Arabian Journal for Science and Engineering (2020) 45:2929–2943
https://doi.org/10.1007/s13369-019-04218-6

RESEARCH ARTICLE - COMPUTER ENGINEERING AND COMPUTER SCIENCE

A Service Context-Aware QoS Prediction and Recommendation of Cloud Infrastructure Services

Rajganesh Nagarajan[1] · **Ramkumar Thirunavukarasu**[2]

Received: 8 June 2019 / Accepted: 24 October 2019 / Published online: 8 November 2019

Abstract
Service recommendation is an active research area in cloud computing since mapping of right services as per the desired requirements of user is a challenging task. With the rapid development of cloud services, the recommendation techniques are facing the problem in predicting the better QoS values because of neglecting the contextual information of cloud services. The paper proposes a service context-aware-based cloud broker that extracts the service details by considering the contextual information of cloud services and computes service similarities on the basis of QoS values. Further, it tackles the cold start problem by adopting the matrix factorization principle and predicts better QoS values for newly arrived services. To validate our approach, we have conducted experimental works on benchmark datasets and the result shows that the proposed approach outperforms better results than the model-based approaches. In particular, our proposed system produces improved response time for the dataset of sparse nature.

Keywords Cloud service · QoS prediction · Service context · Matrix factorization

1 Introduction

Nowadays, cloud providers offer different types of cloud services and identification of appropriate services as per the requirements of the user is a challenging task. To tackle the problem, the implication of recommendation systems has been greatly recognized by the cloud service research community. In the context of cloud computing, a recommender system [1] has been regarded as an information filtering technique used to predict the right services on behalf of user. This system uses the concept of 'rating' to measure the user's preferences about the services. Many commercial service providers such as Amazon, Netflix, Facebook, LinkedIn, Yahoo are using the implication of recommendation system for the effective delivery of services. In general, the recommender system is classified into three different categories on the basis of the techniques adopted: (i) content-based recommended services which are similar to user's liked services in the past, (ii) collaborative filtering (CF)-based recommended services if they are similar in nature (item-based) or if the users are having similar taste (user-based) and (iii) hybrid method—a combination of content and collaborative methods for recommending the services.

Recently, context-based recommendation systems have been recognized to a greater extent in the cloud service life cycle for improving the quality of service prediction. The term 'context' is a multifaceted concept and perceived differently from the perception of domain experts. In order to narrow down the meaning of context, Dourish [2] categorized them into two kinds: representational view and interactional view. In representational view, context attributes and their structure are fixed and it does not change its behavior over time. In contrast, the interactional view assumes that an underlying context induces the user or service behavior. As a result, context-aware recommendation system gets significant attention to make decisions by incorporating available contextual information. Context awareness is the potential demand of recommender systems, especially in cloud computing, for better quality of service (QoS) provisioning.

While considering QoS factors in context-aware service recommendation, the existing collaborative filtering

✉ Rajganesh Nagarajan
rajganesh@avccengg.net

Ramkumar Thirunavukarasu
ramkumar.thirunavukarasu@vit.ac.in

[1] Department of Information Technology, A.V.C. College of Engineering, Mayiladuthurai, Tamilnadu, India

[2] School of Information Technology and Engineering, Vellore Institute of Technology, Vellore, Tamilnadu, India

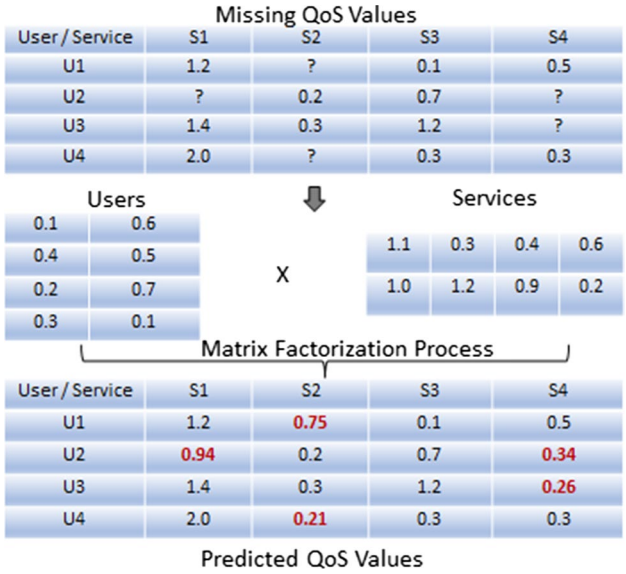

Fig. 1 An example for QoS prediction using matrix factorization

techniques [3–7] have certain limitations such as cold start problem. In addition, they are not integrating contextual information during the recommendation process. Hence, matrix factorization method [8] is widely adopted to deal the QoS-based context-aware recommendation [9]. In this method, latent features about the user and the services are discovered (Fig. 1). Matrix factorization characterizes users/items by vectors of factors inferred from item rating patterns. Recently, these methods have become popular by combining good scalability with accuracy and offer flexibility for modeling various real-life situations.

