Hindawi
Journal of Control Science and Engineering
Volume 2021, Article ID 6669798, 11 pages
https://doi.org/10.1155/2021/6669798

Research Article

A Service Recommendation Method Based on Requirements for the Cloud Environment

Liangmin Guo,[1,2] **Kaixuan Luan**,[1,2] **Xiaoyao Zheng**,[1,2] **and Jing Qian**[1,2]

[1]*School of Computer and Information, Anhui Normal University, Wuhu 241003, China*
[2]*Anhui Provincial Key Laboratory of Network and Information Security, Wuhu 241003, China*

Correspondence should be addressed to Liangmin Guo; glm1999@ahnu.edu.cn

Received 6 October 2020; Accepted 14 March 2021; Published 25 March 2021

Academic Editor: Giulio Ferro

In the cloud computing environment, there are huge amounts of functionally similar cloud services. Additionally, user requirements can change. Therefore, it is difficult to recommend services that meet users' requirements. To overcome the problems, a service recommendation method based on requirements is proposed. First, we form user communities by clustering to reduce the recommended range. Second, we use the reported QoS (Quality of Service) values and the evaluated QoS values to predict the QoS requirements of users. Third, based on the requirements, the matching degree of users to services is obtained. Finally, based on the similarity between the target user and the user's neighbors and the difference in their matching degree of service and the ratings of services by the neighbors, we can obtain a list of service recommendations for the target user. Compared to the traditional collaborative filtering method and the deviation-based method, our method improves the recommendation accuracy without lowering the efficiency.

1. Introduction

The rapid development of Internet technology gave rise to the emergence of cloud computing, which has rapidly developed into one of the hot research fields. Cloud computing has superpowerful computing power and storage capacity. For example, in the mobile Internet, new service models and mobile applications based on mobile cloud computing show rocketing development trends [1]. Cloud computing utilizes the resource pool based on virtualization technology to provide on-demand services (i.e., cloud infrastructure, development platform, and software), which can be paid for dynamically, can be easily extended, and are fast and flexible. Users can get cloud services at any time and any place from cloud platforms. Faced with a vast amount of cloud services with similar or even identical functions, it is difficult for users, especially ordinary users, to choose the services that truly meet their requirements.

Service recommendation technology has been shown to be a valuable tool. The core idea of service recommendation technology is to identify and predict the needs of users and provide corresponding recommendations to users. Existing popular recommendation methods are roughly divided into several categories: content-based recommendation [2, 3], collaborative filtering recommendation [4–12], recommendation based on association rules [13–15], knowledge-based recommendation [16–18], and hybrid recommendation methods [19, 20]. Content-based methods recommend similar items to a user according to the items that the user has liked in the past. These methods can reduce the influence of users' subjective factors and data sparsity on recommendations. Collaborative filtering methods utilize neighbors' known preferences to predict a user's potential preferences for items. These methods rely too heavily on the evaluations of neighbors and have the cold start and data sparsity problems. Recommendation methods based on association rules count the relationships in which different rules appear based on historical data. Knowledge-based methods recommend items to a user in an interactive manner. Hybrid recommendation methods combine multiple methods to overcome the disadvantages of a single method. The above methods usually make recommendations

based on historical circumstances, and it is difficult for these methods to respond to the changing needs of users.

However, in fact, user requirements change over time, for example, a user with no specialist knowledge cannot accurately select services according to his or her requirements in the beginning. As a result, services recommended entirely based on historical records may not be suitable. Therefore, we need to predict the requirements of users to accurately recommend services that meet user requirements. In this paper, a service recommendation method based on requirements is proposed. We combine the evaluated QoS values of the users who are in the same community as a target user, the average rating of services evaluated by the target user, and the QoS value reported by the corresponding provider to predict the QoS required or expected by the target user. That is to say, we use the difference in ratings between the target user and similar users to embody the change.

The main contributions are as follows: (1) a method for predicting the user requirements that is based on the reported QoS value and the evaluated QoS value is proposed to better reflect the actual requirements of users, (2) a method for calculating the matching degree of a user to a service that is based on the requirements is proposed to improve the recommendation accuracy, and (3) the construction of community based on similarity can reduce the recommended range to improve the recommendation efficiency. The rest of the paper is organized as follows. Section 2 explores the related works. A service recommendation method based on requirements is presented in Section 3. Section 4 evaluates our method using experiments and complexity analysis. Finally, Section 5 summarizes all of our research.

2. Related Work

Recently, researchers have proposed many recommendation methods for the cloud environment.

2.1. Collaborative Filtering Methods. Considering the frequent changes in cloud service quality over time, Ding et al. [8] proposed a time-aware cloud service recommendation method using similarity-enhanced collaborative filtering and the autoregressive integrated moving average model to obtain the time feature of the user similarity, address the data sparsity, and predict the QoS values. To improve the prediction accuracy and recommendation efficiency, Afify et al. [9] proposed a hybrid collaborative filtering method based on the model and memory that uses user statistics, service ratings, and user interests to measure the similarity. Liu et al. [10] proposed a web recommendation method based on the locations of users and services to improve the prediction accuracy and computational efficiency. In [11], a service recommendation method based on keywords was proposed to meet users' personalized requirements, which can improve the accuracy and scalability. Zheng et al. [12] proposed a collaborative filtering recommendation method based on the Spearman coefficient to predict the QoS of the active user for an unrated item, which can get more reliable rankings than the method based on the Pearson coefficient. Liu et al. [21] used similarity-enhanced collaborative filtering for QoS prediction, which considers the personalized influence of similar users and services. Chen et al. [22] proposed a model-based service QoS prediction method, which uses the known QoS values to learn a predictive model.

2.2. Association Rules Methods. Zhang et al. [14] proposed a frequent pattern growth association rules algorithm and a multilevel FP-Tree based on the association rules to extract the higher-level relationships and then combined user interests with the improved clustering algorithm to improve the accuracy of cloud service recommendation. Ming et al. [15] proposed a service recommendation method based on multitag and association rules, which can recommend services that are close to users' potential interests.

2.3. Knowledge-Based Methods. Vera-del-Campo et al. [17] proposed a recommendation system that focuses on the protection of all participants against attacks. In the system, the recommenders use the intermediate nodes to recommend services to users. Amin et al. [18] proposed a recommendation method for cloud infrastructures based on a software approach to achieve an energy-efficient cloud platform.

