International Journal of Engineering Research in Computer Science and Engineering (IJERCSE) Vol 3, Issue 3, March 2016

Kibitzer: Media Recommendations using Big Data Analytics

[1] Ashesh Kumar Singh [2] Aditya Shetye [3] Sanket Nalavade [4] Tarun Pable [5] Archana Shirke
[1] B.E., Student [2] Professor
[1][2][3][4][5] Department of Information Technology
Fr. C. Rodrigues Institute of Technology, Vashi Navi Mumbai
[1] User501254@gmail.com [2] adityashetye04@gmail.com [3] sanket.nalavade123@gmail.com
[4] tarun150595@gmail.com [5] archanashirke25@gmail.com

Abstract: **Recommendation systems can be found all over the web, from e-commerce websites like Flipkart and Amazon to social entertainment platforms like YouTube. All of them provide recommendations to users and have completely redefined the online experience. Recommendation systems however need quality data to work with. This is where big data comes into the picture. The volume, variety and velocity of big data provides the opportunity for greater user modeling, adaptation and personalization. The idea here, is to explore this domain to give recommendations on entertainment media, more specifically – movies, music and books. We propose a system that makes use of item-item similarities and matrix factorization together on existing datasets to recommend items in multiple departments.**

***Keywords*—Big data, Collaborative Filtering, Recommendations, item-item**

I. INTRODUCTION

There are a number of applications which supply, to their users, some form of recommendation. A lot of times, users are presented with products, that they "may want to buy", movies they "may want to watch", music they "may want to listen", books "they may want to read" or people they "may want to befriend". Such applications make use of recommendation systems. Recommendation systems (RSS) have become a part and parcel of one's internet experience. The goal of a recommender system is to generate meaningful recommendations for some target user, of items that might interest them. Considering an online setting, say an e-commerce website which is almost always filled with an overwhelming number of products, its highly likely that a user may not have enough judgment to appropriately select from a number of alternatives [1]. RSS are primarily directed towards such individuals. Now, generating these recommendations requires processing of ever increasing, fast flowing and often unstructured data in an efficient manner. This is where big data can be made use of. Big data is not a single market. Rather, it is a combination of various technologies and techniques that have evolved over time. Big data solutions enable organizations to store, manage, and manipulate vast amounts of data at the right speed and at the right time to gain the right insights.

It's convenient to simplify big data into the three vs — *volume*, *variety*, *velocity*. However this can be misleading and overly simplistic [2]. For example, one may be managing a tiny volume of very complex data or one may be processing a huge volume of very simple data, neither of them may be handled equally well through traditional techniques, tools and technologies. Even more important are the *value* & *veracity* associated with the data. Recommender systems are increasingly incorporating big data into themselves to take advantages of the evolving approaches in the area. The tools and techniques required for such and investment in big data has relatively decreased and this has further encouraged even the smaller companies to go the extra mile [3]. As the world enters the era of big data, the recommender systems face greatly increased complexities. Previous computational models and experience on data do not always hold well today, thus, how to build an efficient and robust system has become an important issue for many practitioners of such systems.

Using various statistical models and algorithms, RSS predict preference that users would give to a product/service they had not yet considered [4]. Suggestions for books on amazon, movies on netflix, music on soundcloud, friends on facebook are some real world examples of the operation of industry-strength recommender systems. The design of such RSS depends on the domain and the particular characteristics of the data available or collected. The system may have access to user-specific and item-specific profile attributes such as personal information and product descriptions respectively. Recommender systems differ in the way they analyze these data sources to develop notions of affinity between users and items which can be used to identify what should be recommended [5]. *Collaborative filtering* is one such way

which is the process of filtering for information or patterns using techniques involving collaboration among multiple agents, viewpoints, data sources, etc and has continually attracted attention in this field.

In the sections that follow, we discuss some work done in this field while providing an in-depth background on some algorithms used under collaborative filtering techniques since we incorporate those in our proposed system. Finally we outline our proposed system and draw conclusions.

