Hindawi
Complexity
Volume 2022, Article ID 6958596, 12 pages
https://doi.org/10.1155/2022/6958596

Research Article

Affinity Propagation-Based Hybrid Personalized Recommender System

Iqbal Qasim,[1] **Mujtaba Awan**,[2] **Sikandar Ali**,[3,4] **Shumaila Khan**,[1] **Mogeeb A. A. Mosleh**,[5] **Ahmed Alsanad**,[6] **Hizbullah Khattak**,[7] **and Mahmood Alam**[1]

[1]*Department of Computer Science, University of Science and Technology, Bannu, Pakistan*
[2]*Department of Software Engineering, Riphah International University, Islamabad, Pakistan*
[3]*Department of Information Technology, The University of Haripur, Haripur 22621, Khyber Pakhtunkhwa, Pakistan*
[4]*Beijing Key Lab of Petroleum Data Mining, China University of Petroleum-Beijing, Beijing 102249, China*
[5]*Faculty of Engineering and Information Technology, Taiz University, Taiz 6803, Yemen*
[6]*STC's Artificial Intelligence Chair, Department of Information Systems, College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia*
[7]*Department of Information Technology, Hazara University Mansehra, Mansehra, Khyber Pakhtunkhwa, Pakistan*

Correspondence should be addressed to Sikandar Ali; sikandar@cup.edu.cn, Mogeeb A. A. Mosleh; mogeebmosleh@taiz.edu.ye, and Ahmed Alsanad; aasanad@ksu.edu.sa

Received 15 June 2021; Revised 21 September 2021; Accepted 7 January 2022; Published 27 January 2022

Academic Editor: Muhammad Ahmad

A personalized recommender system is broadly accepted as a helpful tool to handle the information overload issue while recommending a related piece of information. This work proposes a hybrid personalized recommender system based on affinity propagation (AP), namely, APHPRS. Affinity propagation is a semisupervised machine learning algorithm used to cluster items based on similarities among them. In our approach, we first calculate the cluster quality and density and then combine their outputs to generate a new ranking score among clusters for the personalized recommendation. In the first phase, user preferences are collected and normalized as items rating matrix. This generated matrix is then clustered offline using affinity propagation and kept in a database for future recommendations. In the second phase, online recommendations are generated by applying the offline model. Negative Euclidian similarity and the quality of clusters are used together to select the best clusters for recommendations. The proposed APHPRS system alleviates problems such as sparsity and cold-start problems. The use of affinity propagation and the hybrid recommendation technique used in the proposed approach helps in improving results against sparsity. Experiments reveal that the proposed APHPRS performs better than most of the existing recommender systems.

1. Introduction

Recommender systems (RSs) play a vital role in the adaptive web utilizing sophisticated algorithms to reduce the ever-growing information load. Ricci et al. [1] have defined recommender systems, software tools, and techniques, which provide choices to the user for a product selection. During the past decade, the recommender system has been used in several research domains, for example, information retrieval, cognitive science, e-commerce applications, knowledge management systems, and approximation theory [2, 3]. With the popularity of social media, such product recommender systems are getting intelligent, as they often incorporate user's comments about a particular product as input and update their underlying algorithms accordingly. However, due to the speedy growth of Internet technologies, web data is growing enormously; therefore, fetching relevant data from heterogeneous resources has become a difficult task [4]. Recommender system is an effective technology in helping users to tackle this problem by automatically recommending the

related information based on their personalized preferences, which are usually stored in a user profile.

Personalized recommendations are commonly presented as the items being rated. While carrying out this rating, the RS attempts to predict the most appropriate items on the basis of the user's preferences along with other recommendation parameters that vary from portal to portal. To predict an appropriate list of choices, RSs fetch user rating information and translate that to the user-item matrix.

The job of a current day recommender system is to predict item ratings and then make a recommendation close enough to the user preference. The ever-first system of recommendation named Tapestry [5] was developed in the mid 1990s. The most common research papers are focused on movie recommendation studies [6]. However, a great volume of the literature about the RS can be found to recommend music [7–9], best television show [9], best book to read [10], related document fetching in e-learning [11], knowledge system management [12], e-commerce [13], applications in markets [14], and web search [15].

RSs can be divided into three main categories, including collaborative filtering (CF) and content-based and hybrid methods [16]. Content-based filtering (CB) recommendations are based on the past decisions of users. The content-based filtering is efficient in finding text and items related to a certain interest by using techniques such as Boolean queries, though it has some limitations too. The content-based techniques often have the problem of lack of diversity in items recommendation [17]. In literature, content-based approaches are widely used in information retrieval [3] and information filtering [18] research. Some approaches used in content-based systems are News Weeder [19], Info Finder [20], and News Dude [21].

Collaborative filtering (CF) gathers ratings of users as opinions about certain items. The recommendation depends on the opinion of the user similarities and dissimilarities to the active user (neighbor) [22]. They aim that the collected information can be very effective for new recommendations. The collaborative filtering techniques are found to be much effective; however, it has some limitations too. One of its limitations is that the user rating analysis may be minimum; thus, the quality to recommend something would be poor. Also, collaborative filtering has common issues of sparsity, cold-start problem, and new user problem [23, 24]. Collaborative filtering algorithms are further categorized into model-based and memory-based algorithms [25]. Memory-based techniques like user-based KNN [26] use the whole user-item rating to locate k nearest neighbors for the active user and then use the ratings of these neighbors to generate recommendations. However, memory-based collaborative filtering (CF) algorithms are not scalable as the size of users grows. In contrast, model-based techniques like clustering [27] and Bayesian model [28] build a model first from the rating matrix and then use it for generating recommendations. The downside of the model-based algorithm is that the model must be rebuilt after adding a new to the rating matrix. To combine the strengths of the two stated approaches, a hybrid of model-based and memory-based techniques like Region KNN [25] and clustering-based KNN [28] can be used.

Hybrid recommender systems are the systems that combine the features of two or more recommendation techniques; for example, content-based and collaborative filtering [29] are combined for the aim of using the powers of these techniques and overcoming the weaknesses of any individual technique [30, 31]. The hybrid recommender system can be much efficient for increasing prediction accuracy.

In this paper, a hybrid approach has been proposed that combines collaborative filtering and content-based filtering for improving prediction accuracy and valid recommendations. Moreover, our proposed system uses both the model-based and memory-based features of collaborative filtering for clustering the user similarity and user-item rating matrix, respectively. The algorithm we have proposed for our system is a clustering algorithm named affinity propagation [32]. Affinity propagation algorithm produces clusters in a more efficient and accurate manner and hence provides better recommendations as compared to the baseline clustering algorithms. The main contributions of this work are as follows:

(i) The use of affinity propagation algorithm that is effective in case of sparse datasets.

(ii) A hybrid recommendation technique to improve the prediction and recommendation accuracy.

