Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 115 (2017) 853–862

Procedia Computer Science

www.elsevier.com/locate/procedia

7th International Conference on Advances in Computing & Communications, ICACC-2017, 22-24 August 2017, Cochin, India

Handling Natural Noise in Multi Criteria Recommender System utilizing effective similarity measure and Particle Swarm Optimization

Priyankar Choudhary[a], Vibhor Kant[a,*], Pragya Dwivedi[b]

[a]Department of Computer Science & Engineering. The LNM Institute of Information Technology, Jaipur 303031, India
[b]Department of Computer Science & Engineering. Motilal Nehru National Institute of Technology, Allahabad 211004, India

Abstract

Multi criteria recommender systems generate quality recommendations to users by incorporating criteria ratings into recommender system using collaborative filtering because ratings over multiple criteria can capture user preferences efficiently. However, aggregation of similarities computed on multiple criteria is still a major concern. Moreover, the concept of natural noise is an emerging trend that is related to inconsistent behaviour of users. Our work in this paper is an attempt towards developing multi criteria recommender systems that deals with inconsistent ratings and uses particle swarm optimization to learn optimal weights for a user over different criteria in the aggregation process.

© 2017 The Authors. Published by Elsevier B.V.
Peer-review under responsibility of the scientific committee of the 7th International Conference on Advances in Computing & Communications.

Keywords: multi criteria recommender system; collaborative filtering; noisy ratings; similarity aggregation; particle swarm optimization

1. Introduction

Recommender system (RS), one of the most promising web personalization tools, assists users by providing suitable recommendations based on their preferences after extracting useful information from a large corpus of available

* Corresponding author. Tel.: +91-9413172602
E-mail address: vibhor.kant@gmail.com

1877-0509 © 2017 The Authors. Published by Elsevier B.V.
Peer-review under responsibility of the scientific committee of the 7th International Conference on Advances in Computing & Communications
10.1016/j.procs.2017.09.168

information [1]. This facility of web application is widely used by various e-commerce websites for recommending books (Amazon.com), movies (MoviesLens, Netflix), restaurants (Entree) and jokes (Jester) [2]. Usually, Collaborative Filtering (CF) and Content Based Filtering (CBF) are employed to build RS. CF generates recommendations to users based on their similar users while CBF recommends items similar to ones that use has liked in the past [3], [4].

Most of the existing recommender systems suggest items to users based on single criterion i.e. the overall rating. However, users may notice multiple criteria while rating an item, so overall rating does not reveal the actual user behavior. Therefore, integration of ratings on multiple criteria into RS can lead to capture users' behavior efficiently and can generate more effective recommendations to users.

Multi-criteria Recommender Systems (MCRS) generates effective recommendations to users by incorporating various criteria ratings by using CF. To compute total similarity between users, various aggregation functions (maximum, average, minimum) are used. But these functions are not suitable as they do not reveal the optimal weights of a user on various criteria [5]. Therefore, learning the optimal weights of users on various criteria in the process of aggregation is a big challenge. Few researchers have tried to learn these weights using linear regression, genetic algorithms and support vector regression [2], [6]. Further, the success of CF is highly dependent on similarity measures such as Pearson and Cosine. However, these similarity measures are not more effective because they are symmetric and treat positive and negative ratings equally. Numerous benefits can be gained if positive and negative rating are treated independently in a similarity measure [7], [8]. Therefore, selection of appropriate similarity measure is also an area of concern.

In the past, researchers have considered the dataset ratings as ground truth values and free from the noise and did not use any data preprocessing step to deal with noise. The presence of noisy ratings in dataset can lead to improper recommendations. O'Mahony et al. [9] developed an approach which enables the concept administrator to detect noisy ratings in recommender system database for overall rating. Amatriain et al. [10] have studied and characterized the noise caused by user when providing ratings. They analyzed users' ratings at different time interval to deal with the issue of noisy ratings. Toledo et al. [11] proposed an approach to detect and correct noisy rating by using user profile. But in literature survey, we found that researchers are focusing to deal with noise only in only overall rating and they have not tried to incorporate criteria ratings. To the best of our knowledge, no one has tried to manage the natural noise in MCRS. Our work is to develop an effective multi criteria recommendation framework which deals with the issue of noisy ratings associated with criteria ratings and aggregation of similarities.

