[1]Shiba Prasad Dash,
[1]Rajesh Kumar Sahoo,
[1]Chinmaya Ranjan Padhan,
[2]Ram Chandra Barik

A New Variant of Clustering-Based Normalization in Movie Recommendation System

Abstract: - Several communities have experienced considerable growth in the Recommender System (RS) in recent years. Researchers are intrigued by it because of the recent expansion of several businesses engaged in E-commerce, and online video providers like Netflix, YouTube, Hotstar, etc. The collaborative filtering-based RS system strives to recommend to the viewers such movies or videos on their prior history. Typically, a rating matrix is used to represent these data. These ratings are not consistent though some users' reviews are harsh, and others are liberal. Because of this, the RS is unable to recommend tailored movies to demanding viewers. This paper presents a collaborative filtering recommendation system that utilizes movie clustering and normalization to address the aforementioned problem. Using the average user rating for each movie and the number of users who evaluated each movie in the first stage, the suggested technique clusters the movies. In the first phase, it makes clusters of similar movies using Jaccard similarity then the RS calculates the normalized user count for each movie is determined using min-max normalization, Next, the users' scaled average ratings within a certain range are calculated. When the rating matrix creates user rating predictions during the assessment phase, it is divided into training and testing rating matrices.

Keywords: Clustering, Collaborative filtering, Machine Learning, Min–Max normalization, Recommendation system.

I. INTRODUCTION

The recommendation system is an emerging application of Artificial Intelligence especially in digital marketing and e-commerce for recommending additional products to legitimate customers. A recommendation system with AI capabilities essentially presents the user with a selection of products based on their past choices and similarities to other products in the same category. It recommends any product, item, or movie after examining the previous shopping of any product or watching any movies by the corresponding customer. Online readers are given recommendations for news items via recommendation systems, which also help online retailers by suggesting movies to their consumers based on their previous purchases. Recommendation systems may be broadly classified into three categories: recommendation systems based on content, recommendation systems utilizing collaborative filtering, and hybrid recommendation systems. The user's rating for every given product or movie is reflected in the utility matrix.

Recommender systems (RS) are a class of systems that, through data filtering, select relevant items for users from a vast knowledge base. Numerous paradigms, such as knowledge-based, content-based, collaborative filtering-based, and hybrid recommender systems, can be used to classify it [1–8]. Based on the data users have already contributed, the collaborative filtering-based recommendation system (RS) suggests items for their individual collections. This data includes interests, rating behaviors, preferences, historical information, and much more. Neither the goods nor the user features are used by this RS. Here, the product is rated by users or through a rating matrix, where a single element represents the degree of customer satisfaction [9–15]. User ratings can be expressed as positive integers, on a scale from -1 to 1, or a scale from 1 to 5. The content-based RS offers movies that are based on prior user selections and defines a movie in terms of different attributes. The hierarchy of recommendation algorithms is categorized in Fig. 1. The algorithm suggests the newest documentaries to the user if the user has already purchased any documentaries. Our paper focuses on a recommendation system that uses collaborative filtering. Contents of the movies do not have any significant role in making recommendations in this type of CF algorithm. Profile vectors of users are represented as rows in the utility matrix. The main issue of CFRS is the finding similarity between users but these are carried out using Jaccard and cosine distance [17]. This paper also focuses on normalizing user ratings and grouping comparable movies.

[1]Department of Computer Science and Engineering, Biju Patnaik University of Technology, Rourkela, India
[2]Department of Computer Science and Engineering, C. V. Raman Global University, Bhubaneswar, India
Corresponding author: Shiba Prasad Dash (capgs.sdash@bput.ac.in).

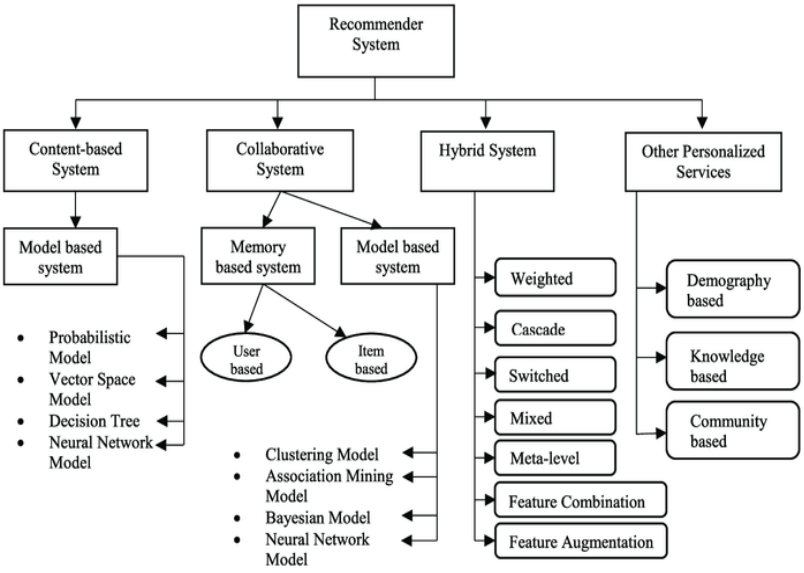

FIGURE 1. **Classification of Recommendation System. It classifies the different types of recommendation techniques that exist.**

The following situations are used to convey the main rationale of this study. (1) Scenario 1: A user rates products between 1 and 3, regardless of the caliber of the goods the user has acquired. Such a person is known as a stringent user. (2) Scenario 2: A user, regardless of the things purchased, assigns ratings in the range of three to five. Such a user is known as a lenient user. (3) Scenario 3: A user rates each movie from 1 to 5 according to its quality. We call this type of customer "average user." There is variation in the user ratings for the movies in these situations. This phenomenon encourages us to take into account uneven user evaluations and deliver users tailored products. We tackle the following issue using collaborative filtering-based RS in this work. This paper tackles the following issues with reinforcement learning based on collaborative filtering. Two issues arise in a rating matrix R consisting of m items and n users. (1) Using a range of possible customized items, develop an RS algorithm that optimizes the total, minimum, maximum, and average number of products suggested for a specific client group. (2) To evaluate the RS technique, the rating matrix is split into two halves, referred to as the training and testing rating matrices. Next, prospective user ratings for the testing rating matrix is estimated, to maximize the Fowlkes-Mallows Index (FMI) and F-Score while minimizing the Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). Here, we offer an effective method for solving this issue that we name clustering-based normalized movie recommendation (CNMR). This algorithm comprises two phases. The processes are called designing and assessing, respectively. The first step involves utilizing a distance, such as the cosine or Jaccard distance, to determine how similar the films are to one another. After finding the similarity between the movies, similar movies are grouped in a single cluster. The min-max normalization is applied to the clusters to obtain the average user ratings for each cluster and to determine which clusters are best for the recommendations.