II. RELATED WORK

All recommendations at their lowest level, whether it be in real world face-to-face interaction or on an e-commerce portal are of two types – *non-personalized* and *personalized*. Non-personalized recommendations are independent of the customer, so each customer gets the same recommendation. These could be average ratings of products, new arrivals; trending or even handpicked and are the simplest type of recommendations. Personalized recommendations on the other hand are offered as a ranked list of items [6]. In performing this ranking, RSS try to predict what the most suitable products or services are, based on the user's preferences and other constraints. In order to complete such a computational task, RSS collect from users their preferences, which are either explicitly expressed, e.g., as ratings for products, or implicitly inferred by interpreting user actions. It is important to note that these may not always be accurate and from a user's perspective are just suggestions which may or may not be the most optimal one for them. Nevertheless personalized recommendations still are more preferable for a unique user experience.

In their book ***recommender systems handbook*** author francesco ricci, lior rokach and bracha shapira list out a range of possible roles that a rs can play on behalf of the service provider and the end user.

Functions of a rs from a service providers perspective:

1. Increase the number of items sold
2. Sell more diverse items
3. Increase the user satisfaction
4. Increase user fidelity
5. Better understand what the user wants

Functions of a rs from an end users perspective:

1. Find some good items
2. Find all good items
3. Annotation in context
4. Recommend a sequence
5. Recommend a bundle
6. Just browsing
7. Find credible recommender
8. Improve the profile
9. Express self
10. Help others
11. Influence others

As these various points indicate, the role of a rs within an information system can be quite diverse [6]. This diversity calls for the exploitation of a range of different knowledge sources and techniques. We have already discussed how big data can help when it comes to such sources. As for techniques, the two major classes of RSS are – *cognitive filtering* and *collaborative filtering*. We discuss them in detail in the following paragraphs.

Cognitive filtering, more popularly know as content-based techniques recommends items based on a comparison between the content of the items and a user profile. The content of each item is represented as a set of descriptors or terms, typically the words that occur in a document. The user profile is represented with the same terms and built up by analyzing the content of items which have been seen by the user. The similarity of items is calculated based on the features associated with the compared items [6]. For example, you have a blog with numerous blog posts of varying topics as show n in figure. 1 where a specified user reads about a post written on the topic linux, you could also suggest them posts written on the topic opensource since those two posts have similar material and are of interest to the user. Here the recommendation is made solely based on the content accessed by a single user. Another thing to note here is that in this example we are considering the number of times a particular target user has read blog posts on some topic, this feedback is implict in nature but we could also consider explicit feed back like ratings given by a single user.

Blogs	**Articles read per user** *(Elise)*	**Similar content** *(Linux)*
Linux	11	Open Source
Open Source	-	Cloud Computing
Cloud Computing	9	
Java Technology	-	
Agile	1	*Ranked blogs*

Fig. 1. ***Simple example of cognitive filtering [9]***

Relevance feedback, genetic algorithms, neural networks, and the bayesian classifier are among the learning techniques for learning a user profile [6][7][8]. The vector space model and latent semantic indexing can both be used by these learning methods to represent

documents. Some of the learning methods also represent the user profile as one or more vectors in the same multidimensional space which makes it easy to compare documents and profiles. Other learning methods such as the bayesian classifier and neural networks do not use this space but represent the user profile in their own way. However these types of systems suffer from problem like cold start, overspecialization and limitations of content analysis [6]. The need of such systems to analyze the actual contents of the item makes them somewhat infeasible for adoption on a large scale where the content may belongs to multiple domains and be tough to examine.

