mathematics

MDPI

Article

Design and Comparative Analysis of New Personalized Recommender Algorithms with Specific Features for Large Scale Datasets

S. Bhaskaran, Raja Marappan * and B. Santhi

School of Computing, SASTRA Deemed University, Thanjavur 613401, India; bhaskaran@mca.sastra.edu (S.B.); shanthi@cse.sastra.edu (B.S.)

* Correspondence: raja_csmath@cse.sastra.edu

Received: 27 May 2020; Accepted: 30 June 2020; Published: 6 July 2020

Abstract: Nowadays, because of the tremendous amount of information that humans and machines produce every day, it has become increasingly hard to choose the more relevant content across a broad range of choices. This research focuses on the design of two different intelligent optimization methods using Artificial Intelligence and Machine Learning for real-life applications that are used to improve the process of generation of recommenders. In the first method, the modified cluster based intelligent collaborative filtering is applied with the sequential clustering that operates on the values of dataset, user's neighborhood set, and the size of the recommendation list. This strategy splits the given data set into different subsets or clusters and the recommendation list is extracted from each group for constructing the better recommendation list. In the second method, the specific features-based customized recommender that works in the training and recommendation steps by applying the split and conquer strategy on the problem datasets, which are clustered into a minimum number of clusters and the better recommendation list, is created among all the clusters. This strategy automatically tunes the tuning parameter λ that serves the role of supervised learning in generating the better recommendation list for the large datasets. The quality of the proposed recommenders for some of the large scale datasets is improved compared to some of the well-known existing methods. The proposed methods work well when $\lambda = 0.5$ with the size of the recommendation list, $|L| = 30$ and the size of the neighborhood, $|S| < 30$. For a large value of $|S|$, the significant difference of the root mean square error becomes smaller in the proposed methods. For large scale datasets, simulation of the proposed methods when varying the user sizes and when the user size exceeds 500, the experimental results show that better values of the metrics are obtained and the proposed method 2 performs better than proposed method 1. The significant differences are obtained in these methods because the structure of computation of the methods depends on the number of user attributes, λ, the number of bipartite graph edges, and $|L|$. The better values of the (*Precision*, *Recall*) metrics obtained with size as 3000 for the large scale *Book-Crossing* dataset in the proposed methods are (0.0004, 0.0042) and (0.0004, 0.0046) respectively. The average computational time of the proposed methods takes <10 seconds for the large scale datasets and yields better performance compared to the well-known existing methods.

Keywords: e-learning; artificial intelligence; machine learning; personalized recommender; recommendation list; collaborative filtering; recommender systems; intelligent optimization

1. Introduction

Recommender systems are broadly utilized to assist users in handling the great amount of information available on the web, particularly in searching for the most appropriate content tailored to

the specific preferences of the user's. Because of the large esteem of this type of organization and required to guarantee that such interfaces provide the user with huge relevance and quality, the mechanisms for making these recommendations should be updated continuously [1]. The recommendations can also be personalized and non-personalized in approach as per the user's characteristics. A list of best-recommended products on the site is provided for a personalized recommendation. The product may be recommended based on an analysis of users' past behavior or the statistical advice provided by the other user, whereas non-personalized are simple to make, as they are independent of user's actions.

Conventional recommender systems are necessarily a content-based and Collaborative Filtering (CF) system [2]. The traditional method utilized for recommendations is CF. Recommender systems based on CF calculate user preferences for products or services by learning past user–item relationships from a group of users sharing the same interests and tastes. An additional standard scheme when designing referral systems is content-based filtering. Content-based filtering schemes are according to the description of the item and profile of the user's preferences. These schemes are well-matched for situations where the known data of an object (name, location, description, etc.) are present, but not in the user list. Despite the achievement of these two filtration techniques, several drawbacks have been recognized. Most of the issues are connected with content-based filtering methods that include limited content analysis, over-specialization, and spacing of data [3]. Moreover, joint approaches reveal cold-start, spacing, and scaling problems. These issues generally decrease the quality of referrals. To alleviate some of the issues identified, hybrid filtration has been proposed combining two or more filtration methods in various ways to enlarge the accuracy and efficiency of the recommender systems [4–18].

The important research in recommender applications is the development of a good recommendation system that is expected to create a better recommendation list, based on the specific needs of the users [13,19–30]. To resolve the problems in the existing methods, this research focuses on the design of two different intelligent optimization methods using Artificial Intelligence (AI) and Machine Learning (ML) for real-life applications that are used to improve the process of generation of recommenders. In the first method, the modified cluster based intelligent CF is applied with the sequential clustering that operates on the values of dataset, user's neighborhood set, and |L|. In the second method, the specific features-based customized recommender that works in the training and recommendation steps by applying the split and conquer strategy on the problem datasets, which are clustered into a minimum number of clusters and the better recommendation list, is created among all the clusters.

Some of the research gaps in the existing recommender methods are: having more deviations in the performance measurements and taking a lot of computational complexity, which results in less accuracy in the generation of the recommendation list [13,23,24,31–39]. Hence, it is necessary to design the new recommender strategies to offset the issues in the well-known methods of solving real-life applications [6,33,38,40–47]. Section 2 focuses on the survey on recent recommender algorithms in the generation of a better recommendation list. The requirement of the design of new recommender methods for real-world applications is also discussed in Section 2. The notations and the definitions applied in the proposed methods are explained in Section 3. The proposed recommender algorithms are presented in Section 4. The simulation of the proposed strategies, along with the analysis of the experimental results, is focused on in Section 5. The conclusions of this research with the future research areas are discussed in Section 6.

2. Literature Survey on Recent Recommenders & the Need for New Recommender Strategies

2.1. User Profile Orientation Recommenders

A few personalized recommenders have been developed recently for real-world applications. The User Profile Oriented Diffusion (UPOD) strategy to learn the user profile is developed [6]. The UPOD strategy makes customized recommendations through diffusion, integrating innovation

with the familiarity of products that are functioning in two different phases such as training and recommendation. The training phase of UPOD performs operations such as data prediction, defining the values of users' features, feature-based clustering of users, creating a bilateral map of interactions, and searching the profile of every cluster. The training phase trains a classifier, which provides features of the target user. The referral phase of UPOD requires an input map, a target user, a set of user attributes, a classifier trained in the training operation, and the size of the referral list. In this strategy, the tuning parameter is automatically adjusted to evaluate the amount of mixing in mass diffusion activity for different sparse datasets. This strategy generates recommendations based on certain features of the user's profile. Most importantly, this strategy includes the user's profile information for refining and personally recommending the content to the users. This strategy is simulated with the parameters: $\lambda = 0.5$, the size of the recommendation list $|L| = 30$, size of the neighborhood is 30, *k-minimum* to 100, *k-maximum* to 200.

2.2. Content-Based Recommenders

The content-based recommenders using the Convolution Neural Network (CNN) model are developed [7]. CNN can be utilized to find hidden factors from the textual data of media assets. To practice CNN, its input and output should initially be settled. For its input, the language model is utilized. For its discharge, structure the latent factor scheme, which is obliged by the L1-calculation. Also, the split Bregman iteration scheme is designed to solve the system. The main improvement of the designed recommendation method is that the text information is utilized straightly to perform the content-based recommendation without tagging.

2.3. Hybrid Recommenders

The hybrid scheme that exploits genomic tags of movies integrated with the content-based filter to recommend related movies is developed [8]. It utilizes Principal Component Analysis (PCA) and correlation coefficient methods to decrease the tags, which are superfluous and demonstrate a minimum proportion of variance. The designed system using content-based filtering on the average rating of the movie, it will suggest top N movies to the users. The e-learning personalization, according to the hybrid recommendation strategy and learning style identification, is discussed [9]. This describes the recommendation module of a programming training method. In this work, the system can automatically adjust to the learner's interests and knowledge levels. The designed system recognizes dissimilar styles of learning style and learner's habits by testing the learning style of the learners and extracting their server records. First, it progresses clusters according to different learning styles. Then, it investigates the habits and interests of learners by extracting scenes frequently. Finally, the system completes a customized recommendation of the learning content following these continuous visualizations provided by the recommender system.

2.4. Filter-Based Recommenders

The personalized travel route recommenders with the help of CF based on Global Positioning System (GPS) trajectories are designed [10]. The presented methods consider users' personal travel preferences according to their historical GPS routes. In this work, first, compute the frequencies of the user's travel behavior using the common filtering technique. A path with the highest probability of user travel behavior is then computed according to the innocent Bayes scheme. The extended version of Collaborative Travel Route Recommendation (CTRR), (CTRR+) scheme, progresses the performances of CTRR by considering cold start users and combing distance with the user travel behavior probability. The investigational outcome shows that the introduced scheme attains good output performance for travel route recommendations matched with the shortest distance path scheme.

2.5. Features-Based Recommenders

The feature-based recommender provides personalized recommendations to users in solving ERP System and E-Agribusiness datasets by performing some configuration functions initially [48]. The evaluation process requires some offline evaluation of parametric optimization and splitting of the dataset into disjoint training and test dataset. The choice of the method has a great impact on the recommendation quality. Modified CF is proposed for both user-based and object-based cases in solving the *MovieLens* dataset [49]. The group recommendation model is proposed based on factors such as sparsity, dynamics, and timeliness [50]. When more additional features are considered, intelligent optimization strategies are further required to minimize the Root Mean Squared Error (RMSE). However, for large scale datasets, to provide a better recommendation, the proposed intelligent optimization methods are competitive to some of the well-known existing methods [51].

The similarity-based targeting along with the baseline approach and latent factor models, which are treated with adaptive regularization technique that provides personalization to both users and items, is discussed in [52]. The recommendation system for users to recommend books is presented in [53]. The bottlenecks of the paper are to test the system with other intelligent techniques to improve the performance of the system. The building recommendation system for online news is described in [54]. The recommendation strategy mainly focuses on user personalization and browsing history. It was just a micro service recommendation system. The architecture of an intelligent and autonomous recommendation system to be used in any virtual learning environment to efficiently recommend digital resources is presented in [55]. The architecture extracts information from the context of the students, identifying variables such as individual learning styles, socioeconomic information, connection log information, location information, among others. It uses the learning styles of the students, the context information, the social networks, among other sources, to select the best digital resources. However, the integration of recommendation methods was not analyzed. The approach consists of some functional components that assist in determining users like active users and also to find the value of |L| [56]. This recommender system uses a specialized weight calculation block that assists to place the various items at different positions of L. For small values of L, it has been found that the precision metric for the traditional benchmark is very low. The machine learning model that recommends a suitable candidate's resume to the human resources based on the given job information is presented [57]. This model operates in two stages: Initially, the resume is categorized. Then, the recommendation is applied based on the similarity index measurement with the given job information. Further enhancement of this model can be enhanced by applying deep learning strategies. The system for the contextual collaborative recommendation that addresses the issues of the n-dimensional contextual complexity models with new users and items is discussed [58]. This model is simulated in a healthy food field where just a few proposed methods are interested in research in this area. They have used a very small sample size of 524 users only. More intelligent techniques can be applied along with large scale datasets [59–61].

The recommender framework for personalization and relevance feedback for some online applications is developed with a size of the dataset of 2500 videos in [62]. This recommender framework focuses on the development of enriched multimedia content which is targeted to the user's preferential information. The recommendation is implemented to extract the video information through the collection of relevance feedback mechanisms from the user interactions. However, the proposed recommender algorithms are designed to provide a better recommendation for large scale datasets with different categories of features such as users, items, interactions, age, location, gender, country, etc. [19]. The proposed recommender algorithms can also be extended to support the recommendation based on the video information for online applications [63–65].

The context-aware video recommender system is developed to improve recommendation performance by incorporating contextual features along with the conventional user-item ratings used by video recommender systems [66]. The CF algorithm is discussed to confront the sparsity problem in the resulting graph partitions that may improve the prediction performance of parallel implementations

without strongly affecting their time efficiency [67]. The parallel hardware implementation based algorithm is developed for embedded CF applications with large datasets [68]. The online recommendation algorithm is designed, which combines clustering and CF techniques to improve the accuracy of online recommendation systems for group-buying applications [69]. The recommender system development is discussed that uses several algorithms to obtain groupings [70].