2.4. Trust-Based Methods. Deng et al. [23] proposed a service recommendation method based on a social network. It considers users' historic behaviors, users' preferences, the trust relationships among users in a social network, and users' comments. The results show that this method can improve the recall rate and precision. Considering the impact of trust on the recommendation accuracy, Yin et al. [24] proposed a cloud service recommendation method based on the trust path model and loop trust model, and the results show that this method improves both the MEA and RMSE by 2%. Due to the existence of unreliable QoS data, Su et al. [25] proposed a trust-based method for reliable personalized QoS prediction to recommend services to users.

2.5. Other Methods. Ding et al. [26] proposed two modified cloud service recommendation algorithms that simultaneously consider the accuracy and diversity by viewing service recommendation as a multiobjective optimization problem, which can improve prediction accuracy and guarantee greater diversity. Zhang et al. [27] proposed a real-time QoS-aware decision-making technique based on the analytic hierarchy process method, which is suitable for selecting infrastructure services for users according to the design time and real-time QoS constraints defined by users. This method acquires the best-fit combinations of cloud services using the matching degree between the constraints and the knowledge base. Considering the historical user-service quality data on multiple platforms, Qi et al. [28] proposed a distributed cloud service recommendation approach based on locality-sensitive hashing, which can

improve recommendation accuracy, scalability, and privacy preservation. Karthikeyan et al. [29] proposed a cloud service recommendation system based on semantic technologies, which uses a natural language processing method to parse the cloud service description documents and process the user queries. Wu et al. [30] proposed a QoS prediction method based on the deviation for cloud and IoT services, which can improve the accuracy and overcome the existing difficulties of heuristic collaborative filtering methods.

However, most of the cloud service recommendation methods do not consider the following fact: the requirements of users change. In particular, a user with no specialist knowledge can hardly accurately select satisfactory services according to his or her requirements at the beginning. Therefore, in this paper, we propose a service recommendation method based on requirements, which can improve the recommendation accuracy without lowering the efficiency.

3. Cloud Service Recommendation Method Based on Requirements

3.1. Description. After a service provider registers its services (i.e., items) on the cloud platform, it reports the quality of service (QoS) that can be provided. The quality of service is described using multiple dimensions, such as the response time, network bandwidth, throughput, and failure rate. The user gives a rating for each dimension after a user interacts with a service.

The data source of our recommendation system is $D = (U, P, Q, R)$, where $U = (u_1, u_2, \ldots, u_n)$ is a set of users in the system, $P = (p_1, p_2, \ldots, p_m)$ is a set of services containing the basic information of services, and $Q = (q_{p_1}, q_{p_2}, \ldots, q_{p_m})$ is the QoS value reported for each service at the time of its registration. q_{p_j} $(j = 1, 2, \ldots, m)$ is a ω-dimensional QoS vector, where $q_{p_j} = (q^1_{p_j}, q^2_{p_j}, \ldots, q^\omega_{p_j})$ and $0 \le q^x_{p_j} \le 1$ $(x = 1, 2, \ldots, \omega)$. Each dimension describes the QoS of a service attribute. R is an $m \times n$ order matrix (as shown in equation (1)), that is, the user's evaluation matrix for services, as follows:

$$R = \begin{pmatrix} r_{u_1 p_1} & \cdots & r_{u_1 p_m} \\ \vdots & \ddots & \vdots \\ r_{u_n p_1} & \cdots & r_{u_n p_m} \end{pmatrix}, \tag{1}$$

where $r_{u_i p_j}$ is the rating of user u_i for service p_j, that is, $r_{u_i p_j} = (r^1_{u_i p_j}, r^2_{u_i p_j}, \ldots, r^\omega_{u_i p_j})$. $r^x_{u_i p_j}$ $(x = 1, 2, \ldots, \omega)$ is the rating of the xth dimension and $0 \le r^x_{u_i p_j} \le 1$. When $r^x_{u_i p_j} = 0$, it indicates that the xth dimension QoS of p_j completely not meet the requirements of u_i; and when $r^x_{u_i p_j} = 1$, the x th dimension QoS of p_j can completely meet the requirements of u_i.

The recommended framework is shown in Figure 1. It is mainly composed of four parts. (1) Part one is the construction of community. k user communities are constructed by a k-means clustering algorithm to narrow the recommendation range and improve the service recommendation efficiency (see Section 3.2). (2) Part two is obtaining the neighbors and screening the services (see Section 3.4). (3) Part three is the prediction of the QoS requirement of a target user u_i. For the screened services, we combine the evaluated QoS values of users for services, the average rating of services evaluated by u_i, and the QoS value reported by the corresponding provider to predict the QoS required or expected by u_i (see Section 3.3). (4) Part four is the service recommendation. We calculate the matching degree between the predicted QoS values and the reported QoS values of the screened services. Then, we obtain the service recommendation list according to the matching degree, the similarity, and the service ratings by the neighboring users (see Section 3.4).

3.2. Construction of Community. In the construction of community, we first need to determine the number of communities. We select k users with the highest interactions as the centers of the initial k communities $\{C_1, C_2, \ldots, C_k\}$, which can better reduce the isolated points. After that, the center of each community is redetermined in each calculation round until the criterion function converges. The criterion function is shown as follows:

$$F^x = \sum_{y=1}^{k} \sum_{u_i \in C_y} \sum_{p_j \in P_{u_i}} \left| r^x_{u_i p_j} - r^x_{u_{(c_y)} p_j} \right|^2, \tag{2}$$

where F^x is the sum of the squared errors of all ratings in the xth dimension, P_{u_i} represents a set of services that user u_i interacts with, and $r^x_{u_{(C_y)} p_j}$ is the x th dimension rating of center $u_{(c_y)}$ of community C_y on service p_j.