Collaborative Filtering is the most successful recommendation technique to date and also widely popular. Tapestry is one of the earliest implementations of collaborative filtering-based recommender systems [10]. This system relied on the explicit opinions of people from a close-knit community, such as an office workgroup. In a typical scenario, there is a list of m users $U = \{u_1, u_2, u_3, \ldots, u_m\}$ and a list of n items $I = \{i_1, i_2, i_3, \ldots, i_n\}$. Each user u_i has a list of items I_{ui}, which the user has expressed their opinions about. Opinions can be explicitly given by the user as a rating score, generally within a certain numerical scale, say between 1 to 5 or can be implicitly derived from purchase records, by analyzing timing logs, by mining web hyperlinks and so on. Note that $I_{ui} \subseteq I$ and it is possible for I_{ui} to be a null-set, in which case it means that the user has given no opinion yet. There exists a distinguished user $u_a \in U$ active user for whom the task of a collaborative filtering algorithm is performed. For example, again taking up the blog where multiple users read multiple blog posts on varying topics as shown in Figure 2 we can make a clustering model for grouping users together based on their taste and then suggests users blog posts that other uses within the same cluster have read but our target user hasn't. Here too we could consider explicit feedback like rating for articles in place of number of times users in a cluster reads an article.

Blogs	**Articles read per user**		
	Marc	Megan	Elise
Linux	13	3	11
OpenSource	10	-	-
Cloud Compting	6	1	9
Java Technology	-	6	-
Agile	-	7	1
Cluster	1	2	1

Fig. 2. ***Simple example of collaborative filtering [9]***

Now depending on the algorithm used collaborative filtering techniques further breaks down into two primary approaches *memory-based* and *model-based* [12][13]. Memory-based algorithms utilize the entire user-item database to generate a prediction [6]. These systems employ statistical techniques to find a set of users, known as neighbors, that have a history of agreeing with the target user (i.e., they either rate different items similarly or they tend to buy similar sets of items). Once a neighborhood of users is formed, these systems use different algorithms to combine the preferences of neighbors to produce a prediction or *top-n* recommendation for the active user. The techniques, also known as *nearest-neighbor* or user-based & item-based collaborative filtering are more popular and widely used in practice [14]. Model-based methods build models based on modern machine learning algorithms discovering patterns in the training data and hence only need a subset of database [6]. Algorithms in this category take a probabilistic approach and envision the collaborative filtering process as computing the expected value of a user prediction, given their ratings on other items. Bayesian models, clustering models, and dependency networks, all have been investigated extensively under such model based collaborative filtering algorithms [6]. Despite of its benefits and the interestingness of recommendations offered, collaborative filtering algorithms have challenges of dealing with spares & noisy data, privacy issues and shilling attacks [11]. Many of these challenges are active research topics and new findings show that they can be overcome in some way or other [15][16].

Besides cognitive filtering and collaborative filtering numerous other classes of RSS have come into existence. These RSS classes are – *demographic*, *knowledge-based*, *community-based* and *hybrid*. They are discussed briefly below:

Demographic RSS recommend items based on the demographic profile of the target user. In other words, they take into consideration the structure of the population that a particular user belongs to and accordingly makes some appropriate recommendation [17][18]. The idea here is that people from different demographic groups have preference for different items. Such RSS could take into consideration the age, gender, nationality of the user to make recommendation. For example, the contents of website could be arranged based on the age of the visitor.
Knowledge-based RSS recommend items based on specific domain knowledge about how certain item features meet target user's needs and preferences and, ultimately, how the

item is useful for the user. These RSS try to reason and can be either *case-based* or *constraint-based*. Case-based recommenders determine recommendations on the basis of similarity metrics treating similarity of items as knowledge or utility whereas constraint-based recommenders predominantly exploit predefined knowledge bases that contain explicit rules about how to relate customer requirements with item features [19][20]. For example a new portal using knowledge-based rs would show news articles based on its knowledge of current happenings and urgency.

Community-based RSS recommends items based on the preferences of the target user's friends. This technique follows the epigram "tell me who your friends are, and i will tell you who you are". [6][21]. These types of RSS have seen a huge increase in implementation due to the popularity of social networking sites like facebook, google plus and twitter in the past few years. Community-based RSS rely heavily on the social relations of users.

Hybrid RSS are actually the combination of any the above mentioned techniques [6]. In its simplest form, only two classes of RSS are combined but hybrid RSS may very well be a combination of even three or more classes. This combination is done with the believe that the advantages of one class of recommenders can compensate for the disadvantages of the other making the overall system more robust and effective. For instance, collaborative filtering methods suffer from new-item problems, i.e., they cannot recommend items that have no ratings. This does not limit content-based approaches since the prediction for new items is based on their description (features) that are typically easily available [16]. Hence we could combine those to get recommendation for newer items.

