Hybrid Approach for Recommendation System

Rohan Passi, Surbhi Jain and Pramod Kumar Singh

Abstract The primary objective of recommendation systems (RSs) is to analyze user's fondness and taste and recommend similar items to him/her. There exist various methods, e.g., user/item collaborative filtering (CF), content-based filtering (CBF), association rule mining (ARM), hybrid recommender system (HRS), for recommendations. Though these methods possess excellent characteristics, they are inefficient in providing good recommendations in particular situations. For example, item CF produces recommendations for cold-start objects; however, it typically has low accuracy compared to user CF. Conversely, user CF often provides more accurate recommendations; however, it fails to provide recommendations for cold-start objects. The hybrid methods aim to combine different approaches coherently to yield better recommendations. This paper presents an HRS based on user CF, item CF, and adaptive ARM. The proposed HRS employs ARM as a fundamental method; however, it considers only a set of users who are nearest to the target user to generate association rules (ARs) among items. Also, the support levels to mine associations among items are adaptive to the number of rules generated. Results of the study indicate that the proposed HRS provides more personalized and practical suggestions compared to the traditional methods.

Keywords Hybrid recommendation system · User collaborative filtering · Item collaborative filtering · Adaptive association rule mining

R. Passi (✉) · S. Jain · P. K. Singh
Computational Intelligence and Data Mining Research Laboratory,
ABV-Indian Institute of Information Technology and Management,
Gwalior 474015, India
e-mail: rohanpassi94@gmail.com

S. Jain
e-mail: surbhijain.iiitm@gmail.com

P. K. Singh
e-mail: pksingh@iiitm.ac.in

A. J. Kulkarni et al. (eds.), *Proceedings of the 2nd International Conference on Data Engineering and Communication Technology*, Advances in Intelligent Systems and Computing 828, https://doi.org/10.1007/978-981-13-1610-4_12

1 Introduction

RSs are information agents that mine patterns to provide recommendations for items that could be of interest to a user. Various approaches, e.g., user CF, item CF, CBF, ARM, HRS, have been proposed for recommendations. However, each one has its own benefits and shortcomings. Nonetheless, here, we discuss user CF, item CF, and ARM in brief which are of interest for this work.

1.1 User Collaborative Filtering

User CF attempts to discover users who are a close match to the target user (user for which the recommendations are being made) and uses their ratings to compute recommendations for the target user. Similarity among users is calculated using similarity scores, e.g., Euclidean distance, Pearson's correlation coefficient [1]. Various issues with user CF are as follows.
Sparsity in database: Most RSs have a significant number of users and items. However, on an average, a user rates only a small fraction of items. It leads to s sparsity in the user–item matrix. Therefore, user-based CF approach would be unable to provide any recommendations for some users [2].
False nearest neighbor: False nearest neighbor is one of the nearest neighbors who tend to be the closest match to user's preferences. However, there exists some other user who indeed is an exact match to user's behavior, and any metric used for computing nearest neighbor does not evaluate this user as the nearest neighbor. As a result, recommendations provided by the system will not be of high quality [3].
Gray sheep: Gray sheep refers to the users who have unusual taste compared to the rest of the users. They have very low correlations with other users, and their opinions do not persistently agree or disagree with any other group of users. Due to this, the gray sheep users may not receive good recommendations and may even have an adverse impact on the recommendations of other users [4].

1.2 Item Collaborative Filtering

In the item CF, the similarity between each item is computed by applying one of the similarity metrics, and then, these similarity values are used to make recommendations for a user [5]. Various issues with the item CF are as follows.
Similar recommendations: Only products similar to the input product are recommended. However, it is highly unlikely that a user shall buy a similar product again. Suppose a user buys X, then it is highly unlikely that he/she will again buy X. However, item CF would only recommend similar products; that is, it will only recommend laptops of another company [6].

Non-personalized recommendations: No preferences from the user are taken into account in the recommendation process. It will recommend a product irrespective of whether the user liked that product or not. As a result, the recommendations provided may not be of the user's interest [5].

