I.J. Image, Graphics and Signal Processing, **2022, 4, 32-45**
Published Online August 2022 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijigsp.2022.04.03

An Improved Popular Items Extraction for Covering Reduction Collaborative Filtering

Abubakar Roko
Computer Science Unit, Usmanu Danfodiyo University, P.M.B 2346, Sokoto – Nigeria
Email: abroko@yahoo.com

Umar Muhammad Bello
Computer Science Unit, Usmanu Danfodiyo University, P.M.B 2346, Sokoto – Nigeria
Email: umarmuhammadbello@gmail.com

Abba Almu
Computer Science Unit, Usmanu Danfodiyo University, P.M.B 2346, Sokoto – Nigeria
Email: almul2003@yahoo.com

Received: 05 February 2022; Revised: 15 March 2022; Accepted: 24 April 2022; Published: 08 August 2022

Abstract: Recommender Systems are systems that aid users in finding relevant items, products, or services, usually in an online setting. Collaborative Filtering is the most popular approach for building recommender system due to its superior performance. There are several collaborative filtering methods developed, however, all of them have an inherent problem of data sparsity. Covering Reduction Collaborative Filtering (CRCF) is a new collaborative filtering method developed to solve the problem. CRCF has a key feature called popular items extraction algorithm which produces a list of items with the most ratings, however, the algorithm fails in a denser dataset because it allows any item to be in the list. Likewise, the algorithm does not consider the rating values of items while considering the popular items. These make it to produce less accurate recommendation. This research extends CRCF by developing a new popular item extraction algorithm that removes items with low modal ratings and similarly utilizes the rating values in considering the popular items. This newly developed method is incorporated in CRCF and the new method is called Improved Popular Items Extraction for Covering Reduction Collaborative Filtering (ICRCF). Experiment was conducted on Movielens-1M and Movielens-10M datasets using precision, recall and f1-score as performance metrics. The result of the experiment shows that the new method, ICRCF provides a better recommendation than the base method CRCF in all the performance metrics. Furthermore, the new method is able to perform well both at higher and lower levels of sparsity.

Index Terms: Covering Reduction, Popular Items, Sparsity, Popular Items Extraction Algorithm, Collaborative Filtering.

1. Introduction

The advancement of the internet has made it possible for so much information to be available in every domain. This makes it difficult for users to find relevant information or product from the vast amount of data. This problem is referred to as information overload [1-4]. This prompted the need for systems that help users in finding what they are looking for, such systems are called Recommender System [5]. Recommender Systems (RS) have been successfully applied to solve information overload problem in many domains such as; tag recommendation, television program recommendation, webpage recommendation, news and document recommendation, movie recommendation, video recommendation and music recommendation [6]. The three types of recommender systems are: Content-Based (CB) systems, Collaborative Filtering (CF) systems and hybrid systems [7-11]. CB recommender systems provide recommendation based on the features of the past items rated by the user. CF recommender systems make recommendation based on the ratings provided by similar users. Hybrid systems are a combination of CB systems and CF systems.

CF systems have grown substantively to become most common recommendation method. Companies such as Netflix and Amazon utilize CF systems [7,12]. In fact, as of 2006, Netflix are willing to pay a prize of $1,000,000 for any improvement on their current CF method [13-15]. CF systems are further divided into User Based Collaborative Filtering (UBCF) and Item Based Collaborative Filtering (IBCF).

Numerous studies have been conducted for improving CF algorithms to solve the data sparsity problem and provide accurate recommendations [16-26]. Among these methods, [26] is the best research in attempting to solve the data sparsity problem called Covering Reduction Collaborative Filtering (CRCF). Despite this, the following problems are unresolved: The algorithm fails as sparsity reduces, since each item can be regarded as a popular item, therefore, falling under redundant users to be removed. This is because CRA removes users with few ratings that concentrate on popular items. This leads to inaccurate prediction. Similarly, when choosing a popular item, the algorithm does not care about the actual rating values, leading to finding inaccurate popular items, hence producing less accurate prediction.

To improve the accuracy of the CRCF and in particular PIEA, this study develops a new PIEA which is used to generate a better list of popular items. The new PIEA is incorporated in CRCF. This can be achieved by utilizing the rating values of items in the recommender system. The new method is called Improved Popular Items Extraction for Covering Reduction Collaborative Filtering (ICRCF). The method removes items from the popular item list with low modal ratings. It also incorporates item rating values while considering the popular items to improve the accuracy of the recommendation.

The paper is organized as follows; section 2 introduces the basic concepts of recommender systems. Section 3 discusses existing literatures on collaborative filtering. In Section 4 presents the research framework, it also describes the existing method and the new method developed. Section 5 presents the evaluation results while section 6 gives a conclusion.

2. Basic Concepts of Recommender System

The following are some of the basic concepts in recommender system

A. Users

A user in a recommender system is an individual who utilizes the recommendation system. In Content-Based Filtering, features such as user demographic information are also used in the process of recommendation. In collaborative filtering, users are usually represented with a single id code. The formal definition of users is given below:

$$Let\ U : U = \{u_1, u_2, \ldots\ldots u_m\}\ be\ the\ set\ of\ users \tag{1}$$

B. Items

Items are the objects that are to be recommended to a user. The type of item in RS completely depends on the RS domain. For e.g.; in movie recommendation, the items are movies while in music recommendation, the items are music. According to [5] , items can either have low complexity and values (e.g.; web pages, music, movie) or high complexity and values (e.g.; digital camera, mobile phones).

$$Let\ I : I = \{i_1, i_2, \ldots\ldots i_n\}\ be\ the\ set\ of\ items \tag{2}$$

Items are usually represented with single-id code just like users. Furthermore, in Content Based recommender systems, attributes of items are also used. Attributes of items are dependent of the items. Examples of attributes in movie recommendation system include; genre, movie director, etc.

C. Rating

A rating is a stored transaction between a user and RS. The are many types of transactions that occur in RS, but ratings are the most popular [5]. There are four types of ratings in RS according to [27]. They are listed below:

- Numerical Ratings – e.g.; the 1-5 rating of Amazon, Netflix and Movie lens datasets.
- Ordinal Ratings – ratings that are in the form of questionnaire e.g.; “agree, strongly agree, disagree”.
- Binary Ratings – ratings in which users are decide if an item is good or bad
- Unary ratings – e.g.; indication of where a user purchased an item, play count of a music, etc.

The first three types of ratings described above are sometimes called explicit ratings while the fourth type is called implicit rating.

