Hybrid Sparrow Clustered (HSC) Algorithm for Top-N Recommendation System

Bharti Sharma 1, Adeel Hashmi 1, Charu Gupta 2,*, Osamah Ibrahim Khalaf 3, Ghaida Muttashar Abdulsahib 4 and Malakeh Muhyiddeen Itani 5

Maharaja Surajmal Institute of Technology, New Delhi 110058, India; bhartisharma@msit.in (B.S.); adeel.hashmi@msit.in (A.H.)
1 Bhagwan Parshuram Institute of Technology, New Delhi 110089, India
2 Al-Nahrain Nanorenewable Energy Research Center, Al-Nahrain University, Baghdad 64074, Iraq: usama81818@nahrainuniv.edu.iq
3 Department of Computer Engineering, University of Technology, Baghdad 10066, Iraq: 30834@uotechnology.edu.iq
4 General Education Program, Dar Al-Hekma University, Jeddah 22246, Saudi Arabia; mitani@dah.edu.sa
5 * Correspondence: charugupta@bpitindia.com or charu.wai1987@gmail.com

Abstract: Recommendation systems suggest relevant items to a user based on the similarity between users or between items. In a collaborative filtering approach for generating recommendations, there is a symmetry between the users. That is, if user A has similar interests with user B, then an item liked by B can be recommended to A and vice versa. To provide optimal and fast recommendations, a recommender system may generate and keep clusters of existing users/items. In this research work, a hybrid sparrow clustered (HSC) recommender system is developed, and is applied to the MovieLens dataset to demonstrate its effectiveness and efficiency. The proposed method (HSC) is also compared to other methods, and the results are compared. Precision, mean absolute error, recall, and accuracy metrics were used to figure out how well the movie recommender system worked for the HSC collaborative movie recommender system. The results of the experiment on the MovieLens dataset show that the proposed method is quite promising when it comes to scalability, performance, and personalized movie recommendations.

Keywords: collaborative filtering; clustering; MapReduce; recommender system; swarm intelligence

1. Introduction

With the rapid expansion of information on the Web, locating relevant information has become a prominent issue. E-commerce has grown significantly in the Internet era, enabling the selling of various products. E-commerce users face difficulties when it comes to selecting a product among millions of options. The recommendation system assists the e-commerce user in making selections among millions of products [1]. A recommender system (RS) gathers information about the goods a consumer is interested in and makes recommendations about those items or products [2]. Nowadays, practically every e-commerce website makes use of RS, supporting millions of users. E-commerce sites such as Netflix and MovieLens [3] for movies, Amazon [4] for books, CDs, and a variety of other things, Entree for restaurants, and Jester [5] for jokes all make use of a recommender system to aid their customers. The recommender system’s output is used by both e-commerce organizations and users [6]. That is, RS not only assists the consumer in obtaining the desired item, but also increases the organization’s revenue by selling more products.

A recommender system’s objective is to give relevant product recommendations to the customer. Algorithms for recommendation systems fall into three broad categories: collaborative, content-based, and hybrid [2]. Collaborative filtering (CF) is the simplest and most efficient of the three algorithms, having been implemented in a variety of real-world applications, including Netflix and Amazon. According to the prediction approach utilized,
CFs can be classified as item-based or user-based. Collaborative filtering identifies individuals who share common interests and generates predictions about them. On the other hand, the item-based approach proposes the most related things to the user. The symmetric relationship of collaborative filtering have been exploited in SRMC [7] and PCA-SVD transformed K-means++ [8] approach. Content-based filtering (CBF) [9,10] makes a suggestion based on what the same person did before. The CBF is based on the item's description and the user’s background [11,12]. For example, when you use the demographic filtering technique [13], you look for people who have the same nationality, age, gender, and more. In [14] hybrid filtering is discussed, which combines both the filtering techniques [15,16]. In order to improve the performance of CF, different clustering techniques are used to make better recommendations based on what users like [17,18]. Clustering is a way to group people together based on how well they rate the same things. A cluster of users is sent to an active user who likes the same things as the people in that cluster. Clustering methods like k-means and fuzzy c-means are available in user-based CF. The quality of the recommendation system is still an open question.

Several techniques have been devised for the development of recommender systems, including swarm-based algorithms [19]. These algorithms are nature-inspired and based on the decision-making capabilities of decentralized and self-organized clans of organisms in a community. These organisms obtain local information from the environment as well as other agents. These organisms’ co-evolutionary behavior resembles the iterative nature of swarm algorithms, such as the firefly algorithm (FA), cuckoo search algorithm (CSA), grasshopper algorithm (GA), whale algorithm (WSA), sparrow search algorithm (SSA), etc. [20]. SSA is based on the group wisdom, foraging, and anti-predator behavior of sparrows [21]. This is why it is used in research to find the best way to improve [22–30].