(iii) While utilizing the features of CF techniques, unlike classical CF recommenders, we combine the density of a cluster with the similarity measure to select a range of clusters for the generation of recommendations. This way of picking clusters gives the active user a decent set of auxiliary recommendations.

The remainder of this paper is organized as follows. In Section 2, some of the related works are presented. Section 3 briefly describes the affinity propagation technique. Section 4 is reserved for the proposed system. In Section 5, we have explained the experimental setup and findings of our proposed APHPRS. Finally, in Section 6, the paper is concluded.

2. Related Work

Recommender systems focus on presenting the most relevant information to the users. In literature, different recommender systems are available that are developed for various application domains. These recommender systems are based on different filtering techniques and utilize various recommendation methods. Bilal and Hamad in [33] proposed a recommendation system for mobile application development. The system was designed to help the mobile application developers by recommending attractive designs and artifacts. They use the collaborative filtering technique by proposing a unique measurement that takes into account both similarity and trust information to demonstrate prediction accuracy. In another work, an ontology-based recommender system was proposed for advertisement on social sites [34]. They applied shared ontology for representing advertisement as well as the interest of users. The use of ontology in recommender systems is one of the new trends in recommender system research. A system for improving the recommendation accuracy of any recommender system

was proposed in [35]. They had presented an approach named Dual Training Error based Correction (DTEC) for the improvement of recommendation performance.

Clustering is one of the techniques that can be utilized for accurate prediction in a recommender system. Clustering can be defined as the group of a certain set of objects based on their characteristics, aggregated according to their similarities. Clustering algorithms have been shown to surpass similarity metrics in terms of locating users that are similar to the target user. Clustering techniques also contribute to the resolving of data sparsity and high dimensionality issues [36]. Pham et al. [37] used the hierarchical clustering algorithm [38] for clustering users based on the social information of users and then used traditional collaborative filtering for rating predictions. A movie recommender system using the performance comparison of seven different clustering algorithms was proposed in [39]. They had optimized the k value of different clustering techniques for the movie recommender system. Moreover, they had applied social network analysis for verification of the recommendation quality of their proposed system. Shindi [40] proposed the centering-bunching based clustering (CBBC) algorithm for recommender systems, that is, "hybrid personalized recommender system using centering-bunching based clustering algorithm (CBBCHPRS)." CBBC algorithm is used to cluster the rating matrix. This algorithm is better than traditional k-mean and k-medoid in which centroids are initially calculated appropriately, resulting in the suitable formation of clusters; however, the technique still needs the number of clusters to be prespecified as input. Similarity measures are then used to choose the most suitable cluster/s for the generation of rating predictions. Bedi et al. [41] used an ant colony-based clustering algorithm to cluster the user-item rating matrix, and then the most similar cluster/s to the active user is/are selected for the generation of recommendations. Shindi [31] proposed a fast k-medoid algorithm to cluster the rating matrix and select the cluster that is most related to the active user for the generation of rating predictions. In this work, we propose a machine learning algorithm known as affinity propagation [32] for our proposed hybrid (of model-based and memory-based) personalized recommender system, which can automatically find the appropriate clusters in a given data set without the need for clusters number to be prespecified. The details about affinity propagation are presented in section 3.

3. Affinity Propagation

Affinity propagation is a novel semisupervised machine learning algorithm that is used for clustering and is referred to as "clustering by passing messages between data points" [32]. It has been observed that affinity propagation can locate relevant information clusters with much fewer errors and in a short time compared with the other available techniques. The central point of a cluster, which is itself a data point, is called an exemplary. A common approach for clustering is learning a group of exemplars so that the total squared differences among data points with their closest exemplars can be minimized. Affinity propagation considers every data point to be equally probable for being selected as an exemplar and take it as a node in the network of message passing system. This approach takes as input similarities between data points. Based on the input similarities, the transfer of messages between data points took place, and eventually, an adequate number of clusters with their corresponding exemplars are generated. In contrast to other techniques, affinity propagation automatically generates the number of clusters and never requires the number of clusters to be specified in advance. The similarity between two points $s(i, k)$ indicates how similar two points are to each other by calculating a negative Euclidean distance between them:

$$s(i,k) = -\| Ui - Uk\|^2, \quad (1)$$

where $i \neq k$.

Affinity propagation takes a numbered value $s(k, k)$ as a similarity between each data point k, and the points that have a higher value of $s(k, k)$ are more chances to become an exemplar. This numbered value is known as "preferences." The number of clusters emerges according to the input preferences but also arises during the message-exchange process. Since affinity propagation assumes every data point as an exemplar initially, the preferences are kept as a common value. This preference may be set to the minimum of all input similarities that will produce clusters less in number, or it can be set to the median of all input similarities that will produce a moderate number of clusters.

Moreover, the messages that are to be sent between the data points are categorized into two types: responsibility and availability [32]. The responsibility messages (i, k) are those messages that are sent from data point i to the data point k representing how well it is appropriate for data point k to be the exemplar for data point i in consideration of other probable exemplars for a point I (Figure 1(a)), while the availability messages $a(i, k)$ are the messages that are sent from data point k to the data point i representing how well it is suitable for point i to pick point k as it is exemplary considering the support from other points that point k should be an exemplar (Figure 1(b)). At any point in time, the two messages can be consolidated for determining the exemplars. Affinity adds a value $\lambda \in [0, 1]$ called damping factor in message passing to periodic variations of in certain circumstances. The process of message passing between data points is ended when the exemplar decision does not change for a certain number of iterations, usually 10.

Figure 2 shows the message-exchange process of affinity propagation. (A) Sending responsibilities $r(i, k)$ are the messages sent from data points to candidate exemplars to specify the favoring value of each data point for candidate exemplar compared to other exemplars. (B) Sending availabilities $a(i, k)$ are those messages that are sent from candidate exemplars to data points indicating the degree by which candidate exemplar can be selected as a cluster center for the data point.

In the initial iteration of affinity propagation, the availabilities are initialized as zero, that is, $a(i, k) = 0$. The responsibilities in this step are then calculated by the rule (equation (2)). In succeeding iterations, while few data

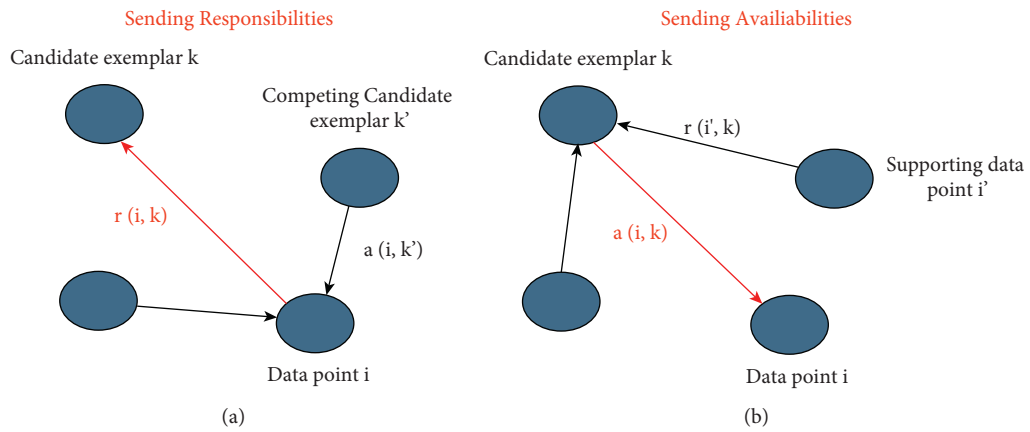

Figure 1: Message-exchange process of affinity propagation [32].