1.3 Association Rule Mining

The ARM is used to obtain Frequent Patterns (FPs), associations, and correlations among the set of objects contained in a transaction database (DB) [7]. The ARs are generated using criteria such as support, confidence. Various issues with the ARM are as follows.

Computationally inefficient: In the case of Apriori algorithm, the cost of computation is high as it requires DB scan and candidate set generation at every iteration for finding FPs. Though FP-growth algorithm is far better than Apriori, computational time increases as the size of transaction DB increases. Therefore, FP-growth algorithm is also inefficient for large DB [7].

Weak recommendations: Sometimes very few or no recommendations are provided corresponding to a particular product. This is because very few rules are generated with high confidence, and it becomes highly infeasible to recommend a product from the rules generated [7].

Non-personalized recommendations: The ARM provides recommendations based on the previous transactions of all the users in the DB. It does not take into account the ratings given by users to different items. It evaluates the recommendations to a user by taking the transactions of those users who are conflicting to that user's behavior [7].

2 Related Work

The CF system is the earliest and the most successful recommendation method [5]. This RS is extensively used in many e-commerce industries such as Amazon and Netflix [8]. This system recommends objects to a user based on the objects that are not rated by that user but have been rated by some other similar users. In [5], the author described CF system as a person-to-person correlation system because they recommend items to a user based on the degree of interconnection between the given user and other users who have bought that item in the past. CF systems are implemented in various domains such as newsgroup articles domain—Grouplens; music domain—Ringo; movies domain—Bellcore's Video Recommender; books domain—Amazon.com; and other product domains. One of the shortcomings of the CF approach is that it must be initialized with huge data containing user preferences to provide meaningful recommendations [5]. As a consequence, this approach faces the cold-start problem when new items or new users enter the system [9].

Item-based RS is an item-to-item correlation system as it recommends similar items based on the content similarity to the items that the user liked before [10]. Examples of such systems are newsgroup filtering system—NewsWeeder; Web page recommendation system—Fab and Syskill & Webert; book recommendation system—Libra; funding recommendation system—ELFI. One of the drawbacks of item-based recommendation approach is a new user problem [11, 12]. Another drawback is over-specialization; that is, item-based approach favors objects that are similar to the objects rated by that user in the past [10]. Singh and Gupta [13] asked the user for the required features of an item to solve cold-start problem and recommended items close to their choice.

Chellatamilan and Suresh proposed a plan of recommendations for e-learning system using ARM, offering students the exceptional selection of e-learning resources [14]. Bendakir and Aïmeur presented a course RS based on ARM. The system integrates a data mining procedure with user ratings in recommendations. This system allows students to assess the previous recommendations for system enhancement and rules up gradation. On the other side, this system does not take into account student's academic background [15].

The hybrid RS integrates multiple approaches to improve recommendation performance and avoid the weaknesses of a single recommendation approach [16]. Content/collaborative hybrids are widely deployed because the rating data is already available or can be inferred from data [17]. However, this combination does not mitigate the cold-start problem as both the methods rely on rating DB given by the user. Ansari et al. [18] proposed a system based on Bayesian network approach that incorporates CF with item-based filtering approach. A Bayesian approach is also used in the system presented in [19] that recommends movies by taking into account information such as the actors starred in each movie. The advantage of this RS is that it can supply personalized recommendations even if it is deprived of a large DB of previous transactions [9]. The ARM has been widely applied in CF system to enhance the recommendation results and to solve the existing system's problems. For example, Garcia et al. [20] combined the ARM and CF recommendation methods to improve e-learning courses.

3 Architecture of the Proposed Recommendation System

The proposed approach works in four phases: computing nearest neighbors to target user, computing the similarity matrix for all the items, generating ARs, and finding recommendations based on generated ARs.

3.1 Computing Nearest Neighbors Corresponding to Target User

The first phase of the architecture is based on the user CF. It involves computation of nearest neighbors of the target user using K-Nearest Neighbor (KNN) algorithm.