In this paper, we incorporate swarm intelligence-based optimization to generate optimal cluster(s) for a recommendation system by using a sparrow search algorithm. The proposed algorithm exploits the symmetric relationships between the users to generate better recommendations. Swarm-based recommendation systems work in two phases, which include the training phase and recommendation phase. Phase I is an offline process in which a rating matrix is produced from the collected data and clusters are obtained by using a swarm clustering algorithm. Phase II is an online process in which the recommendation for an active user is performed.

The rest of the paper is organized as follows: Section 2 provides the literature survey related to swarm intelligence and recommender systems, Section 3 presents the methodology used for generating recommendations, Section 4 deals with experiments and results, and Section 5 gives the conclusion and future scope.

2. Related Work

Some important things to ponder when evaluating the CF recommender systems are covered in [31]. These include the user’s task, the dataset used, and how to judge a recommender system’s accuracy. One way to make a recommendation is to use clustering. The cluster that is formed must have a minimum number of inter- and the maximum number of intra-similarities [14]. Clustering techniques are used a lot in CF-based clustering systems to make them work better.

According to Feng et al. [32], co-clustering can increase the performance of top N suggestions. They demonstrate a recommendation system called user–item community detection-based recommendation (UICDR). The primary idea is to create a bipartite network of data about how people and objects interact. Individuals and objects are classified into several categories. When we have clusters of people and things, we may apply its same collaborative filtering algorithms to each cluster. The results demonstrate that the suggested method can greatly improve the performance of the top N suggestions for a variety of well-known collaborative filtering techniques.

In [33], the clustering method is used to make traditional recommender systems work better with multi-criteria ratings. It used Mahalanobis distance to figure out how similar people in the same cluster are, which made recommendations more accurate.
The authors of [34] came up with a way to improve the quality of recommendations by using a weighted clustering method viz. spherical k-means. If there are a lot of users and a lot of items, this method doesn’t get more complicated (in contrast to existing incremental methods). Therefore, the proposed collaborative filtering system is better for dynamic settings, wherein there are a lot of databases and information changes quickly. In the experiments, real-world datasets were used to see how well and how quickly the method works in terms of scalability and how well it makes recommendations.

Numerous research publications have demonstrated that traditional clustering approaches (such as k-means clustering) can provide local bests. Thus, clustering and algorithms inspired by nature-inspired global optimization can be employed to enhance the performance of recommendations. Reference [35] developed a hybrid movie recommendation system that incorporated both genetic algorithms (GAs) and k-means clustering. It employs principle component analysis (PCA) to reduce the size of the movie population space, hence speeding up the calculations. The studies (conducted on the MovieLens dataset) demonstrate that the suggested method is capable of being extremely accurate and providing more dependable and tailored movie recommendations than existing methods. The authors of [36] used the PSO to enhance the performance of a recommender system. They employed a k-means approach to obtain the initial parameters for particle swarm optimization in the suggested method (PSO). PSO begins with a seed and enhances fuzzy c-means (FCM) to do soft grouping of data (users) rather than hard clustering as in k-means.

The author of [37] devised a method for making a top-N recommendation based on ant behavior. This approach is divided into two sections. Users’ opinions are clustered into categories with a preset number of clusters by using an ant-based clustering technique. The results are then kept in a database for future recommendations and can be used to assist individuals in locating what they’re looking for. The second phase involves making a recommendation to the user who is currently using the app. The new method’s efficiency was evaluated by using the Jester dataset and was compared to a conventional collaborative filtering-based approach. The proposed solution outperforms the usual recommender system, as demonstrated by the findings.

The authors of [38] create a collaborative recommender system for movies by using a gray wolf optimization algorithm and fuzzy c-means (FCM) clustering. To obtain the initial clusters from the MovieLens dataset, the gray wolf optimizer technique is employed. FCM is used to categorize users in a dataset according to how similar their ratings are. The authors of [39] made recommendations by using the collaborative filtering technique, k-means clustering, and cuttlefish optimization. The proposed solution begins with the k-means algorithm, which groups people with similar tastes together, and then uses the cuttlefish algorithm to generate the best recommendations based on the k-means algorithm’s results.

In [40], the authors combine the k-means and cuckoo search algorithms to create recommender systems, whereas in [41], the k-means clustering technique is combined with an algorithm for optimizing artificial bee colonies. In the first phase of the proposed strategy, k-means is employed to classify people into distinct clusters. The second phase employs the artificial bee colony optimization technique to enhance the k-means procedure’s outcomes [42]. The proposed method’s results demonstrate significant progress in terms of scalability and performance. Additionally, they provide more accurate personalized movie recommendations by avoiding the cold start issue. In [43], an urban travel recommendation is made by using a hybrid quantum-induced swarm intelligence clustering. The authors demonstrate a novel method for grouping users in order to create a collaborative filtering-based recommender system that takes advantage of quantum-behaved particle swarm optimization. Because clustering algorithms were combined with collaborative filtering to put individuals with shared characteristics into clusters, the recommendation’s efficiency was boosted.