D. User-Item Rating matrix (R)

This is a matrix which contains all the users, items and their ratings. Users and items can either be the rows or columns while the ratings are the values provided by a user on an item or vice-versa. The formal definition of the ratings in R is given below;

$$\textit{The rating function } f : U \times I \to R,\ r_{ui} = f\left(u, i\right) \textit{ represents the rating score of user } u \textit{ on item } i$$

where;

$$r_{ui} = \begin{cases} \gamma, & the\ rating\ score \\ 0, & no\ rating \end{cases} \tag{3}$$

Table 1. An example of a user-item rating matrix

Users U	Items I								
	I1	I2	I3	I4	I5	I6	I7	I8	I9
u1	3	4	1	1	1	2	0	0	0
u2	5	3	1	3	1	3	1	2	5
u3	5	4	1	2	0	4	3	3	3
u4	1	0	1	4	4	3	4	4	2
u5	2	4	1	4	3	2	3	3	0
u6	0	0	0	0	0	0	0	0	5

E. Density

Density is the percentage of the populated ratings on the rating matrix and it is the number of non-empty ratings divided by the number of users multiplied by the number of items. It is calculated as given below:

$$Density = \frac{no.\,of\ \ non\ empty\ ratings}{no.\,of\ \ users \times no.\,of\ \ items} \times 100 = \frac{|R|}{m \times n} \times 100 \tag{4}$$

For Table 1.,

$$Density = \frac{40}{6 \times 9} \times 100 = \frac{40}{54} \times 100 = \ 70.4\%$$

F. Sparsity

Sparsity is the opposite of density. It is the percentage of empty ratings out of all possible ratings. It is given by the formula below:

$$Sparsity = 100 - Density \tag{5}$$

So basically, the higher the sparsity the lower the density and vice-versa. The matrix R is usually very sparse because only few users rate items. High Sparsity is one of the major problems in recommender system. A sparse dataset makes finding neighbours difficult thereby leading to inaccurate prediction.

G. Popular and Niche Items

Some recommender systems [23, 24, 26] introduced the concept of popular and niche items. In simple terms, Popular items are items that are rated more frequently by users while niche items are all the items that are not popular.

3. Related Works

This section describes the recent collaborative filtering methods. All of the reviews in the section are attempting to solve the problem of data sparsity. For each review, the strengths and weaknesses are provided as follows:

M. A. Ghazanfar and A. Prügel-Bennett developed a recommender system which combines Naïve Bayesian classification and Collaborative Filtering [16]. The Naive Bayes classifier is used to approximate the missing values in the user-item rating matrix. This makes the matrix dense and then CF is then applied for the final prediction. When CF fails, the system uses the calculated value from the Naïve Bayesian classification. In other words, the system switches between the two methods depending on the criteria. The system was experimented using 5-fold cross validation methodology on Movie lens dataset with Mean Absolute Error (MAE) as performance metric. The system was shown to perform better than the individual CF and Naïve Bayesian classification; however, the result of the classifier is biased towards higher ratings. Moreover, in the case where CF fails, the ratings provided by Naïve Bayesian classification are only positive integers, this could increase the error in prediction.

M. K. K. Devi et al. proposed a recommender system using Probabilistic Neural Network (PNN) and Self Organizing Map (SOM) which is a clustering technique [17]. The system works in two phases; in the first stage which is the online stage, trust calculations between users are made using Euclidean distance. In this stage, users are modelled using the SOM. The second stage which is the offline stage, trust is calculated with the users in the trusted cluster, and then recommendation is provided by rating prediction using the trust values. The system was experimented on Movie lens dataset using MAE and computation time as performance metric. The results show that the system outperforms other systems such as Single Value Decomposition and User Based Collaborative Filtering (UBCF). The system also overcomes the problem of biasness towards high ratings, however, the system produces higher error when the sparsity is less, leading to inaccurate prediction.

Y. Chen et al. used association retrieval to make recommendation [18]. Association Retrieval is a statistical method which takes users and items as a set of nodes and then uses the bipartite graph representing the relationship between users and items. Predictions are computed from graph and the original rating matrix. The system was experimented on Movie lens dataset using precision, recall, coverage and f1 score as performance metrics. The result show that it performs better than UBCF. The performance of the system is less affected by sparsity, however, the bipartite graph in the association rule ignores actual rating value of items which may lead to inaccurate prediction. Moreover, the system might produce high errors because the prediction provided are not bounded i.e.; it can produce a higher value than the given scale of rating.

J. Bobadilla et al. developed a UBCF recommendation system using a new similarity measure [19]. To find similarity between two users, the new similarity measure treats the individual ratings of each user as vectors. Another vector is then used to represent the ratio of items rated by both users. The three vectors described are used to compute the new similarity. Genetic algorithm is used to determine the optimal similarity computed from the vectors. The system was experimented on Movie lens, Film Affinity and Netflix using MAE, precision, recall, execution time as performance metrics. The results show that the new similarity measure is better in all metrics with other systems compared. In particular, the new similarity measure is 42% faster than the cosine similarity measure. Unlike the system described above, it can be used in a decimal-point based datasets, however, genetic algorithm comes with its problems-it requires a lot of parameters to be tuned correctly e.g.; fitness level probability, cross over probability and mutation probability. Selecting unsuitable values for any of the above parameters leads to inaccurate prediction.

G. Guo et al. developed a recommendation system called ‘merge’ which combines UBCF and user trust information [20]. After finding neighbors in UBCF, the system uses the trusted neighbors explicitly defined by user to produce the final recommendation. The system was experimented on Film Trust, FlixStar and Epinions using MAE, recall and precision as performance metrics. Merge was shown to produce better results in all the metrics. It does not ignore the actual rating values of items nor does it contain a lot of parameters like genetic algorithm, however as clearly stated, merge requires an active user to specify the users he trusts which maybe time consuming and could also lead to inaccurate prediction.

H. Sobhanam et al. proposed a new collaborative filtering recommendation method using association rule and clustering techniques to produce recommendations [21]. The system starts by pre-processing of the sparse dataset where, a software called Weka is used to pre-process the data and then store it in Microsoft Access. The purpose of pre-processing is to fill in the missing values in the sparse dataset. Association rules used to add new items with ratings by finding similar items. These added new items are then clustered using k-means algorithm to produce the final recommendation. The system does not require user to specify any information however, the association rule described is done using external tools which is could also be time-consuming.

S. Augustin developed a new recommendation method called Usage context Based Collaborative Filtering (UC-BCF) [22]. The system focuses on providing a wide of recommendation of items that are not mainly popular. The system uses a form of association rule where items can be considered similar even if they never occur together. The system was experimented on Netflix and Movie lens datasets using aggregate diversity and accuracy as performance metric. The results show that it performs better than UBCF, IBCF and matrix factorization techniques in aggregate diversity. However, the result also shows that the accuracy of recommendation is lost. Moreover, using the wrong usage context in other domains such as music leads to inaccurate prediction.

