RECOMMENDATION SYTEM ANALYSIS AND EVALUATION

A THESIS SUBMITTED TO THE GRADUATE SCHOOL OF APPLIED SCIENCES OF NEAR EAST UNIVERSITY

By
MINASE NETSEREAB TEKLEAB

In Partial Fulfillment of the Requirements for the Degree of Master of Science in Software Engineering

NICOSIA, 2019

MINASE ETSEREAB TEKLEAB

RECOMMENDATIONS SYTEM ANALYSIS AND EVALUATION

NEU 2019

RECOMMENDATION SYTEM ANALYSIS AND EVALUATION

A THESIS SUBMITTED TO THE GRADUATE SCHOOL OF APPLIED SCIENCES OF NEAR EAST UNIVERSITY

By
MINASE NETSEREAB TEKLEAB

In Partial Fulfillment of the Requirements for the Degree of Master of Science in Software Engineering

NICOSIA, 2019

MINASE NETSEREAB TEKLEAB: RECOMMENDATION SYTEM ANALYSIS AND EVALUATION

Approval of Director of Graduate School of Applied Sciences

Prof. Dr. Nadire Cavus

We certify that this thesis is satisfactory for the award of the degree of Master of Science in Software Engineering

Examining Committee in Charge:

Asst. Prof. Dr. Yöney Kırsal Ever	Head of the Department of Software Engineering, NEU
Assoc. Prof. Dr. Kamil Dimililer	Head of the Department of Automotive Engineering, NEU
Asst. Prof. Dr. Boran Şekeroğlu	Supervisor, Department of Information System Engineering, NEU

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

Name, Surname: Minase Netsereab, Tekleab

Signature:

Date:

ACKNOWLEDGEMENTS

This Master's thesis, Recommendation System Analysis and Evaluation is the concluding piece of my two-year Master's degree of SE with NEU.

The project took six months. For other researchers interested in the field of Analysis and Evaluation recommender system, I believe my work could be a good summary of the state-of-the-art research results.

The thesis could not be done without the help of many dedicated people. First I would like to thank Assist. Prof. Dr. Boran Şekeroğlu, my thesis supervisor, who provided timely support and invaluable feedback and ideas for this research.

I also would like to thank NEU for generously offering me the opportunity and scholarship to study in TRNC. This two-year international experience would definitely change my future. And last but not least, I would like to thank all my friends and family who helped me through the tough time of these two years and encouraged me to finish this work.

To my parents …

ABSTRACT

Recommendation systems are popularly discussed in research literature aimed at solving the problems of information overload in a variety of contexts and application fields. When developing such applications, there are a wide range of choices regarding what approaches, algorithms and techniques to employ.

In this thesis I will provide a detailed analysis of different recommender systems' techniques (Content-based, Collaborative and Hybrid), which have been proposed in the recent literature.

Finally, evaluation methods and metrics to measure the performance of those systems will be discussed. I will explore the properties and potentials of various metrics and protocols in recommendation engines which will serve as a compass for conducting research and practice in the area of recommendation engines. Furthermore, an experiment will be conducted to measure their effectiveness on two recommendation models using precision-recall metrics which is applied on offline public dataset.

Keywords: Evaluation; recommender systems; content-based filtering; collaborative filtering; hybrid filtering.

ÖZET

Tavsiye sistemleri, popüler olarak, çeşitli bağlamlarda ve uygulama alanlarında aşırı bilgi yükü problemlerini çözmeyi amaçlayan araştırma literatüründe tartışılmaktadır. Bu tür uygulamalar geliştirilirken, hangi yaklaşımların, algoritmaların ve tekniklerin kullanılacağına ilişkin çok çeşitli seçenekler vardır.

Bu tezde, farklı literatürde öne sürülen farklı tavsiye sistemleri 'tekniklerinin (İçerik tabanlı, İşbirlikçi ve Karma) tekniklerinin ayrıntılı bir analizini sunacağım.

Son olarak, bu sistemlerin performansını ölçmek için değerlendirme yöntemleri ve ölçümleri tartışılacaktır. Tavsiye motorları alanında araştırma ve uygulama yapmak için pusula görevi yapacak olan tavsiye motorlarında çeşitli ölçüm ve protokollerin özelliklerini ve potansiyellerini keşfedeceğim. Ayrıca, çevrimdışı kamu veri setine uygulanan hassas hatırlama ölçümleri kullanarak iki öneri modelindeki etkinliğini ölçmek için bir deney yapılacaktır.

***Anahtar Kelimeler*:** Değerlendirme; öneri sistemleri; içerik esaslı filtreleme; işbirlikçi filter; hibrit filtre.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

RS:	Recommendation System
CF:	Collaborative Filtering
CBF:	Content Based Filtering
IR:	Information Retrieval
GUI:	Graphical User Interface
PCA:	Principal Component Analysis
ML:	Machine Learning
RSSE:	Recommendation systems in software engineering
CBRS:	Content Based Recommender System
AI:	Artificial Intelligence
SVD:	Singular Value Decomposition
MSE:	Mean Square Error
RMSE:	Root Mean Square Error
MAE:	Mean Absolute Error
MSD:	Millions Song Dataset
PCA:	Principal Component Analysis
LSA:	Latent Semantic Analysis

CHAPTER 1

INTRODUCTION

The increasing significance of the internet as a platform for electronic and business transactions has served as a driving force for the advancement of recommendation systems technology (Aggarwal, 2016). The field of RS was appeared first when Tapestry was developed and implemented using collaborative filtering by (Goldberg, Nichols, Oki, and Terry, 1992) in 1992. As the RS field introduced, researchers studied the utilization of algorithms from machine learning (ML), an area of artificial intelligence (AI).

Nowadays, RSs are applied in numerous information-based organizations such as Google (Liu, Dolan, and Pedersen, 2010), Twitter (Ahmed, Kanagal, Pandey, Josifovski, Pueyo, and Yuan, 2013), LinkedIn (Rodriguez, Posse, and Zhang, 2012), Netflix and in the field of software engineering (Robillard, Walker, and Zimmermann, 2010).

Recommender system as explained by (Deshpande and Karypis, 2004), is a personalized information filtering technology used to predict whether a specific user will be interested in a particular item or to recognize a set of N items that will be preferred by a certain user (Prabha and Duraisamy, 2016).

Stored data, input data, and algorithm (Burke, 2002) are the basic building blocks of a recommendation system. According to (Bobadilla, Ortega, Hernando, and Gutiérrez, 2013) recommendation algorithms are classified into, collaborative filtering, content-based filtering and hybrid filtering.

Collaborative filtering (CF) requires information from the user on the item to start recommending items to the target user. Users express their interest on the item by giving certain level of rating to the item i.e. according their taste. The more they like the item the high rating they will give. So based the rating another items related to the previous item they preferred will be offered by computing the similarity with the item. Many researchers have put effort on CF to develop it. Consequently, it has been applied by various online-shopping sites. While this is true, CF approach has limitation such as cold-start and data sparsity.

Content-based filtering (CBF) in contrast to CF it does not need any previous information about the items. A user profile is created by taking the features of the item. Without the item contents the recommendation will not get into effect. Once the system has the user profile constructed, similarity metrics computes the similarity of the contents of items in the profile with the item contents in the database. That way it recommends new items to the user. As CBF depends on the item properties, there are conditions where the contents are not able to extract for instance image contents. In addition, it keep offering similar items. There is no item variety.

Hybrid filtering algorithms (Hybrid) was basically come to exist to deal the limitation of CF and CBF. For example, CBF can overcome cold-start of CF and CF can handle the overspecialization problem of CBF. Cascading (Burke, 2002) is one of the methods of combing algorithms which merges scores of other techniques with their weights.

The outcomes of the filtering approaches introduced on top are required to be evaluated for their performance. We need to know, how important the recommendation was to the user. For example for an e-commerce, is the company selling many items and the revenue of the company improved. And several factors has to be considered.

Evaluation protocols and metrics assists us on knowing the performance in various ways before the system commences its actual task. Different datasets are applied to conduct the process as the performance differs from one dataset to another. As we aim perfection by evaluation, the research area is yet a very challenging (Gunawardana, 2011; Herlocker, Konstan, and Terveen, 2004). There are several factors that contributes to the challenge:

- A scalability of dataset is one of the major factors. The algorithms performs distinctly for different datasets. The size of dataset also greatly influences the performance. The accuracy and speed of the algorithm reduces as the dataset size increments.

• We have various number of evaluation metrics with varied properties. Some of them contradict to one another. Many tradeoffs are faced. For example, when the precision of the recommendation system is improved its recall decreases.

• Some evaluation metrics requires different evaluation protocols. For instance, serendipity is tested using user study while accuracy prediction leverages offline method.

1.1 Motivation of the Work

In this modern era of technology the critical issue we are facing is information overload which is causing a lot of challenges retrieving relevant data. It is challenging separating relevant from irrelevant information. Virtual environments like the Internet become more and more intricate and rich while comparing with real environment in respect to the amount of information and its complexity. For the last twenty five years, recommender engines have been assisting and easing these complexity barriers by presenting the internet users information they are really interested in smartly.

Recommender Systems main goal is to assist users dealing with information overload as introduced in (section 1), finding or extracting relevant information from irrelevant in a vast space of resources. Research on the area of RS have been active field since the first recommendation system evolved and some books and articles that survey different algorithms and application domains have been published recently. However, these researches have not discussed in depth the different techniques utilized in Recommender System, and only some of them have reviewed the different types of evaluation process to assess the effectiveness of *RS*. Thus to narrow this gap, in this thesis i present introduction of recommendation systems in general, and then we focus on presenting details of the main techniques of RS and evaluation methods and discuss metrics from different perspectives that have been active in the research literature.

This thesis will directly provide help to academics and practical professionals to get idea about recommendation systems, how they work and implemented, what techniques are leveraged and how they are evaluated. Recommendation systems are taking over the e-commerce in particular. So inducing understanding on the user is a critical aspect as they have to put trust and use them in their day to day activities.

1.2 Research Question

This thesis will answer, What are the most used techniques in recommender systems, the main performance evaluation metrics and methodologies used in the recommender systems field and Which Recommendation system model (popularity and item based) is better from Information Retrieval (IR) perspective?

1.3 This Research aims and Contributions are to:

- present a systematic analysis of recommendation approaches and their implementation process;
- present highlights of the limitations and possible solutions of each techniques discussed;
- it systematically examines the recommender systems evaluation metrics from three perspectives and
- finally conduct an experiment comparing item-based to popularity-based recommendation models.

1.4 Structure of the Thesis

This thesis includes of five chapters. Chapter 1, introduces a short background to the Recommendation engines, describes shortly the recommendation systems headlines and gives an overview about the objective and the structure of the thesis.

In Chapter 2, the literature review about recommendation engine algorithms and evaluation metrics is presented.

In Chapter 3, this thesis first analyzes the three categories of RSs namely CBF, CF and hybrid filtering (Adomavicius and Tuzhilin, 2005) and try to present the findings in an easy to digest manner, in order to provide a concrete understanding of available approaches for potential users.

In Chapter 4, the thesis first introduces recommendation system performance evaluation protocols, and highlights their pros and cons. Secondly; the thesis discusses three perspectives of evaluation metrics of recommendation systems from the perspectives of information retrieval, human-computer interaction, and machine learning. Finally, perform a quantitative comparison on two RS models (Item-based and popularity-based) built on a real word MillionsSong dataset.

In Chapter 5, this thesis is concluded by providing answer to the research questions. The future work including suggestions for the further development is summarized.

CHAPTER 2

LITERATURE SURVEY AND RELATED WORKS

2.1 Recommendation Systems

Recommendation system is generally described as a system that offers suggestion or recommendation for subjects to deal with the complex information overload (Rashid, Albert, Cosley, Lam, and McNee, 2002) and in the area of online shopping, assists users by finding items from a database that are similar to their interests and preference (Schafer and Konstan, 1999). Recommender systems provides users with their individual tastes and services of recommendation (Isinkaye and Folajimi, 2015) which overcomes the problem of retrieving users' needs due to information overload. There are different ways of building recommendation systems utilizing techniques such as collaborative algorithm, content-based algorithm or a combination of both hybrid algorithm (Acilar, 2009; Jalali, Mustapha, Sulaiman and , 2010).

2.2 Filtering Techniques

2.2.1 Collaborative Techniques

This technique suggests items to users by searching like-minded subjects with identical preference based on their preference it present suggestions to the target user which is referred item-based. Various application areas have employed CF approaches. Existing CF approaches generally categorized into: model-based and memory-based (Adomavicius and Tuzhilin, 2005). Neighborhood-based is splitted into, user-based (Huang, Wang, Liu, Ma, and Chen, 2015) and item-based (Shi and Larson, 2010), which makes predictions based on historical ratings related to similar item or users. On the other hand, model-based methods uses vectors to represent the items and users in a vector space. A dimensionality reduction technique, Matrix factorization, a well-performing approach, latent factor models used (Weimer, Karatzoglou and Le, 2007) and proposed Co-Rank. ListRank-MF provided by (Shi and Larson, 2010) creates features with MF.

News-based system, GroupLens, is one of the CF applications which suggests users articles from a massive news dataset. Topic diversification algorithms are used by Amazon which bettered its predictions (Huang, Wang, Liu, Ma, and Chen, 2015). The Application leverages CF technique to handle the problem of scalability by creating a matrix of related items offline using item-item matrix. The application predicts items to the user that matches to those already bought. However, collaborative methods has limitations such as ramp-up (Montaner, López, and de la Rosa, 2002), scalability and sparsity issues.

2.2.2 Content Based Filtering

CBF techniques match item-content to user features. CBF presents prediction by only considering the user's features it does not regard other user's interests unlike to collaborative techniques (Wang, Sun, and Gao, 2014).