For dynamic recommendation systems, [44] discusses an evolutionary clustering method that groups objects based on their time characteristics. The clustering algorithm creates comparable user groups and increases them over time. This graph illustrates how user performance evolves over time in terms of accuracy and relevance. The authors of [45]
demonstrate how to use evolutionary clustering to forecast ratings based on user collaborative filtering. The primary objective of the strategy demonstrated is to put people with similar interests together and to assist them in finding things that best suit their inclinations. Collaborative filtering is used in each cluster following the clustering procedure. The target rating is determined collaboratively by the users in the cluster. The authors of [46] use hierarchical particle swarm optimization to do clustering (HPSO-clustering). It is based on the PSO method and offers both hierarchical and partitional clustering advantages. The authors of [47] employ three techniques: visual clustering recommendation, a blend of visual clustering, and user-based methods, and a hybrid of visual clustering and item-based methods. To aggregate user stuff together, a genetic algorithm is applied.

3. Materials and Methods

The approach used in this work for generating recommendations is presented in Figure 1. The dataset is passed to the sparrow search algorithm for generating optimal clusters. For generating recommendations for the current user, the cluster with the nearest centroid is selected. In the selected cluster, the k-nearest neighbors are selected by using the kNN algorithm. The items rated by these k-nearest users are averaged, and the items with highest average ratings are suggested to the active user.

Figure 1. Generation of the recommendation system.

3.1. Clustering Using Sparrow Search

A sparrow is a small but very clever bird with a good memory. There are two types of captive house sparrows: producers and scavengers [48]. The producers look for food sources themselves, while the scavengers get food from producers. The sparrows often change their behavior and switch between making and scavenging [49–52]. Sparrows keep an eye on the behavior of the sparrows around them.

Individuals’ energy reserves also play a role in how the sparrow chooses to forage. The sparrows with low energy reserves look for food more often [53–55]. It’s important to point out that the birds that live on the outside of the population are more likely to be attacked by predators and are always trying to get a better place [56,57]. Keep in mind that the animals that are in the middle may move closer to their neighbors in order to reduce their area of danger [58]. We also know that all sparrows have a natural curiosity about everything, and they are always on the lookout for danger. For example, when a bird sees a predator, one or more of them chirp and the whole group flees [59]. The behavior of sparrows can be summarized as follows:

1. The producers usually have a lot of energy and give scavengers places or directions to look for food. They are in charge of finding places where a lot of food can be found. The amount of energy each person has is based on how well they are thought of by others.

2. As soon as the sparrow sees the predator, the birds start to chirp as a warning. To keep people safe, producers have to lead all scavengers to the safe area when the alarm value is higher than the safety level.

3. Each sparrow can become a producer as long as it looks for the best places to get food, but the overall population doesn’t change.
(4) The sparrows with the most energy would be the ones who made the food. Several people who are starving are more likely to fly to other places to get food so that they can have more energy to do things.

(5) The scavengers follow the person who can give them the best food as they look for food. In the meantime, some scavengers may keep an eye on the people who make the food and compete with each other to get more.

(6) The sparrows at the edge of the group move quickly to a safe area when they sense danger. The sparrows in the middle move around so they can be near other people.

Let $X_{i,j}$ denote the $i$th sparrow’s $j$th coordinate of location ($t$ denotes the iteration). The site of the producer is updated as per Equation (1),

$$X_{i,j}^t = \begin{cases} 
X_{i,j}^t \cdot \exp\left(\frac{-i}{\text{iter}}\right) & \text{if } R_i < ST \\
X_{i,j}^t + Q \cdot L & \text{if } R_i \geq ST
\end{cases}$$

(1)

where, $ac(0,1)$ is a random number, $R_2 \in [0,1]$ represents the alarm value, $ST \in [0.5, 1.0]$ represents the safety threshold, $Q$ is a random number from a normal distribution, $L$ is a $1 \times d$ matrix for which each element inside is 1.

The location of the scavenger is updated as per Equation (2),

$$X_{i,j}^{t+1} = \begin{cases} 
Q \cdot \exp\left(\frac{x_{\text{worst}} - x_{i,j}^t}{\tau}\right) & \text{if } i > \frac{n}{2} \\
x_{i,j}^{t+1} + \left[ X_{i,j}^t - X_{p}^{t+1} \right] \cdot A^+ \cdot L & \text{otherwise}
\end{cases}$$

(2)

where, $x_{\text{worst}}$ denotes the current global worst location, $X_p$ is the optimal position occupied by the producer, $A$ is a $1 \times d$ matrix with each element being $+1$ or $-1$, and $A^+ = A^T (AA^T)^{-1}$.