Now that we have given some background on the working of a rs, we can discusses some of the industrial RSS that are used on a daily basis:

1. Apple's itunes, a digital store for music takes ratings from each user's playlists and compares the ratings with those of other itune's users who also have rated their own music colletion. Based on the abundant file information, the system can finds out the taste of the user, predict what else the user would like, and recommend potential music items of interest. The existing music recommender systems like the one of itunes know nothing about the nature of music files. They only rely on the rating values or the action of users. Only depending on such statistic data has limitations in finding more accurately matching music files.
2. Amazon, an online shopping portal uses recommendation algorithms to personalize the online store for each customer. The recommendation items are continuously being changed based on customer interests. It easily detects what the customer's job is, in what situation the customer is, and to what area the customer's taste is changed, and then, it suggests the precisely adequate items to the customer. The system will need to observe user's actions such as emails, coupons, friends & other interesting people, and feedback about products. Amazon is also a well known advocate of item-to-item cf technique.
3. Netflix, a popular online digital video streaming and rental service makes use of over 107 algorithms to generate recommendations. This ensemble of these algorithms results in a number of recommendations. It is also famous for the having held competitions that invite users to help improve their system.
4. Google playstore, which is a mobile app store for getting apps, movies, music, magazines etc on android devices (possibly) makes use of community and demographic RSS. It is capable of giving recommendations based on what apps are liked by a user's contacts and also the popular ones in their locality.

III. ALGORITHMS USED

There are a number of crude as well as state-of-the-art algorithms that can be used in making a collaborative filtering (cf) rs. Choosing the correct one however involves a proper understanding of the purpose and the available dataset. In this section we discuss item-to-item model based algorithm and als model based matrix factorization algorithm that we use in our system to make recommendations.

Item based similarity *CF:* In the algorithm, the similarities between different items in the dataset are calculated by using one of a number of similarity measures like Correlation, and then these similarity values are used to predict ratings for user-item pairs not present in the dataset [22]. A general description of the algorithm is as follows:

1. For every pair of Item i_a and Item i_b, find all the people who rated both i_a and i_b.
2. Use these ratings to form a Item I_a vector and a Item I_b vector.
3. Calculate the similarity between these two vectors.
4. Whenever someone sees a Item, recommend the Items most similar with it.

Users \ *Items*	i_1	i_2	i_3
u_1	3	3	4
u_2	3	-	2

u_3	4	2	5
u_4	-	3	-
u_5	3	1	-

Table 1. user-item rating

For items i_1 and i_2, common raters are users u_1, u_3 and u_5. The two rating vectors are [3,4,3] & [3,2,1].
For items i_2 and i_3, common raters are users u_1 and u_3. The two rating vectors are [3,2] & [4,5].
Taking cosine as a measure, similarity between items are as follows:

$$Cosine(I u_1, I u_3) = 0.916698497028$$
$$Cosine(I u_2, I u_3) = 0.952925780013$$

Hence for user u_4, who has rated item i_2 the algorithm would recommend item i_3 over i_1.

Alternating Least Square (ALS): Alternating Least Squares is based on matrix factorization. The intuition behind using matrix factorization to solve this problem is that there should be some latent features that determine how a user rates an item [11]. Hence, if we can discover these latent features, we should be able to predict a rating with respect to a certain user and a certain item, because the features associated with the user should match with the features associated with the item. ALS rotates between fixing one of the unknowns I_a or U_b [23]. When one is fixed the other can be computed by solving the least-squares problem. This approach is useful because it turns the problem into a quadratic one that can be solved optimally. A general description of the algorithm for ALS algorithm for collaborative filtering taken from Zhou et. al is as follows:

1. Initialize matrix *I* by assigning the average rating for an Item as the first row, and small random numbers for the remaining entries.
2. Fix *I*, solve *U* by minimizing the error function.
3. Fix *U*, solve *I* by minimizing the error function similarly.
4. Repeat Steps 2 and 3 until convergence.