Nowadays, the new recommender algorithms are required for real-world applications, because of the following reasons [1–18,21–24,29–32,43–47,66–70]:

- One of the main reasons why we need a recommender system in modern society is that there are many ways for people to use the Internet. For example, Netflix has an enormous collection of movies. Despite the increase in the amount of information available, a new problem arose due to the difficulty of selecting the items that people wanted to see;
- A recommender system attempts to assess and predict user content preferences related to games, stories, or videos. The system draws from data usage history aimed at making recommendations based on the user's (current) interests;
- In the e-commerce system, recommender systems improve revenue because they are the best way to sell more products;
- A company with a list of thousands and thousands of products will be hard-pressed for all its products with hardcore product recommendations, and such standard recommendations will soon be outdated or inappropriate for many customers by using a variety of methods for filtering; you can find business hours to recommend new products that you can buy (whether on their site, via email, or otherwise);
- The recommender system should provide more precise and personalized recommendations than the existing systems. The outcome of the recommender should prove the correctness of the recommendation based on the specific needs of the users.

3. Notations & Definitions

This research focuses on the design of two different intelligent optimization strategies using AI and ML for real-life applications that are used to generate the better recommendation. The proposed recommender algorithms used the following notations and definitions:

3.1. Correlation Coefficient

$r(x,\ y)$, the correlation coefficient between two random variables or users X and Y for n pairs of observations checks the existence of a linear relationship between them, and it is computed using $Cov(X,\ Y)$, the covariance between random variables X and Y; and $\sigma_x,\ \sigma_y$, the standard deviations of random variables X and Y, respectively, and is defined as follows [32]:

$$r(x,\ y) = \frac{Cov(X,\ Y)}{\sigma_x \sigma_y} \tag{1}$$

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

In general, $0 \le r(x, y) \le 1$. The values of $r(x,\ y) = -1$ and $+1$ signify the perfect negative and perfect positive correlation, respectively. $Cov(X,\ Y)$, the covariance between variables X and Y becomes zero when X and Y are independent. $Cov(X,\ Y)$ is defined as follows:

$$Cov(X,\ Y) = \mathrm{E(XY)} - \mathrm{E(X)E(Y)} \tag{3}$$

The expectation of a random variable X is given by $E(X) = \overline{x}$. Then, the equation $r(x,\ y) = 0$ represents the random variables X and Y, which are independent.

The mean of random variables X and Y are defined as follows:

$$\bar{x} = \frac{1}{n} \sum_{j=1}^{n} x_j \text{ and } \bar{y} = \frac{1}{n} \sum_{j=1}^{n} y_j \tag{4}$$

The unbiased estimators of σ_x, σ_y are the following:

$$\sigma_x = \sqrt{\frac{1}{n-1} \sum_{j=1}^{n} \left(x_j - \bar{x}\right)^2} \tag{5}$$

$$\sigma_y = \sqrt{\frac{1}{n-1} \sum_{j=1}^{n} \left(y_j - \bar{y}\right)^2} \tag{6}$$

3.2. Pearson Correlation Coefficient

The correlation coefficient between two random variables x and y according to Pearson is given by

$$PearsonCorrCoeff(x,\ y) = \frac{\sum_{j \in I_{xy}} \left(r_{x,j} - \bar{r}_x\right)\left(r_{y,j} - \bar{r}_y\right)}{\sqrt{\sum_{j \in I_{xy}} \left(r_{x,j} - \bar{r}_x\right)^2} \sqrt{\sum_{j \in I_{xy}} \left(r_{y,j} - \bar{r}_y\right)^2}} \tag{7}$$

where I_{xy} defines the set of items rated by variables of two users x and y [32]. The rating values of jth item are defined as $r_{x,j}$ and $r_{y,j}$ for users x and y, respectively. The average ratings of all items that interacted with users x and y are defined as $\bar{r}_x$ and $\bar{r}_y$, respectively. $PearsonCorrCoeff(\mathrm{x},\ \mathrm{y}) \in \mathbb{R}$ and $-1 \leq PearsonCorrCoeff(x,\ y) \leq 1$.

3.3. Item Prediction

The prediction of item j (or ratings for items) for a user x is given by

$$Prediction\ (x,\ j) = \bar{r}_x + \frac{\sum_{a \in S} PearsonCorrCoeff(x,\ a)\left(r_{a,j} - \bar{r}_a\right)}{\sum_{a \in S} PearsonCorrCoeff(x,\ a)} \tag{8}$$

where S is the set of users similar to user x concerning the values of the neighborhood [31]. $PearsonCorrCoeff(x,\ a)$ defines the similarity between two variables x and a. The rating value of jth item is defined as $r_{a,\ j}$ for the user a. The mean rating of all items interacted with users a is defined as $\bar{r}_a$. Typical values of the cardinality of the set S lie between 20 and 40 to maximize the item prediction for a better recommendation.

3.4. Mass Diffusion Resource Values

Let $degree(j)$ and $degree(v)$ represent the vertex degrees corresponding to the item j and the user v, respectively. Then, the resource values, according to Mass Diffusion algorithm, are defined as follows [46]:

$$r'_{Mass\ Diffusion}(v,\ j) = \sum_{j \in I} \frac{r(u,\ j)}{degree(j)},\ v \in U,\ j \in I \tag{9}$$

$$r''_{Mass\ Diffusion}(v,\ j) = \sum_{v \in U} \frac{r'_{Mass\ Diffusion}(v,\ j)}{degree(v)},\ v \in U,\ j \in I \tag{10}$$

3.5. Heat Spreading Resource Values

The resource values, $r'_{Heat\ Spreading}(v,\ j)$ and $r''_{Heat\ Spreading}(v,\ j)$, are computed according to the Heat Spreading algorithm, and are defined as follows [46]:

$$r'_{Heat\ Spreading}(v,\ j) = \sum_{j \in I} \frac{r(u,\ j)}{degree(v)},\ v \in U,\ j \in I \tag{11}$$

$$r''_{Heat\ Spreading}(v,\ j) = \sum_{v \in U} \frac{r'_{Heat\ Spreading}(v,\ j)}{degree(j)},\ v \in U,\ j \in I \tag{12}$$

3.6. Mass Diffusion Heat Spreading Resource Values

Let λ be a tuning parameter that lies between 0 and 1. Then the resource values, $r'_{MDHS}(v,\ j)$ and $r''_{MDHS}(v,\ j)$, are calculated according to Mass Diffusion Heat Spreading algorithm and are defined as follows: [36]

$$r'_{MDHS}(v,\ j) = \left(\sum_{j \in I} \frac{r(u,\ j)}{degree(i)^{\lambda}} \right) / degree(v)^{1-\lambda},\ v \in U,\ j \in I \tag{13}$$

$$r''_{MDHS}(v,\ j) = \left(\sum_{v \in U} \frac{r'_{MDHS}(v,\ j)}{degree(v)^{\lambda}} \right) / degree(i)^{1-\lambda},\ v \in U,\ j \in I \tag{14}$$

3.7. Dissimilarity (Y, X)

Let Y and X represent the vectors of different attribute values of users corresponding to the values of categories, where $y_j \in Y$ and $x_j \in X$ are the values of every data attribute of users Y and X, respectively. Then the metric of dissimilarity between Y and X in k-modes computation for m attributes is defined as follows [6]:

$$Dissimilarity\ (Y,\ X) = \sum_{i=1}^{m} \delta(y_i,\ x_i) \tag{15}$$

where $\delta(y_i,\ x_i) = 0$ when $x_i = y_i$ and $\delta(y_i,\ x_i) = 1$ when $x_i \neq y_i$. For example, consider the two vectors X and Y with sample attribute values such as a_1 = profession, a_2 = age, and a_3 = sex and assume the sample values of vector X and Y are x_1 = engineer, x_2 = 20–25 and x_3 = female and y_1 = doctor, y_2 = 25–35 and y_3 = female respectively. The dissimilarity measurement between X and Y is obtained as follows: $Dissimilarity\ (Y,\ X) = 1 + 1 + 0 = 2$, resulting in only one attribute; a_3 = sex and is common to both vectors X and Y. This measurement finds the number of identical and non-identical attributes in both vectors X and Y. This function does not define a distance metric. This measurement is applied in the k-modes clustering algorithm in assigning the classifiers for each object in each cluster. The mode of clusters is evaluated using $Dissimilarity\ (Y,\ X)$ & $Dissimilarity\ (Q,\ X)$ measurements. Then the effective data partition is evaluated for each cluster.

3.8. Dissimilarity (Q, X)

Let $X_1, X_2, X_3 \ldots X_n$ be the set of n objects corresponding to the categories of vector X. Let Q be the attribute vector of the categories of these n objects. Let Q have the least value of $Dissimilarity\ (Y,\ X)$ in the cluster hand, and hence, it decreases the following function [6]:

$$Dissimilarity\ (Q,\ X) = \sum_{j=1}^{n} Dissimilarity\left(Q,\ X_j\right) \tag{16}$$

3.9. Column Entropy

Let $X = \{x_1, x_2, x_3 \ldots x_n\}$ be a vector of the dataset, which consists of c columns and n instances. Define a column vector to describe the instance x_i as ($x_i = \{x^i{}_1, x^i{}_2, x^i{}_3 \ldots x^i{}_c\}$). Then the column values of x_i are assigned from the finite number of unique categorical values of the domain set A_i. For a value $v \in A_i$, the probability of $x_i = v$ is given by $P(x_i = v)$. Let $p(v|X)$ be the empirical probability of $x_i = v$, which is evaluated in the dataset X. Then, the column entropy of domain A_i is defined as follows [16]:

$$H(A_i|X) = -\sum_{v \in A_i} p(v|X) log_2 p(v|X) \tag{17}$$

3.10. Expected Entropy

Partition the data set X into k clusters of objects as $C^k = \{C_1, C_2, C_3 \ldots C_k\}$. Let $(n_1, n_2, n_3 \ldots n_k)$ be the objects of the subset of C^k. Define $H(C^k)$ as the entropy of the cluster, which depends on the dataset. Then, $E\left(C^k\right)$, the expected entropy for $C^k = \{C_1, C_2, C_3 \ldots C_k\}$, is given by [16]

$$E\left(C^k\right) = \frac{1}{n} \sum_{k=1}^{k} n_k H(C_k) \tag{18}$$

3.11. Entropy-Based Clustering Criterion

The entropy-based clustering criterion is given by the function, $Optimize\left(C^k\right)$, which minimizes the value of expected entropy $E\left(C^k\right)$ [16]. For a constant c, this optimization criterion is defined as

$$Optimize\left(C^k\right) = \frac{1}{c}\left(H(X) - E\left(C^k\right)\right) \tag{19}$$

3.12. Recall (List, User)

Let $g(L)$ be the items count that is associated with the valid target user, $Valid_{Users}$, in the recommendation list L and testing set. Let I_{User} be the total items count that are related to the valid user's $User \in Valid_{Users}$. Then *Recall* (*List*, *User*) defines the proportion of items in the testing set $valid_{users}$ that are similar to the items in the recommendation list L that is created for a target user u and is given by [38]

$$Recall\ (List,\ User) = \frac{g(L)}{I_{User}} \tag{20}$$

3.13. Precision (List, User)

Let $|L|$ be the number of elements in the recommendation list L. Then, $Precision\ (List,\ User)$ defines the measure of the ratio of items in L that are corresponding to items connected to the target user *User* in the testing set. This metric is defined as follows [38]

$$Precision\ (List,\ User) = \frac{g(L)}{|L|} \tag{21}$$

3.14. Rank (i)

Rank (*i*), the rank of item i determines a position where items connected to the target user in the testing set appear in L and are defined as follows [38]

$$Rank\ (i) = \frac{\text{The item position in L}}{\text{The number of items initially unknown to the user}} \tag{22}$$

3.15. Ranking Score (User)

Ranking Score $(User)$, the ranking score of a user is defined as follows [39]:

$$Ranking\ Score\ (User) = \sum_{j \in I_{User}} \frac{Rank\ (j)}{|I_{User}|} \tag{23}$$

3.16. Sparseness (Dataset)

Given the total number of users, the number of data items, and number interactions between the users and data items, then the sparseness of a given dataset is computed as follows [27]:

$$Sparseness\ (dataset) = 1 - \frac{\text{Number interactions between the items and user}}{(\text{Number of items} \times \text{Number of users})} \tag{24}$$

3.17. Root Mean Square Error (RMSE)

For n, the number of ratings present in the test set, the quality of the predicted rating is obtained using RMSE [50]. This metric compares the prediction ratings with probe test set and is defined as follows:

$$RMSE = \sqrt{\frac{\sum_{j=0}^{n}(PredictionRate\ (u,\ i) - Rate(u,\ i))}{n}} \tag{25}$$

4. Proposed Recommender Algorithms

This research focuses on the design of two different intelligent optimization methods using Artificial Intelligence and Machine Learning for real-life applications that are used to improve the process of generation of recommenders. In the first method, the modified cluster based intelligent CF is applied with the sequential clustering that operates on the values of the dataset, user's neighborhood set, and the size of the recommendation list. This strategy splits the given dataset into different subsets or clusters, and the recommendation list is extracted from each group for constructing the better recommendation list. In the second method, the specific features-based customized recommender that works in the training and recommendation steps by applying the split and conquer strategy on the problem datasets, which are clustered into a minimum number of clusters and the better recommendation list, is created among all the clusters. This strategy automatically tunes the tuning parameter λ that serves the role of supervised learning in generating the better recommendation list for the large datasets.