Fab an example of CBF algorithm mostly depends on various users' ratings in order to form a training data. Some other recommenders like Letizia (Letizia, 1995) use CBF to assist users to find the information that interest them on the Internet. The application adopts a GUI which enables customers searching the web; it tracks the users' browsing pattern in order to suggest web pages that may like. Similarly (Pazzani, 1999) used Naive Bayesian classifier to build an intelligent agent. The system has the capability of providing training instances to the user by rating several web sites as important or not.

Regardless the success of CBF technique, it suffers from several limitations. Limited feature extraction, over-specializing predictions and data-sparsity (Adomavicius and Tuzhilin, 2005) can be listed. Such limitations affect the accuracy of predictions.

2.2.3. Hybrid Filtering

The idea of combining recommendation algorithms, hybrid filtering, was proposed to mitigate the limitations identified and to improve the accuracy and performance of recommendations (Adomavicius and Tuzhilin, 2005). Doing so, the strength is harnessed while leveling out their corresponding weaknesses (Al-Shamri and Bharadwaj, 2008). (Mican, 2010) classified into hybrid filtering into seven types; weighted, feature-augmentation, mixed, feature-combination, switching, cascading and meta-level based on their operations.

Most widely used hybrid techniques are built by combining CF and CBF, their output is aggregated later or adding CBF to CF features or vice versa. Ultimately, a model that integrates features of both the techniques could be designed (Ziegler and Lausen, 2004). A simple hybrid merging characteristics of CF with CBF together was proposed by (Cunningham, Bergmann, Schmitt, Traphoner, and Breen, 2001).

Cascade hybrid technique was recommended by (Ghazantar, 2010), combining the ratings, properties and demographic data of items address the sparsity as well as cold-start issues. Hybrid CF technique proposed by (Ziegler and Lausen, 2004) generates profiles by applying the technique super-topic score and topic diversification that exploits bulk taxonomic information which in return overcomes sparsity limitation of CF.

2.3 Recommendation System Evaluation

Various evaluation on many recommendation techniques using distinct dataset was conducted by (Breese, John, and David Heckerman, 1998). The experiments on that research paper are a corner stone to the current research literature.

The research done by (McNee and Riedl, 2006) reveals, that accuracy metrics are not sufficient to for choosing the right algorithm. The researchers highlighted considering the non-accuracy metrics such as serendipity of the items being recommended. An extensive study metrics targeting for measuring CF recommendation system was provided by (Herlocker, 2004). An experiment was also done on the similarity of various metrics in a perceptive way and finally decided that the analyzed metrics can be classified in to three main classes.

Some researchers like (Herlocker, 2004) do not accept MAE as a metric evaluating recommendations. They backed their idea by giving a similar example as a user's rating does not mean the user is probably to listen a music. Other researchers also advice considering the purpose of the recommendation algorithm in general. For instance (Del Olmo and Gaudioso, 2008), built a recommendation system framework that splits the recommendation system into two parts, filter and guide, for calculating predictions distinctly. And suggested to employ metrics that focus on the fact whether the recommendations provided by the system are actually found to be relevant to the users' needs and on the RS's goal.

(Cremonesi, Turrin, and Lentini, 2009) also proposed an evaluation approach for CF RSs. A further research by (Celma and Herrera, 2008) applied accuracy metrics and classification metrics for comparing two CF techniques using MovieLens1 dataset. Limitations and challenges of RS evaluation is discussed on a paper (Herlocker, 2009). It also focuses on usage of methodologies, dataset and metrics.

(Kohavi, Longbotham, and Sommerfield, 2009b) presented comprehensive study on evalution and provided a hands-on guide for carrying experiments on a web. In the next paper Crook et al. advices to emphasize on the importance of evaluation criteria's that meet the business goals (Crook, Frasca, and Kohavi, 2009), in their book encompass a section that overviews RS evaluation (Jannach, Zanker, and Felfernig, 2010). Similarly, Shani and Gunawardana, has contributed very insightful RS evaluation chapter to (Ricci, Rokach, and Shapira, 2010) handbook and outline the necessary aspects in conducting offline, online and user-study experiments.

In the literature, Information retrieval is another valuable source of evaluation metrics and measures. Basically it is aimed at providing relevant search results and contributes metrics for RS evaluations (for example (Measures, 2009)). Davis and Goadrich depicts that there is a deep correlation between Receiver Operator Characteristic (ROC) space and Precision-Recall (PR) space (Davis and Goadrich, 2006).

CHAPTER 3

ANALYSIS OF RECOMMENDATION SYSTEM TECHNIQUES

3.1. Recommendation Systems

Recommendation Systems (RS) intend to suggest a user or a group in a system to select or purchase items from a large number of item or information space (Aggarwal, 2016). Methods or algorithms adopted from the fields of ML, AI, and statistics are widely used in RS. Amazon for instance sorts and suggests books by employing ML. RS also contributes a significant role helping users when they are having a problem of deciding which item to select from a mass of items (Ricci, Rokach, Shapira, and Kantor, 2011), assisting users to maximize profits (Prabha and Duraisamy, 2016) or minimize risks (Said and Bellogín, 2014).

Research on recommendation systems has been going on both on academic and industry for almost twenty five years now, but with the increase in the number of e-commerce applications, online users, vendors and increasingly complex products and services, the demand for new intelligent recommendation techniques has also increased linearly.

Recommendation systems development varies from domain to domain and the type data to work on. For instance, five-star is used in Netflix, like/dislike in Facebook and soon. Which means the user feedback is recorded into a data source in such a way. The data filtering process aiming at finding the matching pairs also differs.

Generally, all recommendation engines apply a similar process (Hiralall, 2011) to offer recommendations to a target user, as illustrated in Figure 3.1.

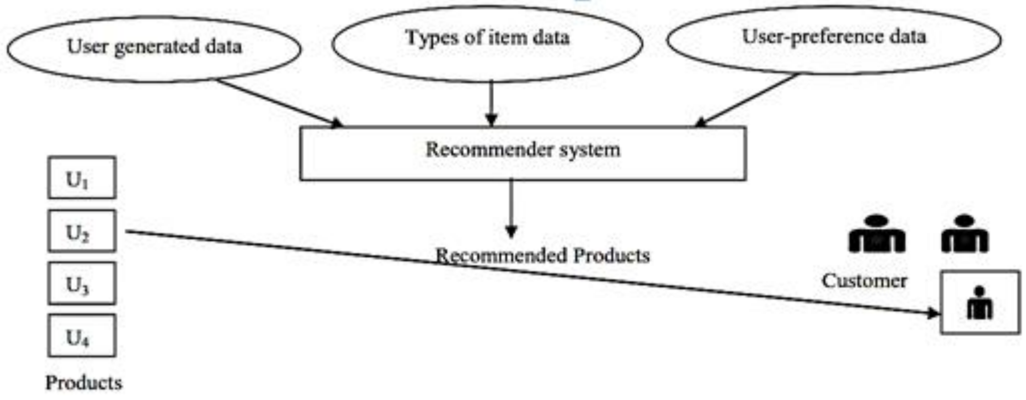


Figure 3.1: Framework of Recommendation Process

(Adomaviciu and Tuzhilin, 2005) classified Recommendation systems into three namely collaborative filtering, content-based, and hybrid filtering (see Figure 2.1), based on the information utilized to provide the suggestions. Detail RS techniques analysis is presented in section 3.1

- RSs with a CF approach measure the like mindedness of two users by comparing their inclination for items which they have evaluated. The intuition is similar-users have similar-item rating. This degree of similarity is then exploited while choosing the set of users whose views influence the final recommendations. Thus similarity computation is the crucial part of CF process. For example, by getting access to user profiles in an online movie database, the RS has get access to all of the person records, including the age, country, city, and films purchased. Based on this information, the system can identify users that share the same music preference, and then suggest movies purchased/watched by similar users.

Generally, CF-based techniques suffer from new user or item as it depends on a history of ratings of the user/item to compute the similarities, for the determination of the neighborhood.

- RSs with a CBF depends on item attributes to accomplish the recommendation process. For instance, a user is on flight-reservation web site to book flights searching flights to a certain destination. The system will ask the user to provide attributes such as from-to airports and calendar. The system then matches these user flight-attributes to the flight-attributes in the flight database and present flights exactly to the attributes or similar to them. Different types of algorithms are used to find the similarity between items. The commonly used are; Term Frequency Inverse Document Frequency (TF/IDF) (Mooney and Roy., 2000), Naïve Bayes Classifier, Decision Trees, and Artificial Neural Networks (ANN.

- The third classification, Hybrid recommendation systems combine two or more recommendation techniques to handle their unique limitation and advantage from their unique strength. Netflix is a well-known model of hybrid RSs. The application provide suggestions by analyzing the watching and searching habits of related users (i.e., CF) and by recommending movies that share traits with movies that a customer has rated highly (CBF). CF and hybrid filtering recommendation system require data from the user prior to presenting recommendations. To achieve such task, feedback from users can be gathered using explicit or implicit methods.

Explicit feedback: This type of feedback is given directly by the user through ratings. The most common example is when a user rates a watched movie on a scale from 1 to 5 or when users express their preferences by like/dislike on Facebook. The system therefore receives an explicit preference score for a given user-item pair, based on which a ranking of items can be determined.

Implicit feedback: is collected implicitly from various user interactions on a website, such as product page views, purchases, or additions to cart. Monitor user click and keystroke logs. The feedback that the system receives when such an event is registered as a result of successful recommendation takes the form of values. This implies binary preference, for example, value = 1 if bought, value= 0 if not bought.

In addition to the commonly used recommendation approaches, in which users are provided with items that might like, recommendations can be done in other ways. Trust-based recommendations (Bobadilla, Ortega, and Hernando, 2013) take into consideration the trust relationship that users have between them. A trust relationship is a link in a social network to a friend or a following connection. Suggestions based on trusted friends are worth more than those that do not have trust links. Context-aware recommendations (Melville and Sindhawani) completely depend on the context or the situation the user is in.

A context is a set of information that characterize the current activity or state of the user, such as the user's current location (museum, church, office), or the current activity (idle, running, cleaning). Despite their remarkable role, context-aware require high computation time to process the contextual dataset which makes them very challenging in the research area. Another context based approach is, risk-aware recommendations (Bruke, 2002), considers a state where critical information is available such as patient's vital symptoms. As its name indicates, it is sensitive to risk because a wrong decision may risk/threaten a user's life or cause damage. For instance, recommending pills the patient should take or stocks the customer should buy or, sell.

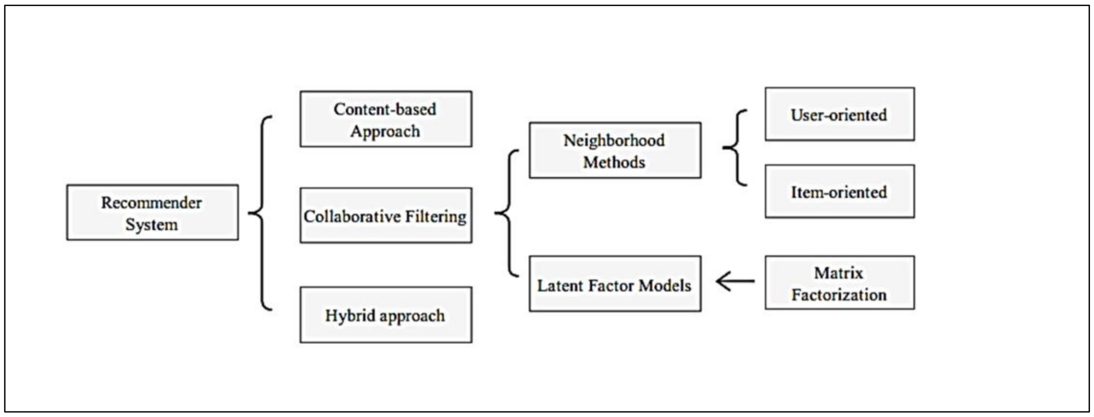


Figure 3.2: Traditional Recommendation Systems approaches main category

3.2 Content Based Filtering

CBF is also called cognitive Filtering. Cognitive filtering systems were basically designed to filter relevant content and suggest from items mainly text-based like e-mail messages. It is successfully implemented on text mining related system.

Nowadays, CBF are popularly used in the area of RS. These systems make predictions on the basis of past user selections history (Bobadilla, Ortega, Hernando, and Gutiérrez, 2013; Lu, Dianshuang, Mao, and Wang, 2015; Lu, Medo, Yeung, Zhang, and Zhang, 2012; Pujahari and Padmanabhan, 2014; Wintrode, Sell, Jansen, Fox, and Garcia-Romero, 2015).

In cognitive system first a user profile (Onoda and Murata, 2006) is created based on the information provided by the user such as age, gender, and soon. A profile for the item the user liked or watched also generated. Related items to profile generated are then recommended to the user (Lops, Marco, and Semeraro, 2011). Pandora.com is one of the many applications of a CBRS, as it profiles songs by attributes, and then recommends users or listeners with songs that are similar to those the user liked in the past. It does so, by matching or searching the features within songs not user profile of neighbor candidates.

Researchers considers cognitive systems as Information retrieval (Balabanovic and Shoham, 1997) and generally It employs techniques from Information Retrieval such as classification, clustering and text analysis (Mooney and Roy, 2000). For instance, in NewsWeeder (Lang, 1995), documents in the rating categories are represented by word vectors using TF - IDF, and then each user is given a weight for each category by averaging tf-idf word vectors.

Skyskill and Newsweeder are most common CBF based recommendation systems. Skyskill recommendation system recommends Web documents (Pazzani, 1999) and Newsweeder suggests news articles (Lang, 1995). And (Zhang, Callan and Minka, 2002) proposes an application which identifies relevant documents with new information and without by implementing a Bayesian approach (discussed in later section).