The rest of this paper is organized as follows: Section 2 defines background details related to our proposed approach which is discussed in Section 3. Section 4 presents the experimental evaluation of our proposed approach. Last Section concludes our work having some future remarks

2. Background

All This section briefly discusses about multi criteria recommender system, noise in recommender system and fundamentals of particle swarm optimization (PSO).

2.1. Multi Criteria Collaborative Filtering (MCCF)

CF is the most popular and widely used technique in recommender system. It computes similarities among the preferences of various users to recommend the suitable items by automating the process of "Word of Mouth". In MCCF, rating function can be expressed as user × item -->$R_0 \times R_1 \times R_2 \ldots R_k$, here R_0 is overall rating [2-5] and R_i is the rating for individual criteria where $i \in \{1,2,3,\ldots k\}$. In MCRS recommendations can be generated using three phases namely, total similarity computation, neighborhood generation and prediction of unseen items.

2.1.1 Total Similarity Computation

MCCF computes the similarities between active user u_x and another user u_y based on each criterion using following widely accepted similarity functions such as Pearson and Jaccard which are expressed in the equation [7], [8]:

$$\text{sim\_pearson}^c(u_x, u_y) = \frac{\sum_{s \epsilon S_{x,y}} (r_{x,s} - m_x)(r_{y,s} - m_y)}{\sqrt{\sum_{s \epsilon S_{x,y}} (r_{x,s} - m_x)^2} \sqrt{\sum_{s \epsilon S_{x,y}} (r_{x,s} - m_x)^2}} \quad (1)$$

where $S_{x,y}$ is the set of mutual items which are rated by both users. $r_{x,s}$ is the rating given by user u_x on co-rated item s. $r_{y,s}$ is the rating given by user u_y on co-rated item s. m_x, m_y are mean ratings for user $u_x$and u_y respectively. Formula for Jaccard similarity to compute similarity between active user u_x and another user u_y is described as follows.

$$\text{sim\_jaccard}^c(u_x,u_y) = \frac{(\frac{p}{p+q+r}) + (\frac{n}{n+q+r})}{2} \quad (2)$$

where p is the number of ratings that are positive for both user u_x and u_y, q is the number of ratings that are positive for user u_x and negative for u_y, r is the number of ratings that are negative for user u_x and positive for u_y, n is the number of ratings that are negative for both user u_x and u_y.

After computing similarity based on each criterion, MCCF aggregates these similarities using various aggregation functions namely, average, minimum and maximum. However, these aggregation functions are not appropriate i.e. average gives equal priority to each criterion. Many researchers have tried to aggregate these similarities by learning optimal weights through various machine learning approaches. We have tried to learn these optimal weights for a user using PSO.

2.1.2 Neighbourhood Set Formation

After computing similarities among users, MCCF selects top k similar users in the neighborhood set N for an active user.

2.1.3 Prediction and Recommendations

Based on neighborhood set N, it predicts a rating $r^k{}_{x,i}$ to an active user u_x for item i by using the following formula [15]:

$$r^k{}_{x,i} = m_x + \frac{\sum_{y \epsilon N} (r_{y,i} - \overline{r_y}) * similarity(u_x, u_y)}{\sum_{y \epsilon N} |similarity(u_x, u_y)|} \quad (3)$$

where N is the set of neighbors who have rated the item i. $\overline{r_y}$ is the mean rating of user u_y. Finally, top predicted items will be recommended to an active user u_x.

2.2. Noise in Collaborative Filtering

Noise in recommender system is mainly divided into two categories namely, malicious noise and natural noise. Malicious noise is introduced by an attacker to bias the recommendations having wicked intentions while natural noise is introduced unintentionally by a user depending upon many factors. Numerous work has been done to deal with the issue of malicious noise but area of natural noise is less investigated. Categorizing of malicious noise is an easy task as attacker has some certain pattern. On the other hand, there are many potential reasons behind generation of natural noise. Therefore, handling these natural noises is a challenging task in RS.