The results of several simulations performed using the proposed method, which considers a broad class of people and objects, are compared using a collaborative filtering-based RS. This is the structure of the paper. In section 2, related work is presented. The research work's problem statement is explained in Section 3. Section 3 presents the proposed CNMR algorithm. In Section 4, the findings and discussion are summarized. A review of prospective next initiatives is included in Section 5's conclusion.

II. RELATED WORK

Ahn H. J. et al. introduced proximity-impact-popularity, a novel heuristic similarity metric for collaborative filtering that outperformed cosine and Pearson's correlation in cold-start circumstances. Better recommendations are provided by the suggested method. However, the majority of the aforementioned study was not taken into account while normalizing the advice (Ahn H. J. et al., 2008). Bobadilla et al. have created a large number of equations for RS that use collaborative filtering and memory. To make recommendations and apply them to the e-learning RS, they took into account the users' degree of understanding. The degree of knowledge of the users was considered in a variety of equations for memory-based collaborative filtering-based

recommendation systems that were used for e-learning recommendation systems. Other collaborative filtering techniques and recommendation systems with four categories of filtering algorithms are presented as: collaborative, content-based, hybrid, and demographic. Both memory-based and model-based methods have been explored by them. These filtering algorithms were created to make recommendations for products or services based on user reviews. Utilizing the MAE, RMSE, coverage, precision, recall, normalized mean average error, and receive operational characteristics, the recommendation system's accuracy is determined (Bobadilla, J., 2009, 2013). Chen et al. have researched how to employ CF for internet marketing, advertising, and suggestion systems. Additionally, Chen et al. (2012) recommends the Multi-Collaborative Filtering Trust Network approach, which is an enhanced version of the CF algorithm intended to operate on the Web 2.0 platform. Duet al. have created a trusted network and proposed trust links among different users in the 0–1 domain. They have proposed a potential expansion of identifying untrustworthy users in the trusted network (Du, Y., 2016). Hong et al. proposed cross-cultural contextualization and created a model utilizing matrix factorization and clustering techniques to determine the cross-cultural factor impacting users' preferences (Hong, M. et al. 2019). In their paper, Isinkaye et al. demonstrate how the issue of information overload during recommendation has been studied, and the various features and potentials of various prediction approaches in recommendation systems have been explored. These initiatives operate as a guide for recommendation systems research and application (Isinkaye et al., 2015). Karidi et al. have suggested a strategy for showing relevant tweets on user timelines called a semantic suggestion. They have done this by using a knowledge graph, which represents ideas, instances, people, places, and their connections (Karidi et al, 2017). Kim et al. have improved accuracy by using the clustering and normalization concepts. This method also made the pre-existing prediction systems perform better. They utilized them as a feature vector and clustered the ratings based on the following metrics: mean user rating, proportion of most common to second most frequent rating, frequency of ratings, and standard deviation from mean rating. To accomplish prediction, they have modified the neighbor users' ratings throughout the normalization procedure. To anticipate the ratings of nearby users, they have utilized the highest frequency of ratings as the average rating of a user. In their manuscript, they have implemented the normalization method using the user's minimum, average, and maximum ratings (Kim et al., 2013). Koren et al. have presented two collaborative filtering approaches. The two different approaches given by them are the neighborhood model and the latent factor approach. The neighborhood model primarily focuses on the relationship between various users, whereas the latent factor model converts people and products into the same space. In their research, they have concentrated on an important aspect, which is the temporal factor. Here 5 scale rating system is used with an extra time factor which is the day of rating an item by the user. Linden et al. presented a cooperative filtering methodology for item correlations and compared it with three other approaches: search-based strategies, cluster-based strategies, and the traditional CF. In the worst-case situation, their proposed method requires O(m2n) for n users and m items (Linden et al., 2003). Lu et al. surveyed applications for recommendation systems and their advancements. It focuses on both the theoretical and practical facets of RS research. They described a variety of recommendation system approaches and categorized them into eight groups: contextually aware recommendation techniques, collaborative filtering-based recommendation techniques, hybrid recommendation techniques, social network techniques for recommendations, computational intelligence approaches for making recommendations, and content-based recommendation techniques. For the assessment, eight categories like e-business, e-group, e-learning, e-commerce, e-government, e-resource, and e-library have been defined. They claimed that the knowledge-based approach that is most employed in all disciplines is e-learning and that specific users are more likely to report e-resources (Lu et al., 2015). They have covered dimensionality reduction strategies, meta-approaches, social filtering strategies, diffusion-based strategies, and similarity-based strategies. There are two types of similarity-based methods: item-based and user-based. Luo et al. suggested a regularized matrix factorization (RAF) driven collaborative filtering recommender system over two large datasets (Luo et al., 2012). Majeed et al. have concentrated on providing tailored advice to travelers. They conducted two investigations using semi-structured interviews and mobile ethnography. Furthermore, they created a hybrid RS using implicit user feedback (Majeed et al., 2019). Nayak et al. focused on a novel algorithm that chooses k-comparable people by figuring out how many other people have the most in common with a specific item. They also implement their proposed approach and compare it with the kNN methodology (Nayak et al., 2018). Panda S et al. suggested a new collaborative filtering RS method for item suggestion that is based on normalization. Panda et al. have employed a min-max normalization strategy in their research.

TABLE I

EXISTING RECOMMENDATION MODELS USED FOR MOVIE RECOMMENDATIONS

Author	Contribution	Clustering	Normalization Technique
S. Panda et al. [12]	A normalization-based collaborative filtering recommendation algorithm	NO	Min-Max Normalization
Z. Liang et al. [22]	An algorithm for recommending movies using weight normalization optimization and three-way neural interaction networks	NO	weight-normalization
J. Zhang et al. [18]	Customized Real-Time Video Suggestion System: Workable Model and Assessment	K-means clustering	NO
H. Jing [23]	Implementing Enhanced K-Means Algorithm in Cooperative Recommendation Framework	K-means clustering	NO
W. Ma et al. [20]	Normalizing Context-Aware Scaled Baseline Predictor-Based Item-Based Collaborative Filter	NO	Baseline predictor
N. Ifada et al. [19]	Multi-Criteria Collaborative Filtering Approach for Recommendation System based on Normalization	NO	Decoupling normalization
M. Hong et al. [13]	Contextualization across cultures for recommender systems	cross-cultural factors	Z-score Normalization
Y. Yang et al. [11]	An item similarity model based on group preferences: contrasting clustering methods in ambient and context-aware recommender systems	KNN clustering	NO
S. Kim et al. [9]	Enhancement of rating normalization for collaborative filtering	Two-step clustering	Z-score Normalization
Proposed Model	Normalization in a Movie Recommendation System through Clustering	Hierarchical clustering	Min-Max Normalization