Next, we calculate the similarity between the other users and each center of the k communities. If the similarity between a user u_i and the center $u_{(c_y)}$ of community C_y is the highest, then u_i belongs to community C_y. The similarity between user u_i and user $u_{(c_y)}$ is calculated as follows:

$$\mathrm{sim}\left(u_i, u_{(c_y)}\right) = \frac{1}{\omega} \sum_{x=1}^{\omega} \frac{\sum_{p_j \in P_{u_i} u_{(c_y)}} \left(r^x_{u_i p_j} - \overline{r}^x_{u_i} \right) \left(r^x_{u_{(c_y)} p_j} x - \overline{r}^x_{u_{(c_y)}} \right)}{\sqrt{\sum_{p_j \in P_{u_i} u_{(c_y)}} \left(r^x_{u_i p_j} - \overline{r}^x_{u_i} \right)^2} \sqrt{\sum_{p_j \in P_{u_i} u_{(c_y)}} \left(r^x_{u_{(c_y)} p_j} - \overline{r}^x_{u_{(c_y)}} \right)^2}}, \tag{3}$$

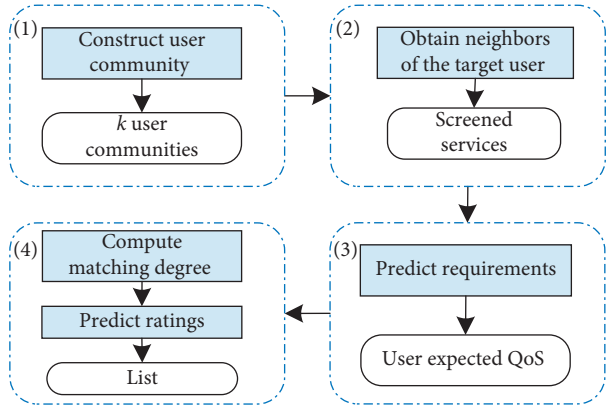

Figure 1: Recommendation framework.

Input: data D, centers of the initial k user communities $u_{(c_1)}, u_{(c_2)}, \ldots, u_{(c_k)}$
Output: k user communities $C_1, C_2, \ldots, C_k$
(1) **repeat**
(2) **for** each user u_i that does not belong to the set $\{u_{(c_1)}, u_{(c_2)}, \ldots, u_{(c_k)}\}$ **do**
(3) max sim = 0;
(4) **for** each $u_{(c_y)}$ $(y = 1, 2, \ldots, k)$ **do**
(5) compute the similarity $\mathrm{sim}(u_i, u_{(c_y)})$ between u_i and $u_{(c_y)}$ using (3);
(6) **if** $\mathrm{sim}(u_i, u_{(c_y)}) \geq \max$ **then**
(7) $\max = \mathrm{sim}(u_i, u_{(c_y)})$;
(8) $\max\, y = y$; //record the center
(9) **end if**
(10) **end for**
(11) Add u_i to the community $C_{\max y}$;
(12) **end for**
(13) **for** each C_y $(y = 1, 2, \ldots, k)$ **do**//the center of each community is redetermined
(14) **for** each service p_j $(j = 1, 2, \ldots, m)$ **do**
(15) $r_{C_y:\, p_j} = (\overline{r}^1_{C_y:\, p_j}, \ldots, \overline{r}^\omega_{C_y:\, p_j}) = ((1/|C_y|)\sum_{u \in C_y} r^1_{\mathrm{up}_j}, \ldots, (1/|C_y|)\sum_{u \in C_y} r^\omega_{\mathrm{up}_j})$;
(16) **end for**
(17) $\min c = 1$;
(18) **for** each user u in C_y **do**
(19) $(d_{\mathrm{up}_1}, \ldots, d_{\mathrm{up}_m}) = ((1/\omega)\sum_{x=1}^{\omega} |r^x_{up_1} - \overline{r}^x_{C_y:\, p_1}|, \ldots, (1/\omega)\sum_{x=1}^{\omega} |r^x_{up_m} - \overline{r}^x_{C_y:\, p_m}|)$;
(20) $\overline{d_u} = (1/m)\sum_{j=1}^{m} d_{\mathrm{up}_j}$;
(21) **if** $\overline{d_u} \leq \min c$ **then**
(22) $\min c = \overline{d_u}$;
(23) $u_{(c_y)} = u$; //update the center
(24) **end if**
(25) **end for**
(26) **end for**
(27) **until** each $F^x (x = 1, 2, \ldots, \omega)$ converge

Algorithm 1: Construction of user community.

where $\overline{r}^x_{u_i}$ $(\overline{r}^x_{u_i} = (1/|P_{u_i}|)\sum_{p_j \in P_{u_i}} r^x_{u_i p_j})$ and $\overline{r}^x_{u_{(c_y)}}$ $(\overline{r}^x_{u_{(c_y)}} = (1/|P_{u_{(c_y)}}|)\sum_{p_j \in P_{u_{(c_y)}}} r^x_{u_{(c_y)} p_j})$ represent the average values of the xth dimension rating of services evaluated by user u_i and $u_{(c_y)}$, respectively, and $P_{u_i u_{(c_y)}}$ represents a set of services that are interacted with by both users u_i and $u_{(c_y)}$, that is, $P_{u_i u_{(c_y)}} = P_{u_i} \cap P_{u_{(c_y)}}$.

Algorithm 1 describes the construction process of user communities. After each round of clustering, the center of each community is redetermined (lines 13–27 in Algorithm 1). For a community C_y, we first calculate the average value of each dimension of the service ratings to obtain a vector $(r_{C_y:\, p_1}, r_{C_y:\, p_2}, \ldots, r_{C_y:\, p_m})$ (lines 14–16 in Algorithm 1). Then, if the evaluation vector $(r_{up_1}, r_{up_2}, \ldots, r_{up_m})$ of user u is the closest to the average vector, then we consider user u to be the new center of community C_y (lines 17–26 in Algorithm 1).

3.3. Prediction of User Requirement. When users select services, they are unsure of their requirements in the beginning or their requirements are constantly changing.

Input: dataset D
Output: a recommendation list including L services
(1) Construct k user communities via Algorithm 1;
(2) Find the community C that the target user u_i belongs to;
(3) **for** each user u ($u \neq u_i$) in C **do**
(4) Obtain the similarity $\mathrm{sim}(u_i, u)$ using (3);
(5) **end for**
(6) Obtain N neighboring users with higher similarity from C;
(7) $P^* \leftarrow$services that the N users interact with;
(8) Remove the services with lower ratings from P^* to obtain P^{**};
(9) **for** each service p in P^{**} **do**
(10) Obtain the requirement vectors $E_{u_i p}$ and $E_{u_{(j)} p}$ ($j = 1, 2, \ldots, N$) using (4);
(11) Calculate the matching degrees Md^x_{up} and $Md^x_{u_{(j)}p}$ ($x = 1, 2, \ldots, \omega$) using (5);
(12) Calculate the predicted rating $\hat{r}^x_{u_i p}$ of the x th dimension QoS using (6);
(13) **end for**
(14) Sort the predicted ratings of all services in P^{**} in descending order, and select the first L services to form a list of recommendations.