IV. PROPOSED SYSTEM

The proposed system will be an application capable of providing cross domain recommendations to its users using big data analytics. The system will be capable of dealing with the accompanying challenges of volume, velocity and variety that big data presents to provide acceptable value and trustable veracity on recommendations. This system is intended to be cross domain with respect to movies, music and books.

such existing systems are rare and also inaccurate. A statistical survey was conducted and the results obtained suggested that:

- About half of them could not make sense of the usefulness of recommendations given.
- Most had not come across a single recommendation system which suggests across multiple domains.
- Almost all of them had received bizarre or useless recommendations at some or the other time.
- Almost all of them had received extremely obvious recommendations at some or the other time.
- Almost all of them had received extremely obvious recommendations at some or the other time.

The core system will be moldable and adaptable due to having being based off cf in a way that would allow it to be effortlessly employed by businesses or organizations looking to provide interesting and useful recommendations to their clients or users.

The system is composed three individual module but are capable of working together to attain our goal of giving recommendations. We have divided the task of data collection, rating and filtering within these modules. Keeping a modular structure also makes the system more manageable since if a particular module needs some changes it will effect components only within that particular module in most cases and not others.
These individual modules are:

A) User interface module

This module supplies the user interface for end customers that are the people who have an account an are logged in to obtain recommendations. A user can view recommendations on the website, and give feedback. This is available only to the front end consumer.

B) Data supplier module

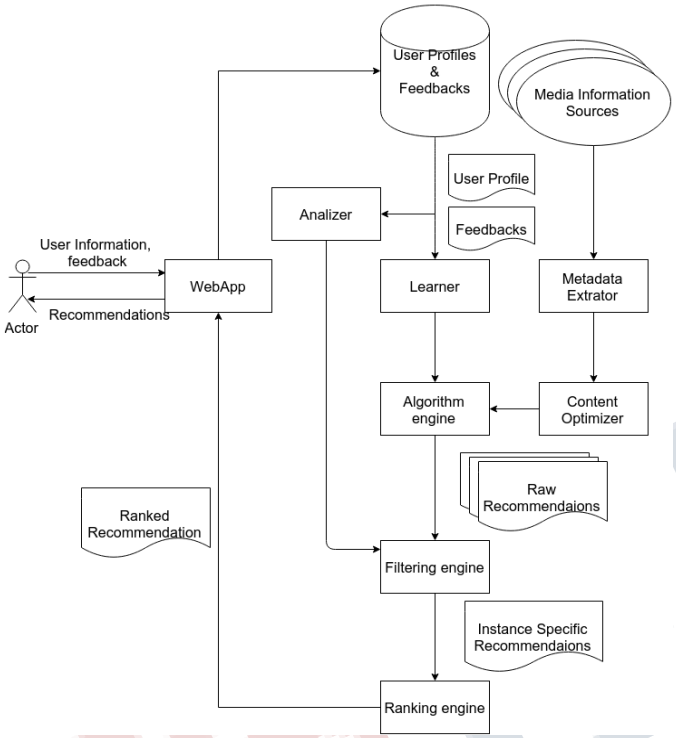
This module is only for getting relevant data into the system. Since our system heavily depends on supplied data, it becomes necessary to maintain a separate module for explicitly dealing with the prepossessing which is accessible only to the back-end users and not the customers.

C) Recommendation module

This module is the one which does all the heavy lifting since it is directly responsible for generating ***Recommendations.*** This is used by both customers (unknowingly) and the backend service providers. Customer cannot do many operations, but their ratings and other habits along with existing pointers are very important to create a relevant recommendation. Users can only use give rating operation. This subsystem has its part in user

interface only as a "rating for this recommendation" part. This provides feedback for improvements in our recommendations. On the background of this subsystem, suppliers can get customer profile and statistics.