Initially, an attempt was made to obtain the mean scores for every film as well as the mean score for every user. The missing user ratings are then located by applying min-max normalization to the average user ratings. As per the newly generated missing user ratings, the best possible movies are recommended to the users (Panda Set al., 2020). According to Pelanek et al., the concept of modeling the students has influenced the thorough examination of two performance indicators: RMSE and area under the curve. Additionally, they have offered several methods for computing these metrics, including global, student-by-student, and skill-by-skill averaging, which presents students and skills as two fundamental dimensions. In addition, a fattened matrix is used to represent both students and skills. Regardless of pupils or skills, the values of this matrix are equivalent in the overall computation. When averaging across students, the values of a student are taken into consideration and averaging across all talents. Comparable involves determining the typical user rating for each product. On the other hand, when averaging across talents, the skill ratings of all pupils are considered. Keep in mind that it is the same as determining the user average rating (Pelanek et al., 2017). Porteous et al. introduced a collaborative filtering technique based on a Bayesian matrix factorization model. This CF technique fills in the gaps created by the missing ratings in the matrix by examining the available ratings. For this, they have integrated the user data (implicit feedback, confidence levels, temporal impacts, etc.) with the observations that are already in place. This paper (Porteous et al., 2010) proposes a memory-based CF method without providing any additional details. Raheem et al. surveyed to find out about practical recommender system techniques that help users make decisions by displaying options according to the system's forecast. Sarwar et al. have talked about a number of

recommendations based on item methods such as regression, correlations between items, cosine similarity, and weighted sum, and experimentally given a comparison of them. Sarwar et al. (2001) found that item-based algorithms perform better than user-based algorithms in terms of quality and scalability. A recommender system with dual consideration for user ratings and item ratings from all users was presented by Xie et al. They have constructed a vector utilizing the mean, variance, and range to represent the user/item rating. However, according to Xie et al. (2012), their algorithm does not consider the attributes of the product or the consumer profile. With a focus on context-aware reinforcement learning, Yang et al. released the UGPS-1 and UGPS-2 models. While UGPS-2 employs K-NN, UGPS-1 uses K-means clustering.

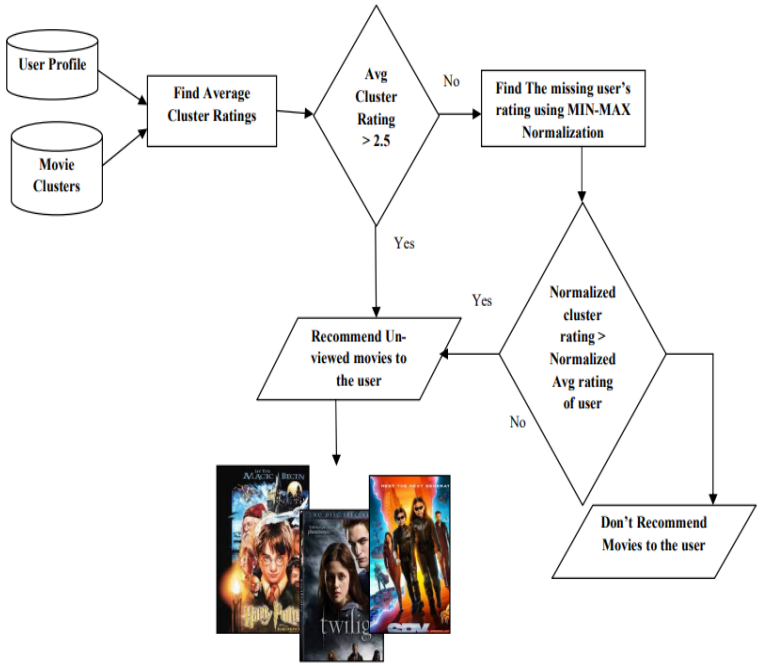

FIGURE 2. **Data flow diagram of the proposed methodology.**

In relation to the number of groups and item sets, UGPS-1 increases recommendation efficiency while UGPS-2 decreases it (Yang et al., 2018). In their survey, they separated the CF-based algorithms into memory-based and model-based strategies. Additionally, a technique based on user ratings and behavior was developed to assess both algorithms using two case studies (Yang et al., 2016).

III. PROPOSED METHODOLOGY

The clustering-based normalized movie recommendation (CNMR) algorithm, movie clustering, pre-processing, and recommendation finding are the four primary phases of the suggested method. The data flow diagram for the recommended methodology is shown in Figure 2.

In order to recommend movies using the CNMR method, the normalization-based collaborative filtering recommendation approach (NCFR) was previously developed. The algorithm forecasts the user's absent ratings for the various films to present the user with a list of potential viewings. The design stage and the assessment stage are the two phases into which we have separated our procedure.

There are five fundamental phases in the designing stage. Partitioning many movies into different clusters is the initial stage. The final step is to calculate the aggregate number of users who evaluated each film, as well as the average user rating. Then, it normalizes the number of users per cluster using the widely used min-max normalization technique. The number of users in each cluster will be scaled from 1 to 5 using this min-max normalization.

It also makes it easier to find the most popular movie series. It then retrieves the maximum, minimum, and average ratings for each user and applies normalization to them in the second step. The goal of this is to categorize people into three groups strict, tolerant, and normal, and utilize normalization to equalize each group within a given range. As a result, it is not specifically illustrated in the suggested algorithm. For instance, if a person thinks a movie is extremely satisfying, they will give it a minimum rating of 3. On the other side, another

customer rates their unhappiness with a minimum of 1. Similarly, maximum ratings for users can also differ. As a result, normalization is crucial for bringing the ratings into parity. The final stage discovers the sequence of the individualized movie clusters and suggests a collection of them to the users. Normalized user average ratings are used for dynamic thresholds, and normalized user numbers are used as weight values for the items.

During the evaluation stage, this method is used to separate the rating matrix's training and testing sections. First, the training rating matrix produces the user count per movie, the user counts per cluster, the normalized average user rating, and the average user rating per cluster. The testing rating matrix functions in movies with ratings depend critically on the normalized user count for each film. Three different cases exist. (1) If the normalized user count for each cluster is on or above the higher threshold, the average user rating for each cluster is selected as the anticipated rating.