With respect to service QoS factors, matrix factorization methods faced the following issues: (i) similarity results for user and item are affected due to the changes in QoS values of services, (ii) individual similarity computation is needed for each QoS factor of the services, (iii) the similarity values become inaccurate in case of sparse records. Hence, it is important to build more effective model with the inclusion of contextual information into the existing matrix factorization methods. Motivated by these, we have incorporated the contextual information about services for achieving a better QoS recommendation.

Besides, we have an idea to converge all those activities under a single term called 'broker,' which is responsible to perform the service identification, extraction of service context for their similarity checking, prediction of missing QoS values for the recommendation purpose. From the perspective of cloud computing, a cloud broker [10–12] is a software application designed to serve as an umpire between consumer and provider by offering value-added services.

In summary, the paper makes the following contributions from our end:

(1) Proposes a cloud broker framework with a user portal for feedback collection and service recommendation process.
(2) Extracts the service details along with its context values for the evaluation purpose.
(3) Calculates service similarities and group them according to their QoS values.
(4) Advocates service context-based matrix factorization method to predict the missing QoS for the effective recommendation of cloud services.

The rest of this paper is organized as follows: In Sect. 2, we account the related work in cloud service recommendation system. Section 3 introduces our broker-based context-aware recommender system for effective QoS prediction and recommendation of services. In Sect. 4, we briefly explain the experimental works. Finally, Sect. 5 concludes our proposal with future direction.

2 Related Work

Service recommendation has been an active research area, and many researchers have proposed their works using collaborative filtering, content-based filtering and hybrid approaches. Few of the works used the matrix factorization approach for QoS prediction and recommendation of services. This section highlights various existing research works with respect to the service recommendation and QoS prediction.

Collaborative filtering (CF) uses preferences of similar users in the same reference group as a basis for recommendation. The CF is classified into three categories, namely item-based, user-based and hybrid methods. In item-based method, the similarities are relatively stable than the user-based one. Customers with little domain knowledge have difficulties in choosing the suitable providers who can match their requirements. With the growth of public cloud offerings, multiple services with the same function but different quality attributes are available and it is important to evaluate them for finding the most suitable one. Therefore, Gao and Yu [4] developed a collaborative filtering-based system to predict the services for the user. This personalized recommendation system measures and rates the cloud providers in terms of the customer preferences. Due to the hybrid nature of customer preferences and insufficient services at the providers end, this system results a mismatched service and fails to retain the service quality.

Mezni and Abdeljaoued [5] proposed a collaborative filtering-based recommendation system for cloud services using fuzzy formal concept analysis to address the problem of data sparsity and cold start. They used the lattice representation to describe the cloud environment for the service

recommendation. However, the dynamic service recommendation is not achieved here.

Afify et al. [6] proposed a recommender system that comprises the collaborative filtering and content-based methods to recommend the services as per the user's requirements. The authors concentrated on QoS factors of the recommendation system, which uses only objective time-weighted feedbacks. However, the automatic discovery of service details from the providers' database has not been addressed well in this work. In addition, the trustiness of the recommendation provided by the third parties has also to be validated. Similarly, Hawalah and Fasli [7] used the user-based collaborative filtering to generate appropriate recommendations. The user preferences about the services are captured using keyword and processed by Hadoop MapReduce. As a result, they developed an ontological structure to represent the user's context for service recommendation.

Osadchiy et al. [13] proposed a system with pairwise association rule to accumulate the required services as per the user profiles. Through this method, the author attempted an alternate solution rather than the collaborative and content filtering. By exploring the big data mining concepts, the system gets equipped well to perform the association with large volume of user profiles. Similarly, Meng et al. [14] developed a preference-aware service recommendation method based on MapReduce and used collaborative filtering to recommend the services. The variation in the user's context sometimes affects the efficiency of the overall system. Therefore, service-based context can be a better solution for this issue.