The proposed recommender algorithms can also be extended to support the recommendation based on the multimedia information for online applications by updating the user profile, dataset resources, and domain knowledge base components of the proposed recommender systems [62]. These recommender algorithms are also applied in generating recommendations for hybrid online and big data in social and complex network applications [63–65].

The proposed algorithms are discussed in the following subsections.

4.1. Novelty in the Proposed Methods

The proposed recommender algorithms are designed using the following new strategies:

- Applying the split and conquer strategy on a large scale datasets into different clusters and generating better recommenders from each cluster.
- Updating the similar knowledge needs of other users in all clusters in a database and storing the better recommendation lists in a database for all the clusters.

- Applying the Machine Learning in the identification of knowledge requirements for every partition by extracting the better previously-stored knowledge information from the database to reduce the computational complexity.

The proposed strategy works well for large scale datasets compared to the feature-based recommender developed in solving ERP System and E-Agribusiness datasets, which require the computation of some configuration functions initially and its evaluation process requires some offline evaluation of parametric optimization [48]. Compared to this strategy, the proposed strategy obtains the better recommendation list in each of the clusters and it is updated in the database for future recommendation purposes. The favorite items are combined in the recommendation list based on the user profile of a target user.

The proposed methods are compared with the CF developed for both user-based and object-based cases in bipartite networks where the filtering is based on the degree of nodes in the bipartite network [49]. However, in the proposed CF, the split and conquer strategy predicts the overall ratings for all unrated items and recommends the best list for each cluster, and then the better recommendation from the ratings of the entire list is chosen artificially from the database. The proposed methods work well in solving the *MovieLens* dataset compared to the other methods.

The group recommendation model is proposed based on factors such as sparsity, dynamics, and timeliness [50]. However, the proposed collaborative recommender is designed to update the similar knowledge needs of other users in all clusters of the same group, and the information is updated in a database for providing the overall better recommendation. Even when more additional features are considered, this intelligent optimization strategy further minimizes the RMSE for large scale datasets to provide a better recommendation.

For the datasets that require a higher number of categories, latent class methods are computationally slow and provide an infeasible solution using *k*-modes clustering developed in [51]. Hence for large datasets that involve more categorical variables to improve the performance of cluster analysis, the *k*-modes clustering with specific features-based personalized recommender algorithm is required. Then *k*-modes method is applied to construct the different clusters by preprocessing the updated training dataset. The proposed *k*-modes strategy is a frequency-based method which evaluates the mode of clusters using $Dissimilarity\ (Y,\ X)$ & $Dissimilarity\ (Q,\ X)$ measurements. In the proposed *k*-modes strategy, for each object from the *classifier*: (*training-dataset*, *attributes-set*), the object is assigned to the cluster whose mode is the nearest to it according to $Dissimilarity\ (Y,\ X)$ & $Dissimilarity\ (Q,\ X)$ measurements. This strategy is used to evaluate the effective data partition for each cluster in obtaining a better recommendation. The training and recommendation steps are applied with some preprocessing and automatically tune the parameter λ to serve the role of supervised learning.

4.2. Modified Cluster Based Intelligent Collaborative Filtering Algorithm (Method 1)

The key idea of CF is that the items which are liked by many users can be liked by any other user. The proposed modified cluster based intelligent CF is operating on the dataset, neighborhood set of the user and |L| [25–28]. The method starts with the partition of the knowledge needs of the required information and applies the sequential clustering to identify the required knowledge needs [33–42]. It identifies the necessary current knowledge needs for each of the partitions by extracting the better previously-stored similar knowledge needs from the database artificially. This strategy splits the given data set into different subsets or clusters and the recommendation list is extracted from each cluster, and the combined recommendation list is generated and stored in a database. Then the better recommendation list is chosen at the end by extracting from the previously-stored recommendation lists from the database. The flowchart for proposed method 1 is depicted in Figure 1, and its algorithm is shown in Algorithm 1.

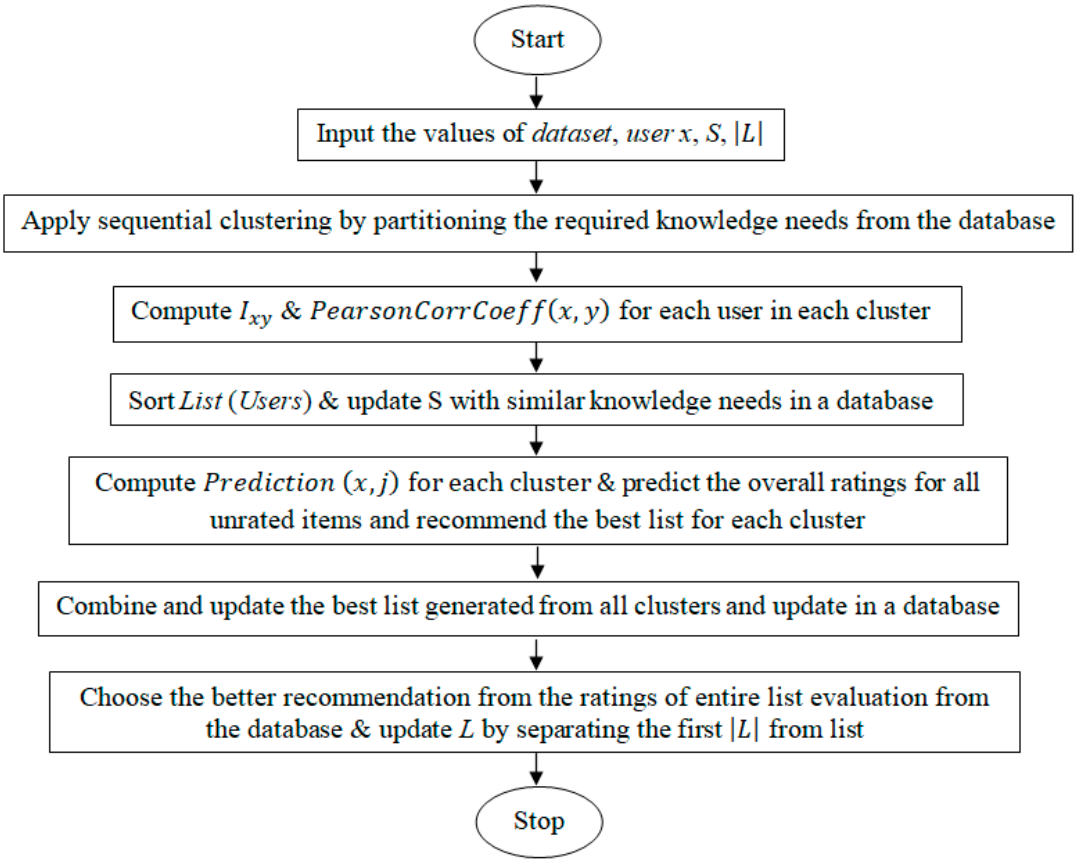

Figure 1. The Flowchart of the Proposed Method 1.

Algorithm 1: Modified Cluster based Intelligent CF

Inputs: *dataset*, *user x*, *S*, |*L*|
1: Define the *Ratings-Matrix* from the items and users of the dataset.
2: Partition the knowledge needs of the required information on the datasets
3: Apply sequential clustering to identify the current knowledge needs for each of the partition and extract the better stored similar knowledge needs from the database
4: For each user *y* in the *dataset* and $y \neq x$ in each cluster:
5: Compute I_{xy} by separating the items which are rated by the two users *x* and *y* in the *Ratings-Matrix*
6: Find the value of $PearsonCorrCoeff(x,\ y)$ using Equation (7)
7: End For
8: Reversely sort the list of users, *List* (*Users*) according to $PearsonCorrCoeff(x,\ y)$
9: Update the set *S* by separating |*S*| from *List* (*Users*)
10: Update the similar knowledge needs of other users in all clusters and update in a database
11: Find I_x the number of items interacted with user *x*
12: For each item $j \notin I_x$ in each cluster:
13: Compute $Prediction\ (x,\ j)$ using the neighborhood set *S* using Equation (8)
14: End For
15: Sort all items using $Prediction\ (x,\ j)$ in each cluster
16: Predict the overall ratings for all unrated items and recommend the best list for each cluster
17: Combine and update the best list generated from all clusters and update in a database
18: Choose the better recommendation from the ratings of entire list evaluation from the database
19: Update *L* by separating the first |*L*| from list
20: Return the list *L*

4.3. The Specific Features Based Personalized Recommender Algorithm (Method 2)

The new profile-based customized recommender algorithm works in the training and recommendation steps with the split and conquer strategy in which the datasets are clustered into a minimum number of clusters. The algorithmic design strategy is depicted in Figure 2. The training and recommendation steps are applied to each cluster. The recommendation list is generated from each cluster independently and updated in the database. Then the better recommendation list is generated by combining the generated recommendation lists from the database.

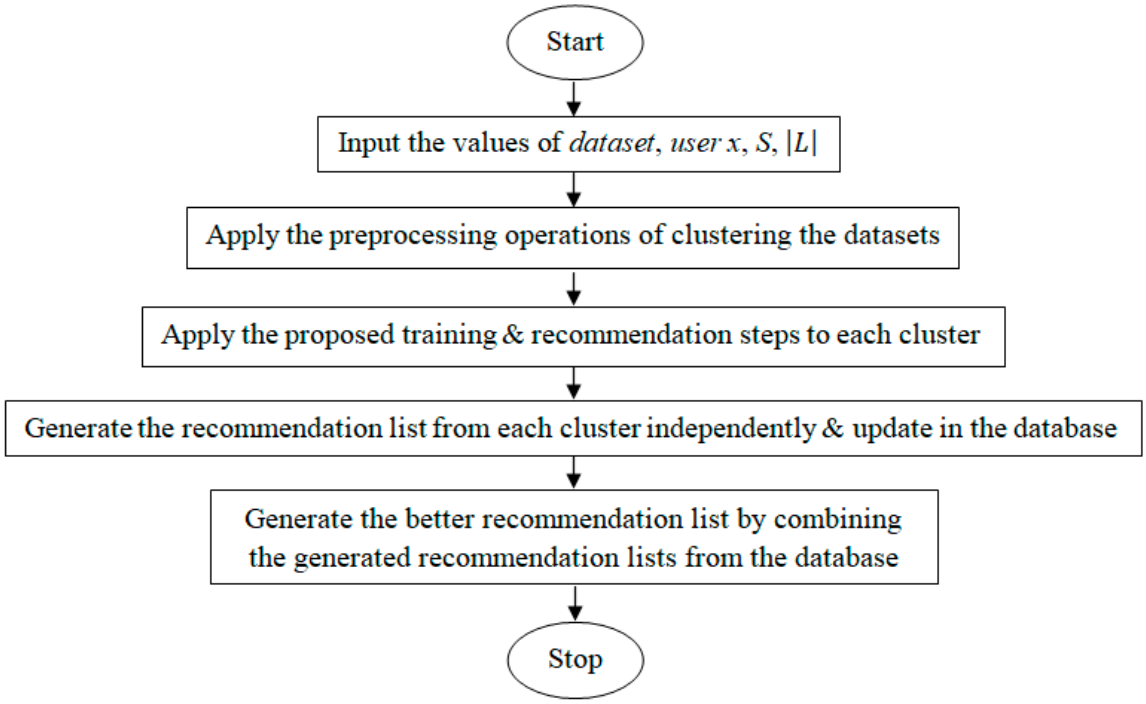

Figure 2. The Flowchart of the Split and Conquer Strategy of the Proposed Method 2.

The proposed *Profile-based Customized Recommender* is modeled as a bi-partition graph G of a user-item with three sets: (I, U, E) where the user set is $\{u_1, u_2, u_3 \ldots u_N\}$, the item set is $\{i_1, i_2, i_3 \ldots i_N\}$ and the edge set is $\{e_1, e_2, e_3 \ldots e_k\}$.