The similarity between each user is measured by applying Pearson's correlation coefficient on the item ratings given by the user. It tends to obtain better outcomes in situations even when the data suffers from grade inflation and is not well-normalized [1]. Consider this dataset $x_1, \ldots, x_n$ and another dataset $y_1, \ldots, y_n$, both containing n values, then the formula for Pearson's correlation coefficient, represented by r, is given by Eq. (1).

$$r = \frac{\sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}}{\sqrt{\sum_{i=1}^{n} x_i^2 - n\bar{x}^2}\sqrt{\sum_{i=1}^{n} y_i^2 - n\bar{y}^2}} \tag{1}$$

All the items rated by a user are considered as transactions corresponding to that user. Next, the transactions of all the nearest neighbors are combined with the transaction of the target user to form the transaction DB for the target user. Thus, the transaction DB formed contains the transactions of only the relevant users.

3.2 Computing Similarity Between Items in the Database

Next step is to compute the similarity of each item with every other item in the transaction DB formed in the above phase using Jaccard index and store these values in a form of the similarity matrix. The Jaccard index is a statistic measure used for comparing the similarity and variability of sample sets. Let U and V be two finite sets, then Jaccard index is defined as shown in Eq. (2).

$$J(U, V) = \frac{|U \cap V|}{|U \cup V|} = \frac{|U \cap V|}{|U| + |V| - |U \cap V|} \tag{2}$$

3.3 Association Rule Mining to Generate Association Rules

Next phase of the proposed approach is to obtain the required ARs via FP-growth approach. The input to the FP-growth algorithm includes the transaction DB of the target user, minimum support count, and minimum confidence. Since it is tough to choose a proper minimum support count before the mining process, adaptivity in the support levels has been introduced. The algorithm adjusts the support threshold such that an adequate number of rules are generated.

3.4 Providing Recommendations to Target User

Once the strong ARs are generated, they are sorted in the descending order of their confidence values. To provide recommendations, consequents of those ARs are added to recommendation list whose antecedent completely match with the input item list. Finally, if the number of recommendations obtained from ARs is less than the required recommendations, the remaining places are filled in with the recommendation list based on the similarity values computed in phase *2*. To provide recommendations, similarity values of items in the input list are added corresponding to every item in the DB. Consequently, items with more similarity values are appended to the recommendation list generated through ARs.

4 Experimental Analysis

For evaluation purpose, MovieLens dataset containing *100,000* ratings given by *943* users to *1682* movies has been used. This section illustrates the proposed algorithm for providing recommendations with an example. Suppose there are *7 users*, namely $U_1, U_2 \ldots U_7$ and *7 movies*, $I_1, I_2 \ldots I_7$. The matrix representation of user–item DB is presented in Table 1, where each row represents a distinct user and each column represents a distinct movie. Each cell contains the rating of the movie on the scale of [*1–5*], where *1* and *5* are lowest and highest rating, respectively. Since all the users do not rate all the items, some cells are vacant. To start with, we first find the nearest neighbors to all the users using Pearson's correlation coefficient. Using Eq. (1), the matrix shown in Table 2 is computed, where each cell represents the similarity value between users. If K is *2* in KNN, nearest neighbors to all the users shall be as below.

$$\begin{array}{llll} U_1 = \{U_4, U_2\} & U_3 = \{U_2, U_5\} & U_5 = \{U_3, U_4\} & U_7 = \{U_2, U_4\} \\ U_2 = \{U_7, U_4\} & U_4 = \{U_7, U_2\} & U_6 = \{U_7, U_1\} & \end{array}$$

Transaction DB for target user U_1 will be the transactions of his *2* nearest neighbors along with his transaction as shown in Table 5. Let the movies be classified into *4* genres, namely A, B, C, and D, Table 3 shows the classification of each movie, where *1* and *0* indicate whether the movie belongs to that genre or not, respectively.