Algorithm: Clustering-based Normalized Movie Recommender (CNMR)
Phase 1: Designing the CNMR
Input: A 2-D matrix R, a set of n users and a set of m movies **Output:** A set of possible movies per user or top-M movies per user
Call ClusterMovie(R, user, movie) Call FindAverageUserRating-NormUserCount(P, n, m) Call FindRatingPerUser-NormRatingPerUser(P, user, cluster) Call FindRecommendation(P, user, cluster, AvgRatingMovies, NormAvgRatingUser, NormUserCount) Call FindOrderRecommendation(P", user, cluster) Find total, average , maximum and minimum number of items recommended

FIGURE 3. **Designing Phase of CNMR**

This is justified by the fact that the majority of people have given the related movie a rating. (2) When the normalized user count for each cluster is less than or equal to the lower threshold, the normalized average user rating is selected as the expected rating. This was chosen because not enough users have given the associated movies enough ratings. (3) If none of the above-specified criteria are satisfied, the weighted average user rating for each movie and the weighted normalized average user rating are blended linearly to anticipate the ratings.

A. Pseudo Code For CNMR

In this section, the designing phase of the CNMR algorithm is discussed. The following figures show the algorithm's design process. Initializing the rating matrix having n numbers of users and m numbers of movies as input to the CNMR is the first step. For the clustering of a large number of movies, CNMR initially calls Procedure 1 (CLUSTER_MOVIES).

B. Procedure 1: ClusterMovie

The algorithm groups a huge number of movies into fewer clusters in this part. For the clustering of the process, we have used a well-known similarity measure technique which is Jaccard similarity. Procedure 1 takes input as a rating matrix R, an array of$m$ movies, and n number of users. The first line of procedure 1 initializes the matrix P, Q, and the cluster matrix. The cluster matrix is to store the same types of movies in their respective clusters. Line 2 to 6 is used to check whether a particular user has rated a movie or not. If the rating exists that

means any number value between 1-5 then to show the rating, we have stored 1 in the Q matrix in the respective user and movie rating place. From line 7 the clustering process begins.

Procedure 1ClusterMovie(R, user, movie)

Input: R[n][m], movies[m], user[n]

```
Set P[n][m], R[n][m], Q[n][m], Cluster[m][m]
for i = 1to n
   for j = 1to m
       If (R[i][j] > 0) then Q[i][j] = 1
   end for
end for
for i = 1to m
Cluster[i][0] = movie[i]
   k = 0
   for j = i+1 to m-1
      Token_1 = movie[i].split()
      Token_2 = movie[j].split()
      if ( textdistance.jaccard(token_1, token_2) >=
0.75 )
Cluster[i][k+1] = movie[j]
          k++
          for l = 1to n
             if ( P[l][j] != 0)
P[l][i] =  P[l][i] +  P[l][j]
Q[l][i] = Q[l][i]+ Q[l][j]
Endif
for( a = j, a < m-1, a++)
P[l][a] = P[l][a+1]
                 j = j-1
              end for
           end for
       end if
   end for
end for
for i = 1to n
   for j = 1to k
P[i][k] = P[i][k] / Q[i][j]
       if P[i][j] ≥ 2.5
          RECOMMEND REMAINING MOVIES OF
THAT CLUSTER TO USER[i]
       end if
   end for
end for
```

FIGURE 4. **Pseudo code for Cluster Movie**

First of all, if any new movie is encountered then line 8 saves that movie as a new cluster and stores it in the Cluster matrix. After that, line 10 to line 23 is used for checking the similarity between movies. First, we have a new cluster movie then we will check whether any other movie is similar in name to it or not.

As we can see in line 10 we will check from the next movie of the cluster movie. For the name similarity checking, we can use Jaccard similarity as it checks both the string bit by bit means it checks each character. The cluster movie is taken as Token_1 and the next movie is taken as Token_2. The similarity is checked and if the text similarity is more than 75% we consider it as the same name movie and store the second movie in the Cluster matrix.

After that, the total number of rated movies is counted i.e. A certain number of movies in the cluster are rated by a particular user. The total rating of the cluster from each user is calculated and stored in the P matrix as in line 19. A movie's column in matrix P is removed if it is kept in an already-existing cluster, as seen in lines 1–23.

Procedure 2: FindAverageUserRating-NormUserCount(P, n, k)

```
Set MaxUserCount = 0, MinUserCount = ∞
Set MaxRange = 5,  MinRange = 1
for j = 1to k
   Set TotalMovieRating = 0, count = 0
   for i = 1to n
   if (R[i][j] > 0)
TotalMovieRating= R[i][j] +  TotalMovieRating
      count ++
   Endif
AvgRatingCluster[j] = TotalMovieRating/ count
UserCount[j] = count
if (UserCount [j] >MaxUserCount)
MaxUserCount= UserCount[j]
   end if
   if (UserCount [j] <MinUserCount)
MinUserCount= UserCount[j]
   end if
end for
for j = 1, 2,…, k
normUserCount[j] = ( UserCount[j] - MinUserCount )/
(MaxUserCount -MinUserCount ) * (MaxRange-
MinRange ) + MinRange
end for
```

FIGURE 5. **Pseudo code for Procedure 2**

Additionally, Line 31: The average rating for each cluster is calculated for each user. Lastly, all of the other movies in that cluster that the user hasn't yet rated are suggested to them if their average rating for that particular movie is higher than 2.5.

Lemma 1: Procedure 1 requires time is O(mn).

C. Procedure 2: FindAverageUserRating-NormUserCount

The normalizing procedure comes next, following the grouping of a large number of videos to produce a smaller number of clusters. The algorithm's two objectives in the first step of normalization are to get the average user rating for each movie cluster and normalize the user count for each cluster. In this process, a min-max normalization algorithm is used. In this case, we additionally require the lowest and maximum ranges of normalization, as well as the maximum and minimum user numbers. The initialization of all necessary data occurs in the first two phases. The overall rating of each cluster and the total number of people who have rated each cluster are shown in lines 3 through 9. In line 10 average ratings of each cluster are calculated. After that, the parameters maximum user count and minimum user count for each cluster are calculated (lines 12-18). Finally, line 20 calculates the normalized user count for each cluster by the mathematical formula as shown in equation 1.

$$NormUserCount[j] = (UserCount[j] - MinUserCount) / (MaxUserCount - MinUserCount) \times (MaxRange - MinRange) + MinRange \quad (1)$$

Procedure 3: FindRatingPerUser-NormRatingPerUser(P, n, k)

```
Set NormMaxRatingUser = 5, NormMinRatingUser = 1
for i = 1to n
   Set TotalUserRating = 0, count = 0
   Set MaxRating = 0,MinRating = ∞
   for j = 1to k
      if ( P[i, j] > 0)
TotalUserRating = TotalUserRating + P[i][j]
         count = count +1
         if (  P[i][j]>MaxRating)
MaxRating=  P[i][j]
         end if
         if (  P[i][j]<MinRating)
MinRating = P[i, j]
         end if
       end if
   end for
MinRatingUser[i] = MinRating
AvgRatingUser[i] = TotalUserRating / count
MaxRatingUser [i]= MaxRating
NormAvgRatingUser[i] = ( AvgRatingUser[i] –
MinRatingUser[i] ) / ( MaxRatingUser[i] –
MinRatingUser[i]) * ( MaxRange – MinRange) +
MinRange
end for
```

FIGURE 6. **Pseudo code for Procedure 3**

Lemma 2: Procedure 2 (FindAverageUserRating-NormUserCount) requires O(nk) time.