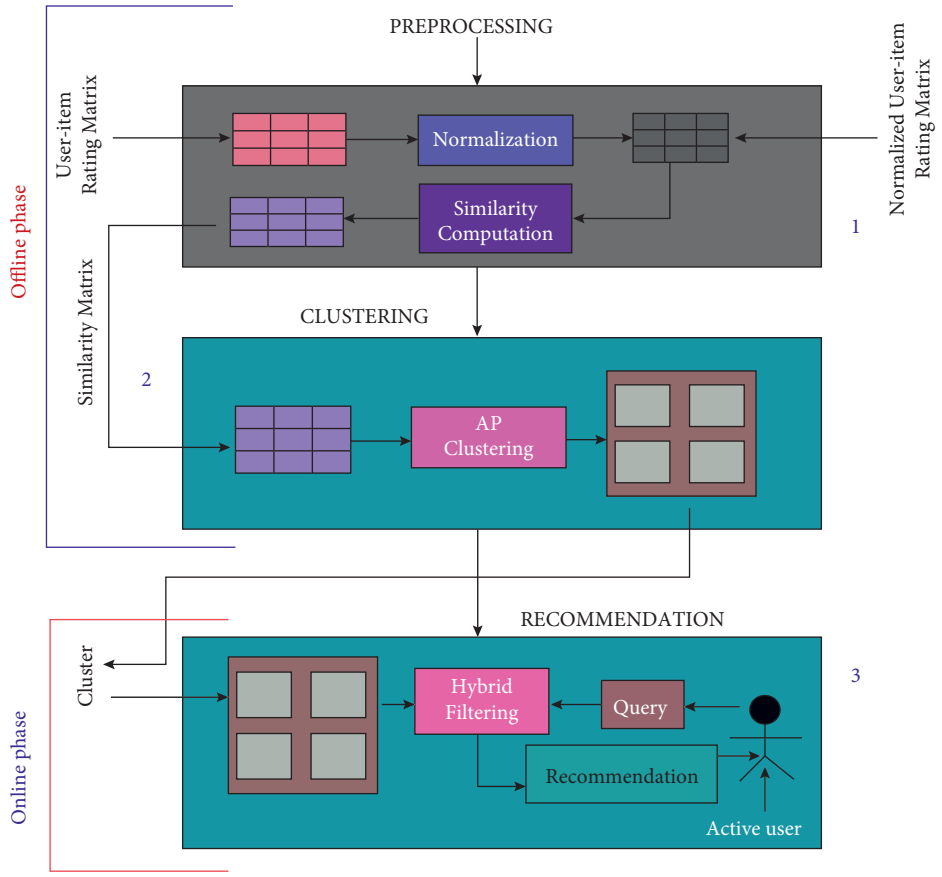

Figure 2: Affinity Propagation-Based Hybrid Personalized Recommender System.

points are efficiently associated with other exemplars, their availabilities decrease lower than zero as approved using the availability update rule (equation (3)). The found negative availabilities brought much decrease in the similarity value provided as input $s(i, k')$ in responsibility update rule and detached the already supportive candidate exemplars. When the value is $k = i$, the value for responsibility $r(k, k)$ is initialized as input preference that point k is selected as an exemplar, $s(k, k)$, minus the largest of the similarities between point i and all other candidate exemplars. The found "self-responsibility" proves point k as an exemplar and is based on the input preference mitigated by how poorly it is to be allocated to a different exemplar [32].

$$r(i,k) = s(i,k) - \{a(i,k') + s(i,k')\}, \quad (2)$$

where $r(i, k)$ represents the update rule for responsibility, $s(i, k)$ represents the input similarity of data point i to its exemplar k, and $\max_{k' \text{ s.t. } k' \neq k} a(i,k') + s(i,k')$ is the maximum similarity of point i with other exemplars.

$$a(i,k) = \{0, r(k,k)+\}. \quad (3)$$

The above rule of availability is used to collect information of data points as if the candidate exemplar might be helpful in making a good exemplar. The availability $a(i, k)$ is set to the self-responsibility $r(k, k)$ plus the sum of the positive responsibilities candidate exemplar k receives from other points. The proposed affinity propagation algorithm has a drawback of high computational cost. Its computational complexity is of the order $O(n^2t)$, where "n" represents the total number of the data points and "t" is the number of all iterations until final clusters are made. Moreover, the proposed algorithm may fail to produce effective results when the similarity matrix is not generated well.

4. The Proposed Affinity Propagation-Based Hybrid Personalized Recommender System

The proposed Affinity Propagation-Based Hybrid Personalized Recommender System (APHPRS) combines the features of content-based methods and collaborative filtering methods for handling the issues like sparsity and cold-start problems. The proposed method works in two phases (Figure 2). In the offline phase, it performs the preprocessing on data. In this phase, a rating matrix is generated and normalized, and then a pairwise similarity matrix for users is generated of the normalized matrix of user-item ratings. The similarity matrix is then loaded into an affinity propagation algorithm for clustering. Once the clusters are formed, they are kept in some database file so that the clusters can be used for generating recommendations in future.

The second phase (phase 2) is about generating recommendations for the active user. In this phase, the similarity metric and the number of user preferences are combined in a specific cluster for finding suitable clusters for the generation of recommendations. Moreover, the quality of item ratings in each cluster is also recorded. Based on this measure, suitable clusters get selected out of the list of chosen clusters for the rating predictions. After this step, the recommendations are generated using the weighted average of item ratings in the chosen clusters. Unlike classical collaborative recommenders, we combine the density of a cluster with the similarity measure to select a range of clusters for the generation of recommendations. This way of picking clusters gives the active user a decent set of auxiliary recommendations. The results are then further refined by choosing cluster/s having maximum quality ratings. The detailed procedure and working of the proposed APHPRS are described in the following by taking the example of the Jester dataset.

Figure 2 represents the proposed APHPRS system. (1) The preprocessing step: user-item rating matric is taken from the Jester dataset and normalized. After normalization, a pairwise similarity matrix between users is calculated. (2) The clustering step: it is the step where the found similarity matrix is provided to the AP algorithm for grouping those users who have similar ratings. (3) The recommendation step: in this step, the recommendations take place for active users. Here, the similarity between active users and clusters is computed to find the best clusters for generating recommendations. The rating quality of each item not rated by an active user is calculated in the selected clusters. To generate the recommendations, clusters are further selected based on the rating quality of an item. Note that step 1 and step 2 are the parts of the offline phase, while step 3 is the online phase of our proposed system.