Pham et al. [12] and Amatriain et al. [13] explained two reasons for existence of natural noises in recommender system datasets: (1) Alteration in user preference over a period of time. (2) Impact of several factors on a user like personal conditions, social influence, emotional states and context. Amatriain et al. also observed degree of inconsistencies in users' ratings for items at different time intervals that varied from one to fifteen days. They concluded that inconsistencies appear due to change in users' priorities and users' imprecision. Many researchers

[12],[14], [15] corrected noisy data by using some additional information beyond ratings. O'Mahony et al. [9] and R.Y. Toledo et al. [17] tried to handle noisy data problem and did not use any information beyond ratings. But all the work has been done on overall ratings which is based on single criteria. Our aim is to handle the issue of natural noise on multi criteria without using any information beyond rating.

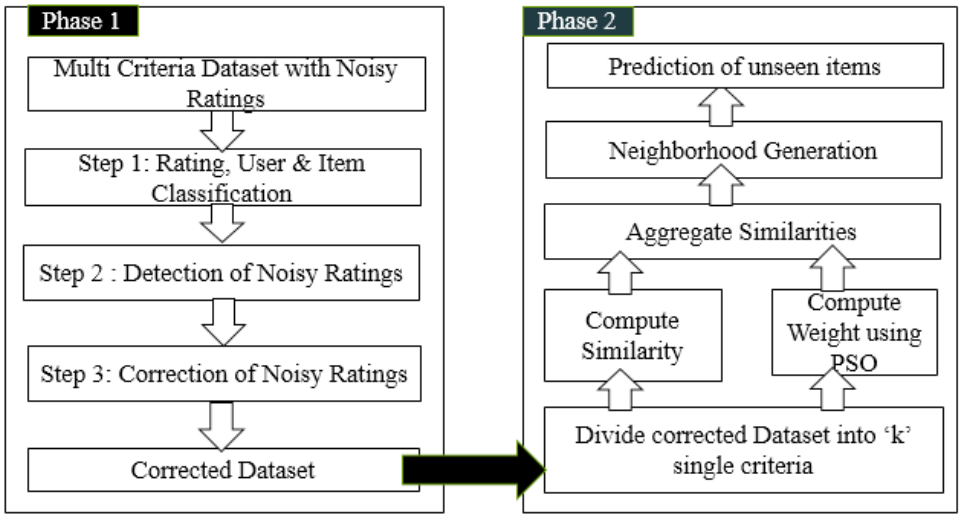

Fig. 1: Various Phases of Proposed Multi Criteria Recommendation Framework

2.3. Particle Swarm Optimization

PSO is based on the paradigm of swarm intelligence and it is inspired by social behaviour of animals like fish and birds. PSO is a simple but powerful algorithm and successfully applied in various fields of science. PSO [16-18] is a population based meta heuristic approach that optimizes a problem by iteratively improving initial candidate solutions with respect to a quality function. It starts with initial population randomly, that is collection of candidate solution called particles. Each particle in a population is expressed in the form of a position as well as a velocity vector. These particles move around the search space over subsequent generations according to some mathematical expression based on the particle's position and velocity. The movement of each particle is influence by its local personal best and global best position each particle keeps track of its personal best position and its velocity. Position and velocity of particle are updated according to personal best of particle and global best value of particles.