D. Procedure 3: FindRatingPerUser-NormRatingPerUser

Next, the CNMR algorithm calls procedure 3 to find the user's minimum, maximum, and average ratings as well as to normalize the ratings. P matrix, user n, and the number of video clusters k are the inputs for procedure 3. The outcome of this process is to normalize the maximum, minimum, and average ratings of each user from matrix P, as well as the user ratings for each movie cluster. First of all, we have set the user's normalized minimum rating at 1 and their normalized maximum rating at 5. The purpose of lines 2 through 19 is to determine both the user's minimum and maximum ratings (line 17). By dividing the user's overall rating by count in line 18, you may determine each user's average rating. The normalized average ratings for each user are also found on line 20.

Lemma 3: Procedure 3 takes O(nk) times for execution.

E. Procedure 4: FindRecommendation

In this process, the CNMR algorithm tries to find the ratings of the unrated or unwatched movie cluster. To get the possible rating of each user for all unwatched movie clusters respectively. First of all, we have to check if any user has not rated any movie cluster. That can be found in line 3 of procedure 4. If the condition is satisfied, then that movie cluster is assigned by the average rating of that cluster for that user. Else we make the previously existing rating of that cluster by the user to 0. If, for any cluster, the user's new rating exceeds the user's normalized average rating, the corresponding rating is then raised by the normalized user count. Otherwise, that rating is replaced by 0.

Procedure 4: FindRecommendation(P, n, k, AvgRatingCluster, NormAvgRatingUser, NormUserCount)

```
for i = 1to n
   for j = 1to k
      if (P[i][j] = 0)
T[i][j] = AvgRatingCluster [j]
       else
T[i][j]  = 0
       end if
      if ( T[i][j]  != 0 &  T[i][j] ≥NormAvgRatingUser
[i])
T1[i][j]  = T[i][j] * NormUserCount [j]
     else
T[i][j]= 0
     end if
   end for
end for
```

FIGURE 7. **Pseudo code for Procedure 4**

Lemma 4: Procedure 4 requires O(nk) times.

F. Procedure 5: FindOrderRecommendation

The input for operation 5 is a T1 matrix, where n represents the total number of users and k the total number of videos. Based on the rating value, the movie cluster is then put back together for the user's recommendation. All the suggestions for the recommendation cluster are kept in the temp matrix in descending order (line 3 to line 17). The lower indices of the temp matrix include the cluster with the higher rating values. For instance, the user should be suggested to watch the movie cluster that shows up first in the temp matrix's index (Line 19 to Line 21). After then, the user will be shown recommendations for further movie clusters, beginning with the temp matrix's second index and continuing forth.

Lemma 5: Procedure 5 (FindOrderRecommendation) takes $O(nk^2)$ time for execution.

G. Phase 2: Evaluating the RS

Phase 2 of the CNMR algorithm, which is the evaluating phase, is carried out after the designing phase to determine the algorithm's effectiveness. The assessing part of the suggested algorithm NCFR accepts as input a rating matrix with n users and m items. The testing matrix, or TE, is then built using some of the ratings from the rating matrix, as shown in line 1 of figure 7. The training matrix makes up the remaining part of the rating matrix, which is a sub-matrix of the testing matrix. Phase 1 of the CNMR algorithm processes the training matrix first, followed by the execution of method 6 in the evaluating phase.

FIGURE 8. **Pseudo code for Procedure 5**

FIGURE 9. **Phase 2: Evaluating the RS**

Procedure 6: FindRatingForTestingRatingMatrix(P, TE, n, m, NormUserCount, AvgRatingMovie, NormAvgRatingUser, τ_1, τ_2, λ_1)

```
for i = 1to n
   for j = 1to m
      if P[i][j] = 0 & TE[i][j] ≠ 0
         if NormUserCount[j] ≥ τ1
T[i][j] = AvgRatingMovie[j]
         else if NormUserCount[j] ≤ τ2
T[i][j] = NormAvgRatingUser[i]
          else
T[i][j]  =  λ1 ×NormAvgRatingUser [i] + (1- λ1)×AvgRatingMovie [j]
          end if
       end if
   end for
end for
```

FIGURE 10. **Find Rating For Testing Rating Matrix**

H. Procedure 6: FindRatingForTestingRatingMatrix

For the evaluation process rather than checking movies cluster-wise, all the ratings are checked individually for each movie. So here we take the testing matrix TE for n users and m movies. Also, after processing the training matrix in procedure 1, again convert the results into a matrix of n users and m movies. This process determines whether or not any rating matrix element is zero. Line 3 also determines whether any component of the testing matrix is not zero. If the criterion is met, re-evaluate to see if the normalized user count exceeds the upper threshold or falls below the lower threshold. The normalized average rating of the user is then stored in the T matrix. If not, the weight values mentioned in Lines 8 and 9 are used to combine the weighted average rating of the user and the weighted average user rating of the item in a linear fashion.

Theorem 1: The design phase of CNMR takes $O(nk^2)$ time.

Theorem 2: Evaluating the phase of the CNMR requires O(nm) time.

Theorem 3: $O(nk^2)$ time is the complexity of the CNMR.

IV. RESULTS AND SIMULATION ANALYSIS

During the design and evaluation stages, a large number of simulations were conducted to show the effectiveness of the suggested algorithm, CNMR. Python Cells are being used for the dataset simulation in this case.

A. EXPERIMENTAL ANALYSIS

According to Table II, the seven users, identified as U1, U2, U3, U4, U5, U6, and U7, are rating ten films, including Spy Kids, Harry Potter, Spy Kids 2, Star Wars episode 1, Star Wars episode 2, Twilight, Star Wars episode 3, Harry Potter 3, Twilight 2, and Spy Kids 2. First of all, the CNMR algorithm calls procedure 1 (CLUSTER_MOVIES). The aim of procedure 1 is to cluster a large number of movies into a few numbers of possible clusters.