4.1. Offline Phase. The proposed APHPRS starts with an offline phase that consists of two different steps: preprocessing and clustering. The steps in this phase are done in offline mode for faster execution of recommendations and to reduce running time. As the proposed algorithm has a little high computational cost (i.e., $O(n^2t)$), the calculation in the offline phase prevents the system from slow recommendations. In this step, the data in the form of user-item ratings is collected and clustered using the proposed algorithm. Moreover, a similarity matrix is also obtained for the user ratings, which is then used as an input for the proposed affinity propagation algorithm. Here, in this phase, we do not need to have online processing as all of its steps are done without connection with an active user.

4.1.1. Preprocessing. In this step, we took Jester dataset as input and normalized it for future processing of our system. We then compute the similarity between different entries available in the Jester dataset and fetch the similarity to the next step. The details of preprocessing are presented in the succeeding steps:

(1) Jester Dataset. The Jester dataset is an online dataset for the Web-based Joke Recommender System. In the Jester dataset, each row represents a different user, while each column (except the first column) shows the rated score given by a particular user. The first column gives a specific number of jokes rated by a particular user. The remaining columns give the ratings for different jokes. The user-item rating matrix collected from Jester data consisted of item ratings on a scale of −10 to 10, and in any cell, the value 99 represents null or no rating of the item (joke). Table 1 shows a sample of the Jester dataset that we have taken only 10 entries to keep the table readable to the reader.

(2) Normalization. In this step, we have taken the Jester dataset (Table 1) as input and removed the first column. Table 1 represents a sample of the Jester dataset, where the first column shows the number of jokes rated by a particular user while the rest of the columns show the rated values of each joke. Note that we have picked only 10 jokes in the below table. We then normalized the ratings to the scale from 0 to 1, where 0 represents null or no rating for a particular joke (Table 2). It is noteworthy that we have kept the positive and null ratings only as the recommendations will be made on the basis of these two factors. Unlike Jester dataset, we have removed the negative ratings in the normalized rating matrix. The normalized values are calculated using

Table 1: A sample taken from Jester dataset before normalization.

Number of jokes rated	J1	J2	J3	J4	J5	J6	J7	J8	J9	J10
74	−7.82	8.79	−9.66	−8.16	−7.52	−8.5	−9.85	4.17	−8.98	−4.76
100	4.08	−0.29	6.36	4.37	−2.38	−9.66	−0.73	−5.34	8.88	9.22
49	99	99	99	99	9.03	9.27	9.03	9.27	99	99
48	99	8.35	99	99	1.8	8.16	−2.82	6.21	99	1.84
91	8.5	4.61	−4.17	−5.39	1.36	1.6	7.04	4.61	−0.44	5.73
100	−6.17	−3.54	0.44	−8.5	−7.09	−4.32	−8.69	−0.87	−6.65	−1.8
47	99	99	99	99	8.59	−9.85	7.72	8.79	99	99
100	6.84	3.16	9.17	−6.21	−8.16	−1.7	9.27	1.41	−5.19	−4.42
100	−3.79	−3.54	−9.42	−6.89	−8.74	−0.29	−5.29	−8.93	−7.86	−1.6
72	3.01	5.15	5.15	3.01	6.41	5.15	8.93	2.52	3.01	8.16

Table 2: A sample of normalized rating matrix in the scale of 0 to 1 taken from Jester dataset.

Users	J1	J2	J3	J4	J5	J6	J7	J8	J9	J10
U1	0.109	0.9395	0.017	0.092	0.124	0.075	0.0075	0.7085	0.051	0.262
U2	0.704	0.4855	0.818	0.7185	0.381	0.017	0.4635	0.233	0.944	0.961
U3	0	0	0	0	0.9515	0.9635	0.9515	0.9635	0	0
U4	0	0.9175	0	0	0.59	0.908	0.359	0.8105	0	0.592
U5	0.925	0.7305	0.2915	0.2305	0.568	0.58	0.852	0.7305	0.478	0.7865
U6	0.1915	0.323	0.522	0.075	0.1455	0.284	0.0655	0.4565	0.1675	0.41
U7	0	0	0	0	0.9295	0.0075	0.886	0.9395	0	0
U8	0.842	0.658	0.9585	0.1895	0.092	0.415	0.9635	0.5705	0.2405	0.279
U9	0.3105	0.323	0.029	0.1555	0.063	0.4855	0.2355	0.0535	0.107	0.42
U10	0.6505	0.7575	0.7575	0.6505	0.8205	0.7575	0.9465	0.626	0.6505	0.908
Active user	0.864	0.7695	0	0	0.289	0	0.9465	0.675	0	0.806

0 indicates item being not rated

$$n_i = \frac{x_i - \min(x)}{\max(x) - \min(x)}, \tag{4}$$

where n_i is the value being normalized, x_i is the value before normalization, $\min(x)$ is the lowest value, and $\max(x)$ is the highest-rated value. The calculated values for each rating are given in Table 2. Table 2 represents the normalized form of Jester data, where the first column has been removed, and the ratings are converted on a scale between 0 and 1. Note that the negative ratings have been skipped, as the recommendation will be made on positive ratings.

(3) Similarity Matrix. Affinity propagation needs a pairwise matrix of similarity values to be fed as input. We have obtained the similarity among users according to their rating for items (Table 3). We have used the negative square error or Euclidean distance for calculating the similarity. The similarity matrix can be found using

$$s(i,k) = -\|Ui - Uk\|^2, \tag{5}$$

where i and k can be any two users and $i \neq k$.

The similarity matrix constructed for the normalized rating matrix has 10 users; hence, the similarity matrix will have $n(n-1) = 10(9)$ entries that will lead us to create a table having 90 entries. For the sake of brevity, a short version of that matrix is reproduced here (Table 3). Here, columns 1 and 2 represent indices of the two users, while the column represents the negative Euclidean distance between the corresponding pair of users.

Table 3: A sample similarity matrix.

U_i	U_k	$s(i, k)$
1	2	−3.3837435
1	3	−3.4048888
1	4	−1.1776203
2	1	−3.3837435
2	3	−5.7244563
2	4	−4.0768918
3	1	−3.4048888
3	2	−5.7244563
3	4	−1.700498
4	1	−1.1776203
4	2	−4.0768918
4	3	−1.700498

4.1.2. Clustering. We used affinity propagation as a clustering algorithm for grouping items based on their similarity. The default damping factor of λ is initialized as 0.5; see details about the damping factor in Section 2. The used clustering algorithms have generated three clusters of the given similarity matrix.