3. Proposed Approach

This section describes our proposed multi criteria recommendation framework MCRS-PDC which deals with the issue of noisy ratings for enhancing recommendation quality using dice coefficient and PSO. Let U = $\{\boldsymbol{u_1}, \boldsymbol{u_2}, \boldsymbol{u_3}, \ldots\ldots, \boldsymbol{u_m}, \}$ be the set of m users, I = $\{\boldsymbol{i_1}, \boldsymbol{i_2}, \boldsymbol{i_3}, \ldots\ldots\ldots\ldots, \boldsymbol{i_n}\}$ be the set of n items and C = $\{\boldsymbol{c_1}, \boldsymbol{c_2}, \boldsymbol{c_3}, \ldots\ldots\ldots, \boldsymbol{c_k}\}$ be the set of k criteria. The rating vector for an active user u for items i is represented as $\boldsymbol{r_{u,i}}$ = $\{\boldsymbol{r^0}_{u,i}, \boldsymbol{r^1}_{u,i} \boldsymbol{r^2}_{u,i} \boldsymbol{r^3}_{u,i} \ldots\ldots\ldots \boldsymbol{r^k}_{u,i}\}$. Where $\boldsymbol{r^0}_{u,i}$ is the overall rating given by user u on item i and $\boldsymbol{r^k}_{u,i}$is the rating given by user u on criteria k of item i. Our proposed multi criteria recommendation framework consists of following two phases which are depicted in figure 1.

Phase 1: Pre-processing Phase – Detection and Correction of Noisy Ratings
Phase 2: Recommendation Phase using corrected dataset

3.1. Phase 1: Pre-processing Phase (Detection and Correction of Noisy Ratings)

In this phase, we classify ratings, users as well as items and follow a mechanism that detects a noisy rating and correct them. It has three phases which are defined below [11].

Step 1: Classification of Ratings, Users and Items

In this step, we classify ratings, users and items into distinct classes.

Rating Classification:

Usually, ratings provided by users are expressed on a scale that show their interest in an item. We classify these ratings into three categories namely, weak, average, strong based on the following threshold value k (weak - average threshold) and v (average - strong threshold). These thresholds will be computed as follows:

$$k = \min R + \text{round}((1/3)\times(\max R - \min R))$$
$$v = \max R - \text{round}((1/3)\times(\max R - \min R))$$

where minR and maxR are the minimum and maximum possible value of ratings on a rating scale respectively. Classification of ratings can be expressed as:

Weak ratings: if $r^k_{u,i} < k$.
Mean ratings: if $k \leq r^k_{u,i} < v$
Strong ratings: if $r^k_{u,i} \geq v$

User Classification:

We classify users into four distinct categories namely, kind user, average user, critical user and inconsistent user based on the following defined sets.

- W_u = Collection of weak ratings provided by user $u = \{r^k_{u,i} | \forall i \in I$ where $r^k_{u,i} < k\}$
- Au = Collection of average ratings provided by user $u = \{ r^k_{u,i} | \forall i \in I$ where $k \leq r^k_{u,i} < v\}$
- Su = Collection of strong ratings provided by user $u = \{ r^k_{u,i} | \forall i \in I$ where $r^k_{u,i} \geq v\}$

User classification is represented in Table 1 based on the cardinalities of these defined sets W_u, Au and Su.

Item Classification:

Here, we classify items into four categories namely, less preferred, highly preferred, average preferred and inconsistent.

- W_i = Set of weak ratings assigned to item $i = \{ r^k_{u,i} | \forall u \in U$ where $r^k_{u,i} < k\}$
- A_i = Set of average ratings assigned to item $i = \{ r^k_{u,i} | \forall u \in U$ where $k \leq r^k_{u,i} < v\}$
- S_i =Set of strong ratings assigned to item $i = \{r^k_{u,i} | \forall u \in U$ where $r^k_{u,i} \geq v\}$

Item classification is represented in Table 1 based on the cardinalities of these defined sets W_i, Ai and Si. These classifications depend on the cardinality of a set with respect to the combined cardinalities of the remaining two sets which is explained in Table 1.

Step 2: Detection of Noisy Ratings

After classification of ratings, users and items, we analyze the presence of contradiction among the classes. So, we define a group of three homologous classes associated with users, items and ratings which are explained in Table 2.

Based on the Table 2, we follow the following three rules for a rating to be a noisy rating [12]:

Rule 1: If user 'u' is critical user, item 'i' is weakly preferred and $r^k{}_{u,i} \geq$ k then $r^k{}_{u,i}$ is a noise.
Rule 2: If user 'u' is average user, item 'i' is averagely preferred and $r^k{}_{u,i}$ < k or $r^k{}_{u,i} \geq$ v then $r^k{}_{u,i}$ is a noise.
Rule 3: If user 'u' is kind user, item 'i' is strongly preferred and $r^k{}_{u,i}$<v then $r^k{}_{u,i}$ is a noise.