Xu et al. [15] developed a recommendation system by combining the user context (geographical information) and service context (affiliation information) to achieve superior QoS prediction accuracy. Hence, the authors improved the QoS prediction while recommending the services to the users. However, consideration of different contexts fails to compose the required services on time and leads unexpected service pack for the inexperienced cloud user. Similarly, Qi et al. [16] utilized the dynamic QoS data of services to build a time-aware user index table and then used the less sensitive user indices to determine the similar time slots of the target user. The diversity of quality criteria with respect to user and service is not focused in this work.

Ding et al. [17] proposed a time-aware service recommendation approach to improve the quality of service during their recommendation. The authors applied the collaborative filtering method to record the user's similarity dynamically. However, the focus toward the unstructured data is missing in their work. In another work, Ma et al. [18] proposed a MCDM-based variation-aware approach via collaborative QoS prediction to select an optimal cloud service according to users' non-functional requirements. However, the authors have concentrated the QoS values (central tendency, variation range, frequency of variation and period) the basic cold start problem has not yet resolved in this work. Similarly, processing of huge collections of data is also not approached properly.

Liu and Chen [19] developed a novel clustering-based and trust-aware approach to predict the QoS values from similar user history. The similarity is calculated by incorporating both explicit textual and rating information. The trust values from the unreliable cloud user are reconstructed with the trust-aware approach to find out the right QoS values. Due to the different nature of cloud user, the effectiveness of the clustering-based recommendation system is moderate.

Wang et al. [20] proposed a cloud service recommendation approach based on collaborative filtering via exploring user usage history. They have included three phases such as user similarity computation, user neighbor selection and possibility prediction for performing the recommendation. The cosine similarity with improved weighting factor is used for the similarity checking and grouping of services. The method is effective only for the active user history and lack in the prediction of QoS for the newly arrived services.

Fan et al. [21] presented a user location-based (context-based) recommendation system to provide the services. The authors have recorded the authors' location and then performed the similarity mining for the updated location. The QoS about the services is predicted by the Bayesian inference. As a result, an ideal service is recommended to the appropriate user. However, the investigation on the correlations between context properties, such as temporal and spatial correlations, to improve the accuracy of QoS prediction is to be dealt.

Kuang et al. [22] predicted the QoS of a service for the users and their context detail. According to that, a cluster is formed with same context information. The authors also recommended the unused services by measuring their similarity with the existing services. In another work, Yu et al. [23] presented a recommendation system based on role mining with the context of user information. By role mining, the authors predicted the user's service consumption behavior before the recommendation. Both the works focused only on user-based service recommendation and not considered the service/item context.

Xue et al. [24] proposed an online to offline recommendation system, which provides suitable services according to user profile as well. This work relies on four categories of data sources such as user data, context data, social data and service data. Due to different categories of data, the performance of the system is reduced while predicting the better services for the cloud user. The analyzing process of data sources certainly increases the system overhead and need of high-end computational components.

Jiang et al. [25] combined collaborative and content-based methods to consider the tag, time and users' social

relations during the recommendation of services. Similarly, Colombo-Mendoza et al. [26] presented a recommendation approach that comprises three kinds of contextual information such as location, time and crowd. Hence, the authors constructed an ontology-based context modeling approach. Their work did not observe the user's feedback about the services while recommending the services.

The user-based methods are not suitable to offer the newly arrived services, and hence, the need for the item-based recommendation has been focused. In addition, the tag details about the services have been varied according to the user's need. Hence, the need for the item based with its contextual information is to be explored. Li et al. [27] developed a recommendation system that considered the contextual information's of services. The details about the services are extracted from the server side for the identification of its QoS value. Then, these values are matched with the client's requirements for recommendation process. The authors are not furnished any solution to obtain the QoS for the dynamic services in the server end.

Sundermann et al. [28] insisted about the use of contextual information (item details) for the recommendation purpose. They proposed an unsupervised method to extract contextual information of the item instead of user. The privileged information of the service/item is considered for the recommendation process. Hence, the items with moderate ratings are not considered for the recommendation.

Jiang et al. [29] proposed a model by utilizing the descriptive texts and tags of cloud services to extract latent relations among cloud services. Their model segments cloud services into clusters based on the latent information and then ranked them for recommendation. Due to the handling of huge amounts of textual and tag information, the need for the data analytics tools must be focused to improve the data analysis process.