We have chosen affinity propagation as in this clustering algorithm, and we do not need the number of clusters to be prespecified. Moreover, the proposed algorithm is effective in the case of sparse data, and it takes advantages of data sparsity when similarities are well-conducted [42]. The details of the found clusters are shown in Table 4.

Table 4 represents the clusters found using affinity propagation. The first column shows three different clusters,

Table 4: Users in each cluster with exemplar.

Cluster	User	Exemplar
1	3, 7	3
2	1, 4, 6, 9	6
3	2, 5, 8, 10	10

while the second column shows the number of users in each cluster. In the third column, the central point (exemplar) of each cluster is recorded.

4.2. Online Phase. In this phase, predictions and recommendations are made for the active user. The steps of this phase are done online, as the active user is considered while processing each step. A detailed description of the online phase is given in the next sections.

4.2.1. Selecting the Best Cluster(s). Selecting the best cluster(s) for the generation of recommendations is dependent on two aspects: (1) the number of users in a cluster and (2) the similarity of that cluster with the active user. The match score of a certain cluster is obtained by

$$\text{Match}_{\text{score}(i)} = \frac{\rho(i) \cdot sm(i)}{\sum_{i=1}^{n} \rho(i) \cdot sm(i)}, \tag{6}$$

where $sm(i)$ is the similarity between the exemplar of i_{th} cluster and the active user, $\rho(i)$ is the density of the i_{th} cluster, and n represents the complete set of clusters formed.

A similarity measure is used to find the cluster of users having preferences that match the most with the profile of the active user. There are numeral different measures used for calculating similarities like Pearson correlation, Euclidean distance measure, and vector similarity measure. We have used Euclidean distance measure to find the similarity between the profile of an active user and cluster. The Euclidean distance can be calculated using equation (7):

$$D(i) = \left\{ \sum_{j=1}^{d} \left| \exp_{i,j} - u_j \right|^2 \right\}^{1/2}, \tag{7}$$

where d is the total number of items or attributes of a user, $\exp_{i,j}$ i_{th} attribute of the exemplar of cluster j, and u_j is the j_{th} user.

Hence, the similarity of the i_{th} cluster with the active user is calculated as follows:

$$sm(i) = \frac{1}{D(i)}. \tag{8}$$

The density of the cluster can be calculated using the following equation:

$$\rho(i) = \frac{\text{population of users in cluster } i}{\text{total population of users}}. \tag{9}$$

The density of a cluster increases with the increase of users in a cluster and vice versa. Those clusters whose match score lies within the range (highest score $- \alpha \leq$ match_score $\leq$ highest score) are selected for recommendations generation. In this paper, $\alpha = 0.2$. By using this range, unlike collaborative filtering, not only the clusters having the highest score but also the other clusters having their scores slightly less than the maximum one are selected. The match score, density ρ, and similarity values calculated are shown in Table 5. Table 5 represents the calculated similarity $sm(j)$ with an active user, the density $\rho(i)$ of each cluster, and the matching score between clusters, where the first column shows the functions for different clusters, while the rest of the column shows the corresponding function measure values for each cluster.

The clusters selected are 2 and 3 because their match score lies within the range $0.4311 - 0.2 \leq$ match score ≤ 0.4311.

4.2.2. Calculation of Rating Quality. Rating quality any item explains how similar the ratings of different users are in any specific cluster. The rating quality can be found using

$$\text{Qty} = \frac{\left(\text{ub}_{\text{rating}} + \text{mean}_{\text{rating}} \right)}{2 * \text{ub}_{\text{rating}}}, \tag{10}$$

where $\text{ub}_{\text{rating}}$ is the maximum rate score of an item and $\text{mean}_{\text{rating}}$ represents the mean value of item ratings in any cluster. If $\text{ub}_{\text{rating}}$ and mean $_{\text{rating}}$ values are equal, we get quality rating Qty as 1 that represents good quality. Hence, a greater value of Qty will lead to higher rating quality and vice versa. The calculated rating quality of different unrated jokes is shown in Table 6. Table 6 shows the rating quality of unrated jokes in different clusters, where the first column represents the active user unrated jokes, while the second and third columns show the corresponding quality ratings, that is, clusters 2 and 3, respectively.

4.2.3. Prediction of New Item Ratings. After finding the quality of each unrated item on the basis of rating quality, the clusters are then selected from the initially chosen clusters (Table 7). Instead of selecting the cluster with the quality for an unrated item, those clusters where the quality "Qty" of each unrated item is in the range [17] are additionally retrieved from the cluster set that was initially chosen for the generation of rating predictions. In this paper, $\alpha = 0.1$. Table 7 shows the ratings predicted for unrated jokes. If only one cluster is selected, then the mean rating of the item in the selected cluster is calculated; otherwise, the rating is calculated as the weighted average. For jokes 4, 6, and 9, the rating is calculated as a weighted average of ratings in clusters 2 and 3, while for joke 3, the average rating is calculated from cluster 3.

Item's rating is then predicted using

$$\text{Rating} = \frac{\sum_{i=1}^{k} \left(\text{Qty}_i * \text{mean}_{\text{rating}} \right)}{\sum_{i=1}^{k} \text{qty}_i}, \tag{11}$$

where Qty_i represents the item's quality in the chosen cluster, $\text{mean}_{\text{rating}}$ is the rating of the item in any selected cluster, and k represents the number of clusters being selected.

Table 5: Cluster selecting procedure.

Function	Cluster 1	Cluster 2	Cluster 3
$sm(i)$	0.5393	0.7002	0.6544
$\rho(i)$	0.2	0.4	0.4
Match score(i)	0.166	0.4311	0.4029

Table 6: Rating quality of unrated jokes in selected clusters.

Unrated jokes by an active user	Quality rating in cluster 2	Quality rating in cluster 3
J3	0.636	0.8685
J4	0.7592	0.8112
J6	0.7413	0.792
J9	0.7429	0.8063

Table 7: Ratings predicted for the active user.

Jokes	Clusters selected	Predicted ratings
J3	3	0.7064
J4	2, 3	0.27
J6	2, 3	0.4403
J9	2,3	0.34

Table 8: Clustering performance on IRIS dataset by the proposed algorithm and baseline algorithms.

IRIS classes	Clustering algorithms			
	k-means	k-mediods	CBBC	Affinity propagation
Setosa	50	50	50	50
Versicolor	34	41	44	47
Virginica	66	59	56	53

4.2.4. Generation of Top-N Recommendations. Once the item ratings are predicted, the next step is to provide the active user with top-N suggestions. For example, if $N = 1$, joke 3 will be suggested. For $N = 2$, jokes 3 and 6 will be suggested, and so on.