Based on the above rules, we form a set of possible noisy ratings.

Table 1: User and Item Classification

User Classification		Item Classification	
Critical User	|Wu| ≥ |Au| + |Su|	Less preferred	|Wi| ≥ |Ai| + |Si|
Average User	|Au ≥ |Wu| + |Su|	Average preferred	|Ai| ≥ |Wi| + |Si|
Kind User	|Su| ≥ |Wu| + |Au|	High preferred	|Si| ≥ |Wi| + |Ai|
Inconsistent User	Doesn't fulfil above conditions	Inconsistent preferred	Doesn't fulfil above conditions

Table 2: Homologous Classes

	User Class	Item Class	Rating Class
Group 1	Critical	Less - preferred	Weak
Group 2	Average	Average - preferred	Average
Group 3	Kind	High - preferred	Strong

Step 3: Noisy Rating Correction

After construction of the set of possible noisy ratings, we correct these noisy ratings using traditional collaborative filtering. we predict a new rating $r^{k*}{}_{u,i}$ for each rating in set of possible noises. We modify a possible noisy rating $r^k{}_{u,i}$ by new rating $r^{k*}{}_{u,i}$ if abs ($r^k{}_{u,i}$ -$r^{k*}{}_{u,i}$) >1 . By following this procedure, we can generate corrected data from original data [12].

3.2. Phase 2 (Recommendation Phase using corrected dataset)

In this phase, we generate recommendations to active users based on the corrected user-item rating matrix. It has following steps

Step 1. Similarity Computation & Aggregation

Our proposed system computes the similarity between active users u_x and other user u_y for each criterion by utilizing dice similarity coefficient which is expressed as follows [7], [8]:

$$\text{sim\_dice}^c(u_x,u_y) = \frac{\left(\frac{2p}{2p+q+r}\right)+\left(\frac{2n}{2n+q+r}\right)}{2} \quad (4)$$

where p is the number of ratings that are positive for both user u_x and u_y, q is the number of ratings that are positive for user u_x and negative for u_y, r is the number of ratings that are negative for user u_x and positive for u_y, n is the number of ratings that are negative for both users.

After computing similarities among users for each criterion, the system will compute total similarity among users by aggregating these similarities using following equation:

$$Total\_similarity\ (u_x, u_y) = \sum_{c\in\{1,\ldots,k\}} w_c \times similarity^c(u_x, u_y) \quad (5)$$

Where w_c is the weight for criteria c. We have used PSO to learn optimal weight w_c for each criterion that is described as follows.

Step 1.1. (Particle Representation and Initial Population)

Each criteria weight is represented by 8 binary digits having range from 0 to 255. Since we have to learn 'k' weights for each criterion therefore the particle is represented by 8k bits. After weight computation, we divide each weight by the total weight to normalize them. Each weight can be obtained by converting binary digit into its corresponding decimal value [16], [17].

Step 1.2. (Particle Dynamics)

PSO contains a population of candidate solution called swarm where each candidate solution is considered as particle. These particles move over the search space by using some velocity. The algorithms update the entire swarm at each time stamp by updating the velocity and position of each particle in every dimension by the following rule [16-18]

$$velocity_i = \text{w}\ velocity_i + c_1.r_1(pos_{PBest,i} - pos_i) + c_2.r_2(pos_{GBest} - pos_i) \quad (6)$$

If $(|\ velocity_i\ | > velocity_{max})$,

$$velocity_i = (velocity_{max}/|\ velocity_i\ ).velocity_i \ ; \quad pos_i = pos_i + velocity_i \quad (7)$$

where pos_i is the position of current particle i , $pos_{PBest,i}$ is the best position attained by particle i, pos_{GBest} is the swarm's global best, $velocity_i$ is the velocity of particle i, w is the random inertia weight between 0.5 and 1, c_1 and c_2 are spring constant whose values are set to 1.494, r_1 and r_2 are random numbers between 0 and 1.