Steps in content-based RS (Pradeep and Bhaskar, 2018), consider, a user is on a book recommender system, the Recommendation System will analyse the content of that book aiming at finding other similar books it can offer as follows:

1) Initially, the books are represented in the form of attributes or descriptors the same as a relational database. Books can be described by Genre (Science fiction, Comedy, Drama), Author's Name, Publisher, Published-date, words used in the book.

2) Represent the values for each descriptor by a vector in a multidimensional vector space.

3) Similarly, a user profile is created for each user based on his purchase history, explicit ratings, and reviews.

4) So now the user is represented with attributes like the genre (List of books they prefer), Author's name (List of books they bought of an Author).

5) Finally map each user to a book similar to his taste using similarity metrics. In CBF Cosine similarity is generally used, which finds the similarity or cosine distance between the item vector and profile vector. Assume we have profile vector *u* and item vector *v*, then their similarity is (See Equation 3.7 and 3.8).

Based on the cosine value, which ranges between -1 to 1, the items are arranged in descending order and one of the two below approaches is used for recommendations:

- Top-n approach: the user is recommended the first top *n* items where the *n* elements are decided by the business.

- Rating scale approach: in this technique a threshold is set and all the items on top of the threshold are offered to the user.

A major drawback of this algorithm is it over-specialize items presented to the user. It will never recommend products which the user has not bought or liked in the past. For instance, If a user has watched or liked only romantic movies in the past, the system will recommend only romantic movies. As such, it missed a feature called Serendipity, which is the main feature CF (discussed in the next section). Content based filtering approach framework as shown in Figure 3.3.

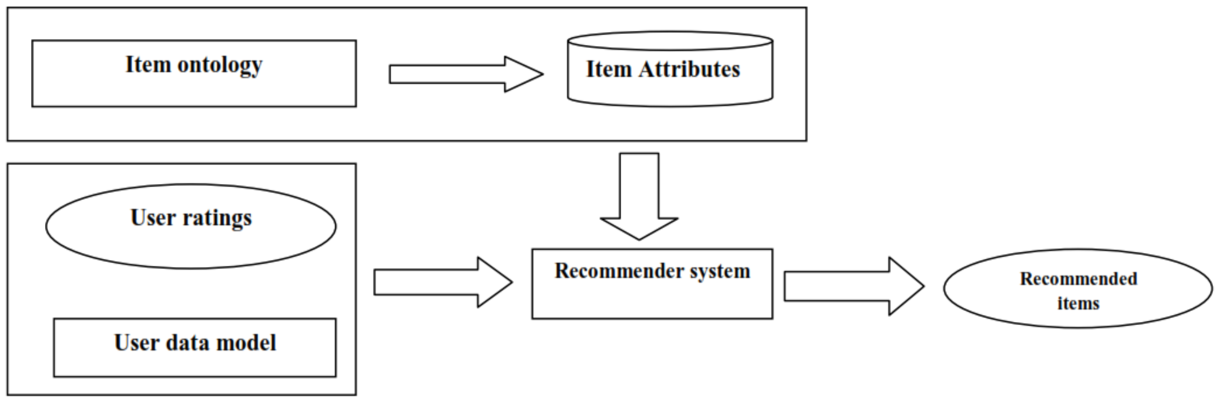


Figure 3.3: Frame-work of content-based approach (Aamir and Bhusry., 2015)

3.2.1 Popular Content-Based Filtering Algorithms.

Various algorithms are being used in content-based models. These techniques finds similarities in the descriptions that can be leveraged to differentiate highly liked items from others (Robles, Larranaga, Pena, Marbán, Crespo, and Pérez, 2003). Generally algorithms are adopted from IR and ML as they are well-suited for text categorization (Sebastiani, 2002). The most used algorithms are reviewed in the section below.

3.2.1.1 Term-Frequency - Inverse Document Frequency (TF - IDF)

TF-IDF, as its name indicates it measures the frequency of a term in documents. The more the term is repeated on the text, more it becomes important. However, the importance reduces if it occurs frequently on the corpus. The weight (Baeza-Yates, and Ricardo, 1999) of a particular term in a text is computed (Chakrabarti, 2002) as,

$$TF(x) = \frac{freq(x)}{sum\ of\ all\ terms - freq(a)} \tag{3.2}$$

Where, $freq(x)$ is the frequency of term *x* in a document.

IDF is a measure that works together with TF. Its main goal is to reduce the weight of a term that appears in the corpus frequently. The importance of the term decreases if it shows up in the collection of documents more often. So it should be assigned a small weight. The IDF Equation is given by,

$$IDF(i) = log \ \frac{N}{n(i)} \tag{3.3}$$

Where N is the corpus size, $n(i)$ the number of documents in the corpus where i occurs.

Therefore, TF-IDF is given by Equation 3.4 :

$$TF - IDF(i,j) = TF(i,j) * IDF(i) \tag{3.4}$$

3.2.1.2 Naive-Bayes Classifier

Naive Bayes is a probabilistic approach to inductive learning, and belongs to the general class of Bayesian classifiers, and its text classification performance was reported by (Maron, 1961). It is treated as one of the exceptionally well-performing text classification algorithm and in consequence many recent works have frequently adopted the algorithm (McCallum, Rosenfeld, Mitchell, and Ng, 1998; Mitchell, 1997; Nigam, McCallum, Thrun, and Mitchell, 1998). It generates a probabilistic model based on previously observed data. So when two random variables are jointly distributed with the value of one unknown then the probability of the other variables is calculated applying Bayes-rule.

The probability, *P(c/d),* is calculated using Bayes theorem Equation 3.5 (Paquale and Semeraro, 2011) as, the probability of c given d, $P(c/d)$, is given by the product of the probablity of a document in class C , $P(c)$, to the probability of d given c, divided by the probability of a document in class d;

$$P\left(\frac{c}{d}\right) = \frac{P(c)P\left(\frac{d}{c}\right)}{P(d)} \tag{3.5}$$

Where, *P(c)* is the probability of a document in class *C*.

To classify the document *d*, the class with the highest probability is chosen:

$$C = argmax_{cj} = \frac{P(c_j)P(d/cj)}{P(d)} \tag{3.6}$$

3.2.1.3 Decision Tree Rule Learner

Decision tree is a data mining technique which also widely adopted by recommendation systems. As its name signifies, it applies a tree like structure for visualizing the classification problem into nodes or new trees. The algorithm recursively (Quinlan, 1986) builds new classes by splitting the training set, in our case the text documents, until the new classes consists only the instances of a single class that is the word or phrase. The algorithm commonly employs entropy as for selecting the most important attributes (Yang and Pedersen, 1997) .

This technique is being widely researched for the use with structured or restricted data. However, many disagree the usage of decision tree bias for unstructured or unrestricted textual classification tasks (Pazzani and Billsus, 1997). As a result, the splitting criteria i.e. information-theoretic employed the algorithm and the inductive bias are useful for small trees with few tests. But, usually textual classification task consists a lot of relevant attributes (Joachims, 1998). On this case, the technique is less applicable as it poorly affects the performance of the textual classification process. Decision trees are easy and understandable when only applied on small structured which improves the performance of content-based models.

3.2.2 Merits and Demerits of CBF

Merits:

- Recommendations are generated using the user preferences alone rather than the user community
- Can be employed in real time as the model does not need to load all the data for processing or generating recommendations
- High accuracy compared to CF as product content is utilized rather than just rating information
- Easily handling of the "cold-start" problem

Demerits:

- limited content analysis
    - domains other than text documents, for example , images are difficult to extract their feature and represent them using keywords.
- Overspecialization
    - no serendipity: the system will keep recommending the user items that are similar to those already rated.
    - diversity of recommendations is needed: the RS keeping recommending similar items

3.3 Collaborative Filtering Recommendation System

Since the first recommendation system Tapesery built in mid 1990s, CF has been the most well-performing and often employed filtering technique and researched (Sarwar, Karypis and Konstan , 2000; Sarwar, Karypi, and Konstan, 2001; Yang and Liu, 1999). Collaborative techniques handles well some of the limitations of the content-based technique discussed in the previous section 3.2. For instance, it works well with items in which CBF has a problem of extracting the items content such as movies by getting other users feedback. The strong side of CF is it depends on the quality of an item not on content. This enables to break the barriers of serendipity and limited content analysis problem of CF.

CF RSs works on a dataset of users and item rating. The target user who is expecting recommendation is known as active user. The active-user is recommended items from similar users by simply searching the database. Based on those users that have similar taste, it will recommend the items like in Amazon.com and www.movielens.org. These e-commerce sites in turn increases the customers' loyalty and sales(Schafer, and Konstan, 1999).

The major tasks being performed in collaborative filtering are user-user or item-item. The workflow (Good, Schafer, Konstan, Borchers, and Sarwar, 2008) for item-item:

1) Expressing a User's preference by rating the items.
2) Finds the users with most similar taste by mapping their rating with other users rating.
3) Then finally, the most highly rated by users are recommended by the system.

And in the user-user:

1) Searching the user neighbors.
2) Discovering the interests of the neighbors of a active customer.

In CF technique, user neighbors are created by looking at the user’s purchasing history and computing their similarity. Then the prediction is performed in either of these two ways, explicit such as item-rating or implicit for instance monitoring the user’s behavior towards the item .

CF employs various approaches such as

- Cosine angle (Qamar, 2010) or neighborhood based algorithm (KNN (DENIYI , and WAI, 2014)) used to compute the cosine distance between two users that is item-item approach.
- Pearson coefficient (Rodgers and Nicewander, 1988) performs well in computing the similarity between two users that is user-user technique.
- For other techniques CF uses Bayesian techniques (Deerwester and Dumais, 1990), matrix factorization (SVD (Golub and Kahan, 1965)), association rules, PCA (Pearson, 1901), and Latent Semantic Analysis (LSA (Golub and Kahan, 1965)).

CF is further classified into:

3.3.1 Memory-Based Filtering

Professor L.Herlocker, from University of Minnesota, proposed this algorithm in late 1990s. Memory-based algorithms employ the complete user-item database loaded in the memory to create a prediction. These approaches use methods borrowed from statistics to get neighbors for the active user. Neighbors are users that either bought similar item or rated items that are different equally.

Neighborhood-based methods works for almost any types of recommendation like books, music, movies and products without the need of feature selection. Nevertheless, it suffers from some limitations like; Cold start (first- rater) problem, Sparsity (huge number of users with little item ratings), and Popularity bias problem.

The next subsections discusses the two subcategories of memory based approaches, Item-based and user based. They follow almost similar intuition, user-based look for users who gave similar credit for an item and item-based for an “item rated similarly by various users".

3.3.1.1. User-Based Collaborative Technique

User based approach searches and determines similarity of users who provided the same rating for items using measures known as similarity metrics (Isinkaye, Folajimi, and Ojokoh, 2015) (discussed in section 3.2.1.3).

Here is the basic workflow of this strategy, let us consider a user *u* and neighborhood of *u as v*,

1. Find a user/ group-of-users whose like/s and dislike/s are similar to the defined user *u*. For example, *u* likes the same movies, the user/ group-of-users like and *u* dislike the movies the user/ group doesn't like. This user/ group-of-users is called neighborhood of *u*.

2. After finding the *v*, then the step following is finding the set of items/movies which are not bought/seen by user *u* but are liked by *v*. Then, recommend those items to user *u*.

User-based approach has a scalability drawback. It is a situation exhibited when the user-matrix contains a lot more users comparing to items, and so plenty of computation time is employed which makes searching much harder over users. For instance, in youtube.com the number of users increases at a very high rate in contrast the items uploaded (Ekstrand, Riedl and Konstan, 2010). This scalability issue leads to the evolution of item-based collaborative approach. Whilst, the domain where we apply these approaches also it determines. In the context of news, for instance, user-based performs exceptionally well.

3.3.1.2 Item-Based Collaborative Technique

Researchers of university of Minnesota proposed this technique in 2001 (Pronk, Verhaegh, Proidl, and Tiemann, 2007) and then adopted by Amazon.com (Linden, Smith, and York, 2003). It is based on a Computation of similarity as user-based but between items. Item-item approach use Pearson Correlation (Xiaoyuan and Taghi, 2007) to calculate the similarities among items and NKK offer prediction to the active user .

In this system finding similarity (Sarwar, Karypis, and Konstan, 2001) among items is the most complex step. To compute the similarity and generate the prediction, the utility-matrix or the database of users has to be scanned now and then which is impractical in real life applications. These solutions below will somehow easy the problem:

1. Find out the nearest items/users in a regular manner.
2. Use clustering to pre-group items into groups and limiting the search space to a cluster.
3. Dimensionality reduction techniques can also be used to reduce the search space.

3.3.1.3 Determining Similarity

Similarity metrics play a critical role in recommendation systems which measure the similarity between user-user or item-item. In this section I will present the most popular approaches, Cosine Similarity and Pearson Correlation (Amatriain and Xavier, 2011; Breese, John, David Heckerman and Kadie, 1998) which compute the similarity that is used as input for getting the user neighbors.