5. Experimental Setup

5.1. Clustering Performance Evaluation. The performance of the proposed clustering algorithm was evaluated using the IRIS dataset that is available in the UCI repository (https://archive.ics.uci.edu/ml/datasets/Iris). The IRIS dataset contains a total of 150 objects divided into three main classes, where each class has 50 objects. The classes are labelled as "Setosa," "Versicolor," and "Virginica." The performance of clusters was evaluated in terms of the accuracy of grouping each and every object according to the true classes. In the experiments, we compared the proposed algorithm with many baseline algorithms used for clustering. The proposed affinity propagation algorithm outperforms the existing algorithm as it clusters the object more accurately (see Table 8). The accuracy has been improved due to the message sending process of the proposed algorithm. Moreover, the proposed algorithm does not need the cluster number to be prespecified, and it automatically generates the cluster centroids (exemplars).

Table 8 shows the evaluation of the proposed clustering algorithm on the IRIS dataset. In this table, the first column represents the various classes of the IRIS dataset and the second column holds the different algorithms and their clustering results. This table clearly reveals that the affinity propagation algorithm generates the clusters with more accuracy.

5.2. Performance Evaluation of the Proposed Recommender System. APHPRS has been implemented in MATLAB version 7.8. The experiment was conducted on the Jester dataset. We also implemented another collaborative filtering-based recommender system [43] manually for performance comparison, using R-precision and MAE metrics. To evaluate the performance, we have used two different sizes of datasets. Dataset 1 is a small subset of Jester data that consists of a user-item rating matrix having 10 users and 10 jokes (Table 2). Dataset 2 is a large subset of the Jester dataset consisting of 70,000 users and 100 jokes. The proposed APHPRS was evaluated by checking its prediction quality and recommendation quality.

Prediction quality is tested by relating the predicted user rating to the test set of actual user ratings. For the evaluation of predictive accuracy, Mean Absolute Error (MAE) is used as a universal metric [44]. Hence, we used the MAE metric for evaluating the prediction quality of the proposed APHPRS. If we represent the set of rating prediction for items as $p_1, p_2, p3, \ldots, p_N$, and represent the set of corresponding actual user ratings as $q_1, q2, q3, \ldots, q_N$, then MAE can be computed using

$$\text{MAE} = \frac{\sum_{i=1}^{N} \left| p_i - \text{q}_\text{i} \right|}{N}. \tag{12}$$

The low MAE value will lead to greater quality of a recommender system, that is, smaller values of MAE, indicating that rating predictions are much similar to real ratings. For the experiment on dataset 1, we have randomly selected 7 different users from the Jester dataset and gradually increased the size of the test set from 1, 2, 3, 4, 5, 6, and 7 (Figure 3). We have compared our proposed APHPRS

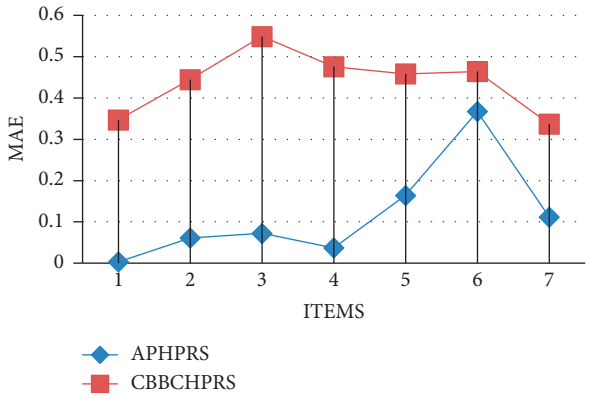

Figure 3: MAE for the two recommender systems for dataset 1.

system with the existing hybrid personalized recommender system using a centering-bunching-based clustering algorithm (CBBCHPRS) [40]. The proposed algorithm and the CBBCHPRS both use the same number of users of the Jester dataset.

Figure 3 represents the MAE calculated for our proposed APHPRS in comparison with CBBCHPRS. The result shows that our proposed system has a smaller MAE that will generate better recommendations as compared to other recommender systems. In the figure, the horizontal numbers show seven different items of the Jester dataset, while the vertical numbers show the corresponding MAE score of both systems for the selected items.

The MAE curves of our algorithm in Figure 3 are superior as compared to the other algorithm, which shows that our algorithm generates quality predictions for item ratings. For recommendation quality, the most widely used metrics are precision and recall. Precision is the recommended items that are relevant out of the total number of recommended items. A recall is the number of relevant recommended items out of the total number of relevant items. These two metrics are problematic in a case when at a given cutoff point, for example, precision@N, the number of relevant recommendations is less than Said et al. [43] have proposed a metric for recommendation quality, called R-precision. It has been shown empirically that R-precision reflects more accurately the quality of recommendation compared to the traditionally used metrics. Hence, we used R-precision for the evaluation of the recommendation quality of our recommender system. R-precision is computed as

$$R-\text{precision} = \frac{r}{R}, \tag{13}$$

where R shows the total number of relevant recommendations, while r is the number of retrieved recommendations that are relevant. Here, R is used as a cutoff point and varies from query to query. For the experiment on dataset 1, we randomly selected 6 different users from the Jester dataset and generated top 1, 2, 3, 4, 5, and 6 recommendations. The results obtained are shown in Figure 4.

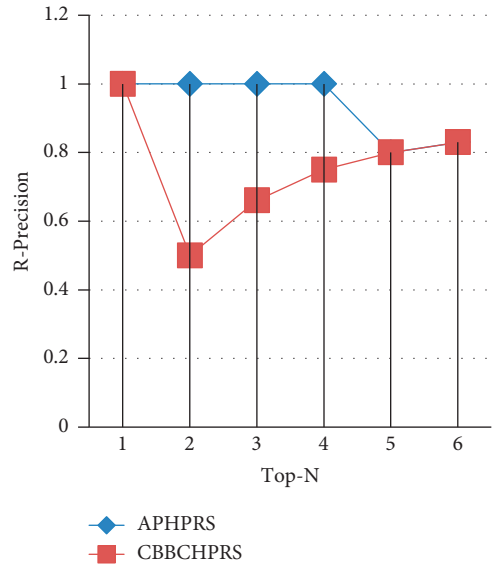

Figure 4: R-precision for the two recommender systems for dataset 1.

Figure 4 shows R-precision generated for the top-N recommendations. The results were compared with CBBCHPRS, which shows that our proposed APHPRS has better performance in comparison with other systems.

Another experiment was performed on dataset 2, where we randomly selected 10 different users from the Jester dataset and gradually increased the size of the test set from 3, 6, 9, 12, 15, 18, 21, 24, 25, and 27. The results obtained are shown in Figure 4.

A second experiment has been conducted using a different dataset (dataset 2) taken from Jester data. For this experiment, we randomly selected 10 active users from the Jester dataset and generated the top 5, 10, and 15 recommendations. The results of the second experiment were compared with Bat Algorithm (BA) used in [45]. The results obtained show that our proposed recommender system produces high R-precision as compared to the existing BA.

Figure 5 represents the MAE calculated for our proposed APHPRS in comparison with the BA algorithm using dataset 2. The results show that our proposed system has a smaller MAE in most cases as compared to other systems. In this figure, the MAE values of the ten users were summed up for each top-N recommendation.