Table 1 Ratings matrix

	I_1	I_2	I_3	I_4	I_5	I_6	I_7
U_1	5	2	–	4	–	1	3
U_2	3	3	4	–	–	3	–
U_3	2	4	3	–	5	–	–
U_4	4	–	5	3	–	2	4
U_5	–	5	1	2	4	–	5
U_6	–	–	–	5	1	5	2
U_7	3	–	4	3	2	–	–

Table 2 Similarity matrix of users

	U_1	U_2	U_3	U_4	U_5	U_6	U_7
U_1	1	0.19	−0.98	0.62	−0.79	−0.28	0
U_2	0.19	1	−0.09	0.75	−0.96	−1	0.94
U_3	−0.98	−0.09	1	−0.56	0.59	−1	−0.64
U_4	0.62	0.75	−0.56	1	−0.37	−0.89	0.89
U_5	−0.79	−0.96	0.59	−0.37	1	−0.84	−0.75
U_6	−0.28	−1	−1	−0.89	−0.84	1	0.79
U_7	0	0.94	−0.64	0.89	−0.75	0.79	1

Table 3 Movies genre matrix

	A	B	C	D
I_1	0	1	0	0
I_2	1	0	1	0
I_3	0	1	1	1
I_4	0	1	0	1
I_5	0	0	1	0
I_6	1	1	0	0
I_7	1	0	1	0

Table 4 Similarity matrix of movies

	I_1	I_2	I_3	I_4	I_5	I_6	I_7
I_1	1	0	0.33	0.5	0	0.5	0
I_2	0	1	0.25	0	0.5	0.33	1
I_3	0.33	0.25	1	0.67	0.33	0.25	0.25
I_4	0.5	0	0.67	1	0	0.33	0
I_5	0	0.5	0.33	0	1	0	0.5
I_6	0.5	0.33	0.25	0.33	0	1	0.33
I_7	0	1	0.25	0	0.5	0.33	1

Table 5 Transaction DB of U_1

User	Transaction
U_1	I_1, I_2, I_4, I_6, I_7
U_4	I_1, I_3, I_4, I_6, I_7
U_2	I_1, I_2, I_3, I_6

Next step is to calculate the similarity between all the movies using Jaccard index. Similarity matrix shown in Table 4 is computed using Eq. (2). Let input item set of user U_1 is $\{ I_1, I_2 \}$ then ARs generated for user U_1 using *minimum support count = 3* and *minimum confidence = 0.7* are: $I_1 \rightarrow I_6$ and $I_6 \rightarrow I_1$. Suppose the number of rules should fall in the range [10, 20]. Since number of rules generated for *support count threshold = 3* do not fall in the specified range, decrease the value of minimum support count to *2* and minimum confidence is still *0.7*. Then, following rules are generated:

$I_1 \rightarrow I_6$	$I_1, I_2 \rightarrow I_6$	$I_4, I_6 \rightarrow I_7$	$I_7 \rightarrow I_4, I_6$
$I_3 \rightarrow I_6$	$I_1, I_3 \rightarrow I_6$	$I_4, I_7 \rightarrow I_6$	$I_3, I_6 \rightarrow I_1$
$I_6 \rightarrow I_1$	$I_1, I_4 \rightarrow I_6$	$I_6, I_7 \rightarrow I_4$	$I_2, I_6 \rightarrow I_1$
$I_2 \rightarrow I_1, I_6$	$I_1, I_7 \rightarrow I_6$	$I_4 \rightarrow I_6, I_7$	
$I_1, I_6, I_7 \rightarrow I_4$	$I_1, I_4, I_6 \rightarrow I_7$	$I_1, I_4, I_7 \rightarrow I_6$	