Algorithm 2: Recommendation process.

However, if the services that users select do not meet their real requirements, then users may give low ratings to the services after they use them. The ratings can reflect the real requirements of users to a certain extent. Users may need to make several selections before obtaining the services that eventually meet users' real requirements.

In this paper, we combine the ratings of users for a service, the average rating of services evaluated by a target user u_i, and the QoS value reported by the corresponding provider to compute the QoS value that u_i expects or predicts. The xth dimension of the QoS value predicted by u_i is calculated as follows:

$$e^x_{u_i p_j} = \left(\sqrt{\frac{\sum_{u \in U_{p_j}} r^x_{\mathrm{up}_j}}{\left| U_{p_j} \right|} - \overline{r}^x_{u_i} + 1} \right) \times q^x_{p_j}, \tag{4}$$

where U_{p_j} represents a set of users interacting with p_j in the community that includes u_i and $\overline{r}^x_{u_i}$ is the average rating of the xth dimension QoS of services by u_i. We can obtain the QoS requirement vector $E_{u_i p_j}$ of u_i, that is, $E_{u_i p_j} = (e^1_{u_i p_j}, e^2_{u_i p_j}, \ldots, e^\omega_{u_i p_j})$.

3.4. Recommendation Algorithm. The basic strategy of our recommendation method is as follows:

(1) We calculate the similarity between u_i and the other users in community C (we suppose the target user u_i is in C) using equation (3). The N users $(u_{(1)}, u_{(2)}, \ldots, u_{(N)})$ with higher similarity are the neighboring users of u_i. Then, we obtain a set P^* of services that these neighbor users interact with, that is, $P^* = \cup_{j=1}^{N} P_{u_{(j)}}$. If the average rating of a service is lower than the specified threshold, then the service will not appear in the recommendation list to ensure that no low-quality service will be recommended. Therefore, according to the ratings of the neighboring users on the services in P^*, the services with lower ratings are further screened out and then the set P^{**} is obtained.

(2) We compute the predicted QoS values of the services in P^{**}.

(3) For each service p ($p \in P^{**}$), we calculate its matching degree of the xth dimension QoS according to the following equation:

$$Md^x_{u_i p} = 1 - \frac{\left| e^x_{u_i p} - q^x_p \right|}{e^x_{u_i p} + q^x_p + 2}. \tag{5}$$

(4) We predict the rating of service p for u_i based on the ratings of p from the neighboring users, the similarity between users, and the difference in the matching degree of p between users. Therefore, we compute the predicted rating $\hat{r}^x_{u_i p}$ of the xth dimension QoS according to the following equation:

$$\hat{r}^x_{u_i p} = \overline{r}^x_{u_i} + \frac{\sum_{j=1}^{N} \mathrm{sim}\left(u_i, u_{(j)}\right) \times \left| Md^x_{u_i p} - Md^x_{u_{(j)} p} \right|^{-(1/2)} \times \left(r^x_{u_{(j)} p} - \overline{r}^x_{u_{(j)}} \right)}{\sum_{j=1}^{N} \mathrm{sim}\left(u_i, u_{(j)}\right) \times \left| Md^x_{u_i p} - Md^x_{u_{(j)} p} \right|^{-(1/2)}}. \tag{6}$$

where $\overline{r}^{x}_{u_i}$ and $\overline{r}^{x}_{u_{(j)}}$ are the average ratings of the xth dimension QoS of services by u_i and $u_{(j)}$, respectively.

(5) Finally, we recommend the L services with the highest predicted ratings to the target user.

The specific process is shown in Algorithm 2.

4. Experiments

4.1. Experimental Environment, Dataset, and Performance Metrics. The hardware and software environment of our experiments is as follows: an Intel (R) Core (TM) i7-6500U 2.5 GHz CPU, 8 GB of memory (RAM), the Windows 10 64-bit operating system, and Dev C++ as the programming language.

In this paper, we use a public dataset WSDREAM (https://github.com/wsdream/wsdream-dataset) [31], which includes data on real-world QoS evaluation results from 339 users on 5825 Web services. In this dataset, there are two dimensions to the evaluation results: the response time and throughput. Therefore, the value of ω is equal to two. For this dataset, we conduct the following pretreatment: (1) if the response time or throughput of a user on a service is equal to −1, then we ignore this data pair; and (2) the other data will be normalized by dividing the maximum. In addition, we assume that the reported QoS value for a service is equal to the average of the evaluation values of all users on the service.

We use the MAE and RMSE to assess the predictive accuracy of the predicted ratings produced by the recommendation system. The MAE and RMSE are the mean absolute error and root mean square error, respectively. The smaller the values of the MAE and RMSE are, the higher the accuracy. Their calculation methods are as follows:

$$\mathrm{MAE} = \frac{1}{N_{\mathrm{data}}} \sum_{(u_i,p_j)\in \mathrm{data}} \left| \frac{1}{\omega} \sum_{x=1}^{\omega} \widehat{r}^{x}_{u_i p_j} - \frac{1}{\omega} \sum_{x=1}^{\omega} r^{x}_{u_i p_j} \right|,$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{N_{\mathrm{data}}} \sum_{(u_i,p_j)\in \mathrm{data}} \left(\frac{1}{\omega} \sum_{x=1}^{\omega} \widehat{r}^{x}_{u_i p_j} - \frac{1}{\omega} \sum_{x=1}^{\omega} r^{x}_{u_i p_j} \right)^2}. \tag{7}$$

4.2. Experimental Results and Analysis. Our method is founded on the user-based collaborative filtering and considers the QoS requirements of users. Hence, to evaluate the performance of our method (abbreviated as RB), we compare it with the following methods:

(i) A traditional collaborative filtering method based on users (abbreviated as CF), which only uses the similarity among users to make recommendations