The sparrows on the periphery have a higher chance of being attacked by predators, so they constantly move inwards. We assume that these sparrows, which can sense danger, account for 10 to 20% of the population. Equation (3) expresses their inward movement,

$$X_{i,j}^{t+1} = \begin{cases} 
X_{best}^t + \beta \cdot \left[ X_{i,j}^t - X_{\text{best}}^t \right] & \text{if } f_i > f_{g} \\
X_{i,j}^t + K \cdot \left[ x_{i,j}^t - x_{\text{worst}}^t \right] \left( \frac{1}{(f_i - f_g) + \epsilon} \right) & \text{if } f_i = f_{g}
\end{cases}$$

(3)

where, $X_{best}$ denotes the current global best location, $\beta$ is a random value from a normal distribution, $K \in [-1, +1]$ is a random number, $f_i$ is the fitness value of the current sparrow, $f_g$ and $f_w$ are current global best and global worst fitness values, and $\epsilon$ is the smallest constant to avoid zero division error.

Like the other swarm-based algorithms, the sparrow search algorithm can also be utilized for data clustering (refer to Figure 2 and Algorithm 1). The fitness function for the clustering process is WCSS (within-cluster sum of squared errors), i.e., the average Euclidean distance of all the points from their assigned cluster head. To make successful recommendations to users based on their interests, we introduce a new recommender system based on a hybrid sparrow clustering recommender system (HSC). This is intended to be more adaptable than typical recommender systems in generating diverse and unique recommendations.
Algorithm 1. Algorithm for clustering using sparrow search

Input
- data: the dataset to be clustered
- $K$: the number of clusters to be generated
- $N$: the number of sparrows
- $P$: the number of producers
- $S$: the number of sparrow who sense the danger
- $G$: the maximum iterations

Algorithm
1. Create $N$ sparrow, each having $K$ cluster heads with random normalized values.
2. Create clusters of data for each sparrow (based on shortest Euclidean distance).
3. Calculate the fitness of each sparrow.
4. Rank the sparrows and find the current best and cueeent worst sparrow.
5. for $i = 1$ to $P$: update sparrow location using Equation (1).
6. for $i = P + 1$ to $N$: update sparrow location using Equation (2).
7. for $i = 1$ to $S$: update sparrow location using Equation (3).
8. Change the location of each sparrow if it is better than the old location.
9. Repeat the step 3–8 for $G$ iterations.
10. Return the cluster- heads of the fittest sparrow.

Output: Optimal clusters

3.2. Generating Recommendation by Using Sparrow Clustered Recommendation System

To generate recommendations for an active user, a cluster (among $k$-clusters) is to be selected. A simple approach is to select the cluster whose centroid has the highest similarity with the active user (e.g., the centroid with the lowest Euclidean distance with the active user). If there are large numbers of clusters, then multiple clusters can also be used for better results. In such a case, the probability that a cluster $i$ is chosen for generating recommendations was given by

$$P_i = \frac{p_i \times d_i}{\sum_{i=1}^{k} p_i \times d_i}$$ (4)
where \( p_i \) is the density of the cluster, and \( d_i \) is the Euclidean distance between the active user profile and the centroid of the cluster.

The density of the cluster (i.e., \( p_i \)) can be calculated as follows:

\[
p_i = \frac{N_i}{\sum_{i=1}^{K} N_i}
\]

where, \( N_i \) are the number of users in the cluster.

The recommendations are provided from the cluster with the highest probability or multiple clusters that lie in a particular probability range. The latter approach may provide the active user recommendations which are different and make him interested in trying something new. After selecting the clusters for recommendations, the next step is to predict the ratings for the active user’s unrated items and recommend the items whose predicted value is high. If there is only one chosen cluster, then the values of unrated items are simply the average of the corresponding item ratings by all the users in the cluster. But if multiple clusters have been selected, then we also consider the quality of ratings in each chosen cluster. A criterion of the rating quality of a cluster is the number of ratings available of each item in the cluster. The higher the densities of ratings, the better the quality of the cluster,

\[
Q_i = \frac{\sum_{p=1}^{t} r_{ip}}{n_i \times t}
\]

where, \( Q_i \) is the quality of cluster, \( n_i \) is the number of users in cluster \( i \), \( t \) is the number of items, and \( r_{ip} \) is the count of ratings available for item \( p \) in cluster \( i \).