To provide recommendations to target user U_1, first, consequents of those rules are added whose antecedents completely matches with the items in the input list. Therefore, I_6 is added to recommendation list owing to the presence of rule $I_1, I_2 \rightarrow I_6$. Then, if any of the subsets of input item set is present in the antecedent of the rules, its consequents are added to the recommendation list. Since rules $I_1, I_4, I_6 \rightarrow I_7$ and $I_1, I_6, I_7 \rightarrow I_4$ contain I_1 in their antecedents, now, the recommendation list shall be $\{ I_6, I_7, I_4 \}$. Since *4* movies are to be recommended to the user U_1, for every item in input item set, their respective rows of Jaccard index obtained in Table 4 corresponding to every movie are added. The list thus obtained is sorted in the descending order of the total value of Jaccard index of each movie. The remaining places in recommendation list are filled up by the items having highest total value. Adding values of Jaccard index for item I_1 and I_2 will result in values as shown in Table 6. Sorting the movies according to their total value of Jaccard index for items in the input list will result in the following order of movies.

$$\{ \{ I_1 : 1\} , \{ I_2 : 1\} , \{ I_7 : 1\} , \{ I_6 : 0.83\} , \{ I_3 : 0.58\} , \{ I_4 : 0.5\} , \{ I_5 : 0.5\} \}$$

Though movies I_1 and I_2 have the maximum similarity value, they are not added to recommendation list as they belong to the input set. As movies I_7 and I_6 are already present in the recommendation list, movie I_3 is added to the list. Therefore, final recommendations provided to the user U_1 are $\{ I_6, I_4, I_7, I_3 \}$.

Table 6 Jaccard coefficient for I_1 and I_2

	I_1	I_2	I_3	I_4	I_5	I_6	I_7
I_1	1	0	0.33	0.5	0	0.5	0
I_2	0	1	0.25	0	0.5	0.33	1
Total	1	1	0.58	0.5	0.5	0.83	1

5 Results

To compare the accuracy, the ranking of various items have been considered rather than rating prediction of the items [21]. In the test data, the ratings given by a user to a particular item are already known. A list of items $list_1$ based on the ratings given by the user to the items is generated in descending order. Another ranked list of items $list_2$ is also generated according to the recommendations provided by the proposed system. Then, two sets S_1 and S_2 are made from the items in the upper half of the ranked list $list_1$ and $list_2$, respectively. A new set S_3 is generated from the intersection of the two sets S_1 and S_2 as shown in Eq. (3).

$$S_3 = S_1 \cap S_2 \tag{3}$$

Consider the cardinality of the set S_3 be n. Now, the accuracy of a RS may be defined as shown in Eq. (4).

$$\text{Accuracy} = \frac{n}{N} \times 100 \tag{4}$$

Figure 2 compares the accuracy of the proposed RS with other existing RSs.

5.1 Comparison with User Collaborative Filtering

The proposed architecture deals with the problem of a false nearest neighbor not only by merely depending upon the nearest neighbors' ratings to a product. But also, the items reviewed by the nearest neighbors have been combined to form a transaction DB of the target user. Then, the ARM has been applied on the relevant set of transactions to get FPs and associations for a particular item. Thus, the results obtained are efficient and more personalized. The proposed architecture also handles the gray sheep problem effectively. The partial behavior of all those users who somewhat resembles the gray sheep user has been considered properly. On an average, the user CF took *0.44* s, while the proposed system took *0.57* s to provide recommendations.

5.2 Comparison with Item Collaborative Filtering

The transactions of nearest neighbors of the target user have been taken into account to solve the problem of non-personalized recommendations. The user's transaction set includes only those users' transactions who are highly coupled to the user's behavior. The issue of vague recommendations has also been solved as every user has its unique transaction set. The recommendations are generated after carefully studying the transaction DB of the user. As the last phase of the proposed system involves ARM, the problem of recommending only similar products is also solved. Recommendations provided are based on what a customer would need next if she has already bought a product, i.e., user's history played a key role in generating recommendations. The results reveal that the proposed architecture outperforms the item CF system regarding accuracy and personalization and, however, has a slight overhead in computation time. On an average, item CF took *0.31* s for generating recommendations, while the proposed RS took *0.57* s.