A simplified diagram of the system with its various



functional components is given below:

Fig. 3. ***Architectural diagram***

The Detailed description of each of the smaller components is as follows:

A. Web App

This component acts as the client front-end to our whole underlying system for the end user. It is responsible for getting basic user information such as their gender, age and also get feedback on the recommendations given to them. Besides this the users may also rate and browse media recommendations that are made available to them. This module also collects both explicit and implicit feedback from the users.

B. Analyzer

It basically informs the algorithm engine what kind of data is available to work with. The major role of this system is to get new users started with basic recommendations since they will surely not have any history to go with.

C. Learner

This module collects data representative of the user preferences and tries to generalize this data, in order to construct the profile type. The generalization strategy is able to infer a model of interests starting from items rated on in the past. The output of this module is fed into the filtering engine and plays a crucial role in getting what users might be interested in.

D. Metadata Extractor

This modules works on a large set of media from internal and external sources if metadata related to them is not readily available. It basically deals with getting meta data into the system to be worked on. This module would also come in handy if at all in future there is a shift towards a content-base backed CF hybrid system.

E. Content Optimizer

Not all metadata on music, movies and books are useful in all possible use cases and moreover since only a handful of these can be properly inferred, others can be safely discarded. It is the job of this module to get rid of all redundancies and keep only the useful attributes of the original information. In an essence it it the preprocessor.

F. Algorithm engine

This is the central module in the overall system and gets its input from multiple modules namely the learner, and optimizer. Its job is to give raw recommendations by performing initial Big Data analytics in two steps.

In the first step, it uses Item-Based similarity CF algorithm to display a list of items similar to those that the user has positively rated in the past. Measures like *Correlation*, *Regularized Correlation*, *Cosine* and *Jaccard* are used to evaluate item similarities using the following mathematical formulas:

$$Corr(A,B)=\frac{n\sum AB-\sum A\sum B}{\sqrt{n\sum A^2-(\sum A)^2}\sqrt{n\sum B^2-(\sum B)^2}}$$

$$\mathrm{RegCorr}(A,B)=\frac{n}{n+n'}\times Corr(A,B)+\left(1-\frac{n}{n+n'}\right)\times PriorCorr$$

$$Cosine(A,B)=\frac{\sum AB}{\|A\|\|B\|}$$

$$Jaccard(A,B)=\frac{A\cap B}{A\cup B}$$

Where,

A and *B* are item pair rating vectors *n* is the size of vector. *Corr* and RegCorr stand for Correlation and Regularized Correlation respectively.

- In the second step CF algorithm, *ALS* is used to give recommendations on movies that are even more interesting.
- Additionally an alternate version of Item-Based similarity CF algorithm is also employed to find the most dissimilar items as to the user's liking and such items are removed if present from the combined result of the above generated recommendation set to further improve the quality and generate a final raw recommendations.
- The reason for performing Item-Based similarity CF first is that it doesn't needs to be done often and comparatively takes less time than ALS. This allows us to have less waiting time to give personalized recommendations.

G. Filtering engine

Up until now the the raw recommendations are only mapping of products to other products and users, not to specific users. This module selects those items that have high likelihood of being preferred by the user. It makes use of the learner module input to look for the closest match based on their profile.

H. Ranking engine

Raw recommendations cannot be presented to users due to the limits of representation. It is possible that due to a high number of matches a number of recommendation could be made to a user. But doing it all at once would be overwhelming and this approach also carries a risk factor if those recommendations don't get good results.

I. Some limitations:

1. Our system is not suitable for extremely spare datasets where items have very few raters in common.
2. It may take a while before new recommendations become available to the user based on their recent behavior.
3. Our system won't take into account the implications of biased/incorrect user feedback and assumes that all user feedback on the quality of recommendations provided as well as the items on the webapp are genuine. The system is susceptible to *shelling attacks* but measure can be taken to minimize it.

V. APPLICATIONS

Aside from its theoretical contribution our system is generally aimed at educating and practically improving commercial RSS in a feasible manner so that all groups of users can take advantage of it. These aspects are relevant to different stages in the life cycle of a RS, namely, the design of the system, its implementation and its maintenance and enhancement during system operation and upgrade.