4. Results

4.1. Phase I: Training Phase

From Tables 1–3, the MovieLens dataset has 100,000 ratings of 943 users for 1682 movies. The movies are classified into 19 genres viz. action, comedy, horror, etc. The dataset is divided into two parts, 80% as training data and 20% as test data. The data is converted into a 943 x 1682 matrix. The dataset need not be normalized as the ratings are in the scale of 1–5. However, the dataset is sparse (only 100,000 ratings out of possible 1,586,126 available), so we need to replace the missing values by 0. The rating matrix containing the rating data is clustered into \( K \) clusters by using the sparrow search technique. \( N \) sparrows are initially generated, each having \( k \) cluster heads. Each cluster head has 1682 dimensions with values in the range of 1–5, generated at random. For each sparrow, \( K \) clusters are generated by assigning each point in the dataset to the nearest cluster head in the sparrow (the similarity measure used is Euclidean distance), and the WCSS (within-cluster sum of squares) is calculated. The sparrow with the lowest WCSS is considered to be the fittest sparrow, and the less fit sparrows are moved toward the fittest sparrow. This process is repeated to a certain number of iterations, and the fittest sparrow after all these iterations is considered to be the final solution (clusters).

Table 1. Snapshot of MovieLens dataset.

<table>
<thead>
<tr>
<th>User #1</th>
<th>Movie #1</th>
<th>Movie #2</th>
<th>.</th>
<th>.</th>
<th>.</th>
<th>Movie #1682</th>
</tr>
</thead>
<tbody>
<tr>
<td>User #2</td>
<td>0</td>
<td>3</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>0</td>
</tr>
<tr>
<td>.</td>
<td>4</td>
<td>0</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>2</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>User #943</td>
<td>0</td>
<td>5</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>0</td>
</tr>
</tbody>
</table>
Table 2. Sample snapshot of the population (20 sparrows with 3 cluster heads each).

<table>
<thead>
<tr>
<th>Sparrow #1</th>
<th>Cluster-Head #1</th>
<th>Cluster-Head #2</th>
<th>Cluster-Head #3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 3. Sample of cluster assigned to each user in fittest sparrow (assuming that there are 3 clusters in each sparrow).

<table>
<thead>
<tr>
<th>User #1</th>
<th>User #2</th>
<th>User #3</th>
<th>User #943</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster #</td>
<td>3</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

4.2. Phase II: Process of Recommendation for Active Users

To generate recommendations for an active user, a cluster (among k-clusters) is to be selected. The selected cluster is the one whose centroid is nearest to the current user. Within the selected cluster, the movies which have highest average ratings are recommended. For performance analysis of our recommendation system framework, we calculate various metrics like MAE, SD, RMSE, and t-value. Different graphs and tables of the calculated results are shown for a better understanding of the framework.

4.2.1. Mean Absolute Error (MAE)

We calculated mean absolute error on the dataset of the MovieLens dataset by using Equation (7),

\[ \text{MAE} = \frac{\sum_{i,j} |p_{ij} - t_{ij}|}{M} \]  

where, \( M \) is the number of movies in the dataset, \( p_{ij} \) is the predicted value for \( i \) user on \( j \) items, and \( t_{ij} \) is the true rating. The results are shown in Table 4 for the calculated MAE for different values of \( K \). The outcome as observed from this table is that as we increase the number of clusters, MAE values gradually decrease.

Table 4. Performance of sparrow-search recommender system based on different cluster size on MovieLens 100 k dataset.

<table>
<thead>
<tr>
<th>S. No.</th>
<th>No. of Clusters (k)</th>
<th>MAE</th>
<th>SD</th>
<th>RMSE</th>
<th>t-Value</th>
<th>Recall</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>0.785</td>
<td>0.179</td>
<td>1.282</td>
<td>3.348</td>
<td>0.329</td>
<td>0.311</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>0.776</td>
<td>0.126</td>
<td>1.280</td>
<td>2.814</td>
<td>0.378</td>
<td>0.361</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>0.765</td>
<td>0.124</td>
<td>1.274</td>
<td>2.791</td>
<td>0.443</td>
<td>0.426</td>
</tr>
<tr>
<td>4</td>
<td>40</td>
<td>0.744</td>
<td>0.133</td>
<td>1.258</td>
<td>2.816</td>
<td>0.491</td>
<td>0.452</td>
</tr>
<tr>
<td>5</td>
<td>50</td>
<td>0.729</td>
<td>0.125</td>
<td>1.248</td>
<td>2.816</td>
<td>0.534</td>
<td>0.498</td>
</tr>
<tr>
<td>6</td>
<td>60</td>
<td>0.712</td>
<td>0.112</td>
<td>1.239</td>
<td>2.816</td>
<td>0.567</td>
<td>0.538</td>
</tr>
<tr>
<td>7</td>
<td>70</td>
<td>0.695</td>
<td>0.117</td>
<td>1.229</td>
<td>2.814</td>
<td>0.604</td>
<td>0.552</td>
</tr>
</tbody>
</table>