Pearson Correlation approach: The Pearson correlation was first suggested as an appropriate similarity metric in the Group-Lens recommender system project in 1994 (Konstan, Miller, Maltz, Herlocker, Gordon and Riedl, 1997). It is the most used approach for user-user collaborative techniques (Breese, John, David Heckerman and Kadie, 1998; Herlocker, Konstan, and Reidel, 2002). In Pearson Correlation, we scale the similarity from -1(low correlation) to +1 (high correlation). Zero is for no correlation. Let $I_{u,v}$ is the set of items rated by users u and v, r_{ui} and r_{vi} are user-rating of u and v for item i, *respectively*. And $\overline{r}_u$ and $\overline{r}_v$ are the mean user rating of *u* and *v*, respectively, the pearson correlation $p(u,v)$ between items i and j is given by the Equation 3.7:

$$p(u,v) = \frac{\sum_{iEI_{uv}} (r_{ui} - \bar{r}_u \ )(r_{vi} - \bar{r}_v) \ ))}{\sqrt{\sum_{iEI_{uv}} (\ ((r_{ui} - \bar{r}_u)^2) \sum_{iEI_{vr}} \ (r_{vi} - \bar{r}_v \ )^2)}} \tag{3.7}$$

Similarly the pearson correlation $p(i,j)$ between items i and j is given by the Equation 3.8:

$$p(i,j) = \frac{\sum_{uEu_{ij}} (r_{ui} - \bar{r}_i \ )(r_{ui} - \bar{r}_j) \ ))}{\sqrt{\sum_{uEu_{ij}} (\ ((r_{ui} - \bar{r}_i)^2) \sum_{uEu_{ij}} \ (r_{ui} - \bar{r}_j \ )^2)}} \tag{3.8}$$

Cosine-based approach: This metric is most suited on Item-Based collaborative approaches (Jannach, Zanker, Felfernig, and Friedrich, 2011). As in Pearson correlation, it uses similar scaling.

In vector form, the Cosine angle is given as shown in Equation 3.9:

$$\cos(\bar{a}.\bar{b}) = \frac{\bar{a}.\bar{b}}{\|\bar{a}\|^2 * \|\bar{b}\|^2} \tag{3.9}$$

Let $sim(u,v)$ is the similarity of users *u* and *v*, the user based Equation 10 is:

$$sim(u,v) = \frac{\sum_{i=1} \ r_{ui} r_{vi}}{\sqrt{\sum_{iEX_u} r^2{}_{ui} \sum_{iEX_v}^{n} r^2{}_{vi}}} \tag{3.10}$$

And $sim(i,j)$ is the similarity of items *i* and *j*, the item based formula can be depicted by Equation 3.11:

$$sim(i,j) = \frac{\sum_{uEu_{ij}} \ r_{iu} r_{ju}}{\sqrt{\sum_{uEu_i} r^2{}_{iu} \sum_{uEu_j}^{n} r^2{}_{ju}}} \tag{3.11}$$

Prediction: Nearest neighbor is the most commonly used prediction algorithm of neighborhood based technique. (KNN) is the de-facto algorithm which is the easiest and understandable and well-performing method. In the following section KNN is summarized.

KNN User-based prediction: Let, user v and u are neighbors i.e. similar, in order to predict item i to user u, the neighbors' of u , v, ratings r_{vi} on i should be analyzed. Therefore item-based equation, $pred(u, i)\ is$:

$$\mathrm{pred(u, i)} = \frac{\mathrm{v \in neihgbours(u) r_{ni}}}{\mathrm{number\ of\ neighbours}} \tag{3.12}$$

KNN item-based prediction:

$$\mathrm{pred(u, i)} = \frac{\mathrm{j \in\ createditems(u) itemsim\ (i, j) * r_{ui}}}{\mathrm{j \in\ createditems(u) itemsim\ (i, j)}} \tag{3.13}$$

3.3.2 Model-Based Algorithm

Techniques adopted from ML, linear algebra and data mining approaches are used for searching the patterns on the training-set and make predictions for real time data to develop model-based CF algorithm. It matches the model for the given rating matrix to issue the recommendations. The method was proposed to deal the limitations of memory-based methods. In contrast with memory based CF, the entire dataset is not used to present predictions for real data.

One of the well-performing techniques used in the recent literature is matrix factorization (Schelter, Boden, Schenck, Alexandrov, and Markl, 2013; Song, Cheng, and Lu, 2015; Zhuang , Chin, Juan, and Lin, 2013). This is commonly implemented through techniques such as Stochastic Gradient Descent or Alternating Least Squares (Gemulla, Nijkamp, J Haas, and Sismanis, 2011; Koren, Bell, and Volinsky, 2009; Schelter, Boden, Schenck, and Alexandrov, 2013; Zhou, Wilkinson, Schreiber, and Pan, 2008). Generally, it out-performs memory-based approach in terms of speed and accuracy. Yet, Matrix factorization needs to be recalculated whenever a new rating is entered. Thus it is expensive to compute and time consuming.

Dimensionality Reduction techniques reduce the problems of sparsity (Sarwar et al. 2009) in RS databases, for instance, Principal Component Analysis (PCA), Singular Value Decomposition(SVD), Probabilistic Matrix Factorization (PMF), Latent Semantic Methods, Lustering and Matrix Completion Technique (Isinkaye, Folajimi, and Ojokoh, 2015). Below we described the most widely employed, PCA, SVD and PMF.

3.3.2.1 Principal Component Analysis (PCA)

This is a powerful technique to reduce the dimensions of the data set, this is considered a realization of the MF (Francesco, Rokach, and Shaira, 2011). The principal component analysis is known by using an orthogonal transformation, since it makes use of the eigenvectors of the covariance matrix. The idea is to transform a set of variables that might be correlated, into a set of new uncorrelated vectors. These new vectors are named the principal components. Given that the main purpose is to reduce dimensions, the set of original variables is greater than the final number of principal components. However, when we reduce dimensions, we also lose some information, but the construction of this methodology allows the retain the maximal variance and the least squared errors are minimized (Girase, Sheetal, and Mukhopadhyay, 2015). Each component retains a percentage of the variance, being the first component the one that retains the most, and the percentage retained starts to decrease in each component. Then the dimensions can be reduced by deciding the amount of variance we want to keep.

3.3.2.2 Probabilistic Matrix Factorization

This methodology is a probabilistic method with Gaussian observation noise (Girase, Sheetal, and Mukhopadhyay, 2015). In this case, the user item matrix (V) is represented as the product of two low rank matrices, one for users and the other for the items. Let us recall our variables, we have *n* users, *m* movies, $v_{i,j}$ is the rating from the user *u* to the movie p_j. Now, let us assume U_i and P_j represent the d-dimensional user-specific and movie-specific latent feature vectors, respectively. Then the conditional distributions in the space of the observed ratings $V \in R^{nxm}$, the prior distribution over the users $U \in R^{dxn}$, and movies $P \in R^{dxm}$, are given by (Bokde, Dheeraj, Sheetal, Girase, and Mukhopadhyay, 2015) Equation 3.14, 3.15 and 3.16.

$$p(V|U,V,\sigma^2) = \prod_{i=1}^{n}\prod_{j=1}^{m}\left[n(V_{ij} \mid {U^T}_i \,.\, p \quad {}_i\sigma^2 \mid)\right] \tag{3.14}$$

$$p(U|\sigma^2) = \prod_{i=1}^{n} n(U_i \mid 0, {\sigma^2}_u \quad I) \tag{3.15}$$

$$p(P|\sigma^2) = \prod_{j=1}^{m} n\left(V_j \mid 0, {\sigma^2}_p \quad I\right) \tag{3.16}$$

where, n(X/ μ , σ^2) represents the Gaussian distribution with mean μ and variance $\sigma 2$, and I_{ij} is the indicator variable, I_{ij} = 1 if the user ui has rated the movie p_j and 0 otherwise .

3.3.2.3 Singular value decomposition (SVD)

Matrix factorization or latent factor methods can be used in recommendation systems to drive a set of hidden factors from the rating patterns and represent both users and items by such vectors of factors. Using SVD was first proposed by (Deerwester, Dumais, Furnas, Landauer, Landauer,and Harshman,1990) as a method to discover the latent factors. In information retrieval settings, this latent semantic analysis (LSA) technique is also known as Latent Semantic Analysis (LSI). The idea then inherited by the domain of recommender systems (Goldberg, Roeder, Gupta, and Perkins 2001; Canny, 1990; and Sarwar, Karypis, Konstan, and Riedl, 2000). The general equation can be given as, X= USV^t. Given an nxm matrix X, then U is an rxr orthogonal matrix with non-negative real numbers on the diagonal, and V is an rxn orthogonal matrix. The elements on the diagonal S are referred as the singular values of X (Kalman and Dan, 1996). Then the user-item marix defined as X (before we named it V) can be expressed as a composition of U, S and V. where U is representing the feature vectors corresponding to the items in the hidden feature space (Schafer, Ben, Konstan, and Riedel, 1999).

$$X_{n\times m} = U_{n\times r} \times S_{r\times r} \times V^t_{r\times m}$$

$$\begin{bmatrix} x_{11} & x_{12} & \dots & x_{1m} \\ x_{21} & & & \\ \vdots & & \ddots & \vdots \\ x_{n1} & & \dots & x_{nm} \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} & \dots & u_{1r} \\ u_{21} & & & \\ \vdots & & \ddots & \vdots \\ u_{n1} & & \dots & u_{nr} \end{bmatrix} \begin{bmatrix} s_{11} & 0 & \dots & 0 \\ & s_{21} & & \\ \vdots & & \ddots & \vdots \\ 0 & & \dots & s_{rr} \end{bmatrix} \begin{bmatrix} v_{11} & v_{12} & \dots & v_{1m} \\ v_{21} & & & \\ \vdots & & \ddots & \vdots \\ v_{r1} & & \dots & v_{rm} \end{bmatrix} \quad (3.17)$$

Now we can make a prediction as in Equation 3.18 by multiplying the matrices U, S, and V^t

$$X^{\wedge} = USV^t \quad (3.18)$$

3.3.3 Discussion

As we have discussed so far, the Memory-based techniques (User and Item based) are in many ways alike, even though the output created are distinct. The approach is easy to use and produce satisfactory results. Nonetheless, it exhibits problem of computing the similarity between items/users due to:

- Ramp up/Cold Start Problem:

New user: when a fresh user registers in a recommender, the systems lacks of information to do prediction.

New item: items have to be liked / disliked or rated by users so that there will not be similarity computation problem. For instance, if I upload a new clip on youtube.com, the clip will not be predicted to other users unless it has sufficient user feedback.

Cold start: so a recommender faces prediction difficulty when the items or users added afresh.

- Sparsity: this issue appears usually when there is cold-start. For example, there is a mass of users and items in a database, however, most of the users have not rated most of the items (Park DH, HK, IY and JK, 2012; Burke, 2002). So, the database or user-item matrix becomes very sparse.

- Reduced coverage problem: Coverage is explained as the number of items that the approach can present as suggestions. Coverage is reduced due to the incomparable very less number of users' ratings to the items in the database which causes the system to get difficulty offering them for the users.

Neighbor transitivity: this issue occurs due to data sparsity, in which likeminded users may not be recognized unless both users have rated any of the same items.

- Scalability: refers to the problem when percentage of users and items in database rise enormously, the computation also grows linearly (DH, HK, IY and JK, 2012). Which rises the algorithm complexity such as time, speed and memory. As internet contain massive information, it is difficult to recommend item in less amount of time because of scalability issue.

- Synonymy: occurs when the recommendation system is unable to distinguish items that are exactly or nearly related items to have distinct entries. This latent association between the items cannot be identified by most of the recommender systems thus consider these products differently. For example, "Comic movie" and "Funny movie" looks different but they are actually the same item. However, model-based approaches dealt well the synonymy problem.

- Popularity bias problem, appears when a user is new to a recommendation system, the system offers the user most popular items regardless the user preference. However, this issue is overcome by CBF approach (Section3.2). This effect is referred to Harry Porter's effect (Sarwar, Karypis, Konstan, and Riedl, 2001). Popularity based method is a resolution for cold start or ramp up problem.

3.4 Hybrid Filtering

Hybrid algorithm (Barragans-Martine, Costa-Montenegro, Burguillo, Rey-Lopez, Mikic-Fonte, and Peleterio 2010; Burke, 2002). A hybrid filtering combining two algorithms *m* and *n* tries make use of the pros of *m* to fix the cons of *n or vice versa*. For instance, ramp-up and popularity bias limitations of CF algorithm. Ramp-up weakness does not affect content-based algorithms since the recommendation for new products is based on their content (features) that are typically available when the new-item enters the system.

In recent literature, the most widely used hybridizing methodology is mixing CF with demographic filtering (Vozalis and Margaritis, 2007) or CF with content-based filtering (Barragans-Martine et al, 2010; Choi, Yoo, Kim, and Suh, 2012), in order to use the merits of each one of these techniques. Various fields of research have contributed so much to the growth of Hybrid filtering. Algorithms from soft-computing such as genetic algorithms (Gao and Li 2008; Ho, Fong, and Yan, 2007)], (HO et al, 2007)], fuzzy genetic (Al-Shamri and Bharadwaj, 2008), neural- networks (Christakou and Stafylopatis, 2005; Lee and Woo, 2002; Ren, He, Gu, Xia, and Wu, 2008), Bayesian networks (Campos, Fernández-Luna, and Huete, 2010), clustering (Shinde and Kulkami, 2012) and latent features (Saranya and Atsuhirto, 2009) have been used and packaged into the family of hybrid techniques.

Integrating different techniques of the same type is also possible, like naïve Bayes based CB with kNN based CB. Hybridizing similar techniques with different datasets can also be possible.