Step 1.3. (The Fitness Function)

The following fitness function is used to compute the fitness value of each particle in the swarm [15], [18]:

$$Fitness = \frac{1}{n_R}\sum_{j=0}^{n_R} |ar_j - pr_j| \quad (8)$$

where n_R is the cardinality of a training set of as given active user. ar_j and pr_j are actual and predicted rating on item j respectively.

Step 1.4. (Termination Condition)

We select a predefined number of iteration as termination criteria. When number of iteration reaches to its maximum value, PSO algorithm terminates.

Step2. (Neighborhood Set Formation)

After computing total similarity between active user and other users using PSO, we will select top K similar user in the neighborhood set. The size of the neighborhood set is selected experimentally.

Step 3. (Prediction and Recommendations)

The selected neighborhood set N is used to predict the ratings of all unseen items for an active user using equation 2. Finally, top predicted items can be recommended to the active user.

4. Experiments and Results

To validate the effectiveness of our proposed approach, we conducted experiments on Yahoo Movie dataset [2].

4.1. Design of Experiment

Yahoo Movie dataset consists of 6078 users and 976 items. Items are rated over 4 criteria and each item has its own overall rating also. For our experiments, we have divided each dataset into five different splits namely split-1, split-2, split-3, split-4, split-5. Each split has 1197 users. We divided the ratings of each user into two disjoint sets, training set (60%) and testing set (40%). Training set was used to train the system and testing set was used to check the effectiveness of the proposed systems. We selected top 20 % most similar users for each active user experimentally.

4.2. Performance Evaluation

We have used following performance measure for the evaluation of our proposed approaches: Precision is defined as measure of correctness; Recall is measure of completeness; F-measure is the combination of precision & recall and coverage is defined as number of unknown ratings that are predicted by the system [19].

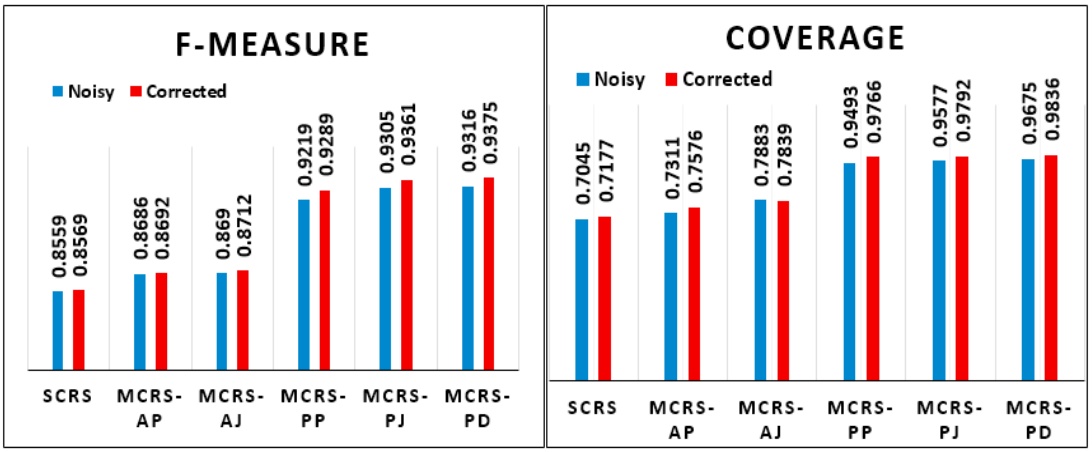

Fig. 2. Analysis of (a) F-measure (b) Coverage of Various approaches

4.3. Experiments

To test the effectiveness of our proposed approach MCRS-PDC, we compared our approach with following approaches.