For the clustering of movies here a similarity measure is used called Jaccard similarity. This Jaccard similarity groups different movies into a unit if there are 75%similarities in their name by comparing each character one by one. So, by using this algorithm above 10 movies are clustered as Harry Potter (Harry Potter, Harry Potter 2, Harry Potter 3), Spy Kids (Spy Kids, Spy Kids 2), Star Wars Episode 1 (Star Wars Episode 1, Star Wars Episode 2, Star Wars Episode 3), and Twilight (Twilight, Twilight 2). The movies of the first cluster are stored in the first row of the Cluster matrix, and the movies of the second cluster are stored in the second row of the Cluster matrix as shown in Table II. Also, the average rating of the users for each cluster is determined and stored in the matrix P as shown in Table IV. Finally, whenever a cluster's average user rating exceeds 2.5 and if there exists any movie that is not watched by that user then those movies are recommended to that user like

in procedure 1, U1 is recommended Star Wars episode 2, and Star Wars episode 3 and U2 is recommended Twilight and Star Wars episode 3 and similarly to all other users.

Next step 2 of the CNMR is executed. The aim of procedure 2 is based on normalization. It finds the average ratings of each user for each movie cluster and the normalized user count for each movie cluster. After calculation, we will get the normalized user count of all the clusters Harry Potter, Spy Kids, Star Wars episode 1, and Twilight are 3, 3, 1, and 5 respectively. The average rating of each cluster is also calculated.

process 3, the methodology specified in procedure 3 of the CNMR designing phase is used to determine the normalized average rating of all users. As the result of step 3, we have normalized the average rating of each user as 3.66, 3.27, 3.21, 3, 2.33, 1, and 3.33 respectively. The testing rating matrix is then transformed into a normalized rating matrix in step 4.

TABLE II

USER-MOVIE RATING MATRIX

	Harry Potter	**Spy Kids**	**Harry Potter2**	**Star Wars Episode 1**	**Star Wars Episode 2**	**Twilight**	**Star Wars Episode 3**	**Harry Potter 3**	**Twilight 2**	**Spy Kids 2**
U1	5		5	3		4		2	4	
U2	1		1	2	3			2	3	
U3	5	5	4			4		5		5
U4		4				3			1	4
U5		3				3	5			3
U6	4		3	3	4		4	4		
U7	1	2	2	2	2	1		1		

TABLE III

CLUSTER OF SIMILAR MOVIES

Cluster Name	**Cluster Movies**		
Harry Potter	Harry Potter	Harry Potter 2	Harry Potter 3
Spy Kids	Spy Kids	Spy Kids 2	
Star Wars Episode 1	Star Wars Episode 1	Star Wars Episode 2	Star Wars Episode 3
Twilight	Twilight	Twilight 2	

The average rating of each cluster in matrix T is substituted for all ratings that include a null, as shown in Table IV. The average ratings of the clusters are 3, 3.25, 3.23, and 2.83. The new rating, which is higher than the user's normalized average rating, is multiplied by the normalized user count and that rating number is substituted. Otherwise, the lesser rating is replaced by the null. Finally, in procedure 5 the CNMR system recommends movies to the users. Here movie cluster is recommended in descending of the updated normalized rating of a user for the different clusters shown in Table IV. Star Wars: Episodes 1 and 2, which are both

suggested in Procedure 1, are the films that are suggested to U1. The sum of the movies suggested in process 1 and the movies in the cluster that are recommended in procedure 5 is the total number of movies that are recommended to a user in this case. Movies recommended to U2 are Twilight, Star Wars Episode 3, Spy Kids, and Spy Kids 2. Spy Kids 2, Star Wars Episodes 1, 2, and 3 are suggested to be watched on U3. U4 is recommended for Star Wars Episode 1, Star Wars Episode 2, and Star Wars Episode 3. The following movies are suggested for U5: Twilight 2, Harry Potter, and Potter 2; Star Wars: Episodes 1 and 2. Also, U6 is recommended for Twilight, Twilight 2, Spy Kids 1, and Spy Kids 2.

TABLE IV

USER-CLUSTER RATING MATRIX AFTER PROCEDURE 3

	Harry Potter	Spy Kids	Star Wars Episode 1	Twilight	Norm avg Rating_user
U1					**3.66**
U2		**3.5**			**3.27**
U3			**9.7**		**3.21**
U4	**9**		**9.7**		**3**
U5	**9**				**2.33**
U6		**3.5**		**14.16**	**1**
U7					**3.33**
Norm_user	**3**	**1**	**3**	**5**	
Count					

	Harry Potter	Spy Kids	Star Wars episode 1	Twilight
U1	**4**		**3**	**4**
U2	**1.33**		**2.5**	**3**
U3	**4.66**	**5**		**4**
U4		**4**		**2**
U5		**3**	**5**	**3**
U6	**3.66**		**3.66**	
U7	**1.33**	**2**	**2**	**1**

TABLE V

RATING MATRIX AFTER APPLYING MIN-MAX NORMALIZATION

V. PERFORMANCE ANALYSIS

A. PERFORMANCE ANALYSIS OF THE DESIGNING PHASE

We have considered different criteria for the analysis of the performance of our proposed algorithm. For analysis purposes, we contrasted our algorithm with the one that is currently in use. Here all the

recommendations are taken into consideration, and we have compared the following data generated in phase 1 and studied in Table-7.

1. Minimum number of movies recommended
2. Total number of movies recommended
3. Average number of movies recommended
4. The maximum number of movies recommended

Phase 1 of the proposed CNMR algorithm recommends a total of 28 movies to the different users in the rating matrix. There is an existing algorithm for item recommendation in which the same normalization processes recommend 18 items.

Here we can see our proposed CNMR algorithm is not recommending any movie to U7. However, the existing algorithm recommends at least 1 item to each user. The reason behind not recommending any movie to user U7 is that U7 has not given a single rating that is more than the average rating which is 2.5. Also, U7 has rated a maximum number of movies as low rating i.e., 1 or 2. So we have designed our algorithm by clustering which results in reducing the rating matrix (P) size to smaller for which there occurs a possibility of no recommendation to any user. A comparison of the performance of ratings of our proposed algorithm and the previously existing algorithm by examining the testing matrix of n users and m movies is shown in Table8.