D. Zhang developed a recommendation algorithm using bi-clustering and Fusion (BiFu) method [23]. The first stage of Bi-Fu is pre-processing where trivial ratings and empty profiles are removed. The result of this is a denser matrix. Bi-clustering then clusters both users and item simultaneously using k-means algorithm. The final prediction is provided form the cluster result. BiFu was experimented on Movie lens and Netflix datasets using MAE and Root Mean Square Error (RMSE) metrics. The results show that BiFu performs better than other systems in all the metrics. Also, it does not involve the use of any usage context, however, BiFu removes trivial ratings which may lead to the loss of sensitive ratings in the original matrix. This may also lead to inaccurate recommendation.

F. Xie et al. introduced three new similarity measures to improve Item Based Collaborative Filtering [24]. The new similarity measures are learning methods initiated by formulating Optimization problem to minimize the squared errors in prediction. The problem is solved using Stochastic Gradient Decent. All of the similarity measures were experimented on Movie lens dataset using MAE, RMSE, precision, recall and F1-score as metrics. The result of the experiment show that the similarity measure performed better than other systems, however, when a single value is changed in the user-item rating matrix, the whole computation has to be done again, thereby very time consuming.

M. K. Najafabadi et al. used clustering and association to make recommendation[12]. The system makes the use of both implicit and explicit data. Explicit data are the ratings provided by user while implicit data are values like music play count in a music recommendation system. Clustering and association rules are both performed based on the explicit data. Final recommendations are provided from the rules extracted. The system was experimented on Million Song dataset with different sparsity levels. Using precision, recall and f1 score evaluation metrics, the system was shown to perform better than other algorithms. However, the system ignores rating values, meaning that the system cannot be applied in all domains.

J. Cheng developed a Jaccard Coefficient-Based Bi-Clustering and Fusion (JCBiFu) recommender system [25]. The system clusters both users and items using a clustering method called Clustering based on Density Peak (CDP). Unlike k-means CDP finds the correct number of cluster centroids automatically, thus improving the clustering accuracy. The system showed better results, however, density peak could remove sensitive ratings in the user-rating matrix, producing inaccurate recommendation.

Z. Zhang developed a new method called Covering Reduction Collaborative Filtering (CRCF) [26]. CRCF is an advancement of traditional UBCF where, after similarity computation between users, the algorithm continues by dividing into popular and niche items. This is done using an algorithm called Popular Items Extraction Algorithm (PIEA). Covering reduction is then applied to remove redundant users from the neighbors of new user. Redundant users are users whose rated items are in the popular items and they made only few ratings. CRCF was shown to perform better than other collaborative filtering algorithms, however in a denser dataset; any item can be regarded as a popular item hence falling under redundant user to be removed. Moreover, PIEA does not consider the rating values in selecting a popular item, hence producing less accurate prediction.

4. Methodology

In this section, an Improved Popular Items Extraction for Covering Reduction Collaborative Filtering (ICRCF) method is presented. The general scheme for the research study framework is depicted in Fig.1.

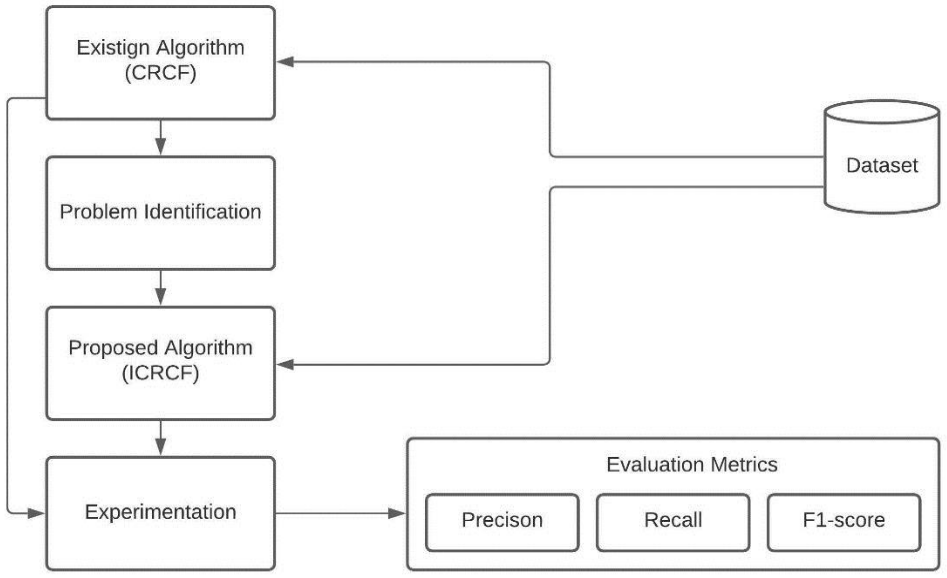


Fig.1. Schematic Diagram of the Research Framework

4.1 Problem Identification

After conducting an intensive review of related works which led to the research gap identification on the existing literature. Several methods, systems, techniques, strengths and weaknesses of the approaches were described. The research problems were identified by studying several related researches mentioned in Section 2. In particular, the main focus here is on the study by [26] in which the unresolved problems were identified as follows: firstly, it does not produce good popular items in a denser dataset (i.e. a less sparse dataset). This is because in a denser dataset, many of the items will have equal number of users who rate term. This makes the algorithm produce less accurate popular items, hence affecting the overall recommendation. Secondly, the algorithm considers the number of users that rate an item to find the popular items instead of the rating values. This also leads to providing less accurate recommendation.

4.2 Coverign Reduction Collaborative Filtering (CRCF)

In this subsection, CRCF is explained, CRCF has four steps; similarity computation, popular and niche items extraction, covering reduction and prediction. To illustrate these procedures, consider Table 1.

4.2.1 Similarity Computation

For the first step, the similarity between an active user and all the users in a given dataset is computed. An active user is a user that is waiting for recommendation at a moment. There are several similarity measures available, CRCF uses Pearson Correlation Coefficient (PCC) which the most commonly used. It is described below;

$$pcc(x,y) = \frac{\sum_{i \in I_x \cap Iy} (x_i - \bar{x}) \times (y_i - \bar{y})}{\sqrt{\sum_{i \in I_x \cap Iy} (x_i - \bar{x})^2} \times \sqrt{\sum_{i \in I_x \cap Iy} (y_i - \bar{y})^2}} \tag{6}$$

PCC produces a value between -1 and 1. PCC is also symmetric, i.e.; $sim(x,y) = sim(y,x)$. Where, x_i is the x-variable in a sample, $\bar{x}_i$ is the mean of the values of the x-variable, y_i is values of the y-variable in sample and $\bar{y}_i$ is the mean of the values of the y-variable.