- Single Criteria Recommender System (SCRS)
- MCRS on Average with Pearson and Jaccard similarities respectively MCRS-AP, MCRS-AJ
- MCRS on PSO using Pearson and Jaccard similarities are respectively MCRS-PP, MCRS-PJ
- Single Criteria Recommender System on Corrected dataset (SCRS-C)
- MCRS on Average with Pearson and Jaccard similarities respectively MCRS-APC, MCRS-AJC
- MCRS on PSO with Pearson and Jaccard similarities are respectively MCRS-PPC, MCRS-PJC

Table 3: Comparison of Precision over various approaches

SCRS	MCRS -AP	MCRS -AJ	MCRS -PP	MCRS -PJ	MCRS -PD	SCRS -C	MCRS -APC	MCRS -AJC	MCRS -PPC	MCRS -PJC	MCRS-PDC
0.8516	0.8572	0.8638	0.8930	0.9142	0.9142	0.8632	0.8652	0.8683	0.9131	0.9196	0.9346
0.8563	0.8642	0.8713	0.8997	0.9141	0.9159	0.8717	0.8775	0.8823	0.9148	0.9212	0.9286
0.8548	0.8645	0.8678	0.9304	0.9248	0.9349	0.8584	0.8752	0.8980	0.9357	0.9385	0.9340
0.8379	0.8645	0.8590	0.8765	0.8919	0.9431	0.8468	0.8772	0.8714	0.9311	0.9173	0.9402
0.8423	0.8672	0.8702	0.9074	0.9387	0.9034	0.8588	0.8854	0.8919	0.9130	0.9486	0.9243
0.8485	0.8635	0.8664	0.9014	0.9167	0.9223	0.8598	0.8761	0.8824	0.9215	0.9290	0.9323

Table 4: Comparison of Recall over various approaches

SCRS	MCRS -AP	MCRS -AJ	MCRS -PP	MCRS -PJ	MCRS -PD	SCRS -C	MCRS -APC	MCRS -AJC	MCRS -PPC	MCRS -PJC	MCRS-PDC
0.8679	0.8507	0.8738	0.9424	0.9464	0.9453	0.8679	0.8700	0.8871	0.9699	0.9704	0.9745
0.8518	0.8901	0.8859	0.9134	0.9476	0.9405	0.8633	0.8823	0.8715	0.9696	0.9720	0.9784
0.8550	0.8844	0.8837	0.9193	0.9650	0.9169	0.8748	0.8972	0.9032	0.9204	0.9366	0.9346
0.8788	0.8719	0.8705	0.9474	0.9701	0.9788	0.8648	0.8917	0.8840	0.9640	0.9582	0.9580
0.8514	0.8722	0.8652	0.9160	0.9118	0.9641	0.8755	0.8720	0.8859	0.9491	0.9722	0.9728
0.8610	0.8739	0.8758	0.9277	0.9482	0.9491	0.8693	0.8826	0.8863	0.9546	0.9619	0.9637

4.4. Results

To validate the effectiveness of our proposed approach MCRS-PDC, we examined the results on precision, recall, f-measure, coverage. In Table 3 and 4, last row indicates the average performance over five splits of various approaches. Based on the tables, we can say that there is remarkable improvement in results after dealing with noisy ratings. Our assumption that noisy ratings can degrade recommendation quality, turns out to be true. We can also analyze from figure 2 that the higher value of f-measure and coverage indicate the better performance of the system. In the figure 2, two adjacent bars indicate the performance of an approach over noisy and corrected dataset respectively over f-measure and coverage. The right most column in Figure 2 (a) and 2 (b) indicates the result of our proposed approach MCRS-PDC over f-measure and coverage. The comparison of experimental results depicted from figure 2 against those obtained using original and corrected dataset clearly indicates the superiority of the proposed approach MCRS-PDC.

5. Conclusion

In this work, we have tried to develop a multi criteria recommender system which deals with natural noise to improve the recommendation quality using users' ratings only. Our system firstly detects the noisy ratings based on the classification of ratings, users and items and then correct these ratings. Further, we generated quality recommendations through CF for each criterion where similarities, captured through dice coefficient, are aggregated using particle swarm optimization for constructing a neighborhood set. Finally, recommendations are generated to users based on these neighbors, the results obtained from experimental section indicate that it is possible to improve the accuracy of recommendation methods through the detection and correction of noisy ratings. Experimental results on YahooMovie dataset demonstrated the effectiveness of our proposed approach MCRS-PDC over other classical approaches in terms of precision, recall, f-measure and coverage.