(ii) A prediction method based on the deviation (abbreviated as DB) [30], which is based on the principle of collaborative filtering and uses deviations of the services and the users to estimate the QoS value

(iii) A collaborative filtering method based on neighbors (abbreviated as NB) [21], which considers the influence of similar users and services

(iv) A collaborative filtering method based on model (abbreviated as MB) [22], which applies the overall known QoS values to learn a predictive model for predictions

4.2.1. Recommendation Accuracy. Figure 2 shows the variations in the MAE and RMSE for different numbers of neighbors with rating matrix densities of 2% and 4%. According to Figure 2, the MAEs and RMSEs of the RB method are the lowest. These results indicate that the accuracy of the RB method is the highest. This is because the RB method removes the services that are given lower ratings by the neighboring users to ensure the recommendation quality and considers the target user's requirement to reduce the impact of data sparsity and improve the accuracy. The CF method uses the user similarity that is calculated by the Pearson correlation coefficient to obtain the predicted ratings. This leads to large errors in datasets with high sparsity. Therefore, its accuracy is the lowest. The accuracy of the DB method is higher than that of the CF method, especially when the density is lower. This is because the DB method considers contextual information. When the matrix density is 2% and the number of neighbors is less than 20, the MAEs of the NB and MB methods are slightly lower than those of the RB method. But their RMSEs are still higher than those of our method. The MB method uses the known data to learn a predictive model, and the NB method only employs enhanced similarity to make the prediction. Therefore, they still suffer from the data sparsity problem.

Figure 3 shows the variations in the MAEs and RMSEs with rating matrix densities of 2%, 4%, 6%, 8%, and 10%. As the matrix density increases, the MAEs and RMSEs of the RB, DB, NB, and MB methods are relatively stable, and those of the CF method have a tendency to decline. This indicates that the RB, DB, NB, and MB methods are more robust. In addition, according to Figure 3, the accuracy of the RB method is obviously higher than those of the other methods under the same matrix density. This is because when predicting the rating of a service, the RB method not only considers the similarity between users but also considers the difference in the matching degrees of a service. The difference in the matching degrees reflects the difference between the requirement of the target user and that of a neighboring user. The smaller the difference in the matching degrees for a service is, the more important the rating given by the neighboring user is when predicting the rating.

Figure 4 shows the variations in the MAE with rating matrix densities of 30%, 40%, 50%, 60%, and 70%. According

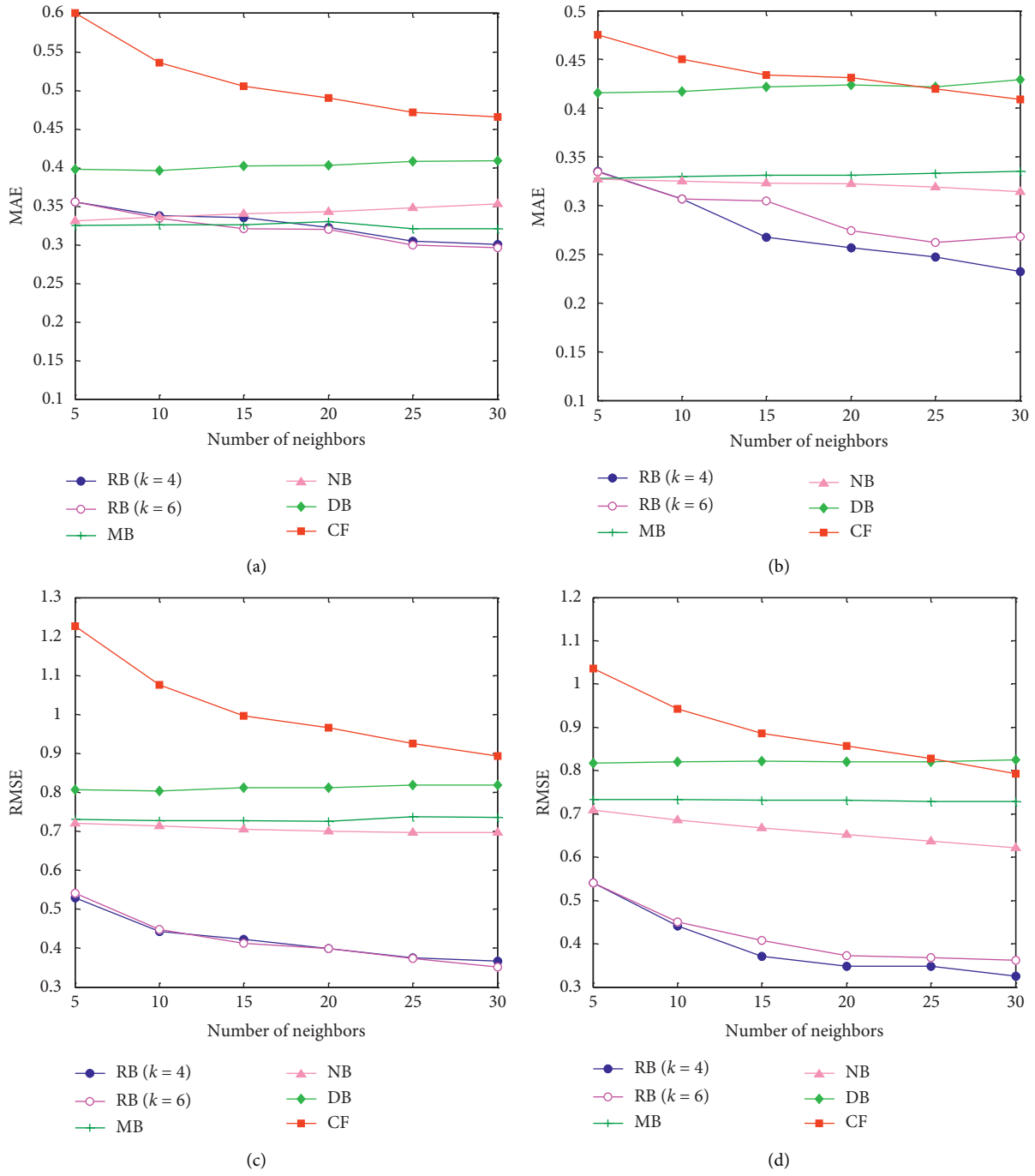

FIGURE 2: MAEs and RMSEs for different numbers of neighbors. (a) MAE (matrix density: 2%), (b) MAE (matrix density: 4%), (c) RMSE (matrix density: 2%), and (d) RMSE (matrix density: 4%).

to Figure 4, the MAEs of each method are lower than the values in Figure 3. This is because the sparsity of the rating matrix is reduced. However, when the matrix density is higher than 50%, the MAE values of the DB, MB, and CF methods are slightly increasing, and those of the RB and NB methods are still dropping and starting to level off. In addition, the RB method outperforms the other methods. This further indicates that the RB method is more robust. The main reason for the results is that the RB method considers the service matching degree and the similarity between users. The DB and MB methods are more suitable for the sparse rating matrix. This is because the DB method uses the deviation-based baseline estimate, and the MB method uses the entire training data to learn a model.