4.2.2 Popular and Niche items Extraction

The following algorithm describes how popular items are computed according [26] and it is unresolved problems are also outlined:

Algorithm 1: Popular Items Extraction Algorithm

```
     Input    User Item Rating Matrix R
     Output   Set of popular items $I^{pop}$
1:   for all $i \in I$ do
2:        $U_i \leftarrow$ Count the number of users $u \in U_i$ such that $r_{ui} \neq 0$
3:   end for
4:   $I^{pop} = \emptyset$
5:   while $\frac{\sum_{i \in I^{pop}} |U_i|}{|R|} \leq rt$ do
6:        $j \leftarrow$ Select an item with the highers value $U_j$ in I
7:        $I^{pop} = I^{pop} \cup \{j\}$
8:        $I \leftarrow I \setminus \{j\}$
9:   end while
10:  return $I^{pop}$
```

To compute popular items according to Algorithm 1, firstly, we need to find the set of users that selected a particular item in Table 1

$U_1 = \{1,2,3,4,5\}$, $U_2 = \{1,2,3,5\}$, $U_3 = \{1,2,3,4,5\}$, $U_4 = \{1, 2, 3, 4,5\}$,$U_5 = \{1, 2,4, 5\}$,$U_6 = \{1, 2, 3, 4, 5\}$, $U_7 = \{2,3,4,5\}$, $U_8 = \{2,3,4,5\}$, $U_9 = \{2, 3, 4, 6\}$,

$I = \{1, 3, 4, 6, 2, 5, 7, 8, 9\}$, sorted according to the length of U_i

$I^{pop} = \emptyset$, $rt = 0.5$

The iterative process of Algorithm 1 is described as follows:

i. $\sum_{i \in I^{pop}} \frac{|U_i|}{|R|} = \frac{0}{40} = 0 \leq \text{rt} \therefore I^{pop} = \{1\},\ I = \{3,4,6,2,5,7,8,9\}$

ii. $\sum_{i \in I^{pop}} \frac{|U_i|}{|R|} = \frac{|U_1|}{40} = \frac{5}{40} = 0.125 \leq \text{rt} \therefore I^{pop} = \{1,3\},\ I = \{4,6,2,5,7,8,9\}$

iii. $\sum_{i \in I^{pop}} \frac{|U_i|}{|R|} = \frac{|U_1|+|U_3|}{40} = \frac{5+5}{40} = 0.25 \leq \text{rt} \therefore I^{pop} = \{1,3,4\},\ I = \{6,2,5,7,8,9\}$

iv. $\sum_{i \in I^{pop}} \frac{|U_i|}{|R|} = \frac{|U_1|+|U_3|+|U_4|}{40} = \frac{5+5+5}{40} = 0.375 \leq \text{rt} \therefore I^{pop} = \{1,3,4,6\},\ I = \{2,5,7,8,9\}$

v. $\sum_{i \in I^{pop}} \frac{|U_i|}{|R|} = \frac{|U_1|+|U_3|+|U_4|+|U_6|}{40} = \frac{5+5+5+5}{40} = 0.5 \leq \text{rt} \therefore I^{pop} = \{1,3,4,6,2\},\ I = \{5,7,8,9\}$

vi. $\sum_{i \in I^{pop}} \frac{|U_i|}{|R|} = \frac{|U_1|+|U_3|+|U_4|+|U_6|}{40} = \frac{5+5+5+5}{40} = 0.5 \leq \text{ rt } \therefore\ I^{pop} = \{1,3,4,6,2\},\ I = \{7,8,9\}$

vii. $\sum_{i \in I^{pop}} \frac{|U_i|}{|R|} = \frac{|U_1|+|U_3|+|U_4|+|U_6|+|U_2|}{40} = \frac{5+5+5+5+4}{40} = 0.6 > \text{ rt } \therefore\ I^{pop} = \{1,3,4,6,2\},\ I = \{7,8,9\}$

$I^{pop} = \{1, 3, 4, 6, 2\}$

Using Algorithm 1, the following problems are identified:

- The algorithm makes *I3* popular even though it has been given a low rating by all the users.
- The accuracy of the algorithm is solely dependent on the on the arrangement of the initial items formed for example; the system might decide to sort items 1, 3, 4, 6 in a different way since all of them are rated equally. This is what happens is dense dataset.

4.2.3 Covering Reduction Algorithm

To understand CRA, firstly we need to define Rough Sets and Covering Reduction.

- Rough Sets

The idea of rough set was first introduced [26] to deal with incomplete and vague information in classification, concept formulation and data analysis. Rough sets have been applied in many areas such as economics, medical diagnosis, biochemistry, environmental science, biology, chemistry, psychology, conflict analysis and recommender systems [12]

- Covering reduction is an extension of classical rough set. There are many types of covering, the following is the definition of covering as given by [28] is given below:

Definition 1: *Let T be the domain of discourse and C be a family of subsets of T. If none of the subsets in C is empty and* $\cup C = T$, *C is called a covering of T.*

Covering Reduction Algorithm is applied in CRCF to remove redundant users from the neighborhood of active user. The algorithm is described by [26] in details

4.2.4 Prediction

The prediction of the items to be recommended to the active user is done using equation (4).

$$pred(u,i) = \frac{\sum_{v \in N \cap v_i} sim(u,v) \times (r_{vi})}{\sum_{v \in N \cap v_i} sim(u,v)} \tag{7}$$

Where *pred(u,i)* is the predicted rating for item *i* by user u, r_{vi} is the user *v* rating on item *i* and $sim(u,v)$ is the similarity value computed using Pearson Correlation Coefficient.

4.3 Motivations for ICRCF

Several researches [23, 25, 26] have shown that dividing items into popular and niche items improves recommendation quality. However, placing a popular item as a niche item will not produce good recommendation. The popular items extraction algorithm described in CRCF has some shortcomings: firstly, it does not produce good popular items in a denser dataset (i.e. a less sparse dataset). This is because in a denser dataset, many of the items will have equal number of users who rate term. This makes the algorithm produce less accurate popular items, hence affecting the overall recommendation. Secondly, the algorithm considers the number of users that rate an item to find the popular items instead of the rating values. This also leads to providing less accurate recommendation.

4.4 Proposed Algorithm

As explained earlier, improvement in PIEA will improve the overall performance of CRCF. The following algorithm presents and Improved Popular Items Extraction Algorithm (IPIEA). When incorporated with CRCF, the overall method is called Improved Popular Items Extraction for Covering Reduction Collaborative Filtering (ICRCF).