This algorithm operates based on user profile information for each cluster. It applies a tuning parameter λ, which lies between 0 and 1. The two important stages in this algorithm are training and recommendation. The stage 1 operations are dataset preprocessing, defining user features, deleting information for invalid users and empty data, performing basic feature-based operations, constructing interactions through graphs, preprocessed data features into clustering using a *k*-modes clustering algorithm, mapping the clusters to λ values, defining the pair (λ, cluster) values, and transforming attributes into categories. The required parameters are n, the data size; k, the cluster size; and t, the number of iterations. The proposed specific features based recommender is depicted in Algorithm 2.

Algorithm 2: Preprocessing—Specific Features based Personalized Recommender

Inputs: *dataset*, *user x*, *S*, |*L*|
1: Partition the knowledge needs of the required information
2: Apply sequential clustering to identify the current knowledge needs and extract the better stored similar knowledge needs from the database
3: Construct the graph *G* using the data set of training that is already known
4: Perform the following operations on each cluster:
5: Assign every user u to the vertex in U
6: Assign every item j to the vertex in I
7: If there is an interaction of a user u with any item j then:
8: Insert the edge e_k into G
9: Make vertex u adjacent to vertex j
10: Perform the following for the target user $\underline{u}$ in each cluster:
11: Assign the value of the resource, $r(u,\ j)$ to every item j in G
12: Set $r(u,\ j) = 1$ if there is an edge between u and j; otherwise set $r(u,\ j) = 0$
13: Apply the propagation process to redistribute the resource values
14: Calculate $r'(v,\ j)$ for every user $v \in U$
15: Redistribute the values of $r'(v,\ j)$
16: Update the new resource value $r''(v,\ j)$ for each item j in G

The flowchart of the training step of the proposed method 2 is depicted in Figure 3. The implementation of a training phase that trains a classifier is shown in Algorithm 3. The best tuning value of λ is extracted based on the target user's features and is returned from Algorithm 3. This algorithm starts with applying initial preprocessing operations on the dataset and it transforms the attribute values into categorical values. For the datasets that require a higher number of categories, latent class methods are computationally slow and provide an infeasible solution [51]. Hence, for large datasets that involve more categorical variables to improve the performance of cluster analysis, the *k*-modes clustering with specific features-based personalized recommender algorithm is required. Then *k*-modes method is applied to construct the different clusters by preprocessing the updated training dataset. The *k*-modes strategy, a frequency-based method that evaluates the mode of clusters using $Dissimilarity\ (Y,\ X)$ & $Dissimilarity\ (Q,\ X)$ measurements, is shown in Algorithm 4. The asymptotic complexity of this strategy is $\Theta(nki)$, a linear complexity. Then the k clusters are evaluated for each value of k ranging from *k-minimum* to *k-maximum*. The effective data partition is evaluated for each cluster using the *k*-modes method [20]. The best value of k, *best* (k), is obtained for the best-clusters, which optimizes the entropy-based clustering criterion function, $Optimize\left(C^k\right)$. Then, this algorithm proceeds by determining the value of λ-*best*, the best λ, for each cluster using Algorithm 5. A training pair for the classifier, (*user-features*, λ-*best* (j)), is constructed, which collects the features of each user in the specific cluster. Finally, the classifier is trained for the *training-set*. The user profile corresponding to the λ-*best* value is indicated by the classifier using the given user attributes. This training algorithm outputs the trained classifier and the bipartite graph G, which represents the interactions between items and users in the *training-dataset*. The specific features based recommendation is depicted in Algorithm 6.

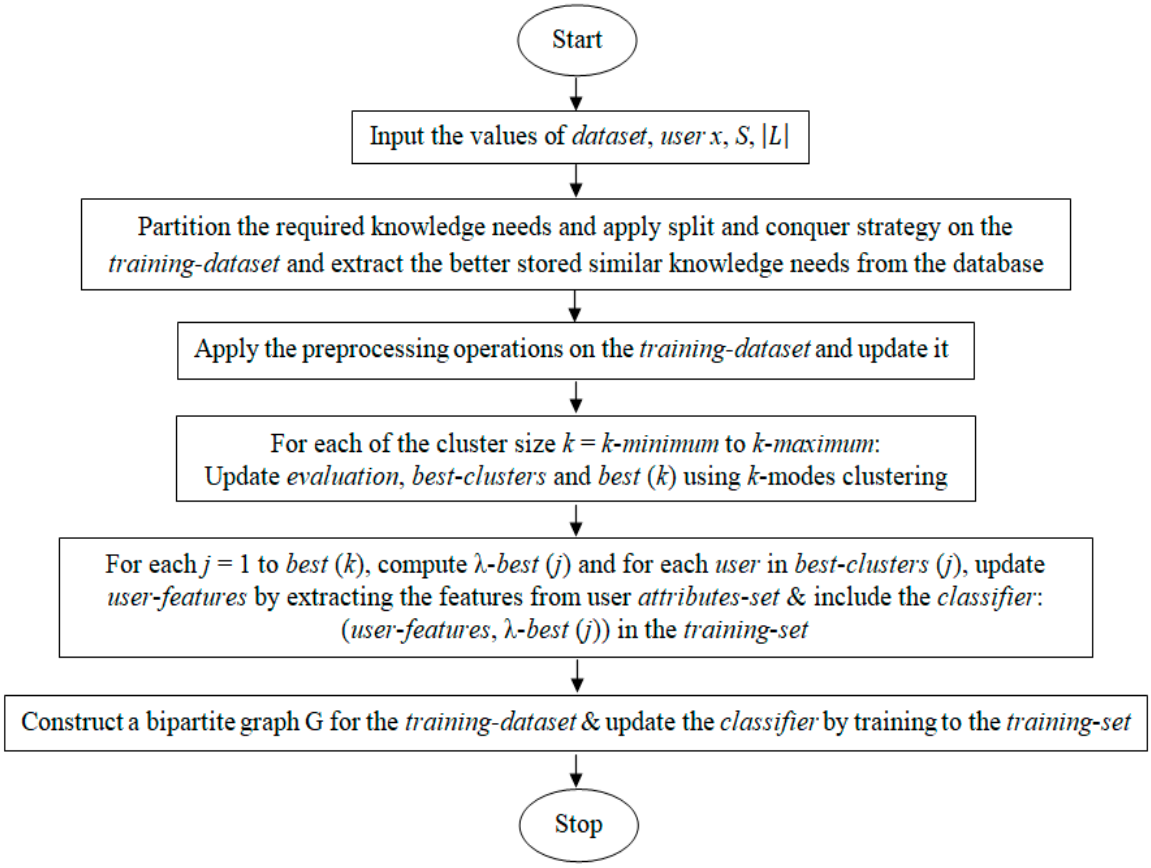

Figure 3. The Flowchart of the Training Step of the Proposed Method 2.

Algorithm 3: Training Steps

Inputs: *training-dataset*, *attributes-set*, *k-minimum*, *k-maximum*
1: Partition the required knowledge needs and apply split and conquer strategy on the *training-dataset* and extract the better stored similar knowledge needs from the database
2: Apply the preprocessing operations on the *training-dataset* and update it
3: Initialize *evaluation* to zero
4: For the cluster size *k* = *k-minimum* to *k-maximum*:
5: Construct *clusters* (*k*), that is, the *k* - clusters by applying Algorithm 4.
6: Evaluate *clusters* (*k*) using entropy-based clustering criterion (Eqn. 19) and update *evaluation* (*k*)
7: Compare *evaluation (k)* with *evaluation* and update *evaluation*, *best-clusters* and *best (k)*
8: End For
9: Construct a bipartite graph G for the *training-dataset*
10: Initialize *training-set* to null
11: For *j* = 1 to *best* (*k*):
12: Compute λ-*best* (*j*) by applying Algorithm 5
13: For each *user* in *best-clusters* (*j*):
14: Update *user-features* by extracting the features from user *attributes-set*
15: Include the *classifier*: (*user-features*, λ-*best* (*j*)) in the *training-set*
16: End For
17: End For
18: Update the *classifier* by training to the *training-set*
19: Return the *bipartite graph* and the *classifier*

Algorithm 4: *k*-modes clustering

Inputs: *training-dataset*, *attributes-set*, *k*
1: Choose *k* initial modes randomly, one for each cluster
2: For each object from the *classifier*: (*training-dataset*, *attributes-set*):
3: Assign the object to the cluster whose mode is the nearest to it according to *Dissimilarity* $(Y,\ X)$ & *Dissimilarity* $(Q,\ X)$ measurements
4: End For
5: For each mode from each cluster:
6: Recheck the objects dissimilarity against the values of current modes
7: Reallocate the object to the cluster
8: Update the clusters modes
9: End For
10: Update *clusters* (*k*) which define the set of *k* different clusters
11: Return *clusters* (*k*)

Algorithm 5: Determining λ-*best*

Inputs: G, *best-clusters* [j], Λ

This algorithm computes the λ-*best*, the best value of the profile of the user's λ for each cluster. It determines λ-*best*, by evaluating each recommendation list, which is generated with each value of λ by applying the metrics *recall*, *precision* and *ranking score*. The set Λ consists of the typical λ values that lie between 0.0 and 1.0.

1: Initialize *evaluation* (λ) to zero
2: For each value of λ in the typical set Λ:
3: *Recommendation-List* (λ) = *Mass Diffusion Heat Spreading* (*best-clusters* [j], λ, G)
4: *evaluation* = *Metric* (*Recommendation-List* (λ))
5: Update *evaluation* (λ) and λ-*best* (*j*) by comparing *evaluation* and *evaluation* (λ)
6: End For
7: Return λ-*best* (*j*)

The output of the trained classifier from Algorithm 3 determines the best λ for a target user in its recommender phase. This step requires the following inputs: a bipartite graph *G*, attributes set, target user, a trained classifier that is the result obtained from Algorithm 3, and $|L|$. The complete operations of the recommendation steps are described in Algorithm 3. The algorithm starts with extracting the target user's feature values and is assigned to user features. Then, λ_{User}, the proper λ value for the user, will be predicted by giving the *user-features* values into the trained *classifier*. The target user profile is reflected in the variable λ_{User} with *Mass Diffusion Heat Spreading* algorithm that selects the novelty-based popular items. The recommendation list is updated by applying the *Mass Diffusion Heat Spreading* algorithm with the tuning parameter λ_{User} and $|L|$. Finally, the lesser-known and favorite items are combined in the recommendation list based on the user profile of a target user. The flowchart of the recommendation step of the proposed method 2 is depicted in Figure 4. The specific features based recommendation process is described in Algorithm 6.

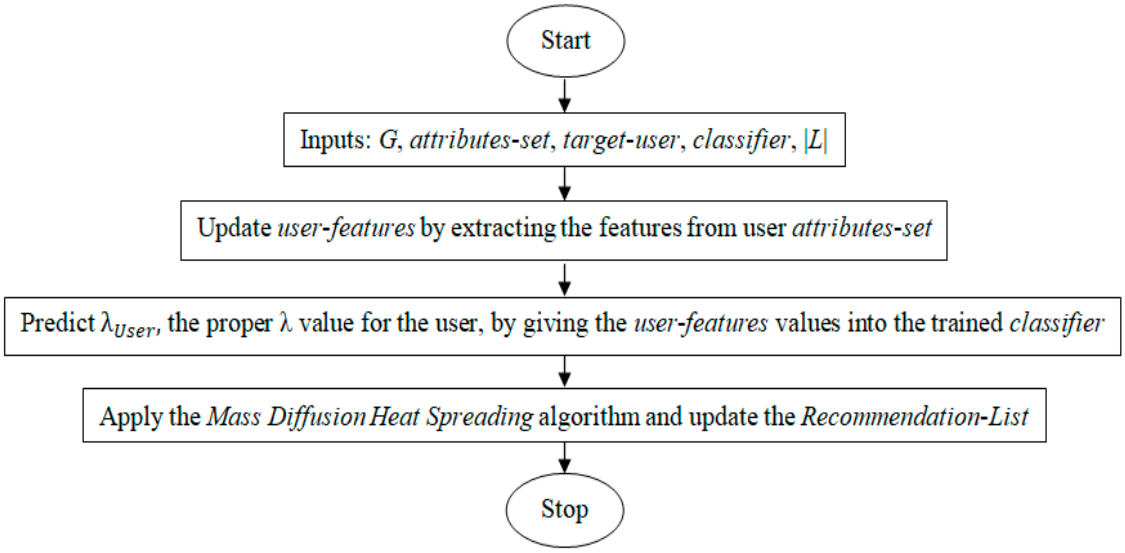

Figure 4. The Flowchart of the Recommendation Step of the Proposed Method 2.

