Massa, P., Avesani, P.: Trust-aware collaborative filtering for recommender systems. In: Proceedings of the International Conference on Cooperative Information Systems, CoopIS, pp. 492–508 (2004)

[70] Groh, G., Ehmig, C.: Recommendations in taste related domains: collaborative filtering vs. social filtering. In: GROUP '07: Proceedings of the 2007

international ACM conference on Supporting group work, pp. 127–136. ACM, New York, NY, USA (2007)

[71] Guy, I., Zwerdling, N., Carmel, D., Ronen, I., Uziel, E., Yogev, S., Ofek-Koifman, S.: Personalized recommendation of social software items based on social relations. In: RecSys '09: Proceedings of the third ACM conference on Recommender systems, pp. 53–60. ACM, New York, NY, USA (2009)

[72] Burke, R.: Hybrid Recommender Systems: Survey and Experiments, In User Modeling and User-Adapted Interaction, Volume 12 Issue 4, Pages 331 – 370, Kluwer Academic Publishers Hingham, MA, USA , 2002

[73] Pazzani, M. J.: A Framework for Collaborative, Content-Based and Demographic Filtering, In Artificial Intelligence Review - Special issue on data mining on the Internet, Volume 13 Issue 5-6, Pages 393 - 408 , Kluwer Academic Publishers Norwell, MA, USA, 1999

[74] C. Basu, H. Hirsh, and W. Cohen, Recommendation as classification: using social and content-based information in recommendation, in Proceedings of the 15th National Conference on Artificial Intelligence (AAAI'98) (Madison, WI), American Association for Artificial Intelligence, 1998, pp. 714–720.

[75] P. Melville, R. J. Mooney, and R. Nagarajan, Content-Boosted Collaborative Filtering for Improved Recommendations, Proceedings of the 18th National Conference on Artificial Intelligence (AAAI) (Edmonton, Alberta, Canada), 2002, pp. 187–192

[76] R. Burke, K. Hammond, and B. Young, The findme approach to assisted browsing, IEEE Expert 4 (1997), no. 12, 32–40.

[77] D. Billsus and M. Pazzani, User modeling for adaptive news access, User Modeling and User-Adapted Interaction: The Journal of Personalization Research 10 (2000), no. 2–3, 147–180.

[78] M. Zanker and M. Jessenitschnig, Case-studies on exploiting explicit customer requirements in recommender systems, User Modeling and User-Adapted Interaction 19 (2009), no. 1–2, 133–166.

[79] R. M. Bell,Y.Koren, and C.Volinsky, The BellKor solution to the Netflix Prize, Tech.Report http://www.netflixprize.com/assets/ProgressPrize2007 KorBell.pdf,AT&TLabs Research, 2007. Accessed July 21 15:07, 2012

[80] A. S. Joydeep, E. Strehl, J. Ghosh, and R. Mooney. Impact of similarity measures on web-page clustering. In Workshop on Artificial Intelligence for Web Search (AAAI 2000), pages 58–64. AAAI, 2000

[81] Strehl, A., and Ghosh, J. 2000. Value-based customer grouping from large retail data-sets. In Proceedings of the SPIE Conference on Data Mining and Knowledge Discovery, 24-25 April 2000, Orlando, Florida , USA

[82] Souvik Debnath, Niloy Ganguly, Pabitra Mitra: Feature weighting in content based recommendation system using social network analysis, In WWW '08 Proceedings of the 17th international conference on World Wide Web, Pages 1041-1042 , ACM New York, NY, USA , 2008

[83] Yolanda Blanco-Fernández, José J. Pazos-Arias, Alberto Gil-Solla, Manuel Ramos-Cabrer, Jorge García-Duque, Rebeca P. Díaz-Redondo, Ana Fernández-Vilas, Belén Barragáns-Martínez and Martín López-Nores: AVATAR: Advanced Telematic Search of Audiovisual Contents by Semantic Reasoning

[84] Smyth B. and Cotter P. A personalized television listings service. Communications of the ACM, 43(8):107.111, 2000.