5.3 Comparison with Association Rule Mining

In the proposed architecture, the DB size has been significantly reduced by using the transaction DB of nearest neighbors and the target user only. This improvement in reduction of DB significantly improves the performance of the proposed RS; the average computation time required by traditional ARM algorithms was *3.23* s while the proposed RS took only *0.57* s. Also, the proposed system takes into account only relevant data; therefore, it generates more efficient and personalized recommendations which are relevant to the context of the user's preferences. Unlike existing ARM that requires minimum support threshold of the rules to be specified previously before the mining process, the proposed RS adjusts the value of threshold at the time of mining so that the number of ARs generated falls within the specified range. This ensures that enough rules are generated so that recommendations are always provided to the user.

Table 7 compares the time required by traditional RS architectures and the proposed RS based on the number of items present in the input item list. Figure 1 compares the average time required by the existing RSs and the proposed RS. The proposed RS takes a little more time than user CF and item CF; however, the recommendations provided are more personalized and accurate. Compared to RSs that uses traditional ARM to provide recommendations, the proposed RS is computationally more efficient and requires less time to provide recommendations to the users (Fig. 2).

Table 7 Time required by recommendation systems

	User CF system (s)	Item CF system (s)	ARM system (s)	Proposed system (s)
1 Items	0.33	0.20	3.12	0.49
2 Items	0.46	0.32	3.25	0.56
3 Items	0.53	0.41	3.32	0.67
Avg. time	0.44	0.31	3.23	0.57

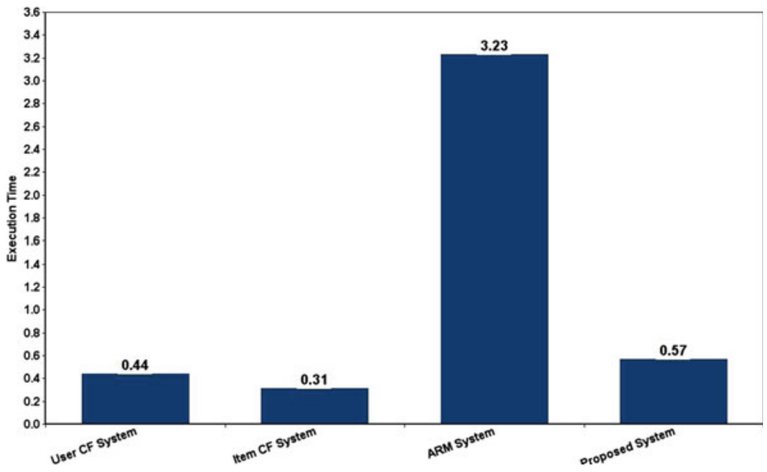

Fig. 1 Average time of recommendation systems

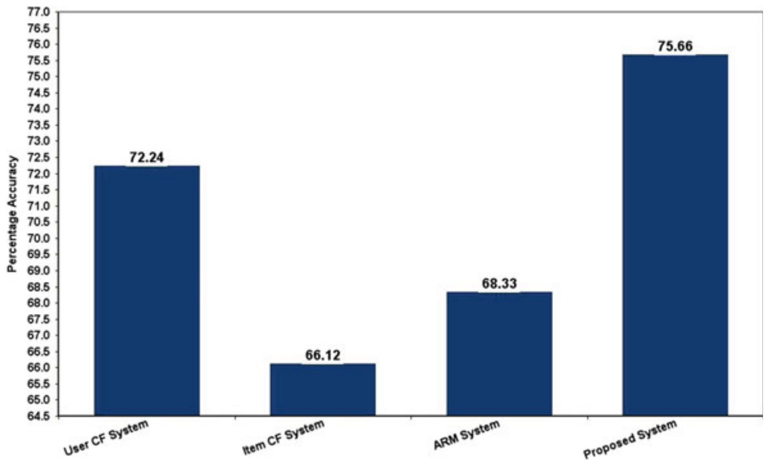

Fig. 2 Accuracy of different recommendation systems

6 Conclusion

All traditional RSs are incapable to the user's taste and preferences. Therefore, an architecture has been proposed that is adequate, as it deals with the problems incurred by all the traditional RSs. The proposed RS combines the positive parts of traditional RSs to overcome the gray sheep problem, the problem of false nearest neighbor, the effect of irrelevant users and recommendations of similar items only. The experiment reveals that the proposed RS successfully overcomes the problems associated with the traditional RSs.