4.2.2. Standard Deviation (SD)

By using Equation (8), we calculate SD on the MovieLens dataset,

\[ \text{SD} = \sqrt{\frac{\sum_{i=1}^{k} \sum_{j=1}^{n_i} \left( \frac{\sum_{l=1}^{n_i} \left[ \frac{t_{ij} - \bar{t}_{ij}}{\bar{t}_{ij}} \right]^2}{n_i} \right)}{n_i}} \]  

(8)
4.2.3. Root Mean Square Error (RMSE)

We calculated RMSE on the MovieLens dataset by using Equation (9),

\[
RMSE = \sqrt{\frac{(p - t)^2}{n}}
\]  

(9)

where, \( p \) is the predicted value, \( t \) is the actual value, and \( n \) is the number of predicted ratings.

4.2.4. t-Value

This \( t \)-value basically depends on the values of the mean obtained for different clusters and their calculated SD values. We calculate the \( t \)-value of the dataset by using Equation (10),

\[
t - value = \sum_{i=1}^{k} \sum_{j=1}^{k} \left( \frac{\bar{X}_i - \bar{X}_j}{\sqrt{SD_i^2/n_i + SD_j^2/n_j}} \right).
\]  

(10)

4.2.5. Precision

Precision is a measure of exactness that determines the fraction of relevant items retrieved from all items retrieved. As an example, precision refers to the proportion of recommended movies that are good,

\[
Precision = \frac{t_p}{t_p + f_p}.
\]  

(11)

\( t_p \) : true positive, \( f_p \) : false positive

4.2.6. Recall

Recall is a measure of completeness. It determines the fraction of relevant items retrieved out of all relevant items. As an example, recall can refer to the proportion of all good movies recommended,

\[
Precision = \frac{t_p}{t_p + f_n}.
\]  

(12)

\( t_p \) : true positive, \( f_n \) : false negative

Results obtained by executing a sequence of experiments are based on four datasets and some metrics with varied parameters. Experiments were performed to evaluate the efficiency of the presented system in a collaborative filtering environment. A system with the configuration of Intel i5 (3.00 GHz), 4 GB RAM, Windows 10 was used for experimental computation, and all the algorithms are executed in the Python programming language (Anaconda environment). Result visualization is done in MS Excel (Microsoft, Redmond, WA, USA).

Four datasets (i.e., MovieLens 100 k, MovieLens 1 million, Jester, and Epinions) are used for result observations considering different parameters like ratings, users, and items.

1. MovieLens 100,000—an original dataset composed from the MovieLens website (movielens.umn.edu, accessed on 4 May 2022) in a time frame of 7 months (i.e., from 19 September 1997 to 22 April 1998) consists of 100,000 ratings of 1682 movies from 943 users. Rating scale ranges between 1 and 5. Data is easily available for experimental computation and analysis.

2. MovieLens 1 million—This Dataset consists of four features: userID, MovieID, rating, and timestamp. It includes 6040 MovieLens users who provided 1,000,209 ratings of 3900 movies. With a rating scale of 1–5 stars, every single user provides a minimum of 20 ratings.
3. Jester—Jester is a dataset of 59,132 users and 150 jokes, and consists of 1.7 million ratings ranging between −10 and 10. It is widely available and is often used for experimental examination of collaborative filtering recommender systems. The dataset features include [userid][itemid] and [rating]. University of California, Berkeley created this dataset, and the data is associated with an online joke-recommendation system.

4. Epinion—A total of 664,824 ratings of 139,738 items from 49,290 users constitutes the Epinion dataset. Open-source data composed from Epinion.com includes userid, itemid, and rating. A scale is set ranging from 1–5 for rating individual items. The performance of the proposed sparrow-based recommendation system is based on a different cluster size.

Table 4 and Figure 3 compare the MAE, SD, RMSE, t-value, recall, and precision values for hybrid sparrow-based clustered algorithms to the collaborative filtering technique (CF) to demonstrate the proposed algorithm’s performance based on different cluster size.

Table 5 compares the performance of the proposed algorithm with state-of-the-art algorithms on the MovieLens 100 k dataset. From the results, it is observed that the proposed algorithm gives better results than other state-of-the-art algorithms and similar swarm-based algorithms.

![Figure 3. Performance of a sparrow search based on cluster size.](image)

Table 5. Performance comparison with other algorithms (k = 70) on MovieLens 100 k.