Hybrid approaches can be implemented in various ways:

1. Apply collaborative and content-based methods individually and aggregate their predictions.

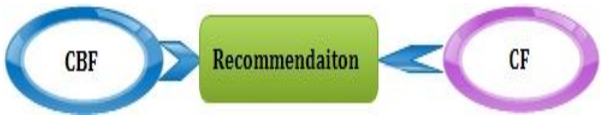
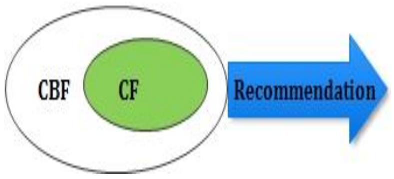


Figure 3.4: Techniques aggregation

2. Integrate some content-based features into a collaborative approach or vice versa,



or

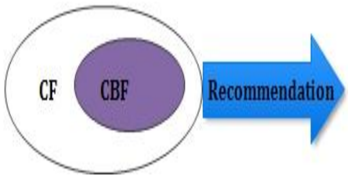


Figure 3.5: Feature integration

3. Construct a unified model that integrate both content-based and collaborative characteristics

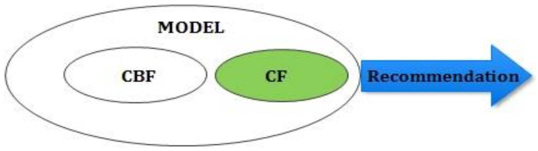


Figure 3.6: Model Unification

Bruke first classified hybrid RS into seven hybridization techniques in 2002:

Weighted hybrid: In this technique each component of hybrid recommender gives score to every item and these scores are aggregated by linear formula. This hybrid system is applied in P-tango (Claypool, Gokhale, Miranda, Murnikov, Netes, and Sartin, 1999) which hybrids both CBF and CF.

Switching hybrid: as its name goes the system switches between two recommender appraoches depending on some parameters. The metrics like precision, recall, confidence, accuracy etc. of a recommender system could be some of the switching criteria's (Burke, 2007). DailyLearner (Pazzani and Billsus, 1997), for instance, employs content-based recommendation first then collaborative recommendation if the collaborative system is un able to offer recommendations with enough evidence.

Mixed hybrid : In this hybridization different recommendations generated are merged to provide a single rating list. PickAFlick (Burke and Hammond, 1997)], PTV system (Smyth and Cotter, 2000) and rofinder (Wasfi, 1999). The reward of this technique is, it avoids the "new item" start-up problem.

Feature combination: in this approach the system treats the collaborative data as simple additional feature connected to every sample and applies CB approaches on the augmented data. For instance, In a research conducted by (Basu and Hirsh, 1998) on movie recommendation system, collaborative filter's ratings and content features were used which improved the precision of a pure collaborative technique.

Feature augmentation: this method is related to the feature combination in which the first recommender's rating output of items is input for the second recommender. The second recommender augments the data with its own contribution. For example, in Libra system (Mooney and Roy, 2000) content-based RS augments and performs recommendation of books using the dataset from Amazon.com by a naive Bayes text classifier.

Cascade hybrid: this approach carries out a staged process of a recommendation process. Where the first recommendation algorithm outputs a coarse prediction of ranking users or items and inputs to the second technique. Then the second algorithm refines the prior recommendation and presents a ultimate recommendation. As an example, EntreeC (Burke, 2002) uses cascade knowledge-based and collaborative algorithms.

Meta-level hybrid: is popularly implemented hybrid technique. In this approach two recommendation techniques are mixed in such a way output of one of the recommendation approach is the input of the other. Meta-level solves sparsity problem of collaborative filtering (Pazzanai, 1999). LaboUr (Schwab and Kobsa, 2001) is one of the applications of Meta-level method, it builds CB user profile which is then cross-checked in a CF way by applying instant-based learning to create.

3.4.1 Merits and Demerits of Hybrid Technique

Merits:

- High accuracy is achieved comparing to CBF and CB.
- Overcomes sparsity and cold-start problems

Demerits:

- A hybridized recommender system is complex to develop, it is not that easy and promising to design the hybrid (Bruke, 2007).

CHAPTER 4

EVALUATION METHODS AND METRICS

4.1 Introduction

Since the coming of the first recommendation system evaluation of recommenders has been proceeding in the recent literature. The research paper by (Herlocker, Konstan, Terveen, and Riedl, 2009) gave us a holistic and extensive study of RS evaluation. Recommendation performance is generally linked to the exactness of rating recommendation. That is, how accurate is the rating estimated against the real rating. Error metrics (discussed later) helps us get the difference between the predicted and real rating. Recently, precision-recall based, quality assessment metrics are taking over as they yield satisfactory results comparing to error metrics. Generally, we care at the quality of a recommendation not ranking. Therefore, precision-recall is widely leveraged in this area as a performance evaluator and used it to conduct the experiment .

This chapter is organized as, first in section 4.2 evaluation methods are reviewed, second in section 4.3 evalution metrics from three perspectives are discussed, finally the experimental setup, dataset and results are provided.

4.2 Evaluation Methods

To test the performance of recommender systems, different approaches are used to evaluate them. Two most commonly applied evaluation protocols are usually considered (Gunawardana and Shani, 2009) online and offline. Simulation evaluation, which allows comparing a many candidate algorithms at a cheap cost (Shani and Gunawardana, 2011). Hence we have utilized in the experiment comparing user-based model to popularity-based. The main goal of recommendation systems is to activate a user to interact with a relevant item because it is shown in the recommendations list.

4.2.1 Online

Also known as “Flights”(Kohavi, Longbotham, Sommerfield, and Henne, 2009a). This method of evaluation is within the real application on real users. As a result, the results are very trustworthy over real users, and measures performance on real application. However, especially in the beginning phases of development this is not a feasible option due to two main reasons. First, it usually takes too much time and effort to test all possible recommendation algorithms with all possible configurations on a real-life data set with strict requirements on response time, a large user and/or item base and within a certain budget (Gunawardana and Shani, 2015). Furthermore, it is applicable to applications that are not launched and it might turn out that one of the tested algorithms is not suitable for the particular use-case. When the evaluation is done in an online setting, users of the system might be negatively influenced by this unsuitable algorithm as it serves them irrelevant recommendations. As this scenario needs to be prevented, offline evaluation is used, which does not change the recommendations in an operating recommender system.

4.2.2 Offline

Offline evaluation is used in the early stages of development evaluation as it is not concerned with a running system, and therefore does not influence the recommendations the subjects of such a system receive (Gunawardana and Shani, 2015). For offline evaluation only historical data of user interactions is needed. Part of this data is used to enable the recommender system to estimate the optimal rating function as closely as possible. This process is known as training the model, and the part of the data set used for this is therefore referred to as training data set. The other part of the data set is used for the actual evaluation of the recommender algorithm but not used during training, and is known as the test data. The dataset is splitted to prevent algorithms to not overfit to the evaluation test data when the same data is used for both training and evaluation. We adopted random splitting option in our experiment.

Simulation is economical to conduct, and it allows comparing several algorithms on various distinct datasets at once. When a dataset is ready, the evaluation process simply runs the test algorithm and compares the predicted results to the \ground truth" from the dataset. The dataset employed for the simulation should be as exact as possible to the data that the recommendation system will use in real life (Herlocker et al, 2009). The limitation of offline simulation lies in the incompleteness of dataset. It is impractical to evaluate the exactness or accuracy of a recommended item to a user if no information about that user-item pair exists in the dataset. Since this evaluation method does not recruit any users, only objective evaluation can be provided.

4.2.3 User Study

A user study regularly recruits a group of users to assess recommendation systems. Quantitative measurement can be collected during subjects performing tasks through observing their behavior, such as how many recommendations are accepted by the user or the ratings indicating how much the user likes the recommended items. On the other hand, qualitative measurement can be collected via questionnaires or surveys. The subjects may be required to answer questions such as whether they have received interesting recommendations, or whether they trust the system (Shani and Gunawardana, 2011).

Unlike online simulation, user studies allow studying user behavior through subjects' interaction with the system and collecting qualitative data which is necessary for estimating the recommendation performance (Hu and Ogihara, 2011). Nevertheless, it is not assured that users will behave in the same way in real life as in the lab (Swearingenv and Sinha, 2001). The outcome can be biased because subjects already know that they are involving in an experiment. If the users know the hypothesis tested, they may tend to support it and satisfy the conductor of the experiment.

Merits:

- Direct answers on the questions at hand
- Collect information unavailable for real users especially in controlled and monitored environments.

Demerits:

- Expensive (typically test subjects get paid).
- Test subjects poorly represent the real system users.
- Measure only a small subset of the system.
- Short term experience mainly.
- Difficult to measure how users will spend real money. Also true for time, attention, and effort.

4.3 Evaluation Metrics

In the next subsections prediction accuracy metrics (MAE, MSE and RMSE), classification accuracy metrics (Precision, Recall, F-measure) and non-accuracy metrics are depicted.

4.3.1 Machine Learning Perspective

Error metrics are employed to quantify the error made by a recommender system on an item rating. A well performing algorithm results in a zero or very less error. The error metrics that recommender system uses are inherited from the area of statistics.

MAE: is the basic means of measuring prediction error. It reads, how much the predicted rating deviates from the real rating (see Formula 4.1). An extensive study is conducted in (Gunawardana and Shani, 2011). Let, p_i and r_i are the predicted and real rating, respectively, N is sum of prediction, MAE is given by the Equation 4.1:

$$MAE = \frac{\sum_{i=1}^{N} |p_i + r_i|}{N} \tag{4.1}$$

MSE: in the term itself the definition lies, first we calculate the error for the test and actual data then square it and finally compute the mean of the errors as shown in Equation 4.2. It is similar to MAE, it just penalizes larger errors since squaring larger numbers has a greater impact than squaring smaller numbers.

$$MSE = \frac{\sum_{i=1}^{N}(p_i - r_i)2}{N} \tag{4.2}$$

RMSE: is another option of MSE, which is one of the most popular metrics (Gunawardana and Shani, 2009) and was applied in Netflix Prize.

$$RMSE = \sqrt{MSE} \tag{4.3}$$

The error results of RMSE and MAE lies in the range between zero and positive infinity. Therefore, for better understanding and interpretation the error results are normalized as Equation 4.4:

$$NMAE = \frac{MAE}{r_{max} - r_{min}} \quad \text{and } NRMSE = \frac{RMSE}{r_{max} - r_{min}} \tag{4.4}$$

4.3.2 Information Retrieval

When options are binary i.e. when the task at hand is to guess if a user will or won’t like an item given a list of items, then classification metrics such as precision and recall are used to evaluate the performance of a recommendation algorithm. When doing such a selection, each item can be categorized into a True-Positive, False-Positive, True-Negative, or False-Negative as showed in Table 4.1. Such a matrix is often referred to as a confusion matrix (Burke, et al., 2011).

Table 4.1: Confusion matrix

	Recommended	**Not Recommended**
Good recommendation	True Positive (tp)	False Negative (fn)
Bad recommendation	False Positive (fp)	True Negative (tn)

From the confusion matrix, Equation of precision and recall is given 4.5 and 4.6, respectively.

$$precision = \frac{Recommended\ items\ that\ are\ relevant}{Recommended\ items} = \frac{tp}{tp+fp} \qquad (4.5)$$

$$recall\ = \frac{Recommended\ items\ that\ are\ relevant}{Relevant\ items} = \frac{tp}{tp+fn}\ , \qquad (4.6)$$

From Eq. 4.5 it can be derived that, Precision forecasts out of all the samples classified as positive, what portion of it was actually correct or relevant to the user (Manning, Raghavan, and Schtze, 2008). For instance, we have a corpus of documents. The corpus contains documents that are relevant and irrelevant to the user. So, while predicting, the precision measures how much junk or irrelevant documents are giving the user.

Therefore, if the system is designed in such a way not to show irrelevant items to the user, precision could indicate the performance satisfactorily.

The complementary form of precision is recall. In practice, both metrics are reported together (Herlocker et al, 2004). Recall or hit-rate (Deshpande and Karypis, 2004) is out of the positive samples, what portion did my RS pick-up. The intuition is how much of the relevant items the RS missed. Recall is a good indicator when the purpose of the RS is to predict only relevant items or the users desires only to see the relevant items.

Precision and recall are inversely related. For instance, If we need 100% recall, the entire corpus will be returned to the user. However, the precision will be abysmal because the RS is returning a lot of irrelevant items. That is the reason why we report them together. To resolve this trade-off, F-measure or the harmonic mean (Ge, Delgado-Battenfeld, and Jannach, 2010) which returns the mean of relevant and irrelevant items is used:

$$\mathrm{F-measure} = \frac{2 \text{ x Precision x Recall}}{\text{Precision x Recall}} \quad (4.7)$$

Remark:

According to (McLaughlin and Herlocker, 2004) measuring an algorithm's performance based on Precision and Recall reflects the real user experience better than MAE does because, usually, users actually receive ranked lists in lieu of predictions for ratings of specific items from a recommender. They determined that algorithms that were quite successful in predicting MAEs for rated items produced unsatisfactory results when analyzing their top-ranked items. (Carenini and Sharma, 2004a) also argue that MAE is not a good indicator from a theoretical perspective, as all deviations are equally weighted. From the user's perspective, however, the only fact that counts is whether an item is recommended. Hence precision-recall metric is adopted in the experiment conducted in section 4.4.

4.3.3 Human Computer Interaction and Experience

The goal of a RS is not really to predict rating of a user to given item but to recommend items that the user might view, buy or listen to. Sometimes, when recommending users' items that have high rating the user might end up to only seeing similar items. At this case, diversity is lost. Diversity measures the RS's ability to offer users items that are unfamiliar and interesting to them. (Lathia, 2010) introduced a metric to measure diversity as Equation4.8.

$$div(l_1, l_2, n) = \frac{|\frac{l_2}{l_1}|}{n} \quad (4.8)$$

Where l_2/l_1 is the portion items in the list l_2 *but* does not belong to the l_1.

Novelty: is another metric, which measure how good a RS is presenting users with items that are new (Shani and Gunawardana, 2011). This metric is also known as unexpectedness. The

RS should surprise (Herlocker et al, 2004; Zhang, Callan, and Minka, 2002) the user with Items that has never watched, purchased or listened to. (Lathia, 2010) developed an Equation 4.9 that calculates the number of new items over a certain period of time**.**

$$novelity(\, l_2, n)= \frac{|\frac{l2}{At}|}{n} \tag{4.9}$$

Where l_2 is the recent list that is to be contrasted against to the set of all items offered to the user date A_t .