TABLE VI

RECOMMENDATION MATRIX WITH THE ORDER OF RECOMMENDATION

	Harry Potter	**Spy Kids**	**Harry Potter2**	**Star Wars Episode 1**	**Star Wars Episode 2**	**Twilight**	**Star Wars Episode 3**	**Harry Potter 3**	**Twilight 2**	**Spy Kids 2**
U1					**1**		**2**			
U2		**1**				**4**	**3**			**2**
U3				**1**	**2**		**3**		**4**	
U4	**1**		**2**	**4**	**5**		**6**	**3**		
U5	**1**		**2**	**4**	**5**			**3**	**6**	
U6		**1**				**3**			**4**	**2**
U7										

TABLE VII

PERFORMANCE ANALYSIS OF PHASE 1

Algorithm	Number of movies recommended			
	Maximum	Minimum	Total	Average
Proposed	6	0	26	4.33
Panda Set al.[18]	6	1	18	2.57

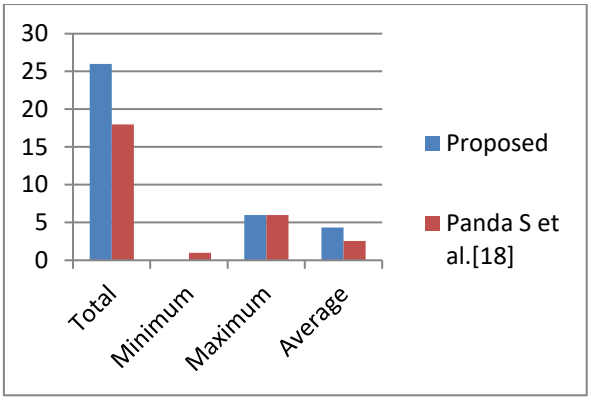

FIGURE 11. **Performance Analysis Graph**

B. EVALUATION OF THE ALGORITHM WITH TESTING MATRIX

In this chapter, we will evaluate our proposed CNMR algorithms using different comparison parameters. Comparing the suggested method to the current algorithm first, we will consider a testing matrix. The testing matrix is a sub-set of the rating matrix which was taken into consideration in the previous chapter. So, after making a testing matrix, remaining the part of the previous rating matrix is considered as a rating matrix. Table VII and Table IX show the testing and rating matrix.

Now phase 1 of the CNMR algorithm is evaluated. After that, the evaluating phase is processed. In phase 2 all the performance measurement parameters as described below are calculated.

a) Mean Absolute Error: The MAE is a statistical measure that illustrates the mean absolute difference between received and anticipated scores. Mathematically it is expressed as

$$MAE = \frac{1}{t}\sum_{i=1}^{n}\sum_{j=i}^{m}\left|T_{ij}^{!!!} - TE_{ij}\right| \times F_{ij} \quad \textbf{(2)}$$

Where, $F_{ij} = \begin{cases} 1 & if TE_{ij} \neq 0 \\ 0 & otherwise \end{cases}$

t = total number of ratings

b) Root Mean Square Error (RMSE): RMSE is the total square difference that separates the actual scores from the predicted ones. It is theoretically calculated as

$$RMSE = \sqrt{\frac{1}{t}\sum_{i=1}^{n}\sum_{j=1}^{m}(R_{ij}''' - TE_{ij})^2 \times F_{ij}} \quad \textbf{(3)}$$

c) Precision (P): It is the total of both false positives and true positives (TP + FP), expressed as a ratio of true positives (TP).

$$P = \frac{truepositives}{truepositives + falsepositives} \quad \textbf{(4)}$$

Precision (P) for CNMR= 0.625

d) Recall (R): It's the sum of accurate positives and inaccurate negatives (FN) divided by the ratio of true positives (TP).

$$R = \frac{truepositives}{truepositives + falsenegatives} \quad \textbf{(5)}$$

e) F-Score: It represents the precision and recall harmonic mean. Mathematically it is represented as

$$F = 2 \times \frac{P \times R}{P + R} \quad \textbf{(6)}$$

f) Fowlkes-Mallows Index (FMI): This shows the geometric means of recall (R) and accuracy (P). In mathematics, it is expressed as

$$FMI = \sqrt{P \times R} \quad \textbf{(7)}$$

TABLE VIII

TESTING MATRIX TE

	Harry Potter	Spy Kids	Harry Potter2	Star Wars Episode 1	Star Wars Episode 2	Twilight	Star Wars Episode 3	Harry Potter 3	Twilight 2	Spy Kids 2
U1	**5**									
U2			**1**		**3**					
U3		**5**								**4**
U4						**3**				
U5										
U6	**4**							**4**		
U7				**2**						

TABLE IX

TRAINING MATRIX R

	Harry Potter	Spy Kids	Harry Potter2	Star Wars Episode 1	Star Wars Episode 2	Twilight	Star Wars Episode 3	Harry Potter 3	Twilight 2	Spy Kids 2
U1			**5**	**3**		**4**		**2**	**4**	
U2	**1**			**2**				**2**	**3**	
U3	**5**		**4**			**4**		**5**		
U4		**4**							**1**	**4**
U5		**3**				**3**	**5**			**3**
U6			**3**	**3**	**4**		**4**			
U7	**1**	**2**	**2**		**2**	**1**		**1**		

TABLE X

PERFORMANCE ANALYSIS CNMR

Algorithm	MAE	RMSE	TP	FP	FN	Precision	Recall	F-Score	FMI
Proposed	1.62	1.94	5	3	4	0.625	0.555	0.588	0.589
Panda Set al.[17]	1.69	2.009	4	3	5	0.571	0.444	0.500	0.503

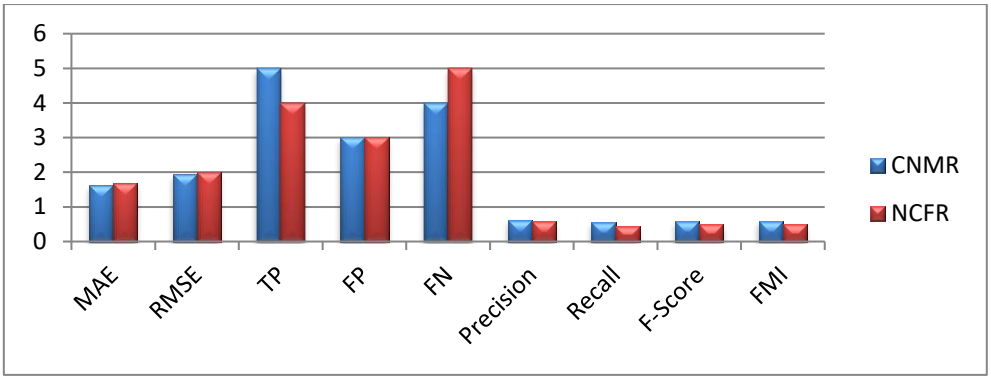

FIGURE 12. **Performance Comparison Graph of proposed model with existing RS model**

The comparison between our suggested CNMR method and the existing min-max normalization-based algorithm (NCFR) in Fig. 10 and Table X demonstrates that the new proposed algorithm outperforms the previous algorithm for movie recommendation. Although CNMR failed to recommend any movie to U7 as the user does not like the movie series in other aspects it performs better than the previous algorithm.