Wu et al. [30] proposed a context-aware (location-based) QoS prediction system for the Web services. Their work is based on the spatial context of the user instead of nature of services involved. Accordingly, the proposed approach classified the location details in order to predict the QoS information. Though the approach attempted to predict the QoS of Web services, recommendation of better QoS services has not been focused, and hence, the need for recommending cloud services with better QoS is emerged.

In our earlier works [11, 12, 31], we have evaluated the trust level of services with a fuzzy-based MapReduce framework. The massive amount of user's feedbacks are simplified with the mappers, reducers and promoted to predict the trust level of services. The system uses the Hadoop to store and process the user's feedback, but it is not applicable for the sparse data. In another work [32], we have developed a fuzzy-based intelligent cloud broker (ICB) for facilitating the service requirements (computing, storage and network components) for an inexperienced cloud user. The broker identified the suitable services based on fuzzy inferencing process, and the importance of the service is expressed with the help of membership values. A fuzzy decision-making model is developed to calculate the information gain for the service attributes by constructing a fuzzy decision tree. Based on that, the opted services are recommended to the inexperienced cloud user. However, the contextual information about the services has not been focused while generating recommendation. In this proposed work, the service details along with its context values are extracted for the evaluation purpose. Then, the proposed system calculates service similarities and groups them according to their QoS values. To tackle the problem of cold start, we advocate a service context-based matrix factorization method to predict the missing QoS for the effective recommendation of cloud services.

3 Proposed Context-Aware Recommendation System with Matrix Factorization

We introduce a broker-based context-aware recommendation system with QoS factors for IaaS type of cloud services, which is shown in Fig. 2. In addition to the user portal, the proposed system consists of three main sub-systems: extraction of service details, evaluation of service and service recommendation. In order to support that, broker includes three repositories: feedbacks, service repository and recommended services repository.

3.1 User Portal for Feedback Collection, Requirement Gathering and Service Recommendation

Normally, all recommendation system has input portal for the user convenience. In addition to the prescribed activities such as service requirements gathering and submission of recommended service list to the user, we have added a provision to record user feedbacks with respect to availed services from the user point. Using these feedbacks, QoS of the services is recorded for further recommendation. Sometimes, the services may not have proper QoS values and not yet consumed by the cloud user for a long period. For this scenario, we have proposed a solution in Sect. 3.3. The following table shows the service requirements of some cloud user (Table 1).

3.2 Extraction of Service Details

Cloud computing provides different types of services such as software as a service (SaaS), platform as a service (PaaS),