4.2.2. Time Complexity Analysis and Execution Time. For simplicity, we assume that n is the number of users and m is the number of services. For the CF method, the time complexity can be estimated as $O(n^2)$. For the DB method,

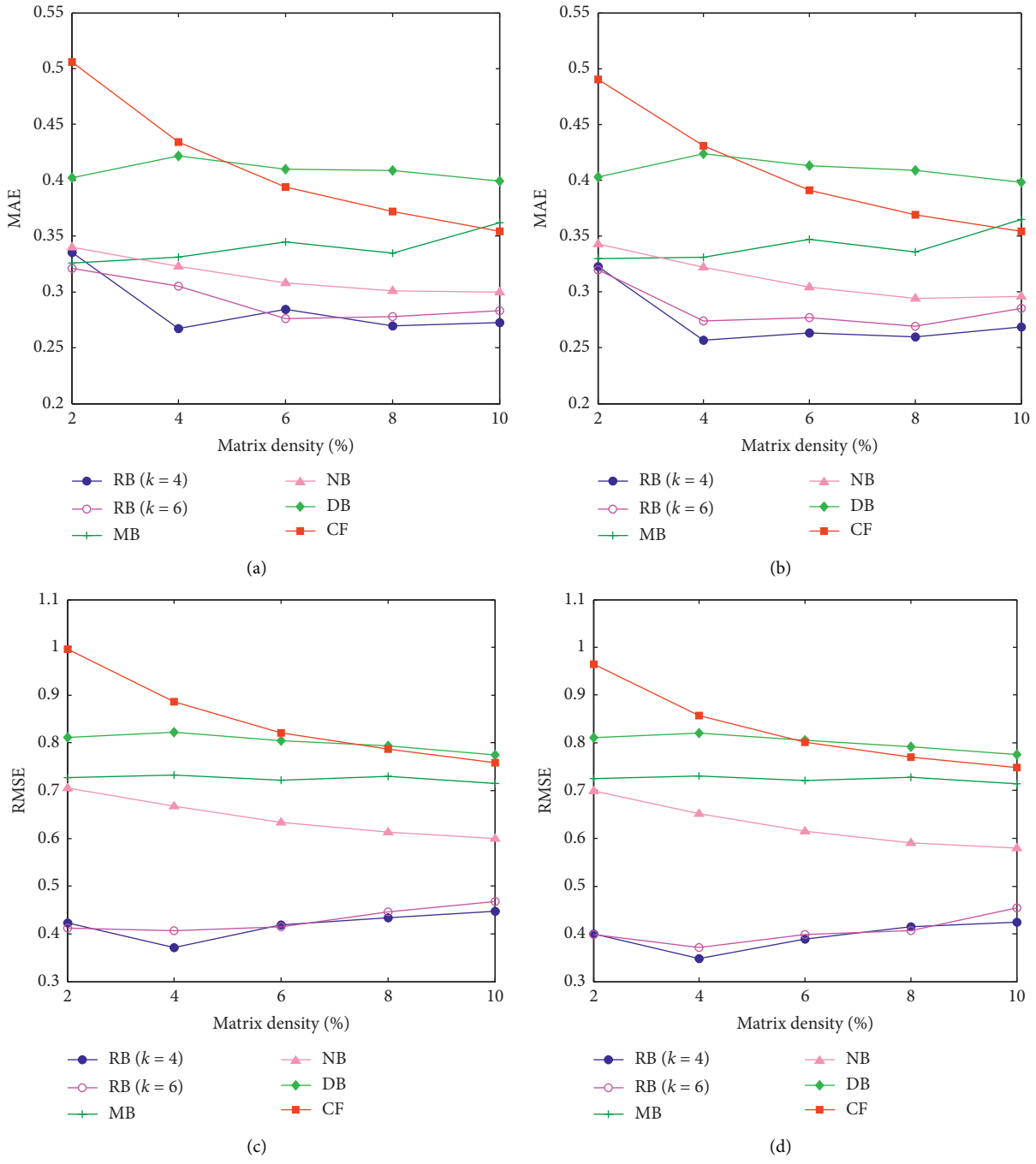

FIGURE 3: MAEs and RMSEs for different matrix densities (2%–10%). (a) MAE (number of neighbors: 15), (b) MAE (number of neighbors: 20), (c) RMSE (number of neighbors: 15), and (d) RMSE (number of neighbors: 20).

the time complexity can be estimated as $O(mnNI_{\mathrm{DB}})$, where I_{DB} is the number of iterations and N is the number of neighbors. For the RB method, the time complexity is $O(knI_{\mathrm{RB}}) + O(n^2) = O(n^2)$ $(k, I_{\mathrm{RB}} \ll n)$, where I_{RB} is the number of iterations in the construction of the user community.

Figure 5 shows the execution time of each method. The execution time of the RB method is close to that of the CF method, and their execution times are less than those of the DB, MB, and NB methods. From the above results, we see that the RB method can improve the recommendation accuracy without lowering the time efficiency.