Algorithm 6: Specific Features based Recommendation

Inputs: G, *attributes-set*, *target-user*, *classifier*, |*L*|
1: Update *user-features* by extracting the features from user *attributes-set*
2: Predict λ_{User}, the proper λ value for the user, by giving the *user-features* values into the trained *classifier*
3: Apply the *Mass Diffusion Heat Spreading* algorithm and update the *Recommendation-List*
4: Return *Recommendation-List*

In general, the worst-case complexity of the proposed recommenders is O (*ni*) for *n* users with the number of data items, *i*. The complexity of model construction of proposed cluster based recommenders with the partition of *c* clusters is O (*cni*), a linear complexity for one rating prediction with the space requirement of O (*ci* + *n*). The asymptotic complexity of the proposed *k*-modes clustering strategy is Θ(*nki*), a linear complexity since the *k* clusters are evaluated for each value of *k* ranging from k-minimum to k-maximum. However, in practical applications, it is expected to have O (*n* + *i*) complexity since for each user only a finite number of items are considered. Since one loop is used on *n* users to compute the similarity and one on the *i* items to compute the prediction.

The implementation cost in realistic recommender systems depends on the number of user attributes, size of the recommendation list, the tuning parameter, and the number of graph edges produced in the bipartite graph. There are some solutions for addressing the implementation cost in realistic recommender systems: discarding the users with minimal required popular items and discarding very popular items, since the items are partitioned into different datasets and clustering the data is applied.

5. Simulations & Results

This section focuses on the simulation of the proposed algorithms on the sample datasets with its outcomes and analysis. The algorithms are implemented in the Java language. It has been experimentally found that the computation time depends on the structure of the computation, number of user attributes, size of the graph edges, and the dataset.

5.1. Datasets

The proposed algorithms are implemented and executed on the following data sets [19]: The *MovieLens* dataset, which consists of the users, items, and interactions values as 910, 1672,

95,579 respectively, with user's features age, location, and gender. *Last.FM* dataset consists of the users, items, and interaction values of 2846, 4995, 14,583 respectively, with users, features country, gender, and age. The dataset *Book-Crossing* consists of the users, items, and interactions values of 3421, 26811, 35,572 respectively with users, features age and location.

5.2. Metrics for Evaluation

The results are evaluated using metrics such as *Recall* (*List*, *User*), *Precision* (*List*, *User*) and *Ranking Score* (*User*) that are applied in validating the values of L that are generated by the recommender system. When *Recall* (*List*, *User*) becomes higher, then the system recommends the testing set items. When *Precision* (*List*, *User*) becomes higher, it indicates that more items in the recommendation list are corresponding to the testing set user items. When the *Ranking Score* (*User*) becomes lower, it indicates that the item is closer to the first position. The sparseness of the data sets is computed using the expression *Sparseness* $(dataset)$. The sparseness values for the data set *MovieLens*, *Last.FM* and *Book-Crossing* are 0.9371, 0.9989, and 0.9996, respectively.

5.3. Comparative Results & Analysis

The proposed methods are compared with the MDHS algorithm [36,46], the nearest neighborhood CF [11,23], UPOD [6], and Dynamic Group Recommender (DGR) algorithms [6,49–51]. The following simulation parameters are applied during the execution of the proposed methods: $\lambda = 0.5$, the size of the recommendation list $|L| = 30$, size of the neighborhood is 30, *k-minimum* to 100, and *k-maximum* to 200. The simulation outcomes are tabulated in Tables 1–6. The results are compared using the statistical *t*-test to find the significance of the proposed methods over the existing methods [6]. The mean μ and standard deviation σ are calculated for the datasets after applying the $Optimize\left(C^k\right)$ (Equation (19)) 100 times. Tables observe that the proposed methods are outperforming the existing techniques based on the metrics applied to evaluate the performance measurements. The data for user profiles can also be extracted from social networks. The average computational time (in seconds) for the proposed methods in the form of (proposed method 1, proposed method 2) for the *Movielens*, *Last.FM*, *Book-Crossing* datasets are (8.2, 4.9), (9.8, 4.89), (8.75, 4.97) respectively.

Table 1. Comparison of μ for the proposed methods with existing strategies—*MovieLens*.

Strategies	*Ranking Score* (*User*)	*Recall* (*List*, *User*)	*Precision* (*List*, *User*)
CF [11,23]	0.497	0.014	0.003
MDHS [36,46]	0.080	0.299	0.086
UPOD [6]	0.074	0.308	0.091
Proposed Method 1	0.070	**0.312**	0.091
Proposed Method 2	**0.069**	**0.312**	**0.092**

Table 2. Comparison of σ for the proposed methods with existing strategies—*MovieLens*.

Strategies	*Ranking Score* (*User*)	*Recall* (*List*, *User*)	*Precision* (*List*, *User*)
CF [11,23]	0.002	0.002	0.001
MDHS [36,46]	0.001	0.008	0.001
UPOD [6]	0.001	0.007	0.002
Proposed Method 1	**0.001**	**0.008**	0.002
Proposed Method 2	**0.001**	**0.008**	**0.003**

Table 3. Comparison of μ for the proposed methods with existing strategies—*Last.FM*.

Strategies	*Ranking Score* (*User*)	*Recall* (*List*, *User*)	*Precision* (*List*, *User*)
CF [11,23]	0.444	0.018	0.001
MDHS [36,46]	0.252	0.119	0.006
UPOD [6]	0.251	0.174	0.009
Proposed Method 1	0.251	**0.175**	**0.010**
Proposed Method 2	**0.250**	**0.176**	**0.011**

Table 4. Comparison of σ for the proposed methods with existing strategies—*Last.FM*.

Strategies	*Ranking Score* (*User*)	*Recall* (*List*, *User*)	*Precision* (*List*, *User*)
CF [11,23]	0.004	0.004	0.000
MDHS [36,46]	0.007	0.006	0.000
UPOD [6]	0.006	0.014	0.001
Proposed Method 1	0.006	**0.015**	**0.001**
Proposed Method 2	**0.004**	**0.019**	**0.001**

Table 5. Comparison of μ for the proposed methods with existing strategies—*Book-Crossing*.

Strategies	*Ranking Score* (*User*)	*Recall* (*List*, *User*)	*Precision* (*List*, *User*)
CF [11,23]	0.406	0.000	0.000
MDHS [36,46]	0.395	0.002	0.000
UPOD [6]	0.394	0.003	0.000
Proposed Method 1	0.395	**0.004**	**0.000**
Proposed Method 2	0.398	**0.004**	**0.000**

Table 6. Comparison of σ for the proposed methods with existing strategies—*Book-Crossing*.

Strategies	*Ranking Score* (*User*)	*Recall* (*List*, *User*)	*Precision* (*List*, *User*)
CF [11,23]	0.007	0.000	0.000
MDHS [36,46]	0.007	0.001	0.000
UPOD [6]	0.008	0.001	0.000
Proposed Method 1	0.008	**0.001**	**0.000**
Proposed Method 2	0.007	**0.005**	**0.000**

The proposed methods are evaluated and compared with the other well-known strategies, such as CF, MDHS, and UPOD. The parameter λ has been set to 0.5 when comparing with MDHS and UPOD methods. The proposed methods are evaluated using the n group cross-validation in which the data set is divided into n groups of equal sizes. $Valid_{Users}$, that is, the users are those who are available both in testing and the training set, are considered for the evaluation.

The performance of the proposed methods with the existing methods is evaluated using the three metrics: *Recall* (*List*, *User*), *Precision* (*List*, *User*), and *Ranking Score* (*User*), and the recommendation list generated by the system is validated. The simulation outcomes are tested using the statistical t-test with a level of significance $\alpha = 0.05$ to check if there is a significant difference between the proposed methods, with the existing methods being statistically significant. The measures k_μ and k_σ are computed for each dataset, after applying the clustering with 10 executions. The inferences are analyzed from the experimental results, which are tabulated from tables Tables 1–6. The figures which are indicated in bold conclude the better performance of the proposed methods over the existing methods.

The comparison of the mean μ and standard deviation σ to the *Movielens* dataset is shown in Tables 1 and 2 respectively. For this dataset, the measures are calculated as $k_\mu = 198$ and $k_\sigma = 4.21$. For this dataset, it has been found that there is no significant difference obtained in terms of parameters μ and σ concerning the *Ranking Score* (*User*) and *Recall* (*List*, *User*) measures in the proposed methods.

However, the proposed method 2 behaves well in terms of *Precision* (*List*, *User*) measurements compared to other methods with the size of the recommendation list $|L| = 30$.

The accuracy of the proposed methods is also evaluated for the *MovieLens* dataset with the same parameters considered in [49]. To test the performance of the proposed recommenders, the *MovieLens* dataset is divided into 90% of the training set and 10% of probe data. The dataset can also be divided into (80%, 20%), (70%, 30%), and so on. The only known information is available in the training set and no prediction is made in the probe set of data. For the jth user u_j, the position of an uncollected object o_j is measured in the ordered queue. Then the position of o_j is obtained by dividing the particular location from the top by the total number of uncollected movies. Hence, a good recommender is expected to produce a small *Ranking Score* (*User*), which shows the better accuracy of the recommender. The performance comparison of the proposed methods with other methods for the *MovieLens* dataset over the three metrics is shown in Figure 5 [49]. The simulation is conducted with 10% of probe data, L = 50. The values corresponding to the proposed methods are better ones concerning all three metrics.

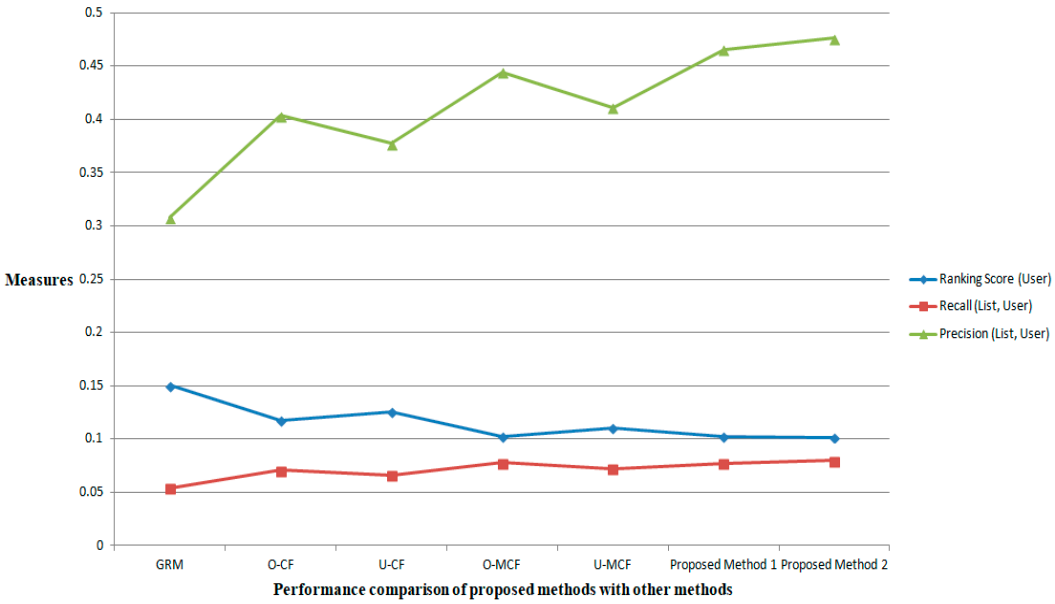

Figure 5. The Performance Comparison of the Proposed Methods with Other Methods [49]: *MovieLens* dataset.

The comparison of the mean μ and standard deviation σ to the *Last.FM* dataset is shown in Tables 3 and 4 respectively. For this dataset, the measures are calculated as $k_\mu = 195$ and $k_\sigma = 2.75$. For the Last.FM dataset, the proposed method 2 performs better than the existing methods in terms of all performance metrics considered. For this dataset, there is no significant difference obtained even when the size of the recommendation list |L| becomes 30, while keeping the size of the neighborhood as <30 in all executions.

For n, the number of ratings present in the test set, the quality of the predicted rating is obtained using RMSE [50]. This metric compares the prediction ratings with the probe test set. The accuracy of the proposed methods is also evaluated for the *Last.FM* dataset and the results are compared with the methods presented in [50]. The minimum RMSE is considered for some of the methods presented in the DGR. The RMSE comparison of the proposed methods with DGR presented in [50] is shown in Figure 6. In the Figure, the size of the groups is plotted on the X-axis and the RMSE is plotted on the Y-axis. The predictions generated by the proposed recommenders are better than the existing methods. The experimental results further conclude that the proposed recommenders consider the features of the individual preferences of the group members and the specific features. It has also been found that

the accuracy is slightly decreasing while the group size is increasing since the group recommendation depends on the diverse set of users and specific personal features.