Based on specific application domains, we can define two major application domain for our recommender system:

1. Entertainment - recommendations for movies, music, books, TV.
2. E-commerce - recommendations for consumers of products to buy such as books, cameras, PCs etc.

VI. CONCLUSION

This system is designed to provide media recommendations using big data analytics. In this document we have surveyed the state of the art on cross-domain recommendation, revising approaches proposed in different research areas, namely user modeling, information retrieval, knowledge management, and machine learning; aiming to characterize, classify and compare such diverse approaches. We believe that hybrid approaches can enhance the multi-domain user preference space with further content-based and contextual information, relations across domains.

REFERENCES

1) Resnick, Paul, and Hal R. Varian. "Recommender systems." Communications of the ACM 40.3 (1997): 56-58.

2) Defining Big Data: Volume, Velocity, and Variety By Judith Hurwitz, Alan Nugent, Fern Halper, and Marcia Kaufman Part of the Big Data For Dummies Cheat Sheet;

3) http://www.dummies.com/how-to/content/defining-big-data-volume-velocity-and-variety.html

4) How Big Data is used in Recommendation Systems to change our lives (15:n36); http://www.kdnuggets.com/2015/10/big-data-recommendation-systems-change-lives.html

5) The Right Recommendation System for Big Data, pg2; http://learn.fractalanalytics.com/rs/fractalanalytics/images/WP%20Big%20data%20recommendation.pdf

6) Melville, Prem, and Vikas Sindhwani. "Recommender systems." Encyclopedia of machine learning. Springer US, 2011. 829-838.

7) Ricci, Francesco, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. Springer US, 2011.

8) Pronk, Verus, et al. "Incorporating user control into recommender systems based on naive bayesian classification." Proceedings of the 2007 ACM conference on Recommender systems. ACM, 2007.

9) Rocchio, Joseph John. "Relevance feedback in information retrieval." (1971): 313-323.

10) M. Tim Jones, Recommender systems, Part 1: Introduction to approaches and algorithms, IBM Developerworks Library (2013).

11) Goldberg, David, et al. "Using collaborative filtering to weave an information tapestry." Communications of the ACM 35.12 (1992): 61-70.

12) Su, Xiaoyuan, and Taghi M. Khoshgoftaar. "A survey of collaborative filtering techniques." Advances in artificial intelligence 2009 (2009): 4.

13) Bell, Robert M., and Yehuda Koren. "Scalable collaborative filtering with jointly derived neighborhood interpolation weights." Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on. IEEE, 2007.

14) Breese, John S., David Heckerman, and Carl Kadie. "Empirical analysis of predictive algorithms for collaborative filtering." Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence. Morgan Kaufmann Publishers Inc., 1998.

15) Al Mamunur Rashid, Shyong K. Lam, George Karypis, and John Riedl. "ClustKNN: a highly scalable hybrid model-& memory-based CF algorithm." Proceeding of WebKDD (2006).

16) Chirita, Paul-Alexandru, Wolfgang Nejdl, and Cristian Zamfir. "Preventing shilling attacks in online recommender systems." Proceedings of the 7th annual ACM international workshop on Web information and data management. ACM, 2005.

17) Melville, Prem, Raymond J. Mooney, and Ramadass Nagarajan. "Content-boosted collaborative filtering for improved recommendations." AAAI/IAAI. 2002.

18) Wang, Yuanyuan, Stephen Chi-fai Chan, and Grace Ngai. "Applicability of demographic recommender system to tourist attractions: a case study on trip advisor." Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology-Volume 03. IEEE Computer Society, 2012.

19) Mahmood, Tariq, and Francesco Ricci. "Towards Learning User-Adaptive State Models in a Conversational Recommender System." LWA. 2007.

20) Bridge, Derek, et al. "Case-based recommender systems." The Knowledge Engineering Review 20.03 (2005): 315-320.

21) Felfernig, Alexander, and Robin Burke. "Constraint-based recommender systems: technologies and research issues." Proceedings of the 10th international conference on Electronic commerce. ACM, 2008.