Novelty and diversity are studied well in (Gunawardana and Shani, 2011; Lathia, 2010; Zhang and Hurley, 2009; Vargas and Castellas, 2011), and algorithms design to offer novel and diversity in item forecast is detailed in (Jambor and Wang, 2010; Onuma, Tong, and Faloutsos, 2009; Weng, Xu, Li, and Nayak, 2007; Zhou, Kuscsik, Liu, Medo, Wakeling, and Zhang, 2010).

Coverage**:** is a most popular non-accuracy that quantifies the percentage of items or users that a recommender system is able to suggest. We have two categories of coverage:

Item-coverage: represents the portion of items listed in the recommendation list per the total set of items (Herlocker et al, 2004). Let *u* is subset of *U* users, *i* is the subset of *I* items and Each item has a content vector C of length *c*, then the item-coverage is given by Equation 4.10 :

$$\mathrm{C_{item}} \;= \frac{\mathrm{i}}{\mathrm{I}} * 100 \tag{4.10}$$

User-coverage: measures the portion of users to whom the recommender system is able to offer items over the set of all users, Equation 4.11. The formula is:

$$C_{user} \;= \frac{u}{U} * 100 \tag{4.11}$$

There are additional metrics that play an important role in measuring the non-accuracy quality of a RS, for instance, privacy, serendipity and scalability, However, these metrics are not much active in the literature et al, 2004; McNee, Riedl, and Konstan, 2006).

4.4 Experimental Setup

In order to run the experiment, the recommender system models (Popularity and similarity based) and evaluation application must actually be built. The Experiment is carried on a public dataset Million Song dataset (see next section) offline.

The experiment is performed by training the recommender models using the generated training sets, and letting them predict the results in the test sets based on this training. By comparing these predictions to the actual results in the test set, the effectiveness of the recommender algorithm can then be evaluated, as discussed in the definition of the evaluation metrics in Section 4.3.

To assess the performance of the recommendation models in this offline experiment the subset of 10,000 songs (1.8GB) is randomly opted from the entire dataset (280 GB) for experimentation purpose. All data sets are splitted randomly such that the first 80% of the interactions forms the training dataset, and the remaining 20% of the interactions forms the test set.

Our RS models are built on the training dataset. The effectiveness of the models is evaluated using the precision and recall metrics as explained in Section 4.3.2.

4.5 Evaluation Datasets

The MillionSongdataset (MSD) which is available from[1] is used for this experiment. As indicated by its name, The MSD contains a collection of one million contemporary top songs (Bertin-Mahieux, Ellis, Whitman, and Lamere 2011; McFee, Bertin-Mahieux, Ellis, and Lanckriet 2012). Since this dataset is linked to several complementary datasets (e.g.,

[1] http://labrosa.ee.columbia.edu/millionsong/

EchoNest, last.fm and MusicBrainz[2]), it contains extensive metadata, audio features, artist tags and level of song, lyrics, and so on. The data corpus used in the competition is termed as the Taste Profile Subset[3], which includes more than forty eight million triplets (song; user; count) collected from user listening histories. The data involves around 1.2 million subjects and more than 380k songs, where all users have minimum 10 songs in their profile.

4.6 Result

The curve for item-based model is larger than popularity model this suggests the item-based model makes better recommendation as shown in Figure. 4.1.

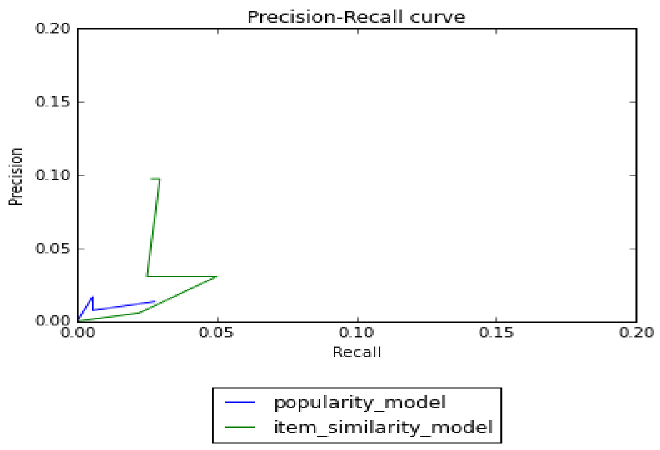


Figure 4.1: Precision-Recall Curve of popularity model and item-based model

[2] http://musicbrainz.org/
[3] http://labrosa.ee.columbia.edu/millionsong/tastepro le

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 Conclusion

Within the scope of this thesis we covered the detailed analysis of the most popular recommendation system algorithms, Collaborative Filtering, Content-based Filtering and Hybrid Approaches. The aim of this research was to understand the RS with their pros and cons of all the algorithms, and then be able to decide which one was the one that fits better for your application.

Furthermore, we gave an overview of evaluation protocols and metrics. Also, two RS models have been implemented, namely Popularity-based and item-based for the experimentation purpose. Both implementations are then evaluated using precision-recall curve. For the evaluation, offline protocol and MSD have been leveraged. The experiments then depict that item-based model is better than popularity-based model.

5.2 Recommendation and Future Works

This thesis recommendation is to use the recommendation techniques and evaluation metrics/ protocols as per the requirements of the business and the domain area as each and every one of them performs well for their targeted domain.

The thesis disregards group recommendation system. The system that provides recommendation to more than one user. This will be discussed on the future works. Moreover, the non-accuracy metrics such as serendipity, novelty and diversity are to be considered in the experiment evaluating the two models employed. The future work will focus on studying and drawing a framework of recommendation systems targeting to the software engineering area.

REFERENCES

Aamir, M., & Bhusry, M. (2015). Recommendation System: State of the Art Approach. *Iinternational Journal of Computer Applications,* 120, 887-975.

Acilar, AM., & Arslan, A. (2009). A collaborative filtering method based on Artificial Immune Network. *Expert System Application*, 8324–8332.

Adomaviciu, G. & Tuzhilin, A. (2005). Towards the Next Generation of recommendation Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Transactions of Knowledge and Data Engineering*, 17 (6), 734-749.

Aggarwal, C.C. (2016). *Recommender Systems. The Textbook p1*. Springer International Publishing Switzerland.

Ahmed, A., Kanagal, B., Pandey, S., Josifovski, V., Pueyo, L. G., & Yuan, J. (2013). Latent factor models with additive and hierarchically-smoothed user preferences. *In Proceedings of the sixth ACM international conference on web search and data mining,* 385-394.

AL-shamri M., Y. H., & Bharadwaj K. K. (2008). Fuzzy-genetic approach to recommender systems based on a novel hybrid user model. *Expert systems with Applications,* 35 (3), 1386-1399.

Amatriain & Xavier. (2011). *Data Mining Methods for Recommender Systems. In Recommender Systems Handbook* (pp. 39–71). Springer: US.

Baeza-Yates, Ricardo, & Ribeiro-Neto B. (1999). Modern information retrieval. ACM press New York, 463.

Balabanovic, M. & Shoham, Y. Fab. (1997). Content-based, collaborative recommendation. *Communications of the Association for Computing Machinery*, 40 (3), 66–72.

Barragans-martine, A. B., Costa-montenegro E., Burguillo J. C., Rey-lopez M., Mikic-fonte F., & Peleterio A. (2010). A hybrid content-based and item-based collaborative filtering approach to recommend TV programs enhanced with singular value decomposition. *International journal of Information Sciences,* 180 (22), 4290-4311.

Basu, C., Hirsh H., & Cohen, W. (1998). Recommendation as classification: using social and content-based information in recommendation. *In Proceedings of the 15th national conference on artificial intelligence* (pp. 714–20). Madison, WI.

Baumann, S., T. Pohle, & S. Vembu. (2004). Towards a socio-cultural compatibility of MIR systems. *In Proceedings of International Society for Music Information Retrieval Conference.*

Bellogín A., Cantador I., Díez, F., Castells, P. & Chavarriaga, E. (2012). An empirical comparison of social, collaborative filtering, and hybrid recommenders. *ACM Transactions on Intelligent Systems and Technology*.

Bellogín, A., Cantador, I., & Castells, P. (2010). A study of heterogeneity in recommendations for a social music service. *In Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems, HetRec* (pp. 1-8). New York, NY, USA. ACM.

Bertin-Mahieux, T., D. P. W. Ellis, B. Whitman, & P. Lamere. (2011). The million song dataset. *In Proceedings of International Society for Music Information Retrieval Conference,* 591- 6.

Billsus, D., & Pazzani, MJ. (1999). A hybrid user model for news story classification. *In Proceedings of the seventh international conference on user modeling* (pp. 99-108). Banff, Canada. Springer-Verlag, New York.

Bobadilla, J., Ortega, F., Hernando, A., & Gutierrez, A. (2013). Recommender systems survey. Knowledge-Based Systems, 46, 109-132.

Bokde, Dheeraj, Girase, S., & Mukhopadhyay, D. (2015). Matrix factorization model in collaborative filtering algorithms: A survey. *In Proceeding Computer Science,* 49, 136-146.

Bradley, K. & Smyth, B. (2001). Improving Recommendation Diversity. *In Proceedings of theTwelvth Irish Conference on Artificial Intelligence and Cognitive Science*.

Breese, John S., Heckerman, D., & Kadie C. (1998). Empirical Analysis of Predictive Algorithms for Collaborative Filtering. *In Proceedings of the Fourteenth Conference on Uncertainty in*

Artificial Intelligence (pp. 43-52). UAI'98. Madison, Wisconsin: Morgan Kaufmann Publishers Inc.

Bruke, R. (2007). Hybrid Web Recommender System. *The Adaptive Web, LNCS, 4321,* 377-408.

Burke, R, Hammond, K, & Young, B. (1997). The Find Me approach to assisted browsing. *IEEE Expert*, 32-40.

Campos L.M., Fernández-Luna J.M., Huete J.F., & Rueda-Morales M.A. (2010). Combining content-based and collaborative recommendations: a hybrid approach based on Bayesian Networks. *International Journal of Approximate Reasoning,* 51 (7), 785-799.

Canny J. (1990). Collaborative filtering with privacy. *In Proceedings of the 2002 IEEE Symposium on Security and Privacy* (pp. 45-57, pp. 391-407). Washington, DC: IEEE Computer Society 2002a.

Celma, O. & Herrera, P. (2008). A new approach to evaluating novel recommendations. *In Proceedings of the 2008 ACM conference on Recommender systems* (pp. 179-186). New York, NY, USA: ACM.

Chakrabarti, S. (2002). Mining the web: Discovering knowledge from hypertext data Science.

Chen, LS., Hsu, FH., Chen, MC., & Hsu, YC. (2008). Developing recommender systems with the consideration of product profitability for sellers. *International Journal of Information Science,* 178 (4), 1032-48.

Choi K., Yoo D., Kim G., & Suh Y. (2012). A Hybrid online-product recommendation system: Combining implicit rating-based collaborative filtering and sequential pattern analysis. *International journal of Electronic Commerce Research and Applications*.

Christakou C., & Stafylopatis A. (2005). A hybrid movie recommender system based on neural networks. *In International Conference on Intelligent Systems Design and Applications.*

Claypool, M, Gokhale, A, Miranda, T, Murnikov, P, Netes, D, Sartin, M. (1999). Combining content- based and collaborative filters in an online newspaper. *In Proceedings of ACM SIGIR workshop on recommender systems: algorithms and evaluation*. Berkeley: California.

Cremonesi, P., Turrin, R., Lentini, E., & Matteucci, M. (2008). An evaluation methodology for collaborative recommender systems. *In AXMEDIS* (pp. 224-231). Washington, DC, USA.

Crook, T., Frasca, B., Kohavi, R., & Longbotham R. (2009). Seven pitfalls to avoid when running controlled experiments on the web. *In Proceedings of the 15th ACM SIGKDD*, 1105-1114.

Cunningham, P., Bergmann, R., Schmitt, S., Traphoner, R., Breen, S., & Smyth B. (2001).WebSell: Intelligent sales assistants for the World Wide Web. *In Proceedings CBR in Ecommerce* (pp. 104-109). Vancouver BC.

Deerwester, S., Dumais, S.T., & Furnas, G.W. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41, 6, 391-407.

Del Olmo, F. H. & Gaudioso, E. (2008). Evaluation of recommender systems: A new approach. *Expert Systems with Applications*, 35 (3), 790-804.

Deniyi A., D., Wai, Z., & Yongquan, Y. (2014). Automated web usage data mining and recommendation system using k-nearest neighbor (KNN) classification method. *International journal of Applied Computing and Informatics*.

Deshpande, M., & Karypis, G. (2004) Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst*., 22 (1), 143-177.

Eighteenth Text REtrieval Conference. (2009). Appendix a: Common evaluation measures. *In The Eighteenth Text REtrieval Conference (TREC 2009) Proceedings*.

Ekstrand, M. D., J. T. Riedl, J. A. & Konstan. (2010). Collaborative Filtering Recommender Systems. *Human-Computer Interaction*, 4 (2).

Fleder, D. & Hosanagar, K. (2009). Blockbuster Cultures Next Rise or Fall: The Impact of Recommender Systems on Sales Diversity. *Manage. Sci.,* 55, 697-712.

Francesco R., Rokach L., & Shapira B. (2011). *Introduction to recommender systems handbook. In Recommender systems handbook*, 1-35.

Gao L. Q., & Li C. (2008). Hybrid personalized recommended model based on genetic algorithm. *In International Conference on Wireless Communication, Networks and mobile Computing*, 9215-9218.

Ghazantar, MA., & Pragel-Benett, A. A. (2010). Scalable accurate hybrid recommender system. *In the 3rd International conference on knowledge discovery and data mining (WKDD)*, IEEE *Computer Society*. Washington, DC: USA.