<table>
<thead>
<tr>
<th></th>
<th>NMF</th>
<th>SVD</th>
<th>Firefly</th>
<th>Cuckoo</th>
<th>Whale</th>
<th>Sparrow</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAE</td>
<td>0.758</td>
<td>0.737</td>
<td>0.695</td>
<td>0.697</td>
<td>0.691</td>
<td>0.685</td>
</tr>
<tr>
<td>SD</td>
<td>0.123</td>
<td>0.121</td>
<td>0.117</td>
<td>0.119</td>
<td>0.115</td>
<td>0.113</td>
</tr>
<tr>
<td>RMSE</td>
<td>0.963</td>
<td>0.934</td>
<td>1.229</td>
<td>1.231</td>
<td>1.228</td>
<td>1.220</td>
</tr>
<tr>
<td>t-value</td>
<td>2.818</td>
<td>2.816</td>
<td>2.814</td>
<td>2.816</td>
<td>2.811</td>
<td>2.792</td>
</tr>
<tr>
<td>Recall</td>
<td>0.594</td>
<td>0.597</td>
<td>0.604</td>
<td>0.592</td>
<td>0.610</td>
<td>0.640</td>
</tr>
<tr>
<td>Precision</td>
<td>0.546</td>
<td>0.549</td>
<td>0.552</td>
<td>0.554</td>
<td>0.560</td>
<td>0.602</td>
</tr>
</tbody>
</table>

Table 6 compares the performance of the proposed algorithm with state-of-the-art algorithms on the MovieLens 1 million dataset. From the results, it is observed that the proposed algorithm gives better results than other state-of-the-art algorithms and similar swarm-based algorithms.
Table 6. Performance comparison with other algorithms (k = 70) of MovieLens 1 million.

<table>
<thead>
<tr>
<th>Method</th>
<th>NMF</th>
<th>SVD</th>
<th>Firefly</th>
<th>Cuckoo</th>
<th>Whale</th>
<th>Sparrow</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAE</td>
<td>0.724</td>
<td>0.686</td>
<td>0.664</td>
<td>0.678</td>
<td>0.660</td>
<td>0.654</td>
</tr>
<tr>
<td>SD</td>
<td>0.123</td>
<td>0.121</td>
<td>0.117</td>
<td>0.119</td>
<td>0.115</td>
<td>0.113</td>
</tr>
<tr>
<td>RMSE</td>
<td>0.916</td>
<td>0.873</td>
<td>0.852</td>
<td>0.862</td>
<td>0.857</td>
<td>0.841</td>
</tr>
<tr>
<td>t-value</td>
<td>2.814</td>
<td>2.809</td>
<td>2.818</td>
<td>2.811</td>
<td>2.803</td>
<td>2.784</td>
</tr>
<tr>
<td>Recall</td>
<td>0.602</td>
<td>0.614</td>
<td>0.638</td>
<td>0.622</td>
<td>0.648</td>
<td>0.673</td>
</tr>
<tr>
<td>Precision</td>
<td>0.531</td>
<td>0.526</td>
<td>0.588</td>
<td>0.554</td>
<td>0.611</td>
<td>0.645</td>
</tr>
</tbody>
</table>

Table 7 compares the performance of the proposed algorithm with state-of-the-art algorithms for the Jester dataset. From the results, it is observed that the proposed algorithm gives better results than other state-of-the-art algorithms and similar swarm-based algorithms.

Table 7. Performance comparison with other algorithms (k = 70) of the Jester dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>NMF</th>
<th>SVD</th>
<th>Firefly</th>
<th>Cuckoo</th>
<th>Whale</th>
<th>Sparrow</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAE</td>
<td>3.51</td>
<td>3.37</td>
<td>3.11</td>
<td>3.41</td>
<td>3.33</td>
<td>3.09</td>
</tr>
<tr>
<td>SD</td>
<td>0.32</td>
<td>0.35</td>
<td>0.39</td>
<td>0.35</td>
<td>0.41</td>
<td>0.44</td>
</tr>
<tr>
<td>RMSE</td>
<td>4.63</td>
<td>4.49</td>
<td>4.26</td>
<td>4.31</td>
<td>4.22</td>
<td>4.01</td>
</tr>
<tr>
<td>t-value</td>
<td>3.11</td>
<td>3.04</td>
<td>2.98</td>
<td>3.02</td>
<td>2.96</td>
<td>2.91</td>
</tr>
<tr>
<td>Recall</td>
<td>0.62</td>
<td>0.67</td>
<td>0.72</td>
<td>0.70</td>
<td>0.73</td>
<td>0.77</td>
</tr>
<tr>
<td>Precision</td>
<td>0.68</td>
<td>0.69</td>
<td>0.74</td>
<td>0.71</td>
<td>0.74</td>
<td>0.78</td>
</tr>
</tbody>
</table>

Table 8 compares the performance of the proposed algorithm with state-of-the-art algorithms for the Epinion dataset. From the results, it is observed that the proposed algorithm gives better results than other state-of-the-art algorithms and similar swarm-based algorithms.