VI. CONCLUSIONS AND FUTURE WORK

This work introduced the normalization-based filtering collaborative filtering strategy for RS. The aforementioned methodology employs the widely recognized min-max normalization technique to address the limitations of the prior collaborative filtering-based approach. To mitigate the drawbacks of the present CF approach, we have used the clustering technique. The time required for the execution of the designing phase is $O(nk^2)$ and the evaluating phase takes O(mn) time. Seven individuals and 10 objects were used to thoroughly explain the algorithm. The findings of the simulation were produced using the following metrics: FMI, precision, recall, MAE, RMSE, maximum, minimum, average, and total number of suggested items.

It is evident from the comparison tables mentioned above in the chapter that our proposed method has fewer errors than the one that is already in use. Additionally, other measurement metrics including precision, recall, F-score, and FMI are superior to the current RS methods. So, the proposed CNMR performs more effectively in the area of movie suggestion.

The proposed work does have some limitations, albeit these will be improved in subsequent work. In the previous chapter, we saw CNMR failed to recommend any movie to U7, so in feature work, we will make our algorithm to reduce the error. In the feature work also, we will try to cluster the users as well as movies using the contents of the movies to make a more effective movie recommendation algorithm.

REFERENCES

[1] H. J. Ahn, “A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem,” Information Sciences, vol. 178, no. 1, pp. 37–51, Jan. 2008.

[2] J. Bobadilla, F. Serradilla, and A. Hernando, “Collaborative filtering adapted to recommender systems of e-learning,” Knowledge-Based Systems, vol. 22, no. 4, pp. 261–265, May 2009.

[3] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, “Recommender systems survey,” Knowledge-Based Systems, vol. 46, pp. 109–132, Jul. 2013.

[4] C. Wei, R. Khoury, and S. Fong, “Web 2.0 Recommendation service by multi-collaborative filtering trust network algorithm,” Information Systems Frontiers, vol. 15, no. 4, pp. 533–551, Sep. 2012.

[5] Y. Du, X. Du, and L. Huang, “Improve the Collaborative Filtering Recommender System Performance by Trust Network Construction,” Chinese Journal of Electronics, vol. 25, no. 3, pp. 418–423, May 2016.

[6] M. Hong, S. An, R. Akerkar, D. Camacho, and J. J. Jung, “Cross-cultural contextualization for recommender systems,” Journal of Ambient Intelligence and Humanized Computing, Sep. 2019.

[7] F. O. Isinkaye, Y. O. Folajimi, and B. A. Ojokoh, “Recommendation systems: Principles, methods and evaluation,” Egyptian Informatics Journal, vol. 16, no. 3, pp. 261–273, Nov. 2015.

[8] D. P. Karidi, Y. Stavrakas, and Y. Vassiliou, “A Personalized Tweet Recommendation Approach Based on Concept Graphs,” Jul. 2016.

[9] S. C. Kim, K. J. Sung, C. S. Park, and S. K. Kim, “Improvement of collaborative filtering using rating normalization,” Multimedia Tools and Applications, vol. 75, no. 9, pp. 4957–4968, Dec. 2013.

[10] Y. Koren, “Collaborative filtering with temporal dynamics,” Communications of the ACM, vol. 53, no. 4, p. 89, Apr. 2010.

[11] Y. Koren, R. Bell, and C. Volinsky, “Matrix Factorization Techniques for Recommender Systems,” Computer, vol. 42, no. 8, pp. 30–37, Aug. 2009.

[12] G. Linden, B. Smith, and J. York, “Amazon.com recommendations: item-to-item collaborative filtering,” IEEE Internet Computing, vol. 7, no. 1, pp. 76–80, Jan. 2003.

[13] J. Lu, D. Wu, M. Mao, W. Wang, and G. Zhang, “Recommender system application developments: A survey,” Decision Support Systems, vol. 74, pp. 12–32, Jun. 2015.

[14] X. Luo, Y. Xia, and Q. Zhu, “Incremental Collaborative Filtering recommender based on Regularized Matrix Factorization,” Knowledge-Based Systems, vol. 27, pp. 271–280, Mar. 2012.

[15] T. Majeed, A. Stämpfli, A. Liebrich, and R. Meier, “What is of interest for tourists in an alpine destination: personalized recommendations for daily activities based on view data,” Journal of Ambient Intelligence and Humanized Computing, vol. 11, no. 11, pp. 4545–4556, Dec. 2019.

[16] S. K. Nayak and S. K. Panda, “A User-Oriented Collaborative Filtering Algorithm for Recommender Systems,” 2018 Fifth International Conference on Parallel, Distributed and Grid Computing (PDGC), Dec. 2018.

[17] S. K. Panda, S. K. Bhoi, and M. Singh, “A collaborative filtering recommendation algorithm based on normalization approach,” Journal of Ambient Intelligence and Humanized Computing, vol. 11, no. 11, pp. 4643–4665, Jan. 2020.

[18] Radek Pelánek, “Measuring Predictive Performance of User Models,” Jul. 2017.

[19] I. R. Porteous, A. Asuncion, and M. Welling, “Bayesian Matrix Factorization with Side Information and Dirichlet Process Mixtures,” Proceedings of the ... AAAI Conference on Artificial Intelligence, vol. 24, no. 1, pp. 563–568, Jul. 2010.

[20] K. Raqim Raheem and I. Hadi Ali, “Survey: Affective Recommender Systems Techniques,” IOP Conference Series: Materials Science and Engineering, vol. 928, p. 032042, Nov. 2020.

[21] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, “Item-based collaborative filtering recommendation algorithms,” Proceedings of the tenth international conference on World Wide Web - WWW ’01, May 2001.

[22] F. Xie, M. Xu, and Z. Chen, “RBRA: A Simple and Efficient Rating-Based Recommender Algorithm to Cope with Sparsity in Recommender Systems,” Mar. 2012.

[23] Y. Yang, D. Hooshyar, J. Jo, and H. Lim, “A group preference-based item similarity model: comparison of clustering techniques in ambient and context-aware recommender systems,” Journal of ambient intelligence & humanized computing/Journal of ambient intelligence and humanized computing, vol. 11, no. 4, pp. 1441–1449, Sep. 2018.

[24] Z. Yang, B. Wu, K. Zheng, X. Wang, and L. Lei, “A Survey of Collaborative Filtering-Based Recommender Systems for Mobile Internet Applications,” IEEE Access, vol. 4, pp. 3273–3287, 2016.

[25] S. P. Dash, R. K. Sahoo, and C. R. Padhan, “Matrix Factorization Based Normalization in Movie Recommendation Systems,” Mar. 2024.