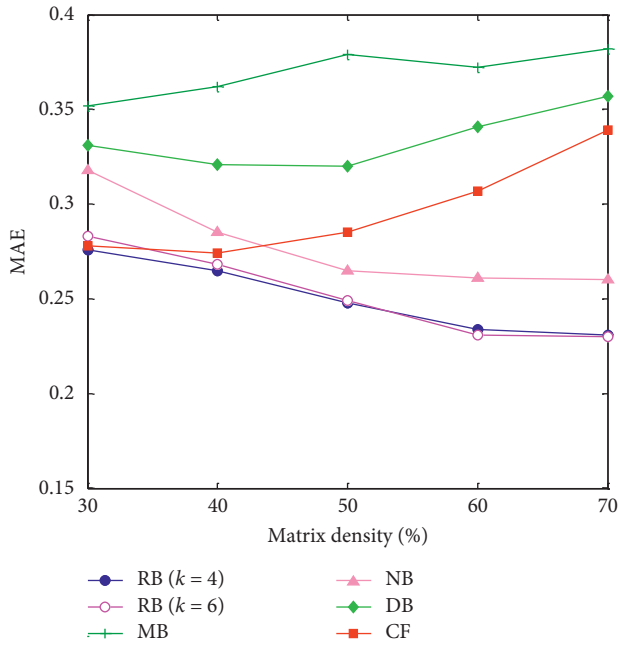

Figure 4: MAE for different matrix densities (30%–70%).

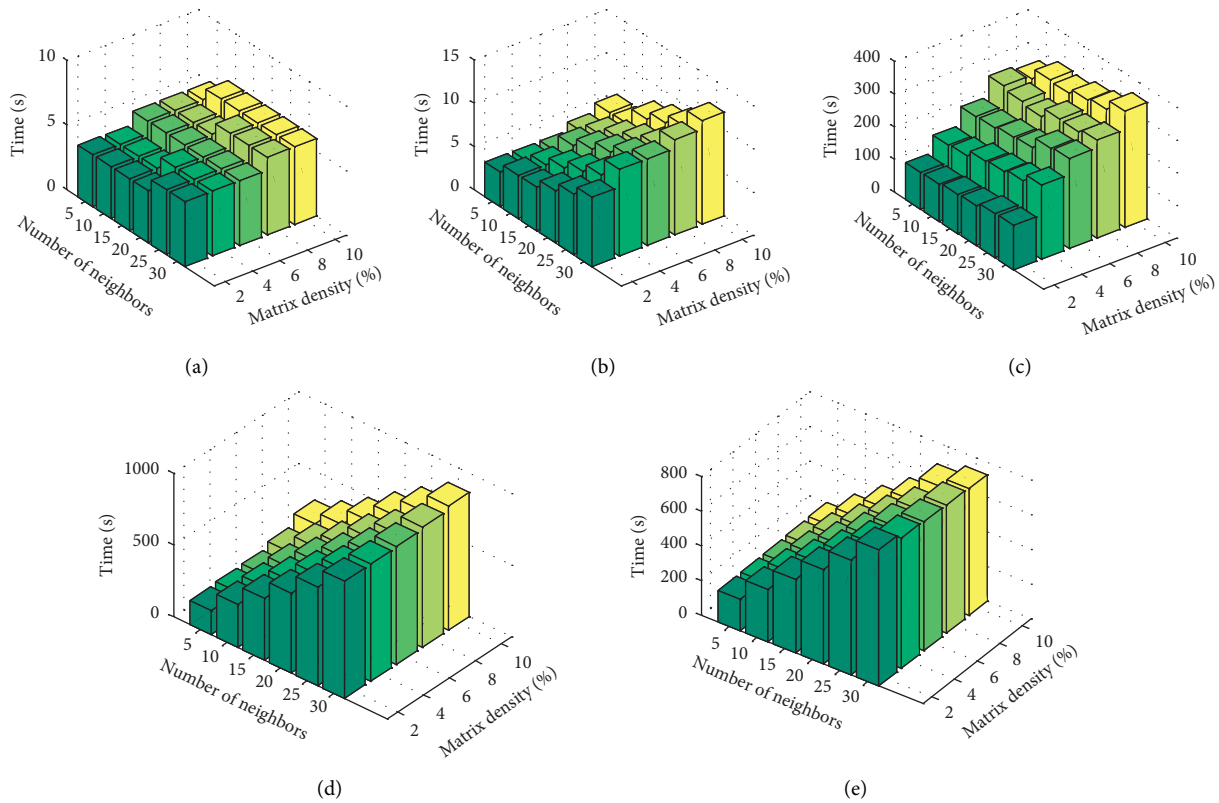

Figure 5: Execution time: (a) CF, (b) RB ($k = 4$), (c) DB, (d) MB, and (e) NB.

5. Conclusions

This paper proposed a service recommendation method based on requirements. First, user communities are constructed by clustering to reduce the range. Second, the reported QoS values and the evaluation QoS values are used to predict the QoS requirements of users. Then, the matching degree of users to services is computed. Finally, service ratings are predicted for the target user to obtain the recommendation list based on the similarity, the difference in their matching degrees, and the ratings of services by the target user's neighbors. In the case of the lower rating matrix densities (2%–10%), the average MAE value (0.28) of our method is less than that of the CF method (0.39), that of the DB method (0.40), that of the NB method (0.31), and that of the MB method (0.34). In addition, the average execution time of our method is less than that of the DB method, that of the NB method, and that of the MB method. Therefore, our method can improve the recommendation accuracy without lowering the efficiency.

In future works, we will further improve the requirement prediction method by mining implicit user information from reviews.

Data Availability

The data used to support the findings of this study are available at https://github.com/wsdream/wsdream-dataset.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (nos. 61672039, 61602009, and 61772034) and the Natural Science Foundation of Anhui Province (no. 1908085MF190).

References

[1] Y. Wang, Q. He, and Y. Yang, "QoS-aware service recommendation for multi-tenant SaaS on the cloud," in *Proceedings of the 2015 IEEE International Conference on Services Computing*, pp. 178–185, New York, NY, USA, June 2015.

[2] Z. H. Huang, J. Zhang, B. Zhang, J. Yu, Y. Xiang, and D.-S. Huang, "Survey of semantics-based recommendation algorithms," *Acta Electronica Sinica*, vol. 44, no. 9, pp. 2262–2275, 2016.

[3] J. Shu, X. Shen, H. Liu, B. Yi, and Z. Zhang, "A content-based recommendation algorithm for learning resources," *Multimedia Systems*, vol. 24, no. 2, pp. 163–173, 2018.

[4] J. B. Schafer, D. Frankowski, J. Herlocker et al., "Collaborative filtering recommender systems," *The Adaptive Web*, vol. 4321, pp. 291–324, 2007.