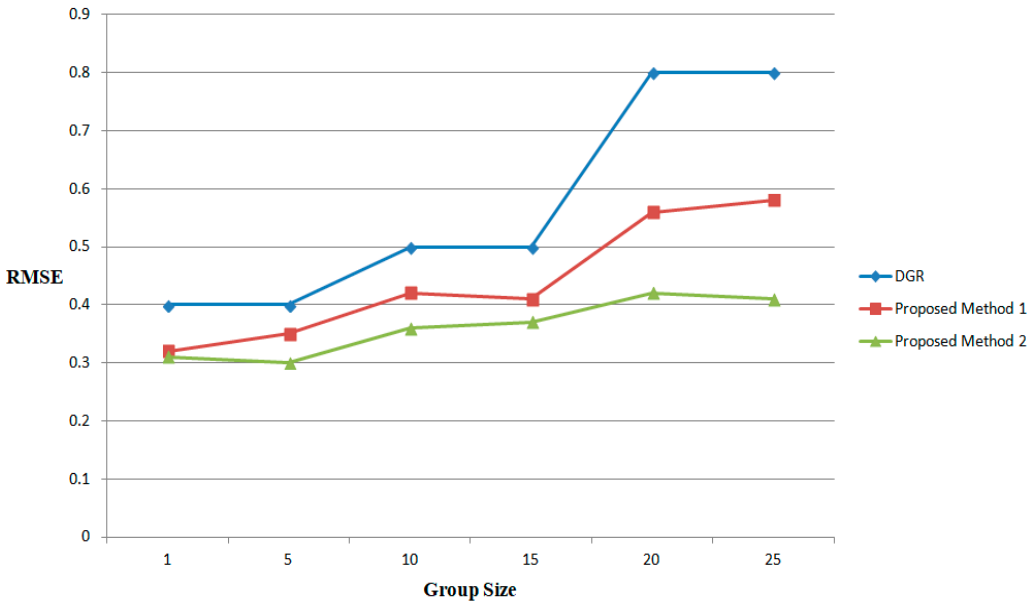

Figure 6. The RMSE Comparison of the Proposed Methods with DGR [50].

The performance of specific feature selection is also analyzed based on varying the values of neighborhood sizes $|S|$. The interesting result is obtained during the simulation when the experiments are conducted for small, medium, and large neighborhood sizes $|S|$ and are shown in Figure 7. The RMSE values of 41% and 37% are obtained in the proposed methods when $|S| < 20$. When the value of $|S|$ increases, RMSE also increases. For large values of $|S|$, there is a significant difference that becomes smaller in the proposed methods.

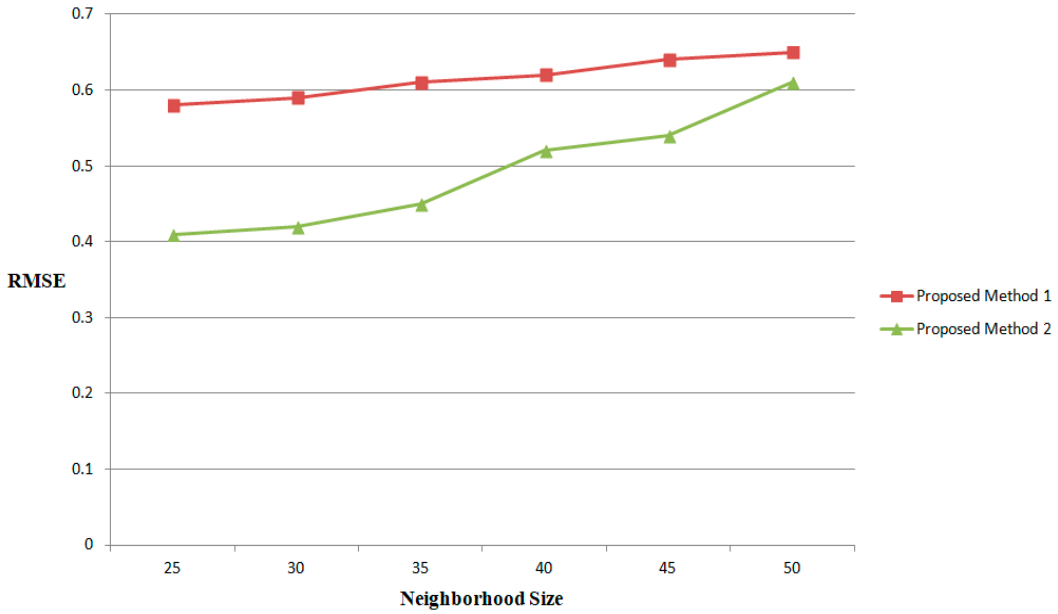

Figure 7. The RMSE Comparison of the Proposed Methods for Different Neighborhood Sizes.

The average (k_μ) & standard deviation (k_σ) of observed ratings for the proposed datasets are shown in Table 7.

Table 7. The average (k_μ) & standard deviation (k_σ) of the observed ratings.

Datasets	k_μ	k_σ
MovieLens	198	4.21
Last.FM	195	2.75
Book-Crossing	194	7.83

The comparison of the mean μ and standard deviation σ to the *Book-Crossing* dataset is shown in Tables 5 and 6 respectively. For this dataset, the measures are calculated as $k_\mu = 194$ and $k_\sigma = 7.83$. For this dataset, it has been found that there is a significant difference obtained in terms of parameters μ and σ concerning the *Precision* (*List*, *User*) and *Recall* (*List*, *User*) measures in the proposed methods. However, the proposed methods are competitive with the existing methods. The proposed method 2 performs well compared to proposed method 1. It has also been found that, for this dataset, there is no significant difference obtained even when the size of the recommendation list $|L|$ becomes 50 while keeping the size of the neighborhood as < 30 in all executions.

The proposed methods are simulated for the large scale data set *Book-Crossing* which consists of 3421 users and 26,811 items on varying the tuning parameter λ for the different values as 0.3, 0.4, 0.5, and 0.6. The significant results are obtained and are shown in Figure 8. It has been found that when $\lambda < 0.5$, the *Ranking Score* (*User*) increases while the *Precision* (*List*, *User*) and *Recall* (*List*, *User*) measures decrease. When $\lambda > 0.5$, the *Ranking Score* (*User*) also increases gradually and no such significant differences are obtained in the *Precision* (*List*, *User*) and *Recall* (*List*, *User*) measures. In this case, the experimental results show that the proposed method 2 can provide a better recommendation based on the defined metrics when $\lambda = 0.5$.

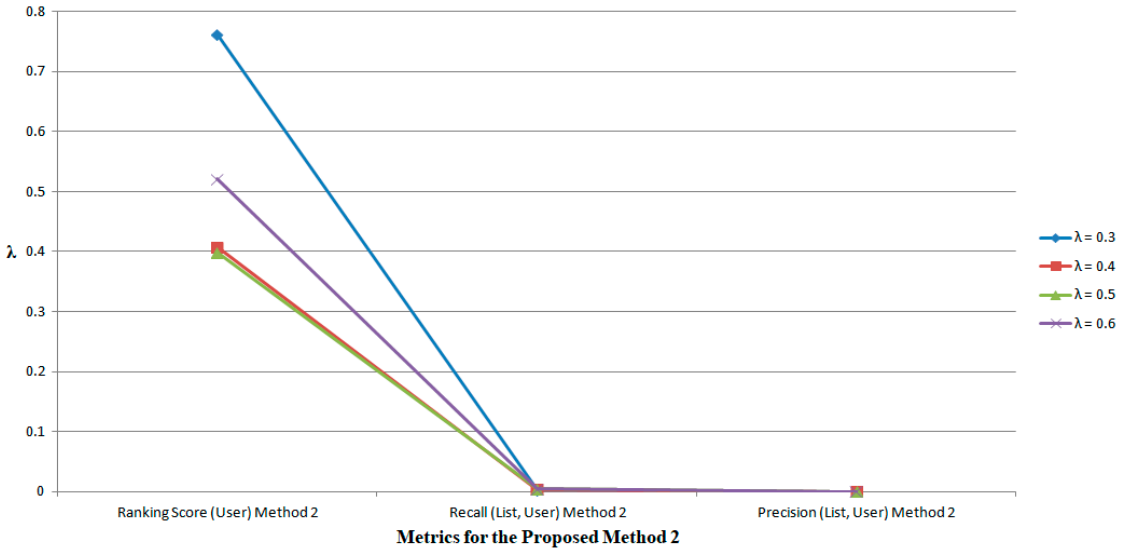

Figure 8. Performance of the Proposed Method 2 for Different Values of λ: *Book-Crossing*.

For the *Book-Crossing* dataset, the metric *Precision* (*List*, *User*) is evaluated for different sizes of the recommendation list, for example, for the values of $|L| = 10$, 20, 30, 40, 50, and 100. The corresponding metric is plotted as shown in Figure 9. When this metric becomes higher, the items that are in L corresponding to the items list that is corresponding to the users present in the testing set values. By keeping the tuning parameter as $\lambda = 0.5$, the proposed method 2 provides better performance while $|L| \leq 30$. When $|L| > 30$, it seems that the proposed method 1 provides better recommendation based on this metric.

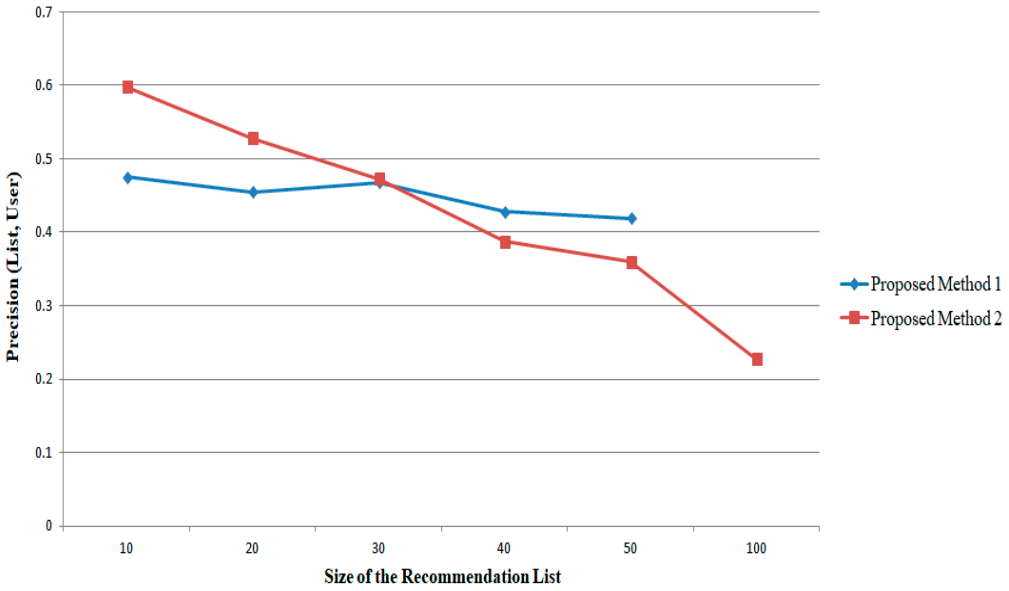

Figure 9. Size of the *Recommendation List* versus *Precision* (*List*, *User*) for *Book-Crossing*.

The experiments are also conducted for the *Book-Crossing* dataset when varying the user sizes, for example, for the values of 500, 1000, 1500, 2000, 2500, and 3000, and the corresponding *Precision* (*List*, *User*) & *Recall* (*List*, *User*) metrics are plotted as shown in Figure 10. The experimental results analyze that for small values of user sizes (≤500), higher values of these metrics are obtained. When the user sizes exceed 500, these metrics decrease gradually and the proposed method 2 performs better than the proposed method 1. The significant differences are obtained in these methods because the structure of computation of the methods depends on the number of user attributes, size of the recommendation list, the tuning parameter, and the number of graph edges produced in the bipartite graph. The maximum values of the (*Precision*, *Recall*) metrics obtained for the *Book-Crossing* dataset in the proposed methods are (0.0004, 0.0042) and (0.0004, 0.0046) respectively.