References

1. Ricci F, Rokach L, Shapira B (2011) Introduction to recommender systems handbook. Springer, Berlin
2. Guo G, Zhang J, Thalmann D (2014) Merging trust in collaborative filtering to alleviate data sparsity and cold start. Knowl-Based Syst 57:57–68
3. Liu H, Hu Z, Mian A, Tian H, Zhu X (2014) A new user similarity model to improve the accuracy of collaborative filtering. Knowl-Based Syst 56:156–166
4. Srivastava A (2016) Gray sheep, influential users, user modeling and recommender system adoption by startups. In: Proceedings of the 10th ACM conference on recommender systems, pp 443–446
5. Shi Y, Larson M, Hanjalic A (2014) Collaborative filtering beyond the user-item matrix: a survey of the state of the art and future challenges. ACM Comput Surv (CSUR) 47(1):3
6. Poriya A, Bhagat T, Patel N, Sharma R (2014) Non-personalized recommender systems and user-based collaborative recommender systems. Int J Appl Inf Syst 6(9):22–27
7. Witten IH, Frank E, Hall MA, Pal CJ (2016) Data mining: practical machine learning tools and techniques. Morgan Kaufmann, Cambridge
8. Herlocker JL, Konstan JA, Borchers A, Riedl J (1999) An algorithmic framework for performing collaborative filtering. In: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval. ACM, pp 230–237
9. Lika B, Kolomvatsos K, Hadjiefthymiades S (2014) Facing the cold start problem in recommender systems. Expert Syst Appl 41(4):2065–2073
10. Ekstrand MD, Riedl JT, Konstan JA et al (2011) Collaborative filtering recommender systems. Found Trends® Hum Comput Interact 4(2):81–173
11. Felfernig A, Burke R (2008) Constraint-based recommender systems: technologies and research issues. In: Proceedings of the 10th international conference on electronic commerce. ACM
12. Wei K, Huang J, Fu S (2007) A survey of e-commerce recommender systems. In: 2007 International Conference on service systems and service management. IEEE, pp 1–5
13. Singh PK, Gupta N (2016) Recommendation model for infrequent items. In: Proceedings of the first New Zealand text mining workshop. TMNZ (Online). Available: http://tmg.aut.ac.nz/tmnz2016/papers/Pramod2016.pdf
14. Chellatamilan T, Suresh R (2011) An e-learning recommendation system using association rule mining technique. Eur J Sci Res 64(2):330–339
15. Bendakir N, Aïmeur E (2006) Using association rules for course recommendation. In: Proceedings of the AAAI workshop on educational data mining, vol 3
16. Sharma L, Gera A (2013) A survey of recommendation system: research challenges. Int J Eng Trends Technol (IJETT) 4(5):1989–1992
17. Nilashi M, bin Ibrahim O, Ithnin N (2014) Hybrid recommendation approaches for multi-criteria collaborative filtering. Expert Syst Appl 41(8):3879–3900
18. Ansari A, Essegaier S, Kohli R (2000) Internet recommendation systems. J Mark Res 37(3):363–375
19. Beutel A, Murray K, Faloutsos C, Smola AJ (2014) Cobafi: collaborative Bayesian filtering. In: Proceedings of the 23rd international conference on World wide web. ACM, pp 97–108
20. García E, Romero C, Ventura S, De Castro C (2009) An architecture for making recommendations to courseware authors using association rule mining and collaborative filtering. User Model User-Adap Inter 19(1–2):99–132
21. Steck H (2013) Evaluation of recommendations: rating-prediction and ranking. In: Proceedings of the 7th ACM conference on recommender systems. ACM, pp 213–220