Gemulla, R., Nijkamp, E., J Haas, P., & Sismanis, Y. (2011). Large-scale matrix factorization with distributed stochastic gradient descent. *In Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 69-77). ACM.

Girase, Sheetal, & Mukhopadhyay D. (2015). Role of Matrix Factorization Model in Collaborative Filtering Algorithm: A Survey.

Goldberg K., Roeder T., Gupta D., & Perkins C. (2001). Eigentaste: A constant time collaborative filtering algorithm. *International journal of Information Retrieval* ***4*** (2), 133-151.

Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *International journal of Communications of the ACM*, 35(12), 61-70.

Golub, G., & Kahan, W. (1965). "Calculating the singular values and pseudo inverse of a matrix". *Journal of the Society for Industrial and Applied Mathematics Series B Numerical Analysis*, 2 (2), 205-224.

Gunawardana A. and Shani G. (2009). A Survey of Accuracy Evaluation Metrics of Recommendation Tasks. *Journal of Machine Learning Res*,10, 2935-2962.

Gunawardana, A. & Shani, G. (2015). *Evaluating recommender systems*. *In Recommender Systems Handbook* (pp. 265-308). Springer.

Herlocker J. L., Konstan J. A., Terveen L. G., and Riedl J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Transaction on Information Systems 22 (1),* 5-53.

Herlocker, J., Konstan, J. A., & RiedlJ. (2002). An Empirical Analysis of Design Choices in Neighborhood-Based Collaborative Filtering Algorithms. *International journal of Information Retrieval*, 5 (4).

Hiralall M., (2011). Recommender systems for e-shops, Business Mathematics and Informatics paper, Vrije Universiteit: Amsterdam.

Ho Y., Fong S., & Yan Z. (2007). A hybrid ga-based collaborative filtering model for online recommenders. *In International Conference on e-Business*, 200 -203.

Hu, Y., & M. Ogihara. (2011). Nextone player: A music recommendation system based on user behavior. *In Proceedings of International Society for Music Information Retrieval Conference*, 103-108.

Huang, S., Wang, S., Liu, T.-Y., Ma, J., Chen, Z., & Veijalainen, J. (2015). List-wise collaborative filtering. *In SIGIR*, 343-352.

Isinkaye, F.O., Folajimi, Y.O., & Ojokoh B.A. (2015). Recommendation systems: Principles, methods and Evaluation. *Egyptian Informatics Journal*, 16, 216-273.

Jalali, M., Mustapha, N., Sulaiman, M., & Mamay, A. (2010). WEBPUM: a web-based recommendation system to predict user future movement. *In Exp. Syst. Applicat.*, 37(9), 6201-6212.

Jambor, T. & Wang, J. (2010b). Optimizing multiple objectives in collaborative filtering. *In RecSys '10: Proceedings of the fourth ACM conference on Recommender systems* (pp. 55-62). New York, NY, USA: ACM.

Jannach, D., Zanker, M., Felfernig, A., & Friedich, G. (2010). *Recommender Systems: An Introduction. Cambridge University Press, 1 edition*.

Kohavi, R., Longbotham, R., Sommerfield, D., & Henne R. M. (2009). Controlled experiments on the web: survey and practical guide. *Data Min. Knowl. Discov.*, 18, 140–181.

Lieberman, H. (1995). Letizia: an agent that assists web browsing. *In Proceedings of the 1995 international joint conference on artificial intelligence* (pp. 924–9). Montreal, Canada.

Linden, G., Smith, B., & York, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering. *In IEEE Internet Computation*, 7(1), 76-80.

Liu, J., Dolan, P., & Pedersen, E. R. (2010). Personalized news recommendation based on click behavior. *In Proceedings of the 15th international conference on Intelligent user interfaces* (pp. 31-40). ACM.

Jannach, D., Zanker, M., Felfernig, A., & Friedrich G. (2011). *Recommender Systems, An Introduction.* Cambridge University Press.

Jesse, D. & Mark, G. (2006). The Relationship Between Precision-Recall and ROC Curves. *Appearing in Proceedings of the 23rd International Conference on Machine Learning.* Pittsburgh, PA.

Joachims, T. (1998). Text Categorization With Support Vector Machines: Learning with Many Relevant Features. *In European Conference on Machine Learning* (pp. 137-142). Chemnitz: Germany.

Kalman & Dan (1996). A singularly valuable decomposition: the SVD of a matrix. *In The college mathematics journal*, 27 (1), 2–23.

Kohavi, R., Longbotham, R., Sommerfield, D., Henne, R.M. (2009). Controlled experiments on the web: survey and practical guide. *In Data Min. Knowl. Discov*. 18(1), 140-181.

Konstan, J. A., McNee, S. M., Ziegler, C.N., Torres, R., Kapoor, N., & Riedl J. (2006). Lessons on applying automated recommender systems to information-seeking tasks. *In AAAI.*

Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Gordon, L. R., Riedl, J. (1997).GroupLens: applying collaborative filtering to Usenet news. *Communications of the ACM*, 40 (3).

Lang, K. (1995). NewsWeeder: Learning to filter netnews. *In Proceedings of the Twelfth International Conference on Machine Learning (ICML-95)* (pp. 331-339). San Francisco, CA.

Lathia, N. (2010). Evaluating Collaborative Filtering Over Time. *PhD thesis*. University of London, Department of Computer Science..

Lee M., & Woo Y. (2002). A hybrid recommender system combining collaborative filtering with neural network. *Lecture Notes on Computer Sciences 2347,* 531-534.

Linden, G., Smith, B., York, J. (2003). Amazon.com Recommendations, Item-to-Item Collaborative Filtering. *IEEE Internet Computing*, 7 (1).

Lops P., de Gemmis, M., & Semeraro, G. (2011). Content-based Recommender System: State of the Art and Trends. *In Springer, Recommender Systems Handbook,* 73-105.

Lu, J., Dianshuang, Wu, Mao, M., Wang, W., & Zhang, G. (2015). Recommender system application developments: a survey. *In Decision Support Systems*, 74, 12-32.

Lu, L., Medo, M., Yeung, C. H., Zhang, Y.C., Zhang, Z. K., & Zhou T. (2012). Recommender systems. *In Physics Reports*, 519(1), 1-49.

Manning, C.D., Raghavan, P., & Schtze H. (2008). *Introduction to Information Retrieval. Cambridge University Press*, New York.

Maron, M. (1961). Automatic Indexing: An Experimental Inquiry. *Journal of the Association for Computing Machinery* ,8(3), 404-417.

McCallum, A., Rosenfeld, R., Mitchell T., Ng, A. (1998). Improving Text Classification by Shrinkage in a Hierarchy of Classes. *In Proceedings of the International Conference on Machine Learning* (pp. 359-367). Morgan Kaufmann Publishers: Madison, WI,

McFee, B., T. Bertin-Mahieux, D. P. Ellis, & G. R. Lanckriet. (2012). The million song dataset challenge. *In Proceedings of 21st International Conference Companion on World Wide Web*, 909-916.

McLaughlin & Herlocker. (2004). Information retrieval. *In Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, 329-336.

McNee, S. M., Riedl, J., & Konstan, J. A. (2006). Being accurate is not enough: how accuracy metrics have hurt recommender systems. *In CHI '06 extended abstracts on Human factors in computing systems, CHI EA* (vol. 06, pp. 1097-1101). New York, NY, USA: ACM.

Mitchell, T. (1997). *Machine Learning. McGraw-Hill.*

Mican, D., & Tomai, N. (2010). Association ruled-based recommender system for personalization in adaptive web-based applications. *In Proceedings of the 10th international conference on current trends in web engineering (ICWE'10)* (pp. 85-90). Berlin: Springer-Verlag.

Mooney, RJ., & Roy, L. (2000). Content-based book recommending using learning for text categorization. *In Proceedings of the fifth ACM conference on digital libraries* (pp. 195-204). ACM.

Mobasher, B. (2007). Recommender systems. *Kunstliche Intelligenz Special Issue on Web Mining* (vol. 3. pp. 41-3). BottcherIT Verlag, Bremen, Germany.

Montaner, M., López, B., de la Rosa, J. & Ll. (2002). Developing Trust in Recommender Agents. *Proceedings of the First International Joint Conference on Autonomous Agents and Multi agent Systems (AAMAS'02)*, Palazzo Re Enzo (Italy).

Murat, G., & Sule, GO. (2010). Combination of web page recommender systems. *Exp Syst Application,* 37(4), 2911-22.

Nigam, K., McCallum, A., Thrun, S., Mitchell, T. (1998). Learning to Classify Text from Labeled and Unlabeled Documents. *In Proceedings of the 15th International Conference on Artificial Intelligence* (pp. 792-799). Madison, WI.

Onoda, T., Murata, H., & Yamada, S. (2006). Support vector machines based active learning for the relevance feedback document retrieval. *In Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*,1822, 389-392.

Onuma, K., Tong, H., & Faloutsos, C. (2009). TANGENT: a novel, "Surprise me", recommendation algorithm. *In KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 657-666). New York, NY, USA: ACM.

Park DH, Kim HK, Choi IY, & Kim JK. (2012). A literature review and classification of recommender systems research. *In Expert Syst. Appl.*, 39 (11), 59-72.

Pazzani MJ. (1999). A framework for collaborative, content-based and demographic filtering. *Artific. Intell. Rev.*, 13, 5(6), 393–408.

Pazzani, M. & Billsus, D. (1997). Learning and Revising User Profiles: The Identification of Interesting Web Sites. *Machine Learning*, 27(3), 313-331.

Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *In Philosophical Magazine*, 2 (6), 559-572.

Prabha, S. & Duraisamy, K.S. K. (2016). A Comparative Analysis of Recommendation Systems Rangasamy College of Technology, Tiruchengode, Tamilnadu, India. *Middle-East Journal of Scientific Research 24 (Techniques and Algorithms in Emerging Technologies),* 346-357.

Pradeep, I. K., & Bhaskar, M. J. (2018). Comparative analysis of recommender systems and its enhancements. *In International Journal of Engineering & Technology*, 304-310.

Pronk, V., Verhaegh, W., Proidl, A., & Tiemann, M. (2007). Incorporating user control into recommender systems based on naive bayesian classification. *In RecSys '07: Proceedings of the 2007 ACM conference on Recommender systems*, 73-80.

Pujahari, A. & Padmanabhan, V. (2014). An approach to content based recommender systems using decision list based classification with k-dnf rule set. *In Information Technology (ICIT)* (pp. 260-263). IEEE.

Qamar, A.M. (2010). Generalized cosine and similarity metrics: a supervised learning approach based on nearest neighbors. *Thesis*, University of Grenoble.

Quinlan, J. (1986). *Induction of Decision Trees. Machine Learning*,181-106.

Ren L., He L., Gu J., Xia W., and Wu F.(2008). A hybrid recommender approach based on Widrow-Hoff learning. *In International Conference on Future Generation Communication and Networking*, 40-45.

Ricci, F., Rokach, L., Shapira, B. & Kantor, P.B. (2011) *Recommender Systems Handbook,* Springer.

Ricci, F., Rokach, L., Shapira, B., & Kantor, P. B. (2010). *Recommender Systems Handbook.* Springer, Berlin, 1st edition.

Rashid, AM., Albert, I., Cosley D., Lam, SK., McNee, SM., & Konstan, JA. (2002). Getting to know you: learning new user preferences in recommender systems. *In Proceedings of the international conference on intelligent user interfaces*, 127-34.

Robillard, M.P., Walker, R.J., & Zimmermann, T. (2009). Recommendation systems for software engineering. *IEEE Software* 27(4), 80-86.

Robles, V., Larranaga, P., Pena, J., Marbán, O., Crespo, J. and Pérez, M.S. (2003). Collaborative filtering using interval estimation naive Bayes. *In Advances in Web Intelligence* (pp. 46-53). Springer: New York, NY.

Rodgers, J.L., & Nicewander, A.W. (1988). Thirteen ways to look at the correlation coefficient, *The American Statistician*, 42 (1), 59-66.

Rodriguez, M., Posse, C., & Zhang, E. (2012). Multiple objective optimization in recommender systems. *In Proceedings of the sixth ACM conference on Recommender systems* (pp. 11-18). ACM.

Said, A. & Bellogín, A. (2014). Comparative Recommender System Evaluation: Benchmarking Recommendation Frameworks. *In Proceedings of the 8th ACM Conference*.

Salter, J. & Antonopoulos, N. (2006). CinemaScreen Recommender Agent: Combining Collaborative and Content-Based Filtering. *IEEE Intelligent Systems*, 21(1), 35–41.

Saranya M., & Atsuhirto T. (2009). Hybrid recommender systems using latent features. *In Proceeding of International Conference on Advanced Information Networking and Applications Workshops*, 661- 666.

Sarwar B., Karypis G., Konstan J., & Riedl J. (2009). Item-based Collaborative Filtering Recommendation Algorithms *In WWW10*. Hong Kong.

Sarwar B., Karypis G., Konstan J., & Riedl J. (2000). Analysis of recommendation algorithms for e-commerce. *In Proceedings of the 2nd ACM Conference on Electronic Commerce (EC '00) (Minneapolis, MN), ACM*, 158-167.

Sarwar, B. M., Karypis, G., Konstan, J. A, & Riedl, J. T. (2000). Application of Dimensionality Reduction in Recommender System. *A Case Study, Architecture*, 1625, 264-8.

Sarwar, B. M., Karypis, G., Konstan, J. A., & Riedl J. (2001). Item-based collaborative filtering recommendation algorithms. *In Proceedings of the 10th International Conference on World Wide Web (WWW ’01),* 285-295.

Schafer, JB., Konstan, J., & Riedl, J. (1999). Recommender system in e-commerce. *In Proceedings of the 1st ACM conference on electronic commerce*, 158-66.