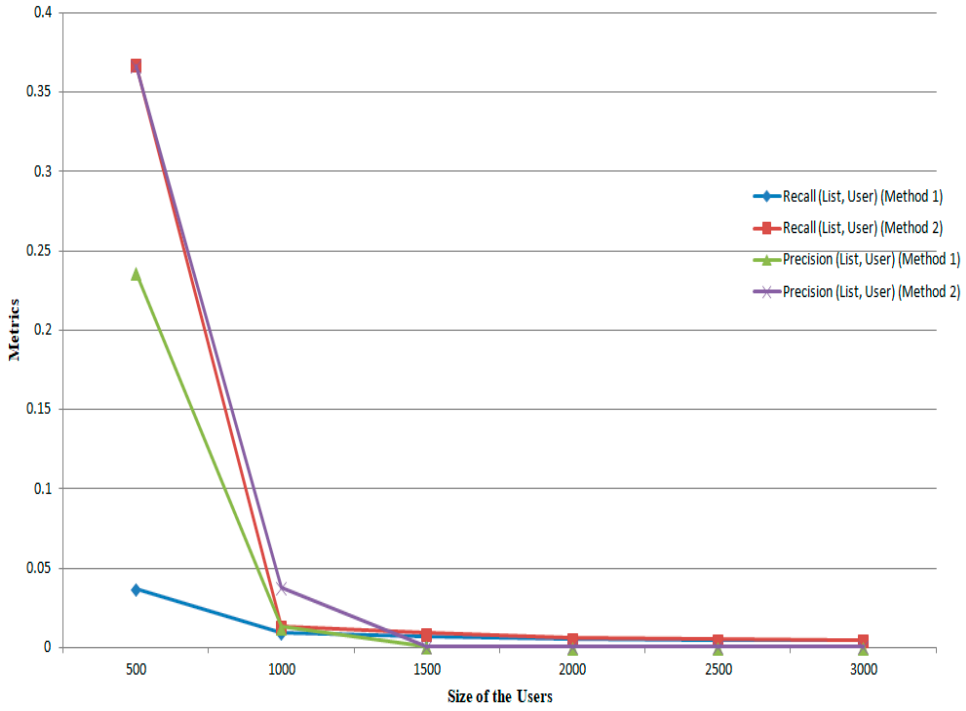

Figure 10. Sizes of the Users versus *Precision* & *Recall* Metrics for *Book-Crossing*.

5.4. Discussion of Important Results

The proposed methods are providing better performance compared to the existing methods. The discussion of the important results of the proposed methods is analyzed as follows:

The statistical measurements mean μ and standard deviation σ are calculated for the considered datasets after applying the $Optimize\left(C^k\right)$ (Equation (19)) 100 times. The expected entropy $E\left(C^k\right)$ is minimized after applying $Optimize\left(C^k\right)$ more than 90 times. The proposed methods are evaluated using the n group cross-validation in which the data set is divided into n groups of equal sizes. $Valid_{Users}$, that is, the users are those who are available both in testing and the training set, are considered for the evaluation. The simulation outcomes are tested using the statistical t-test with a level of significance $\alpha = 0.05$ to check if there is a significant difference between the proposed methods with the existing methods that is statistically significant. The measures k_μ and k_σ are computed for each dataset, after applying the clustering with the number of executions 10. It has been experimentally found that the accuracy of the proposed methods is slightly decreasing while the group size is increasing since the group recommendation depends on the diverse set of users and specific personal features.

The performance of specific feature selection is analyzed based on varying the values of neighborhood sizes $|S|$. The interesting result is obtained during the simulation when the experiments are conducted for small, medium, and large neighborhood sizes $|S|$. The RMSE values of 41% and 37% are obtained in the proposed methods when $|S| < 20$. When the value of $|S|$ increases, RMSE also increases. For large values of $|S|$, there is a significant difference that becomes smaller in the proposed methods. The simulation of proposed methods on the large scale data set, *Book-Crossing*, when varying the tuning parameter λ for the different values such as 0.3, 0.4, 0.5, and 0.6, produces significant results. It has been experimentally found that when $\lambda < 0.5$, the *Ranking Score* (*User*) increases while the *Precision* (*List*, *User*) and *Recall* (*List*, *User*) measures decrease. When $\lambda > 0.5$, the *Ranking Score* (*User*) also increases gradually and no such significant differences are obtained in the *Precision* (*List*, *User*) and *Recall* (*List*, *User*) measures. In this case, the experimental results show that the proposed method 2 can provide a better recommendation based on the defined metrics when $\lambda = 0.5$.

For the *Book-Crossing* dataset, the simulation has been conducted for different sizes of $|L| = 10$, 20, 30, 40, 50, and 100. When this measurement becomes higher, the items that are in L correspond to the items list that is corresponding to the users present in the testing set values. By keeping the tuning parameter as $\lambda = 0.5$, the proposed method 2 provides better performance while $|L| \leq 30$. When $|L| > 30$, it seems that the proposed method 1 provides better recommendation based on this metric. The simulation has been performed on the *Book-Crossing* dataset when varying the user sizes, in multiples of 500, up to the maximum of 3000. The experimental results analyze that for small values of user sizes (≤ 500), higher values of the metrics are obtained. When the user sizes exceed 500, the metrics decrease gradually and the proposed method 2 performs better than the proposed method 1. The significant differences are obtained in these methods because the structure of computation of the methods depends on the number of user attributes, size of the recommendation list, the tuning parameter, and the number of graph edges produced in the bipartite graph. The maximum values of the (*Precision*, *Recall*) metrics obtained for the *Book-Crossing* dataset in the proposed methods are (0.0004, 0.0042) and (0.0004, 0.0046) respectively. The proposed method 2 works well when $\lambda = 0.5$ with the size of the recommendation list $|L| = 30$ and the size of the neighborhood is 30, and it automatically tunes the tuning parameter λ.

6. Conclusions & Future Work

The proposed modified cluster based intelligent CF and the profile based customized recommender method are proposed and analyzed in this research. The proposed method 2 works well when $\lambda = 0.5$ with the size of the recommendation list $|L| = 30$, and the size of the neighborhood is 30, and it automatically tunes the tuning parameter λ. The proposed methods combine the novelty and popularity features based on the user's profile and generate the recommendation list. The experimental

results conclude that the tuning parameter λ serves the role of supervised learning and obtains a better recommendation list for the considered sparse datasets. For the *MovieLens* dataset, it has been found that there is no significant difference obtained in terms of parameters μ and σ concerning the *Ranking Score* (*User*) and *Recall* (*List*, *User*) measures in the proposed methods. For this dataset, the simulation is conducted with 10% of probe data, $L = 50$. The values corresponding to the proposed methods are better ones concerning all three metrics. For the *Last.FM* dataset, the proposed method 2 performs better than the existing methods in terms of all performance metrics considered. The predictions generated by the proposed recommenders are better than the existing methods concerning the RMSE metric. The experimental results further conclude that the proposed recommenders consider the features of the individual preferences of the group members and the specific features. It has also been found that the accuracy is slightly decreasing while the group size is increasing since the group recommendation depends on the diverse set of users and specific personal features. The performance of specific feature selection is also analyzed based on varying the values of neighborhood sizes $|S|$. The interesting result is obtained during the simulation when the experiments are conducted for small, medium, and large neighborhood sizes $|S|$. The RMSE values of 41% and 37% are obtained in the proposed methods when $|S| < 20$. When the value of $|S|$ increases, RMSE also increases. For large values of $|S|$, there is a significant difference that becomes smaller in the proposed methods. For the *Book-Crossing* dataset, it has been found that there is a significant difference obtained in terms of parameters μ and σ concerning the *Precision* (*List*, *User*) and *Recall* (*List*, *User*) measures in the proposed methods. However, the proposed methods are competitive with the existing methods. The proposed method 2 performs well compared to proposed method 1. For the considered datasets, there is no significant difference obtained even when the size of the recommendation list $|L|$ becomes 30, while keeping the size of the neighborhood as < 30 in all executions; in addition, also in the proposed method 2, better recommendations are provided based on the defined metrics when λ is equal to 0.5. By keeping the tuning parameter as $\lambda = 0.5$, the proposed method 2 provides better performance while $|L| \leq 30$. When $|L| > 30$, it seems that the proposed method 1 provides better recommendation based on this metric. The experimental results analyze that for small values of user sizes (≤ 500), higher values of these metrics are obtained. When the user sizes exceed 500, these metrics decrease gradually and the proposed method 2 performs better than proposed method 1. The significant differences are obtained in these methods because the structure of computation of the methods depends on the number of user attributes, size of the recommendation list, the tuning parameter, and the number of graph edges produced in the bipartite graph. The maximum values of the (*Precision*, *Recall*) metrics obtained for the *Book-Crossing* dataset in the proposed methods are (0.0004, 0.0042) and (0.0004, 0.0046) respectively. The average computational time of the proposed methods takes <10 seconds.

The following are the future research directions:

- Enhancing the resource values computation and increasing the item rating values,
- Designing the strategy to filter the better-rated items and ignoring the poorly rated items,
- Applying the local search strategies to improve resource propagation,
- Generating a recommender based on multiple attributes selection.
- Features based trust or reputation in recommendations can be integrated into the proposed algorithms interestingly with some evolutionary operators for e-learning and other real-world applications [59–61,71–73].

Author Contributions: Methodology, software, resources, writing—original draft and formal analysis: S.B.; conceptualization, formal analysis, validation, writing—review and editing, proofreading: R.M.; revise the paper: B.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: The authors would like to acknowledge the support rendered by the Management of SASTRA Deemed University for providing financial support.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bobadilla, J.; Ortega, F.; Hernando, A.; Gutiérrez, A. Recommender systems survey. *Knowl. Based Syst.* **2013**, *46*, 109–132. [CrossRef]
2. Zhang, F.; Gong, T.; Lee, V.E.; Zhao, G.; Rong, C.; Qu, G. Fast algorithms to evaluate collaborative filtering recommender systems. *Knowl. Based Syst.* **2016**, *96*, 96–103. [CrossRef]
3. Adomavicius, G.; Tuzhilin, A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng.* **2005**, *17*, 734–749. [CrossRef]
4. Göksedef, M.; Gündüz-Öğüdücü, Ş. Combination of Web page recommender systems. *Expert Syst. Appl.* **2010**, *37*, 2911–2922. [CrossRef]
5. Mobasher, B. Recommender Systems. In *Kunstliche Intelligenz, Special Issue on Web Mining*; BottcherIT Verlag: Bremen, Germany, 2007; pp. 41–43.
6. Bertani, R.M.; Bianchi, R.A.; Costa, A.H.R. Combining novelty and popularity on personalised recommendations via user profile learning. *Expert Syst. Appl.* **2020**, *146*, 113–149. [CrossRef]
7. Shu, J.; Shen, X.; Liu, H.; Yi, B.; Zhang, Z. A content-based recommendation algorithm for learning resources. *Multimedia Syst.* **2018**, *24*, 163–173. [CrossRef]
8. Syed, M.A.; Rakesh, K.L.; Gopal, K.N.; Rabindra, K.B. Movie recommendation system using genome tags and content-based filtering. In *Advances in Data and Information Sciences*; Springer: Singapore, 2018; pp. 85–94. [CrossRef]
9. Klašnja-Milićević, A.; Vesin, B.; Ivanović, M.; Budimac, Z. E-Learning personalization based on hybrid recommendation strategy and learning style identification. *Comput. Educ.* **2011**, *56*, 885–899. [CrossRef]
10. Cui, G.; Luo, J.; Wang, X. Personalized travel route recommendation using collaborative filtering based on GPS trajectories. *Int. J. Digit. Earth* **2018**, *11*, 284–307. [CrossRef]
11. Beel, J.; Gipp, B.; Langer, S.; Breitinger, C. Research-paper recommender systems: A literature survey. *Int. J. Digit. Libr.* **2016**, *17*, 305–338. [CrossRef]
12. Betru, B.T.; Onana, C.A. Deep learning methods on recommender system: A survey of state-of-the-art. *Int. J. Comput. Appl.* **2017**, *162*, 975–8887. [CrossRef]
13. Ai, Q.; Azizi, V.; Chen, X.; Zhang, Y. Learning heterogeneous knowledge base embeddings for explainable recommendation. *Algorithms* **2018**, *11*, 137. [CrossRef]
14. Celma, O. *Music Recommendation and Discovery*; Springer: Berlin, Germany, 2010; ISBN 978-3-642-13287-2.
15. Chen, B.; Zeng, A.; Chen, L. The effect of heterogeneous dynamics of online users on information filtering. *Phys. Lett. A* **2015**, *379*, 2839–2844. [CrossRef]
16. Chen, K.; Liu, L. Best K: Critical clustering structures in categorical datasets. *Knowl. Inf. Syst.* **2009**, *20*, 1–33. [CrossRef]
17. Deng, X.; Zhong, Y.; Lü, L.; Xiong, N.; Yeung, C. A general and effective diffusion-based recommendation scheme on coupled social networks. *Inf. Sci.* **2017**, *417*, 420–434. [CrossRef]
18. Fu, M.; Qu, H.; Moges, D.; Lu, L. Attention based collaborative filtering. *Neurocomputing* **2018**, *311*, 88–98. [CrossRef]
19. Harper, F.M.; Konstan, J.A. The MovieLens datasets: History and context. *ACM Trans. Interact. Intell. Syst.* **2015**, *5*, 19:1–19:19. [CrossRef]
20. Huang, Z. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Min. Knowl. Disc.* **1998**, *2*, 283–304. [CrossRef]
21. Javari, A.; Jalili, M. A probabilistic model to resolve diversity-accuracy challenge of recommendation systems. *Knowl. Inf. Syst.* **2014**, *44*, 609–627. [CrossRef]
22. Kaminskas, M.; Bridge, D. Diversity, serendipity, novelty, and coverage. *ACM Trans. Interact. Intell. Syst.* **2016**, *7*, 1–42. [CrossRef]
23. Katarya, R.; Verma, O.P. Recent developments in affective recommender systems. *Physica A* **2016**, *461*, 182–190. [CrossRef]
24. Kotkov, D.; Wang, S.; Veijalainen, J. A survey of serendipity in recommender systems. *Knowl. Based Syst.* **2016**, *111*, 180–192. [CrossRef]
25. Lacerda, A. Multi-Objective ranked bandits for recommender systems. *Neurocomputing* **2017**, *246*, 12–24. [CrossRef]
26. Liu, H.; Hu, Z.; Mian, A.; Tian, H.; Zhu, X. A new user similarity model to improve the accuracy of collaborative filtering. *Knowl. Based Syst.* **2014**, *56*, 156–166. [CrossRef]