Table 8. Performance comparison with other algorithms (k = 70) on Epinion dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>NMF</th>
<th>SVD</th>
<th>Firefly</th>
<th>Cuckoo</th>
<th>Whale</th>
<th>Sparrow</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAE</td>
<td>1.06</td>
<td>0.96</td>
<td>0.87</td>
<td>0.91</td>
<td>0.84</td>
<td>0.78</td>
</tr>
<tr>
<td>SD</td>
<td>0.28</td>
<td>0.31</td>
<td>0.35</td>
<td>0.32</td>
<td>0.36</td>
<td>0.39</td>
</tr>
<tr>
<td>RMSE</td>
<td>1.38</td>
<td>1.26</td>
<td>1.15</td>
<td>1.23</td>
<td>1.10</td>
<td>1.02</td>
</tr>
<tr>
<td>t-value</td>
<td>2.09</td>
<td>1.98</td>
<td>1.81</td>
<td>1.88</td>
<td>1.82</td>
<td>1.74</td>
</tr>
<tr>
<td>Recall</td>
<td>0.65</td>
<td>0.69</td>
<td>0.74</td>
<td>0.71</td>
<td>0.74</td>
<td>0.76</td>
</tr>
<tr>
<td>Precision</td>
<td>0.71</td>
<td>0.73</td>
<td>0.74</td>
<td>0.73</td>
<td>0.75</td>
<td>0.77</td>
</tr>
</tbody>
</table>

Figures 4–7 provide the visualization of the comparative results of the proposed algorithm with state-of-the-art algorithms for the MovieLens 100 k, MovieLens 1 million, Jester, and Epinion datasets, respectively.
Figure 4. Performance comparison of the MovieLens 100K dataset.

Figure 5. Performance comparison of the MovieLens 1 million dataset.

Figure 6. Performance comparison of the Jester dataset.
Figure 7. Performance comparison of the Epinion dataset.

From the results, we conclude that the proposed sparrow algorithm for clustering outperforms the existing algorithms in terms of all the metrics. The sparrow algorithm outperformed SVD and NMF algorithms in the range of 5 to 10%, and other swarm-based algorithms in the range of 2–10%.

5. Conclusions and Future Scope

The use of standard clustering algorithms in recommender systems leads to slow results especially when a dataset is large. To speed up the clustering process, swarm intelligence-based clustering is a better option. In this proposed system, we presented a recommender system with four metaheuristic clustered algorithms, namely the firefly clustered algorithm, the cuckoo clustered algorithm, the whale clustered algorithm, and the sparrow clustered algorithm. In the proposed system, the symmetric nature of collaborative filtering is utilized. As far as clustering-based recommendation systems are concerned, getting optimal clusters is imperative in giving good recommendations. To obtain the optimal cluster, we need to minimize the within-cluster sum of squared (WCSS) distance. All algorithms are optimization algorithms and are used for reducing WCSS, which becomes the fitness function to be optimized.

The best clusters are then used to provide relevant and timely recommendations depending on the preferences of other cluster members. After obtaining the optimal cluster, recommendations can be generated by first selecting the right cluster(s) and then selecting the suitable items from the clusters. A straightforward approach is to select the cluster whose cluster head is closest to the active user, average the ratings for each item in the cluster, and then recommend the item with the highest cluster average rating. We evaluated our approach’s performance by using standard statistical measures such as mean absolute error, standard deviation, root mean squared error, and t-value. The results demonstrate that our approach generates highly relevant recommendations. In future work, alternative nature-inspired algorithms, such as multi-objective metaheuristic algorithms, can be utilized in place of the single-objective algorithm.

Author Contributions: Conceptualization, B.S.; methodology, B.S. and A.H.; software, A.H.; validation, B.S., A.H. and C.G.; formal analysis, O.I.K.; investigation, O.I.K. and G.M.A.; resources, M.M.I.; data curation, A.H.; writing—original draft preparation, B.S.; writing—review and editing, A.H.; visualization, B.S.; supervision, C.G.; project administration, O.I.K. and G.M.A.; funding acquisition, M.M.I. All authors have read and agreed to the published version of the manuscript.

Funding: Not applicable.

Institutional Review Board Statement: Not applicable.
Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References


8. Addagarla, S.K.; Amalanathan, A. Probabilistic unsupervised machine learning approach for a similar image recommender system for E-commerce. *Symmetry* 2020, 12, 1783. [CrossRef]