A second experiment has been conducted using dataset 2 taken from Jester data. For this experiment, we randomly selected 10 different users from the Jester dataset and generated the top 5, 10, and 15 recommendations. The experiment was conducted to find the R-precision values. The results obtained show that our proposed recommender system produces high R-precision as compared to the existing BA (Figure 6).

It can be seen in Figure 6 that the R-precision values of our system are higher than those of the other system, which indicates the quality of our recommendations.

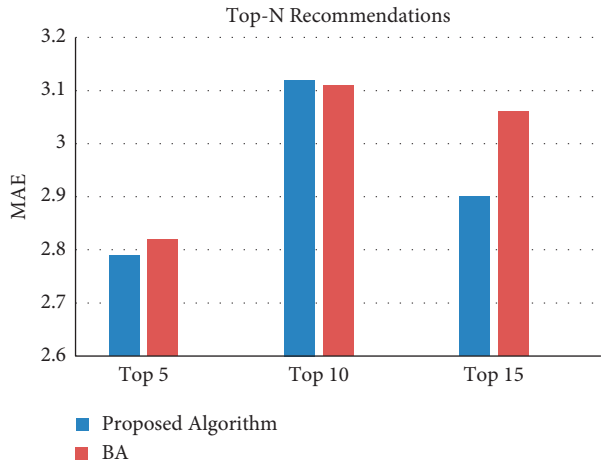

Figure 5: MAE for the two recommender systems for dataset 2.

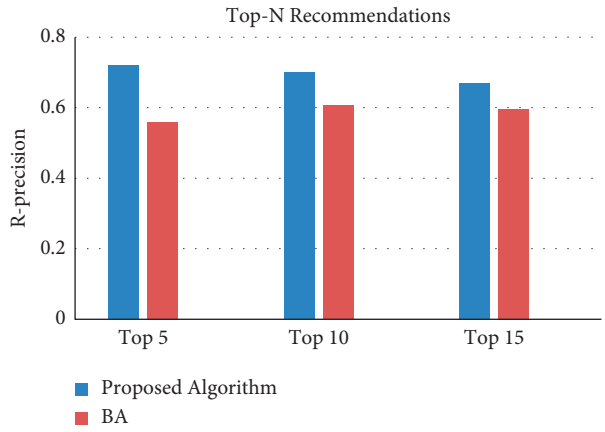

Figure 6: *R*-precision for the two recommender systems for dataset 2.

6. Conclusion

Recommender systems are the tools and techniques used for handling a load of information by filtering a large amount of information and then suggesting a piece of information relevant to the user. This paper proposed APHPRS, an Affinity Propagation-Based Hybrid Personalized Recommender System collecting user preferences of the matrix of items rated by different users, clustered it, and generated recommendations for the user. The clustering method used in this work outperforms other baseline methods as tested on IRIS data; however, this method fails to produce effective results when the similarity matrix is not calculated well. We have combined the features of content-based approaches and collaborative filtering approaches for making the system hybrid. The previous work done in this domain suggests that only similarity score was considered by the existing recommender systems, whereas we incorporated the cluster's quality and density and combined both with the similarity score for selecting a range of quality clusters for recommendations. This helps in discovering the additional clusters which have similarities in accordance with user preference. Evaluation of our proposed APHPRS was made using the Jester dataset. Furthermore, the APHPRS is compared with another hybrid system CBBCHPRS. The performance of our system was evaluated in terms of prediction quality and recommendation quality. The prediction quality was calculated using the MAE score, while recommendation quality was measured in terms of *R*-precision. The slight decrease in the MAE score is evidence that sparsity has less impact on our proposed system. Similarly, the *R*-precision of our proposed system is much higher as compared to the existing systems. The increase in *R*-precision shows that our proposed approach has better results in recommending items.

In future work, we will use the Seed Affinity Propagation (SAP) algorithm that is an advanced version of affinity propagation for finding clusters. We will also consider ontology as a knowledge base for improving prediction accuracy in the future.

Data Availability

The data supporting this research are from previously reported studies and datasets, which have been cited. The processed data are available at (https://goldberg.berkeley.edu/jester-data/).

Conflicts of Interest

The authors declare no potential conflicts of interest.

Acknowledgments

The authors are grateful to the Deanship of Scientific Research, King Saud University, for funding through the Vice Deanship of Scientific Research Chairs.

References

[1] F. Ricci, L. Rokach, and B. Shapira, "Recommender systems: introduction and challenges," in *Recommender Systems Handbook*, pp. 1–34, Springer, Berlin, Germany, 2015.

[2] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a feather: homophily in social networks," *Annual Review of Sociology*, vol. 27, no. 1, pp. 415–444, 2001.

[3] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734–749, 2005.

[4] A. S. Lampropoulos, G. A. Tsihrintzis, Tsihrintzis, and A. George, "A survey of approaches to designing recommender systems," in *Multimedia Services in Intelligent Environments*, pp. 7–30, Springer, Berlin, Germany, 2013.

[5] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," *Communications of the ACM*, vol. 35, no. 12, pp. 61–70, 1992.

[6] W. Carrer-Neto, M. L. Hernández-Alcaraz, R. Valencia-García, F. García-Sánchez, and F. García-Sánchez, "Social knowledge-based recommender system. Application to the movies domain," *Expert Systems with Applications*, vol. 39, no. 12, pp. 10990–11000, 2012.

[7] H.-C. Chen and A. L. P. Chen, "A music recommendation system based on music data grouping and user interests," in *Proceedings of the Tenth International Conference on*

Information and Knowledge Management, pp. 231–238, Atlanta, GA, USA, October 2001.

[8] A. Nanopoulos, D. Rafailidis, P. Symeonidis, and Y. Manolopoulos, "Musicbox: personalized music recommendation based on cubic analysis of social tags," *IEEE Transactions on Audio Speech and Language Processing*, vol. 18, pp. 407–412, 2009.

[9] Z. Yu, X. Zhou, Y. Hao, and J. Gu, "TV program recommendation for multiple viewers based on user profile merging," *User Modeling and User-Adapted Interaction*, vol. 16, no. 1, pp. 63–82, 2006.

[10] R. G. Crespo, O. S. Martínez, J. M. C. Lovelle, B. C. P. García-Bustelo, J. E. L. Gayo, and P. O. D. Pablos, "Recommendation system based on user interaction data applied to intelligent electronic books," *Computers in Human Behavior*, vol. 27, no. 4, pp. 1445–1449, 2011.

[11] C. Porcel and E. Herrera-Viedma, "Dealing with incomplete information in a fuzzy linguistic recommender system to disseminate information in university digital libraries," *Knowledge-Based Systems*, vol. 23, no. 1, pp. 32–39, 2010.

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

[13] R. Katarya and O. P. Verma, "Recommender system with grey wolf optimizer and FCM," *Neural Computing & Applications*, vol. 30, no. 5, pp. 1679–1687, 2018.