[85] Gozde Ozbal, Hilal Karaman, Matchbook, A Web Based Recommendation System For Matchmaking, International Symposium on Computer and Information Sciences (ISCIS'08), Istanbul, Turkey, October 2008.

[86] Philip Bonhard, "Improving Recommender Systems with Social Networking", Proc. of Addendum of CSCW 2004, Nov. 6-10, Chicago, IL

[87] http://lucene.apache.org/core/, accessed July 03, 2012, 17:31

[88] Kantrowitz, M., Mohit, B., Mittal, V.: Stemming and its effects on TFIDF Ranking, In SIGIR '00 Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval, Pages 357 - 359 , ACM New York, NY, USA, 2000

[89] Manning, C. D., Raghavan, P., Schütze, H.: An Introduction to Information Retrieval, Cambridge University Press, 2009

[90] http://www.basarimobile.com/en/index.html, accessed August 2, 2012, 09:28 AM

[91] S. Berkovsky, T. Kuflik, F. Ricci, Cross-representation mediation of user models, User Modeling and User-Adapted Interaction 19 (2009) 35-63

[92] A. McCallum, K. Nigam, A comparison of event models for naïve Bayes text classification. Proceedings of AAAI-98 workshop on learning for text categorization, 1998, pp.41-48.

[93] G. Salton, C. Buckley, Term weighting approaches in automatic text retrieval. Information Processing and Management 24 (1988) 513-523

[94] M. Montaner, B. Lopez, J. L. de la Rosa, A taxonomy of recommender agents on the Internet, Artificial Intelligence Review 19(4) (2003) 285–330.

[95] M.J. Pazzani, A. Meyers, NSF Research Awards Abstracts 1990-2003. http://kdd.ics.uci.edu/databases/nsfabs/nsfawards.html, (2003). Accessed August 02 17:51, 2012

[96] http://snowball.tartarus.org/texts/introduction.html, accessed August 2, 2012, 11:53AM

[97] Evren (Kapusuz) Çilden, Stemming Turkish Words Using Snowball, December 2006, snowball.tartarus.org/algorithms/turkish/accompanying_paper.doc, accessed May 06 10:25, 2012

[98] Cleverdon, C. and Kean, M.: Factors Determining the Performance of Indexing Systems. Aslib Cranfield Research Project, Cranfield, England, 1968

[99] Heng Luo, Changyong Niu, Ruimin Shen, Carsten Ullrich, "A collaborative filtering framework based on both local user similarity and global user similarity", In Proceedings of 2008 European Conference on Machine Learning and Knowledge Discovery in Databases, Part I, 2008

[100] Kok-Seng Wong, Myung Ho Kim: Privacy-preserving similarity coefficients for binary data, In Computers & Mathematics with Applications, Elsevier, 2012

[101] BILLSUS, D. AND PAZZANI, M. J. 1998. Learning collaborative information filters. In Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98). C. Rich, and J. Mostow, Eds. AAAI Press, Menlo Park, Calif., 46–53, 1998

[102] Baker,Kirk: Singular Value Decomposition Tutorial, March 29, 2005, http://www.ling.ohio-state.edu/~kbaker/pubs/Singular_Value_Decomposition_Tutorial.pdf, accessed September 18 22:22, 2012

[103] Sarwar, Badrul M.; Karypis, George; Konstan, Joseph A.; Riedl, John T.: Application of Dimensionality Reduction in Recommender System - A Case Study, CiteSeer, IN ACM WEBKDD WORKSHOP, 2000

[104] K. R. Gabriel and S. Zamir. Lower rank approximation of matrices by least squares with any choice of weights. Technometrics, 21(4):489–498, 1979.

[105] N. Srebro and T. Jaakkola. Weighted low-rank approximations. In ICML, pages 720–727. AAAI Press, 2003.