Algorithm 2: Improved Popular Items Extraction Algorithm (IPIEA)

```
    Input   User Item Rating Matrix R
    Output  Set of popular items $I^{ipop}$
1:  for all $i \in I$ do
2:      $\mathrm{K_i} \leftarrow \{ \mathrm{r_{ui}}$ for $\mathrm{u} \in \mathrm{U_i}$ such thatn $\mathrm{r_{ui}} \neq 0 \}$
3:      if mode$(K_i) \geq$ st then
4:          $\boldsymbol{V_i} \leftarrow \sum_{u \in U} \boldsymbol{r_{ui}}$
5:          $\mathrm{S_i} \leftarrow$ Count the number of users $\mathrm{u} \in \mathrm{U_i}$ such that $\mathrm{r_{ui}} \neq \mathbf{0}$
6:      end if
7:  end for
8:  Sort I in descending order of $\mathrm{V_i}$
9:  $I^{ipop} = \emptyset$
10: while $\frac{\sum_{i \in I^{pop}} |S_i|}{|R|} \leq \boldsymbol{rt}$ do
11:     $\mathrm{j} \leftarrow$ Select an item with the highers value $\mathrm{U_j}$ in I
12:     $I^{ipop} = I^{ipop} \cup \{ j \}$
13:     $I \leftarrow I \setminus \{ j \}$
14: end while
15: return $I^{ipop}$
```

Line 2 gets the set of the rating values of each item. The if-condition on line 3 makes sure that whose modal rating is above a given threshold *st*, where are in the range of rating scale. Line 4 gets the sum of the rating values for a particular item. Line 8 sorts the items by the summation of the rating values. When the algorithm is applied, the results of the popular items are {1, 2, 4, 9}

Algorithm 1 is the original algorithm adopted from [26] which has its limitations as clearly presented. Algorithm 2 solves the problems mentioned of Algorithm 1. All other steps of [26] are kept intact as they are.

5. Experimental Evaluation and Results

This section describes the datasets, experimental environment and evaluation metrics used to assess the effectiveness of the proposed ICRCF as well as the presentation and the description of the results obtained.

5.1 Description of Dataset

Movie lens dataset was used to evaluate the performance of both the proposed and the existing algorithms. The dataset was chosen because is the most common dataset used in recommender system [13]. The dataset is collected and maintained by Group Lens, a research group at the University of Minnesota. There are five versions of the datasets, all of them can be downloaded from [1]. The two versions used in this research are described in Table 2 below:

Table 2. Movie Lens Dataset used in the research

Dataset Version	No. of Users	No. of Items	No. of ratings
ML-1M	6,040	3,900	1,000,209
ML-10M	71,567	10,681	10,000,054

5.2 Data Preparation

For all the two datasets, a user-based 5-fold cross validation technique is used. In this technique, the users in the dataset are divided into approximately 5 equal groups. In each fold, one of the groups is used for testing while the four other groups are used for training. In other words, 80% of the users are used for training while 20% of them for testing. For all the folds, the average is taken as the value for each of the metrics. Furthermore, in a similar way as the experiment by [26]. The ratings of the users in the training set are randomly removed so as to make each of them have a maximum of 20 ratings. This is because new users generally do not have more than 20 ratings.

5.3 Experimental Environment

[1] https://grouplens.org/datasets/movielens/

All the experiments are performed on Intel Core i5-8250U CPU @ 1.60GHz $\times$ 8 processor with 8GB RAM. The software used possessed the following specifications: Ubuntu 20.04 64bit Operating System, Python 3.8.10 general-purpose programming language (Packages used include: Pandas – for loading the dataset and converting it to user-item rating matrix, Sk-learn – for numerical calculations. e.g.; similarity computation, Multiprocessing – for making use of many processor in order to make the experiment run faster and Matplotlib – for plotting the evaluation results) and PyCharm 21.02 – An integrated development environment (IDE) for python programming language.

5.4 Evaluation Metrics

The performance of the proposed ICRCF is evaluated using the following performance metrics:

- Precision: This is a measure of exactness; it determines the fraction of relevant items retrieved out of all items retrieved. It is mathematically given as in Equation (5).

$$Precision\ at\ k = \frac{relevant\ items\ recommended}{all\ items\ recommend} \tag{8}$$

- (ii) Recall**:** This is a measure of completeness; it determines the fraction of relevant items retrieved out of all relevant items. It is mathematically given as in Equation (6).

$$Recall\ at\ k = \frac{relevant\ items\ recommended}{all\ relevant\ items} \tag{9}$$

- F1-Score: This metrics balances between precision and recall because usually, the lower the precision the higher the recall and vice-versa. It is mathematically given as in Equation (7).

$$F1\ Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{10}$$

5.5 Experimental Results and Discussion

In this subsection, the results of the evaluation of both CRCF and ICRCF are presented. Firstly, since another parameter st in introduced in ICRCF, the optimal parameter value of the parameter is computed first. Next, CRCF and ICRCF are compared based on different number of recommendation and sparsity levels.

5.5.1 Optimal Parameter Selection

The experiment by [26] showed that; a rt of 0.5 gives the best result. The research also used 40 as the number of similar neighbors (K). These two parameters are maintained for ICRCF, however, ICRCF introduces another parameter st. The parameter is in the range of rating values of the dataset. To find the optimal value for st, the two parameters are used with different values of st and the number of recommendations is set to 20. The results are shown in Table 3 below:

Table 3. Optimal parameter result for ICRCF

st	Precision	Recall	F1-Score
1	0.203	0.275	0.234
2	0.203	0.275	0.234
3	0.203	0.275	0.234
4	0.205	0.281	0.249
5	0.223	0.295	0.254

The result from Table 5.1. show that ICRCF performs best when $st = 5$. Precision, Recall and F1-Score all increased by 2%. Therefore, the value of st = 5 was used for the experiment

5.5.2 Comparison with Different Number of Recommendations

In this subsection, ML-1M is used with different number of recommendations setting rt = 0.5, k=40 for both CRCF and ICRCF, st = 5 is set for ICRCF. The results for precision, recall and F1-Score is given below:

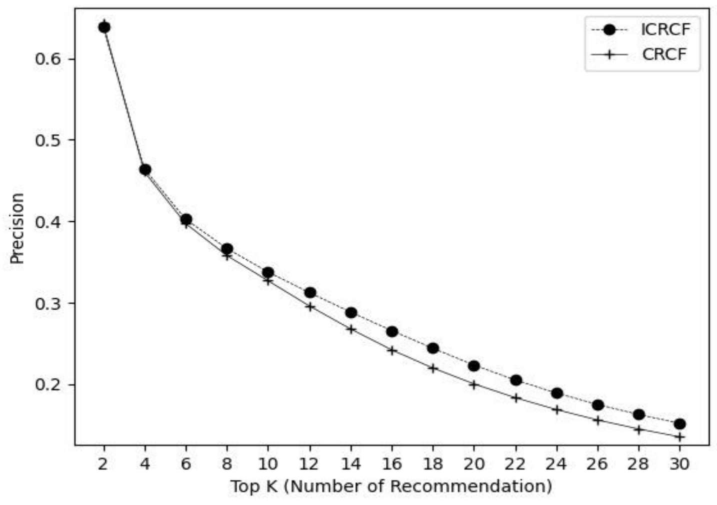


Fig. 2. Precision Values for Different Number of Recommendation in MovieLens-1M

The values of precision (computed using equation 8) in Fig.2. decreases as the number of recommendations increases. This is usually the case in collaborative filtering recommender system. When the number of recommendations is in the range of 2-6, not noticeable difference is observed, however when the number of recommendations is 10 and above, ICRCF improves the precision by 2%. The result suggests that ICRCF performs better as the number or recommendations are increased. The number or recommendation is the value k in equation 8

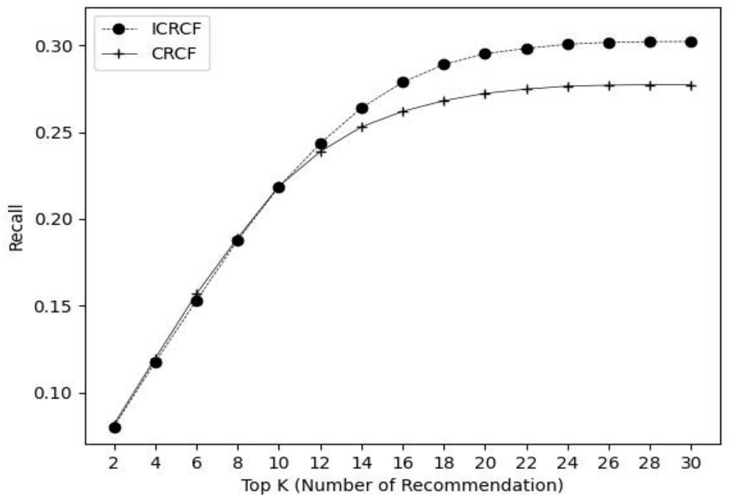


Fig. 3. Recall Values for Different Number of Recommendation in MovieLens-1M

Fig. 3. shows that recall (computed using equation 9) increases as the number of recommendations increase. This is usually the case in collaborative filtering recommender systems. Furthermore, when the number of recommendations is in the range 2-6, no difference is observed. ICRCF improves for values of recommendations more than 12. At top k= 30, ICRCF outperforms CRCF by 3%. This result suggests that the recall values of ICRCF improves as the number of recommendations increase

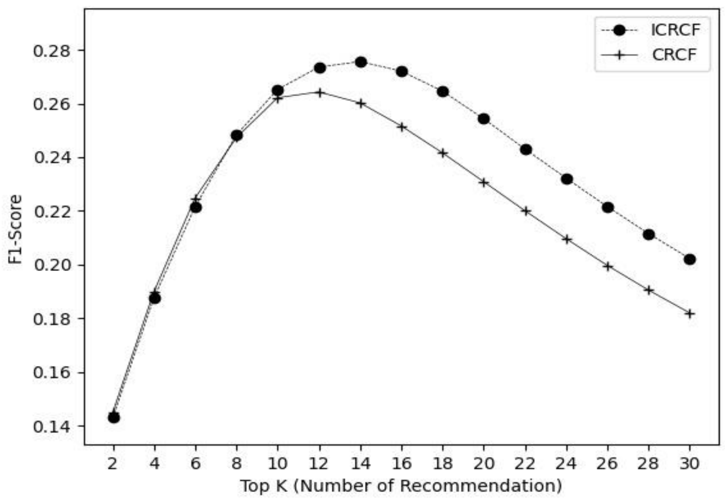


Fig. 4. F1-Score Values for Different Number of Recommendation in MovieLens-1M

The F1-Score (computed using equation 10) values in Fig. 4 behaves in the same way as the recall values. ICRCF improves CRCF by around 3% when the number of recommendations is more than 12. F1-score attempts to balance between precision and recall because one of them maybe significantly lower than the other. However, that is not in our case.

5.5.3 Comparison with Different Levels of Sparsity

To show the performance of the algorithm on different sparsity levels, top 2000 users with the most ratings are selected in ML-10M dataset. Then, ratings were removed randomly to simulate different sparsity levels. The figures below show the results obtained.

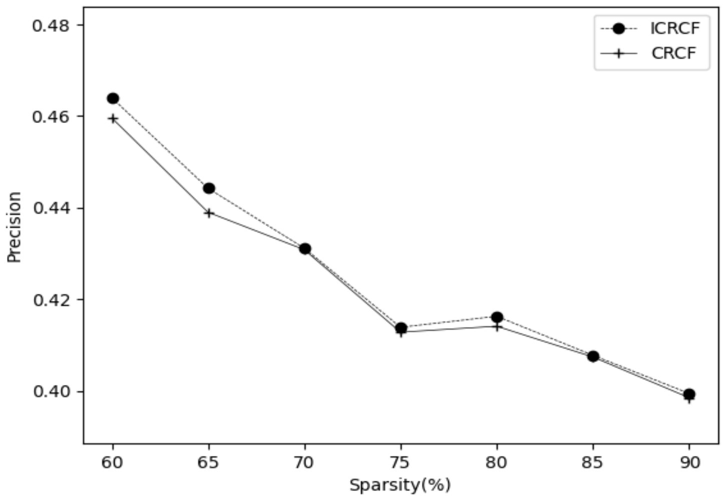


Fig. 5. Precision Values for MovieLens-10M on Different Sparsity Levels

Fig. 5. shows that the precision values decrease as the sparsity increases (this is expected in collaborative filtering recommender systems). The figure shows that (ICRCF) out performs CRCF in precision values at sparsity levels 60% and 65% by 1.1%. While it remains almost identical at other sparsity levels. Basically, 60% and 65% levels of sparsity is denser than higher values of sparsity hence CRCF performs less at the levels. The figure also shows that ICRCF is able to maintain the same values at higher sparsity level as CRCF while increasing the precision at lower sparsity levels.