References

[1] Dwivedi P, Bharadwaj KK. Effective Trust-aware E-learning Recommender System based on Learning Styles and Knowledge Levels. Educational Technology & Society. 2013 Oct 1;16(4):201-16.

[2] Jhalani T, Kant V, Dwivedi P. A Linear Regression Approach to Multi-criteria Recommender System. In International Conference on Data Mining and Big Data 2016 Jun 25 (pp. 235-243). Springer International Publishing.

[3] Adomavicius G, Tuzhilin A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE transactions on knowledge and data engineering. 2005 Jun;17(6):734-49.

[4] Bobadilla J, Ortega F, Hernando A, Gutiérrez A. Recommender systems survey. Knowledge-based systems. 2013 Jul 31;46:109-32.

[5] Adomavicius G, Kwon Y. Multi-criteria recommender systems. In Recommender systems handbook 2015 (pp. 847-880). Springer US.

[6] Parveen R, Kant V, Dwivedi P, Jaiswal AK. Enhancing Recommendation Quality of a Multi Criterion Recommender System Using Genetic Algorithm. In International Conference on Mining Intelligence and Knowledge Exploration 2015 Dec 9 (pp. 526-535). Springer International Publishing.

[7] Al-Shamri MY. Power coefficient as a similarity measure for memory-based collaborative recommender systems. Expert Systems with Applications. 2014 Oct 1;41(13):5680-8.

[8] Liu H, Hu Z, Mian A, Tian H, Zhu X. A new user similarity model to improve the accuracy of collaborative filtering. Knowledge-Based Systems. 2014 Jan 31;56:156-66.

[9] O'Mahony MP, Hurley NJ, Silvestre G. Detecting noise in recommender system databases. In Proceedings of the 11th international conference on Intelligent user interfaces 2006 Jan 29 (pp. 109-115). ACM.

[10] Amatriain X, Pujol JM, Tintarev N, Oliver N. Rate it again: increasing recommendation accuracy by user re-rating. In Proceedings of the third ACM conference on Recommender systems 2009 Oct 23 (pp. 173-180). ACM.

[11] Toledo RY, López LM, Mota YC. Managing natural noise in collaborative recommender systems. In IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS), 2013 Joint 2013 Jun 24 (pp. 872-877). IEEE.

[12] Toledo RY, Mota YC, Martínez L. Correcting noisy ratings in collaborative recommender systems. Knowledge-Based Systems. 2015 Mar 31;76:96-108..

[13] Pham HX, Jung JJ. Preference-based user rating correction process for interactive recommendation systems. Multimedia tools and applications. 2013 Jul 1;65(1):119-32.

[14] Amatriain X, Pujol JM, Oliver N. I like it... i like it not: Evaluating user ratings noise in recommender systems. In International Conference on User Modeling, Adaptation, and Personalization 2009 Jun 22 (pp. 247-258). Springer Berlin Heidelberg.

[15] Al-Shamri MY, Bharadwaj KK. Fuzzy-genetic approach to recommender systems based on a novel hybrid user model. Expert systems with applications. 2008 Oct 31;35(3):1386-99.

[16] Ujjin S, Bentley PJ. Particle swarm optimization recommender system. In Swarm Intelligence Symposium, 2003. SIS'03. Proceedings of the 2003 IEEE 2003 Apr 24 (pp. 124-131). IEEE.

[17] Eberhart R, Kennedy J. A new optimizer using particle swarm theory. In Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on 1995 Oct 4 (pp. 39-43). IEEE.

[18] Wasid M, Kant V. A particle swarm approach to collaborative filtering based recommender systems through fuzzy features. Procedia Computer Science. 2015 Jan 1;54:440-8.

[19] del Olmo FH, Gaudioso E. Evaluation of recommender systems: A new approach. Expert Systems with Applications. 2008 Oct 31;35(3):790-804.