Schelter, S., Boden, C., Schenck, M., Alexandrov, A., & Markl, V. (2013). Distributed matrix factorization with mapreduce using a series of broadcast-joins. *In Proceedings of the 7th ACM conference on Recommender systems* (pp. 281–284). ACM.

Schwab, I., Kobsa, A., & Koychev, I. (2001). Learning user interests through positive examples using content analysis and collaborative filtering. *Draft from Fraunhofer Institute for Applied Information Technology,* Germany.

Sebastiani, F. (2002) Machine Learning in Automated Text Categorization. *In ACM Computing Surveys,* 34 (1).

Shani, G. & Gunawardana, A. (2011). Evaluating recommendation systems. *In Recommender systems handbook.* Springer US, 257297.

Shi, Y., Larson, M., & Hanjalic, A. (2010). List-wise learning to rank with matrix factorization for collaborative filtering. *In RecSys.,* 269 – 272.

Shinde S. K., and KulkamI U. (2012). Hybrid personalized recommender system using centering-bunching based clustering algorithm. *In Expert Systems with Applications,* 39 (1), 1381- 1387.

Smyth, B, & Cotter, P. (2000). A personalized TV listings service for the digital TV age. *In Knowl-Based Syst*, 13(2-3), 53-9.

Song, Q., Cheng, J., & Lu, H. (2015). Incremental matrix factorization via feature space re-learning for recommender system. *In Proceedings of the 9th ACM Conference on Recommender Systems* (pp. 277-280). ACM.

Steck, H. (2013). Evaluation of recommendations: rating-prediction and ranking. *In Proceedings of the 7th ACM Conference on Recommender Systems* (pp. 213-220). ACM.

Swearingen, K., & Sinha, R. (2001). Beyond algorithms: An HCI perspective on recommender systems. *In ACM SIGIR Workshop on Recommender Systems*, 13, 393-408.

Tomlinson, S. (2005). European ad hoc retrieval experiments with hummingbird Search Server TM at. *In CLEF, Working Notes for the CLEF 2005 Workshop*.

Vargas, S. & Castells, P. (2011). Rank and relevance in novelty and diversity metrics for recommender systems. *In Proceedings of the fifth ACM conference on Recommender systems, RecSys "11* (pp. 109–116). New York, NY, USA: ACM.

Vozalis m. g., & Margaritis k. g. (2007). Using SVD and Demographic Data for the Enhancement of Generalized Collabrative Filtering. *Information Sciences*, 177, 3017-3037.

Wang, S., Sun, J., Gao, B. J., & Vsrank, J. Ma. (2014). A novel framework for ranking-based collaborative filtering. *ACM Trans. Intell. Syst. Technol.*, 5(3).

Wasfi, AM. (1999). Collecting user access patterns for building user profiles and collaborative filtering. *In Proceedings of the 1999 international conference on intelligent user, Redondo Beach* (pp. 57–64). CA.

Weimer, M., Karatzoglou, A., Le, Q. V., & Smola, A. J. (2007). Co_Rank: Maximum margin matrix factorization for collaborative ranking. *In NIPS. ems. Computer*, 42(8), 30-37, 200.

Weng, L. T., Xu, Y., Li, Y., & Nayak, R. (2007). Improving Recommendation Novelty Based on Topic Taxonomy. *In Web Intelligence and Intelligent Agent Technology, International Conference on*, 115-118.

Wintrode, J., Sell, G., Jansen, A., Fox, M., Garcia-Romero, D., & McCree A. (2015). Content-based recommender systems for spoken documents. *In Acoustics, Speech and Signal Processing (ICASSP), IEEE International Conference* (pp. 5201-5205). IEEE.

Xiaoyuan, S., & Taghi, M. K. (2009). A Survey of Collaborative Filtering Techniques. *In Advances in Artificial Intelligence*, 421-425.

Yang, Y. & Liu, X. (1999). A re-examination of text categorization methods. *In Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, 42-49.

Yang, Y., & Pedersen J. (1997). A Comparative Study on Feature Selection in Text Categorization. *In Proceedings of the Fourteenth International Conference on Machine Learning, Nashville,* TN, 412-420.

Zhang, M. & Hurley, N. (2009). Statistical Modeling of Diversity in Top-N Recommender Systems. *In Web Intelligence and Intelligent Agent Technology, IEEE/WIC/ACM International Conference*, 1, 490-497.

Zhang, Y., Callan, J., & Minka, T. (2002). Novelty and redundancy detection in adaptive filtering. *In Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR "02* (pp. 81-88). New York, NY, USA: ACM.

Zhou, T., Kuscsik, Z., Liu, J.-G., Medo, M., Wakeling, J. R., & Zhang, Y.-C. (2010). Solving the apparent diversity-accuracy dilemma of recommender systems. *In Proceedings of the National Academy of Sciences*, 107(10), 4511-4515.

Zhou, Y., Wilkinson, D., Schreiber, R., & Pan, R. (2008). Large-scale parallel collaborative filtering for the netflix prize. *In Algorithmic Aspects in Information and Management* (pp337-348). Springer.

Zhuang Y., Chin W., Juan Y., & Lin C. (2013). A fast parallel sgd for matrix factorization in shared memory systems. *In Proceedings of the 7th ACM conference on Recommender systems* (pp. 249-256). ACM.

Ziegler, CN., Lausen, G., & Schmidt-Thieme, L. (2004). Taxonomy-driven computation of product recommendations. *In Proceedings of the 13th international conference on information and knowledge management (CIKM '04)* (pp. 406-15). Washington, DC, USA.

APPENDICES

APPENDIX 1

A SONG RECOMMENDER SAMPLE CODE

```
# Load music data

#Read userid-songid-listen_count triplets

#This step might take time to download data from external sources

triplets_file = 'https://static.turi.com/datasets/millionsong/10000.txt'

songs_metadata_file = 'https://static.turi.com/datasets/millionsong/song_data.csv'

song_df_1 = pandas.read_table(triplets_file,header=None)

song_df_1.columns = ['user_id', 'song_id', 'listen_count']

song_df_2 =  pandas.read_csv(songs_metadata_file)

song_df = pandas.merge(song_df_1, song_df_2.drop_duplicates(['song_id']), on="song_id",
how="left")

song_df = song_df.head(10000)

song_df = song_df.head(10000)

song_df['song'] = song_df['title'].map(str) + " - " + song_df['artist_name']

song_grouped = song_df.groupby(['song']).agg({'listen_count': 'count'}).reset_index()

grouped_sum = song_grouped['listen_count'].sum()

song_grouped['percentage']  = song_grouped['listen_count'].div(grouped_sum)*100

song_grouped.sort_values(['listen_count', 'song'], ascending = [0,1])

train_data, test_data = train_test_split(song_df, test_size = 0.20, random_state=0)

print(train_data.head(5))

pm = Recommenders.popularity_recommender_py()

pm.create(train_data, 'user_id', 'song')
```

```
user_id = users[5]

pm.recommend(user_id)

# Build a song recommender with personalization

# Create instance of item similarity based recommender class

is_model = Recommenders.item_similarity_recommender_py()

is_model.create(train_data, 'user_id', 'song')

#Recommend songs for the user using personalized model

is_model.recommend(user_id)

#Quantitative comparison between the models

#We now formally compare the popularity and the personalized models using precision-recall
curves

start = time.time()

#Define what percentage of users to use for precision recall calculation

user_sample = 0.05

#Instantiate the precision_recall_calculator class

pr = Evaluation.precision_recall_calculator(test_data, train_data, pm, is_model)

#Call method to calculate precision and recall values

(pm_avg_precision_list, pm_avg_recall_list, ism_avg_precision_list, ism_avg_recall_list) =
pr.calculate_measures(user_sample)

end = time.time()

print(end - start)

#Code to plot precision and recall curve

import pylab as pl
```

```
#Method to generate precision and recall curve

def plot_precision_recall(m1_precision_list, m1_recall_list, m1_label, m2_precision_list,
m2_recall_list, m2_label):

    pl.clf()

    pl.plot(m1_recall_list, m1_precision_list, label=m1_label)

    pl.plot(m2_recall_list, m2_precision_list, label=m2_label)

    pl.title('Precision-Recall curve')

    #pl.legend(loc="upper right")

    pl.legend(loc=9, bbox_to_anchor=(0.5, -0.2))

    pl.show()

print("Plotting precision recall curves.")

plot_precision_recall(pm_avg_precision_list, pm_avg_recall_list, "popularity_model",

                      ism_avg_precision_list, ism_avg_recall_list, "item_similarity_model")
```

APPENDIX 2

EVALUATION SAMPLE CODE

```
class precision_recall_calculator():
    def __init__(self, test_data, train_data, pm, is_model):
        self.test_data = test_data
        self.train_data = train_data
        self.user_test_sample = None
        self.model1 = pm
        self.model2 = is_model
        self.ism_training_dict = dict()
        self.pm_training_dict = dict()
        self.test_dict = dict()
    def remove_percentage(self, list_a, percentage):
        k = int(len(list_a) * percentage)
        random.seed(0)
        indicies = random.sample(range(len(list_a)), k)
        new_list = [list_a[i] for i in indicies]
        return new_list
    def create_user_test_sample(self, percentage):
        #Find users common between training and test set
        users_test_and_training =
list(set(self.test_data['user_id'].unique()).intersection(set(self.train_data['user_id'].unique())))
        print("Length of user_test_and_training:%d" % len(users_test_and_training))
        self.users_test_sample = self.remove_percentage(users_test_and_training, percentage)
        print("Length of user sample:%d" % len(self.users_test_sample))
    def get_test_sample_recommendations(self):
        for user_id in self.users_test_sample:
            print("Getting recommendations for user:%s" % user_id)
            test_data_user = self.test_data[self.test_data['user_id'] == user_id]

            self.test_dict[user_id] = set(test_data_user['song'].unique() )

    def calculate_precision_recall(self):

        cutoff_list = list(range(1,11))

        num_users_sample = len(self.users_test_sample)

        for N in cutoff_list:

            ism_sum_precision = 0

            ism_sum_recall = 0

            for user_id in self.users_test_sample:

                ism_hitset =
self.test_dict[user_id].intersection(set(self.ism_training_dict[user_id][0:N]))

                pm_hitset =
self.test_dict[user_id].intersection(set(self.pm_training_dict[user_id][0:N]))

                testset = self.test_dict[user_id]

        return (pm_avg_precision_list, pm_avg_recall_list, ism_avg_precision_list,
ism_avg_recall_list)

    def calculate_measures(self, percentage):

        self.create_user_test_sample(percentage)

        self.get_test_sample_recommendations()

        return self.calculate_precision_recall()
```

APPENDIX 3

RECOMMENDER MODELS SAMPLE CODE

```
class popularity_recommender_py():
train_data_grouped = train_data.groupby([self.item_id]).agg({self.user_id:
train_data_grouped.rename(columns = {'user_id': 'score'},inplace=True)
train_data_sort = train_data_grouped.sort_values(['score', self.item_id], ascending = [0,1])
self.popularity_recommendations = train_data_sort.head(10)
def recommend(self, user_id):
user_recommendations = self.popularity_recommendations
user_recommendations['user_id'] = user_id
cols = user_recommendations.columns.tolist()
cols = cols[-1:] + cols[:-1]
user_recommendations = user_recommendations[cols]
return user_recommendations
class item_similarity_recommender_py():
def get_user_items(self, user):
user_data = self.train_data[self.train_data[self.user_id] == user]
user_items = list(user_data[self.item_id].unique())
return user_items
def get_item_users(self, item):
item_data = self.train_data[self.train_data[self.item_id] == item]
item_users = set(item_data[self.user_id].unique())
return item_users
def get_all_items_train_data(self):
all_items = list(self.train_data[self.item_id].unique())
return all_items
```

```
def construct_cooccurence_matrix(self, user_songs, all_songs):
user_songs_users = []
for i in range(0, len(user_songs)):
user_songs_users.append(self.get_item_users(user_songs[i]))
cooccurence_matrix = np.matrix(np.zeros(shape=(len(user_songs), len(all_songs))), float)
for i in range(0,len(all_songs)):
songs_i_data = self.train_data[self.train_data[self.item_id] == all_songs[i]]
users_i = set(songs_i_data[self.user_id].unique())
for j in range(0,len(user_songs)):
users_j = user_songs_users[j]
users_intersection = users_i.intersection(users_j)
if len(users_intersection) != 0:
users_union = users_i.union(users_j)
cooccurence_matrix[j,i] = float(len(users_intersection))/float(len(users_union))
else:
cooccurence_matrix[j,i] = 0
return cooccurence_matrix
def generate_top_recommendations(self, user, cooccurence_matrix, all_songs, user_songs):
print("Non zero values in cooccurence_matrix :%d" % np.count_nonzero(cooccurence_matrix))
user_sim_scores = cooccurence_matrix.sum(axis=0)/float(cooccurence_matrix.shape[0])
user_sim_scores = np.array(user_sim_scores)[0].tolist()
sort_index = sorted(((e,i) for i,e in enumerate(list(user_sim_scores))), reverse=True)
columns = ['user_id', 'song', 'score', 'rank']
df = pandas.DataFrame(columns=columns)
return df
def create(self, train_data, user_id, item_id):
print("no. of unique songs in the training set: %d" % len(all_songs))
cooccurence_matrix = self.construct_cooccurence_matrix(user_songs, all_songs)
```

```
df_recommendations = self.generate_top_recommendations(user, cooccurence_matrix,
all_songs, user_songs)

return df_recommendations

def get_similar_items(self, item_list):

user_songs = item_list

all_songs = self.get_all_items_train_data()

print("no. of unique songs in the training set: %d" % len(all_songs))

cooccurence_matrix = self.construct_cooccurence_matrix(user_songs, all_songs)

user = ""

df_recommendations = self.generate_top_recommendations(user, cooccurence_matrix,
all_songs, user_songs)

return df_recommendations
```