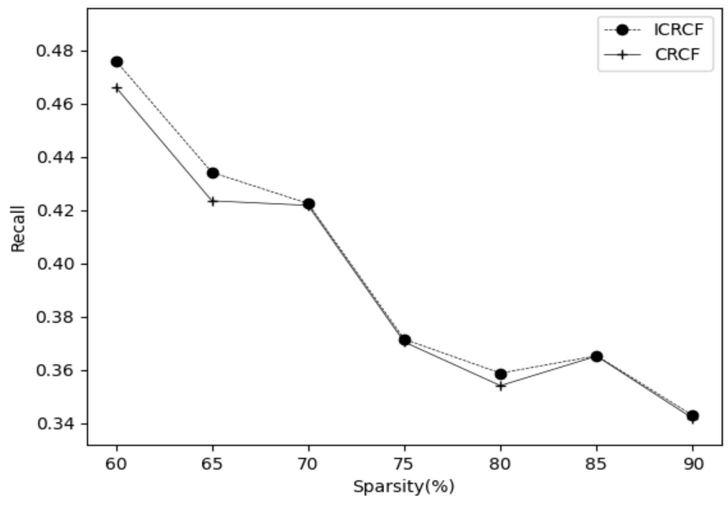


Fig. 6. Recall Values for MovieLens-10M on Different Sparsity Levels

The recall values in Fig. 6. also show that the recall values decrease as sparsity increases, furthermore, ICRCF increases the recall value about 1.3% at 60 and 65% levels sparsity just as in the case of Fig. 5

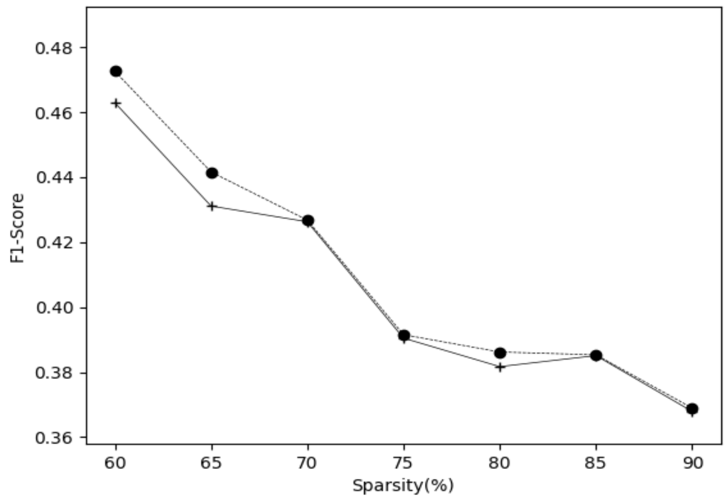


Fig. 7. F1-Score Values for MovieLens-10M on Different Sparsity Levels

The F1-score values in Fig. 7. also decrease as the sparsity increases. Also, ICRCF improves the recall f1-score values at 60 and 65% levels of sparsity.

6. Conclusion

In this research, and Improved Popular Items Extraction for Covering Reduction Collaborative Filtering (ICRCF) is developed to solve of the problems in the existing Covering Reduction Collaborative Filtering (CRCF). CRCF is divided into two algorithms, Popular Items Extraction and Covering Reduction. Popular items extraction in CRCF could produce inaccurate result in a less sparse dataset. Hence a new popular items extraction mechanism is developed. Experimental results show that the newly developed method performs better in the evaluation metrics precision, recall and f1-score. Furthermore, the new method is able to perform both at lower and higher levels of sparsity. However, the method uses a lot of parameters, for future a machine learning method should be introduces to select the best optimal parameters.

References

[1] J. Li *et al.*, “Category Preferred Canopy–K-means based Collaborative Filtering algorithm,” *Futur. Gener. Comput. Syst.*, vol. 93, pp. 1046–1054, 2019, doi: 10.1016/j.future.2018.04.025.

[2] C. Porcel, J. M. Morales Del Castillo, M. J. Cobo, A. A. Ruiz, and E. Herrera-Viedma, “An improved recommender system to avoid the persistent information overload in a university digital library,” *Control Cybern.*, vol. 39, no. 4, pp. 898–923, 2010.

[3] H. Costa and L. Macedo, “Emotion-based recommender system for overcoming the problem of information overload,” *Commun. Comput. Inf. Sci.*, vol. 365, pp. 178–189, 2013, doi: 10.1007/978-3-642-38061-7_18.

[4] A. Roko, A. Almu, A. Mohammed, and I. Saidu, “An Enhanced Data Sparsity Reduction Method for Effective Collaborative Filtering Recommendations,” *Int. J. Educ. Manag. Eng.*, vol. 10, no. 1, pp. 27–42, 2020.

[5] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, *Recommender System Handbook*. 2011.

[6] J. Lu, D. Wu, M. Mao, W. Wang, and G. Zhang, “Recommender system application developments: A survey,” *Decis. Support Syst.*, vol. 74, pp. 12–32, 2015, doi: 10.1016/j.dss.2015.03.008.

[7] J. Wei, J. He, K. Chen, Y. Zhou, and Z. Tang, “Collaborative filtering and deep learning based recommendation system for cold start items,” *Expert Syst. Appl.*, vol. 69, pp. 1339–1351, 2017, doi: 10.1016/j.eswa.2016.09.040.

[8] S. Natarajan, S. Vairavasundaram, S. Natarajan, and A. H. Gandomi, “Resolving data sparsity and cold start problem in collaborative filtering recommender system using Linked Open Data,” *Expert Syst. Appl.*, vol. 149, 2020, doi: 10.1016/j.eswa.2020.113248.

[9] A. M. Jones, A. Arya, P. Agarwal, P. Gaurav, and T. Arya, “An Ontological Sub-Matrix Factorization based Approach for Cold-Start Issue in Recommender Systems,” *Int. Conf. Curr. Trends Comput. Electr. Electron. Commun. CTCEEC 2017*, no. December 2019, pp. 161–166, 2018, doi: 10.1109/CTCEEC.2017.8455147.

[10] A. Angadi, S. K. Gorripati, and P. Suresh Varma, “Temporal community-based collaborative filtering to relieve from cold-start and sparsity problems,” *Int. J. Intell. Syst. Appl.*, vol. 10, no. 10, pp. 53–62, 2018, doi: 10.5815/ijisa.2018.10.06.

[11] B. S. Neysiani, N. Soltani, R. Mofidi, and M. H. Nadimi-Shahraki, “Improve Performance of Association Rule-Based Collaborative Filtering Recommendation Systems using Genetic Algorithm,” *Int. J. Inf. Technol. Comput. Sci.*, vol. 11, no. 2, pp. 48–55, 2019, doi: 10.5815/ijitcs.2019.02.06.