[5] C. D. Wang, Z. H. Deng, J. H. Lai, and P. S. Yu, "Serendipitous recommendation in e-commerce using innovator-based collaborative filtering," *IEEE Transactions on Cybernetics*, vol. 49, no. 7, pp. 2678–2692, 2018.

[6] Z. Hu, G. Xu, X. Zheng et al., "SSL-SVD: semi-supervised learning--based sparse trust recommendation," *ACM Transactions on Internet Technology*, vol. 20, no. 1, pp. 1–20, 2020.

[7] Y. Zhang, G. Liu, A. Liu et al., "Personalized geographical influence modeling for POI recommendation," *IEEE Intelligent Systems*, vol. 35, no. 5, pp. 18–27, 2020.

[8] S. Ding, Y. Li, D. Wu, Y. Zhang, and S. Yang, "Time-aware cloud service recommendation using similarity-enhanced collaborative filtering and ARIMA model," *Decision Support Systems*, vol. 107, pp. 103–115, 2018.

[9] Y. M. Afify, I. F. Moawad, N. L. Badr, and M. F. Tolba, "Enhanced similarity measure for personalized cloud services recommendation," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 8, Article ID e4020, 2017.

[10] J. Liu, M. Tang, Z. Zheng, X. Liu, and S. Lyu, "Location-aware and personalized collaborative filtering for web service recommendation," *IEEE Transactions on Services Computing*, vol. 9, no. 5, pp. 686–699, 2016.

[11] S. Meng, W. Dou, X. Zhang, and J. Chen, "KASR: a keyword-aware service recommendation method on MapReduce for big data applications," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 12, pp. 3221–3231, 2014.

[12] X. Zheng, L. D. Xu, and S. Chai, "QoS recommendation in cloud services," *IEEE Access*, vol. 5, pp. 5171–5177, 2017.

[13] Z. Yang, Q. Sun, Y. Zhang, L. Zhu, and W. Ji, "Inference of suspicious co-visitation and co-rating behaviors and abnormality forensics for recommender systems," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2766–2781, 2020.

[14] C. Zhang, Z. Li, T. Li, et al.,"P-CSREC: a new approach for personalized cloud service recommendation," *IEEE Access*, vol. 6, pp. 35946–35956, 2018.

[15] Y. Ming, "Cloud computing network service recommendation method based on multi tag and association rules," *Journal of Computational and Theoretical Nanoscience*, vol. 13, no. 12, pp. 9548–9552, 2016.

[16] R. L. Rosa, G. M. Schwartz, W. V. Ruggiero, and D. Z. Rodríguez, "A knowledge-based recommendation system that includes sentiment analysis and deep learning," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 2124–2135, 2019.

[17] J. Vera-del-Campo, J. Pegueroles, J. Hernández-Serrano, and M. Soriano, "DocCloud: a document recommender system on cloud computing with plausible deniability," *Information Sciences*, vol. 258, pp. 387–402, 2014.

[18] M. B. Amin, S. Hussain, M. Han et al., *Profiling-based Energy-Aware Recommendation System for Cloud Platforms, Computer Science and its Applications*, pp. 851–859, Springer, Berlin, Germany, 2015.

[19] F. Zhang, Y. Liu, and Q. Xiong, "A novel preferential diffusion recommendation algorithm based on user's nearest neighbors," *International Journal of Digital Multimedia Broadcasting*, vol. 2017, no. 5, 7 pages, Article ID 1386461, 2017.

[20] W. Chen, Z. Niu, X. Zhao, and Y. Li, "A hybrid recommendation algorithm adapted in e-learning environments," *World Wide Web*, vol. 17, no. 2, pp. 271–284, 2014.

[21] J. Liu and Y. Chen, "HAP: a hybrid QoS prediction approach in cloud manufacturing combining local collaborative filtering and global case-based reasoning," *IEEE Transactions on Services Computing*, p. 1, 2019.

[22] Z. Chen, L. Shen, and F. Li, "Your neighbors are misunderstood: on modeling accurate similarity driven by data range to collaborative web service QoS prediction," *Future Generation Computer Systems*, vol. 95, pp. 404–419, 2019.

[23] S. Deng, L. Huang, Y. Yin, and W. Tang, "Trust-based service recommendation in social network," *Applied Mathematics & Information Sciences*, vol. 9, no. 3, pp. 1567–1574, 2015.

[24] C. Yin, J. Wang, and J. H. Park, "An improved recommendation algorithm for big data cloud service based on the trust in sociology," *Neurocomputing*, vol. 256, pp. 49–55, 2017.

[25] K. Su, B. Xiao, B. Liu, H. Zhang, and Z. Zhang, "TAP: a personalized trust-aware QoS prediction approach for web service recommendation," *Knowledge-Based Systems*, vol. 115, pp. 55–65, 2017.

[26] S. Ding, C. Xia, C. Wang, D. Wu, and Y. Zhang, "Multi-objective optimization based ranking prediction for cloud service recommendation," *Decision Support Systems*, vol. 101, pp. 106–114, 2017.

[27] M. Zhang, R. Ranjan, M. Menzel et al., "An infrastructure service recommendation system for cloud applications with real-time QoS requirement constraints," *IEEE Systems Journal*, vol. 11, no. 4, pp. 2960–2970, 2015.

[28] L. Qi, X. Zhang, W. Dou, and Q. Ni, "A distributed locality-sensitive hashing-based approach for cloud service recommendation from multi-source data," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2616–2624, 2017.

[29] S. Balaji, N. K. Karthikeyan, and R. S. Raj Kumar, "Fuzzy service conceptual ontology system for cloud service recommendation," *Computers & Electrical Engineering*, vol. 69, pp. 435–446, 2018.

[30] H. Wu, K. Yue, C.-H. Hsu, Y. Zhao, B. Zhang, and G. Zhang, "Deviation-based neighborhood model for context-aware QoS prediction of cloud and IoT services," *Future Generation Computer Systems*, vol. 76, pp. 550–560, 2017.

[31] Z. Zheng, Y. Zhang, and M. R. Lyu, "Investigating QoS of real-world web services," *IEEE Transactions on Services Computing*, vol. 7, no. 1, pp. 32–39, 2014.