27. Lu, J.; Shambour, Q.; Xu, Y.; Lin, Q.; Zhang, G. A web-based personalized business partner recommendation system using fuzzy semantic techniques. *Comput. Intell.* **2013**, *29*, 37–69. [CrossRef]
28. Lu, J.; Wu, D.; Mao, M.; Wang, W.; Zhang, G. Recommender system application developments: A survey. *Decis. Support Syst.* **2015**, *74*, 12–32. [CrossRef]
29. Ma, T.; Zhou, J.; Tang, M.; Tian, Y.; Al-Dhelaan, A.; Al-Rodhaan, M.; Lee, S. Social network and tag sources based augmenting collaborative recommender system. *IEICE Trans. Inf. Syst.* **2015**, *98*, 902–910. [CrossRef]
30. Ma, W.; Ren, C.; Wu, Y.; Wang, S.; Feng, X. Personalized recommendation via unbalance full-connectivity inference. *Physica A* **2017**, *483*, 273–279. [CrossRef]
31. Patra, B.K.; Launonen, R.; Ollikainen, V.; Nandi, S. A new similarity measure using Bhattacharyya coefficient for collaborative filtering in sparse data. *Knowl. Based Syst.* **2015**, *82*, 163–177. [CrossRef]
32. Pearson, K. Notes on the history of correlation. *Biometrika* **1920**, *13*, 25–45. [CrossRef]
33. Ricci, F.; Rokach, L.; Shapira, B.; Kantor, P.B. *Recommender Systems Handbook*, 1st ed.; Springer: New York, NY, USA, 2010. [CrossRef]
34. Shambour, Q.; Lu, J. An effective recommender system by unifying user and item trust information for B2B applications. *J. Comput. Syst. Sci.* **2015**, *81*, 1110–1126. [CrossRef]
35. Sánchez-Moreno, D.; Gil González, A.B.; Muñoz Vicente, M.D.; López Batista, V.F.; Moreno García, M.N. A collaborative filtering method for music recommendation using playing coefficients for artists and users. *Expert Syst. Appl.* **2016**, *66*, 1339–1351. [CrossRef]
36. Wang, X.; Liu, Y.; Zhang, G.; Zhang, Y.; Chen, H.; Lu, J. Mixed similarity diffusion for recommendation on bipartite networks. *IEEE Access* **2017**, *5*, 21029–21038. [CrossRef]
37. Witten, I.H.; Frank, E.; Trigg, L.; Hall, M.; Holmes, G.; Cunningham, S.J. Weka: Practical machine learning tools and techniques with Java implementations. In Proceedings of the ICONIP/ANZIIS/ANNES'99 Workshop on Emerging Knowledge Engineering and Connectionist-Based Information Systems, Dunedin, New Zealand, 22–23 November 1999; Kasabov, N., Ko, K., Eds.; 1999; pp. 192–196.
38. Yang, Z.; Wu, B.; Zheng, K.; Wang, X.; Lei, L. A survey of collaborative filtering-based recommender systems for mobile internet applications. *IEEE Access* **2016**, *4*, 3273–3287. [CrossRef]
39. Yu, F.; Zeng, A.; Gillard, S.; Medo, M. Network-based recommendation algorithms: A review. *Physica A* **2015**, *452*, 192–208. [CrossRef]
40. Zeng, W.; An, Z.; Liu, H.; Shang, M.-S.; Zhou, T. Uncovering the information core in recommender systems. *Sci. Rep.* **2014**, *4*, 6140. [CrossRef] [PubMed]
41. Zeng, W.; Zeng, A.; Shang, M.S.; Zhang, Y.C. Information filtering in sparse online systems: Recommendation via semi-local diffusion. *PLoS ONE* **2013**, *8*, e79354. [CrossRef]
42. Zhang, F.-G.; Zeng, A. Information filtering via heterogeneous diffusion in online bipartite networks. *PLoS ONE* **2015**, *10*, e0129459. [CrossRef]
43. Zhang, S.; Yao, L.; Sun, A. Deep learning based recommender system: A Survey and new perspectives. *arXiv* **2017**, arXiv:1707.07435. [CrossRef]
44. Zhang, Y.-C.; Blattner, M.; Yu, Y.-K. Publisher's note: Heat conduction process on community networks as a recommendation model. *Phys. Rev. Lett.* **2007**, *99*, 169902. [CrossRef]
45. Zhou, T.; Kuscsik, Z.; Liu, J.-G.; Medo, M.; Wakeling, J.R.; Zhang, Y.-C. Solving the apparent diversity-accuracy dilemma of recommender systems. *Proc. Natl. Acad. Sci. USA* **2010**, *107*, 4511–4515. [CrossRef]
46. Zhou, T.; Ren, J.; Medo, M.; Zhang, Y.C. Bipartite network projection and personal recommendation. *Phys. Rev. E* **2007**, *76*, 1–7. [CrossRef] [PubMed]
47. Ziegler, C.-N.C.; McNee, S.M.S.; Konstan, J.a.J.; Lausen, G. Improving recommendation lists through topic diversification. In Proceedings of the 14th International Conference on World Wide Web, Chiba, Japan, 10–14 May 2005; Volume 5, pp. 22–32. [CrossRef]
48. Juliana, A.P.; Pawel, M.; Sebastian, K.; Myra, S.; Gunter, S. A Feature-based personalized recommender system for product-line configuration. In Proceedings of the 2016 ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences, Amsterdam, The Netherlands, 20 October 2016; pp. 120–131. [CrossRef]
49. Runran, L.; Chunxiao, J.; Tao, Z.; Duo, S.; Binghong, W. Personal recommendation via modified collaborative filtering. *Physica A* **2009**, *388*, 462–468. [CrossRef]
50. Jinpeng, C.; Yu, L.; Deyi, L. Dynamic group recommendation with modified collaborative filtering and temporal factor. *Int. Arab J. Inf. Technol.* **2016**, *13*, 294–301.

51. Chaturvedi, A.; Green, P.; Caroll, J. K-modes clustering. *J. Classif.* **2001**, *18*, 35–55. [CrossRef]
52. Kourosh, M. Recommendation system based on complete personalization. *Procedia Comput. Sci.* **2016**, *80*, 2190–2204. [CrossRef]
53. Yonghong, T.; Bing, Z.; Yanfang, W.; Yue, Z.; Qi, W. College library personalized recommendation system based on hybrid recommendation algorithm. *Procedia CIRP* **2017**, *83*, 490–494. [CrossRef]
54. Marchela, A.; Christos, C. Personalized micro-service recommendation system for online news. *Procedia Comput. Sci.* **2019**, *160*, 610–615. [CrossRef]
55. Julián, M.-P.; Jose, A.; Edwin, M.; Camilo, S. Autonomous recommender system architecture for virtual learning environments. *Appl. Comput. Inf.* 2020. [CrossRef]
56. Anand, S.T. Generating items recommendations by fusing content and user-item based collaborative filtering. *Procedia Comput. Sci.* **2020**, *167*, 1934–1940. [CrossRef]
57. Pradeep, K.R.; Sarabjeet, S.C.; Rocky, B. A machine learning approach for automation of resume recommendation system. *Procedia Comput. Sci.* **2020**, *167*, 2318–2327. [CrossRef]
58. Hanane, Z.; Souham, M.; Chaker, M. New contextual collaborative filtering system with application to personalized healthy nutrition education. *J. King Saud Univ.–Comput. Inf. Sci.* **2020**. [CrossRef]
59. Christian, R.; Michael, W.; Günther, P. State of the art of reputation-enhanced recommender systems. *Web Intell.* **2018**, *16*, 273–286.
60. Pasquale, D.M.; Lidia, F.; Fabrizio, M.; Domenico, R.; Giuseppe, M.L.S. Providing recommendations in social networks by integrating local and global reputation. *Infor. Syst.* **2018**, *78*, 58–67. [CrossRef]
61. Barry, S.; Maurice, C.; Peter, B.; Kevin, M.; Michael, P.O. Collaboration, Reputation and Recommender Systems in Social Web Search. In *Recommender Systems Handbook*; Springer: Boston, MA, USA, 2015; pp. 569–608. [CrossRef]
62. Stai, E.; Kafetzoglou, S.; Tsiropoulou, E.E.; Papavassiliou, S. A holistic approach for personalization, relevance feedback & recommendation in enriched multimedia content. *Multimed. Tools Appl.* **2018**, *77*, 283–326. [CrossRef]
63. Vasiliki, P.; Stella, K.; Eirini, E.T.; Aggeliki, D.; Symeon, P. Personalized multimedia content retrieval through relevance feedback techniques for enhanced user experience. In Proceedings of the 13th International Conference on Telecommunications (ConTEL), Graz, Austria, 13–15 July 2015. [CrossRef]
64. Simon, D. Dynamic generation of personalized hybrid recommender systems. In Proceedings of the RecSys'13: Proceedings of the 7th ACM Conference on Recommender Systems, Hong Kong, China, 12–16 October 2013; pp. 443–446. [CrossRef]
65. Thai, M.T.; Wu, W.; Xiong, H. *Big Data in Complex and Social Networks*, 1st ed.; Chapman & Hall/CRC Big Data Series; CRC Press: Boca Raton, FL, USA, 2016; ISBN-10: 1498726844, ISBN-13: 978-1498726849.
66. Abbas, S.M.; Alam, K.A.; Shamshirband, S. A soft-rough set based approach for handling contextual sparsity in context-aware video recommender systems. *Mathematics* **2019**, *7*, 740. [CrossRef]
67. Sardianos, C.; Ballas Papadatos, G.; Varlamis, I. Optimizing parallel collaborative filtering approaches for improving recommendation systems performance. *Information* **2019**, *10*, 155. [CrossRef]
68. Pajuelo-Holguera, F.; Gómez-Pulido, J.A.; Ortega, F. Performance of two approaches of embedded recommender systems. *Electronics* **2020**, *9*, 546. [CrossRef]
69. Bai, L.; Hu, M.; Ma, Y.; Liu, M. A hybrid two-phase recommendation for group-buying e-commerce applications. *Appl. Sci.* **2019**, *9*, 3141. [CrossRef]
70. Cintia Ganesha Putri, D.; Leu, J.-S.; Seda, P. Design of an unsupervised machine learning-based movie recommender system. *Symmetry* **2020**, *12*, 185. [CrossRef]
71. Bhaskaran, S.; Santhi, B. An efficient personalized trust based hybrid recommendation (TBHR) strategy for e-learning system in cloud computing. *Cluster Comput.* **2019**, *22*, 1137–1149. [CrossRef]
72. Marappan, R.; Sethumadhavan, G. Solution to graph coloring using genetic and tabu search procedures. *Arab. J. Sci. Eng.* **2018**, *43*, 525–542. [CrossRef]
73. Marappan, R.; Sethumadhavan, G. Complexity analysis and stochastic convergence of some well-known evolutionary operators for solving graph coloring problem. *Mathematics* **2020**, *8*, 303. [CrossRef]