[12] M. K. Najafabadi, M. N. ri Mahrin, S. Chuprat, and H. M. Sarkan, “Improving the accuracy of collaborative filtering recommendations using clustering and association rules mining on implicit data,” *Comput. Human Behav.*, vol. 67, pp. 113–128, 2017, doi: 10.1016/j.chb.2016.11.010.

[13] L. Candillier, F. Meyer, and M. Boullé, “Comparing state-of-the-art collaborative filtering systems,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 4571 LNAI, pp. 548–562, 2007, doi: 10.1007/978-3-540-73499-4_41.

[14] M. D. Ekstrand, J. T. Riedl, and J. A. Konstan, “Collaborative filtering recommender systems,” *Found. Trends Human-Computer Interact.*, vol. 4, no. 2, pp. 81–173, 2010, doi: 10.1561/1100000009.

[15] J. Huttner, “From Tapestry to SVD: A Survey of the Algorithms That Power Recommender Systems,” no. May 2009, p. 32, 2009, [Online]. Available: http://thesis.haverford.edu/dspace/handle/10066/3706.

[16] M. A. Ghazanfar and A. Prügel-Bennett, “Building switching hybrid recommender system using machine learning classifiers and collaborative filtering,” *IAENG Int. J. Comput. Sci.*, vol. 37, no. 3, 2010.

[17] M. K. K. Devi, R. T. Samy, S. V. Kumar, and P. Venkatesh, “Probabilistic neural network approach to alleviate sparsity and cold start problems in collaborative recommender systems,” *2010 IEEE Int. Conf. Comput. Intell. Comput. Res. ICCIC 2010*, pp. 286–289, 2010, doi: 10.1109/ICCIC.2010.5705777.

[18] Y. Chen, C. Wu, M. Xie, and X. Guo, “Solving the sparsity problem in recommender systems using association retrieval,” *J. Comput.*, vol. 6, no. 9, pp. 1896–1902, 2011, doi: 10.4304/jcp.6.9.1896-1902.

[19] J. Bobadilla, F. Ortega, A. Hernando, and J. Alcalá, “Improving collaborative filtering recommender system results and performance using genetic algorithms,” *Knowledge-Based Syst.*, vol. 24, no. 8, pp. 1310–1316, 2011, doi: 10.1016/j.knosys.2011.06.005.

[20] G. Guo, J. Zhang, and D. Thalmann, “User Modeling, Adaptation, and Personalization: 20th International Conference, UMAP 2012, Montreal, Canada, July 16-20, 2012. Proceedings,” *User Model. Adapt. ...*, no. Imi, pp. 114–125, 2012, [Online]. Available: http://dx.doi.org/10.1007/978-3-642-31454-4_10%5Cnhttp://link.springer.com/chapter/10.1007/978-3-642-31454-4_10.

[21] H. Sobhanam and A. K. Mariappan, “Addressing cold start problem in recommender systems using association rules and clustering technique,” *2013 Int. Conf. Comput. Commun. Informatics, ICCCI 2013*, pp. 0–4, 2013, doi: 10.1109/ICCCI.2013.6466121.

[22] S. Augustin, K. Niemann, and M. Wolpers, “A New Collaborative Filtering Approach for Increasing the Aggregate Diversity of Recommender Systems,” pp. 955–963, 2013.

[23] D. Zhang, C. H. Hsu, M. Chen, Q. Chen, N. Xiong, and J. Lloret, “Cold-start recommendation using Bi-clustering and fusion for large-scale social recommender systems,” *IEEE Trans. Emerg. Top. Comput.*, vol. 2, no. 2, pp. 239–250, 2014, doi: 10.1109/TETC.2013.2283233.

[24] F. Xie, Z. Chen, J. Shang, W. Huang, and J. Li, “Item similarity learning methods for collaborative filtering recommender systems,” *Proc. - Int. Conf. Adv. Inf. Netw. Appl. AINA*, vol. 2015-April, pp. 896–903, 2015, doi: 10.1109/AINA.2015.285.

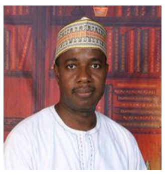
[25] J. Cheng and L. Zhang, *Jaccard coefficient-based bi-clustering and fusion recommender system for solving data sparsity*, vol. 11440 LNAI. Springer International Publishing, 2019.

[26] Z. Zhang, Y. Zhang, and Y. Ren, “Employing neighborhood reduction for alleviating sparsity and cold start problems in user-based collaborative filtering,” *Inf. Retr. J.*, vol. 23, no. 4, pp. 449–472, 2020, doi: 10.1007/s10791-020-09378-w.

[27] B. J. Schafer, D. Frankowski, J. Herlocker, and S. Sen, “Collaborative Filtering Recommender Systems - CollaborativeFilteringRecommenderSystems.pdf,” *Lncs*, vol. 4321, no. January 2007, pp. 291–324, 2007, [Online]. Available: http://www.eui.upm.es/~jbobi/jbobi/PapersRS/CollaborativeFilteringRecommenderSystems.pdf.

[28] Q. Z. Kong and Z. X. Wei, “Covering-based fuzzy rough sets,” *J. Intell. Fuzzy Syst.*, vol. 29, no. 6, pp. 2405–2411, 2015, doi: 10.3233/IFS-151940.

Authors' Profiles



Abubakar Roko is a Reader at the Department of Computer Science, Faculty of Physical and Computing Sciences, Usmanu Danfodiyo Univesity Sokoto, Nigeria. He received his PhD from Univeristy Putra Malaysia in 2016, specializing in the field of XML Retrieval. Currently, his research interest is in the area of text data management and analysis, focusing in particular on query processing in Information retrieval, Recommender Systems, and Sentiment Analysis.

Bello Umar Muhammad received the Bachelor of Science Computer Science from Usmanu Danfodiyo University, Sokoto. He is currently undergoing his Masters Degree in Computer Science from The University. His research interests include; recommender systems, machine learning and computer vison.

Abba Almu received his PhD in Computer Science from Usmanu Danfodiyo University, Sokoto. Masters Degree in Computing: Information Engineering from Robert Gordon University, Aberdeen United Kingdom. He also received his B.Sc. (Hons) in Computer Science from Usmanu Danfodiyo University, Sokoto. He is currently a Senior Lecturer at the Department of Computer Science, Usmanu Danfodiyo University, Sokoto Nigeria. His research interests include information retrieval, recommender systems, decision support systems and machine learning.

How to cite this paper: Abubakar Roko, Umar Muhammad Bello, Abba Almu, "An Improved Popular Items Extraction for Covering Reduction Collaborative Filtering", International Journal of Image, Graphics and Signal Processing(IJIGSP), Vol.14, No.4, pp. 32-45, 2022. DOI:10.5815/ijigsp.2022.04.03