[14] E. Costa-Montenegro, A. B. Barragáns-Martínez, and M. Rey-López, "Which App? A recommender system of applications in markets: implementation of the service for monitoring users' interaction," *Expert Systems with Applications*, vol. 39, no. 10, pp. 9367–9375, 2012.

[15] K. McNally, M. P. O'Mahony, M. Coyle, P. Briggs, and B. Smyth, "A case study of collaboration and reputation in social web search," *ACM Transactions on Intelligent Systems and Technology*, vol. 3, no. 1, pp. 1–29, 2011.

[16] U. Liji, Y. Chai, and J. Chen, "Improved personalized recommendation based on user attributes clustering and score matrix filling," *Computer Standards & Interfaces*, vol. 57, pp. 59–67, 2018.

[17] R. Forsati and M. R. Meybodi, "Effective page recommendation algorithms based on distributed learning automata and weighted association rules," *Expert Systems with Applications*, vol. 37, no. 2, pp. 1316–1330, 2010.

[18] N. J. Belkin and W. B. Croft, "Information filtering and information retrieval," *Communications of the ACM*, vol. 35, no. 12, pp. 29–38, 1992.

[19] K. Lang, "Newsweeder: learning to filter netnews," in *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 331–339, Elsevier, Tahoe, CA, USA, July 1995.

[20] B. Krulwich and C. Burkey, "Learning user information interests through extraction of semantically significant phrases," in *Proceedings of the AAAI spring Symposium on Machine Learning in Information Access*, p. 110, Palo Alto, CA, USA, March 1996.

[21] D. Billsus and M. J. Pazzani, "A hybrid user model for news story classification," in *Proceedings of the CISM International Centre for Mechanical Sciences Um99 User Modeling*, pp. 99–108, Banff, Canada, June 1999.

[22] Y. Shi, M. Larson, and A. Hanjalic, "Collaborative filtering beyond the user-item matrix," *ACM Computing Surveys*, vol. 47, no. 1, pp. 1–45, 2014.

[23] L. H. Son, "Dealing with the new user cold-start problem in recommender systems: a comparative review," *Information Systems*, vol. 58, pp. 87–104, 2016.

[24] F. Zhang, J. Cheng, and Z. Ma, "A survey on fuzzy ontologies for the Semantic Web," *The Knowledge Engineering Review*, vol. 31, no. 3, pp. 278–321, 2016.

[25] Xi Chen, X. Liu, Z. Huang, and H. Sun, "Regionknn: a scalable hybrid collaborative filtering algorithm for personalized web service recommendation," in *Proceedings of the 2010 IEEE International Conference on Web Services*, Miami, FL, USA, July2010.

[26] M. R. McLaughlin, Herlocker, and L. Jonathan, "A collaborative filtering algorithm and evaluation metric that accurately model the user experience," in *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 329–336, Sheffield, UK, July 2004.

[27] L. H. Ungar, Foster, and P. Dean, "Clustering methods for collaborative filtering," *AAAI Workshop on Recommendation Systems*, pp. 114–129, 1998.

[28] A. M. Rashid, S. K. Lam, G. Karypis, and J. Riedl, "ClustKNN: a highly scalable hybrid model-& memory-based CF algorithm," in *Proceedings of the webKDD*, Philadelphia, PA, USA, August 2006.

[29] X. Yang, Y. Guo, Y. Liu, and H. Steck, "A survey of collaborative filtering based social recommender systems," *Computer Communications*, vol. 41, pp. 1–10, 2014.

[30] D. H. Park, H. K. Kim, I. Y. Choi, and J. K. Kim, "A literature review and classification of recommender systems research," *Expert Systems with Applications*, vol. 39, no. 11, pp. 10059–10072, 2012.

[31] S. K. Shinde and U. V. Kulkarni, "Hybrid personalized recommender system using fast K-medoids clustering algorithm," *Journal of Advances in Information Technology*, vol. 2, pp. 152–158, 2011.

[32] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, no. 5814, pp. 972–976, 2007.

[33] B. Abu-Salih, H. Alsawalqah, B. Elshqeirat, T. Issa, P. Wongthongtham, and K. K. Premi, "Toward a knowledge-based personalised recommender system for mobile app development," *JUCS - Journal of Universal Computer Science*, vol. 27, no. 2, pp. 208–229, 2021.

[34] F. García-Sánchez, R. Colomo-Palacios, and R. Valencia-García, "A social-semantic recommender system for advertisements," *Information Processing & Management*, vol. 57, no. 2, Article ID 102153, 2020.

[35] C. Panagiotakis, H. Papadakis, A. Papagrigoriou, and P. Fragopoulou, "Improving recommender systems via a dual training error based correction approach," *Expert Systems with Applications*, vol. 183, Article ID 115386, 2021.

[36] M. Ramezani, P. Moradi, and F. Akhlaghian, "A pattern mining approach to enhance the accuracy of collaborative filtering in sparse data domains," *Physica A: Statistical Mechanics and Its Applications*, vol. 408, pp. 72–84, 2014.

[37] M. C. Pham, Y. Cao, R. Klamma, and M. Jarke, "A clustering approach for collaborative filtering recommendation using social network analysis," *Journal of Universal Computer Science*, vol. 17, pp. 583–604, 2011.

[38] A. Clauset, M. E. Newman, and C. Moore, "Finding community structure in very large networks," *Physical review. E, Statistical, Nonlinear, and Soft Matter Physics*, vol. 70, Article ID 066111, 2004.

[39] L. Debby, S. Jenq, and P. Seda, "Design of an unsupervised machine learning-based movie recommender system," *Symmetry*, vol. 12, no. 2, p. 185, 2020.

[40] S. K. Shinde and U. Kulkarni, "Hybrid personalized recommender system using centering-bunching based clustering algorithm," *Expert Systems with Applications*, vol. 39, no. 1, pp. 1381–1387, 2012.

[41] P. Bedi, R. Sharma, and H. Kaur, "Recommender system based on collaborative behavior of ants," *Journal of Artificial Intelligence*, vol. 2, no. 2, pp. 40–55, 2009.

[42] D. Dueck, *Affinity Propagation: Clustering Data by Passing Messages*, Citeseer, Princeton, New Jersey, USA, 2009.

[43] M. K. Najafabadi, M. N. R. Mahrin, S. Chuprat, and H. M. Sarkan, "Improving the accuracy of collaborative filtering recommendations using clustering and association rules mining on implicit data," *Computers in Human Behavior*, vol. 67, pp. 113–128, 2017.

[44] A. H. Nabizadeh, A. Jorge, and J. P. Leal, "Long term goal oriented recommender systems," in *Proceedings of the WEBIST*, Lisbon, Portugal, May 2015.

[45] S. Yadav, Vikesh, Shreyam, and S. Nagpal, "An improved collaborative filtering based recommender system using bat algorithm," *Procedia Computer Science*, vol. 132, pp. 1795–1803, 2018.