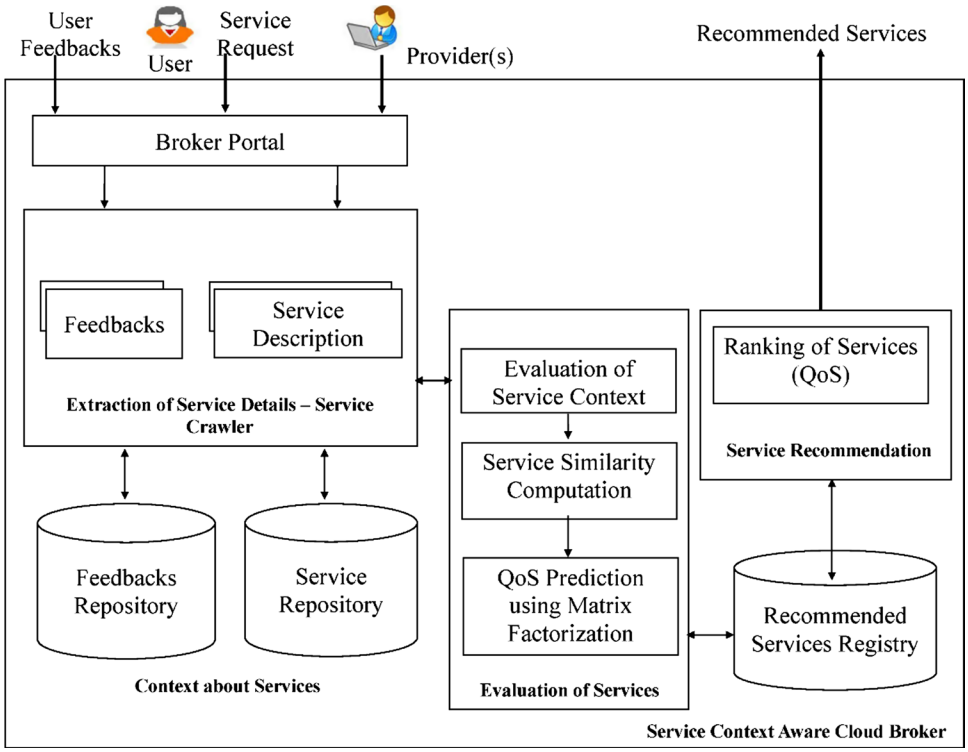

Fig. 2 Proposed system architecture for IaaS service recommendation

Table 1 User requirements

User (*U*)	Type of service (*S*)	Computing speed (GHz)	Storage type with specification		Network bandwidth specification (Gbps)
			Hard disk (TB)	RAM (GB)	
*U*1	Infrastructure services	3.0	1	8	8
*U*2	Infrastructure services	2.5	2	8	16
*U*3	Infrastructure services	3.2	1	16	32

infrastructure as a service (IaaS). In short, anything can be provided as a service, termed as 'XaaS.' In this proposal, we concentrate on IaaS type cloud services only. Computing performance, storage and network facilities are classified under the infrastructure services of cloud, and hence, we denote these facilities as 'service context.'

In this paper, we use service description to identify the context of service. Each cloud service provider has their own service registry and provides the service descriptions in terms of 'description' or 'service level agreement (SLA).' Contextual features describe the functionalities of the service. Hence, we identify the service contexts and perform the similarity computation with our proposed algorithm. In addition to that, the sub-system enables the cloud user to post feedbacks for consumed services. Similar to rating concept of collaborative filtering technique, the matrix factorization utilizes the user feedbacks while predicting the QoS values for the non-rated services. With respect to our earlier work [11], we have recorded the user feedbacks for some QoS factors and processed using a MapReduce framework to reduce the data sparsity in calculation. Here, we address these issues with matrix factorization principle and improved the system quality by predicting the service with the highest QoS values.

In this sub-system, the service details such as service nature, service type, service capabilities and the proposed usage cost are identified with the help of service crawler. It visits the cloud service providers' sites to identify the descriptions of the services. By investigating these descriptions, the basic details about the services can be discovered and updated with the service repository. We referred Amazon,[1] Google,[2] Microsoft Azure,[3] IBM,[4] Oracle,[5]

[1] https://aws.amazon.com/products/.

[2] https://cloud.google.com/terms/service-terms.

[3] https://azure.microsoft.com/en-in/services/cloud-services/.

[4] https://www.ibm.com/software/sla/sladb.nsf/sla/bm.

[5] http://www.oracle.com/us/corporate/contracts/iaas-service-descriptions-1907477.pdf.

Rackspace[6] for the study purpose. Also, WS-DREAM dataset[7] offers a wide collection of service descriptions for the experimental purpose. Table 2 shows the important providers' products with its service names as a response to the user's requirements of Table 1.

3.3 Evaluation of Services

The important task of this sub-system is to preprocess the service context with respect to the posted user's requirements. The functional specification of services is presented in Table 1. By considering the given input requirements, we evaluate the services by its context and then compute the similarities among them. Finally, the missing QoS values are predicted for the recommendation purpose.

3.3.1 Evaluation of Service Context

The service context refers to the characteristic value of the service such as speed, space and access facility of computing device, storage device and network bandwidth, respectively. It is necessary to classify these service characteristics before the service selection. Even though many service providers are transforming their services into cloud computing with standard configuration, the need from the cloud user side is varied with respect to the QoS such as speed, availability and cost. Hence, it is appropriate to preprocess the service request before they are processed against the service selection. The extracted service details of the previous sub-system help to perform the service evaluation process.

We use three important service contexts (computing speed, storage type and network accessibility) from the service description of the respective IaaS cloud service. The service descriptions and SLA's are used to describe the service functionalities with its characteristics. In our proposed approach, we process the service descriptions and extract the vector of representative contents. The detailed processing procedure for the service context evaluation is as follows:

Analyzing of service context The content of a service description file splits into a service vector (S_i) by incorporating the data such as service nature, type, computing characteristics, storage details and the bandwidth information. With respect to our proposal, we have collected a number of services (n) from the most popular cloud service providers (Table 1). All those services have a variety of instances with a variation in the instance configuration. As an example, the following table (Table 3) lists out few service instances for '*computing*' from two cloud service providers. From that,

Table 2 Sample IaaS services from popular cloud service providers

S. No	Service providers	Products category		
		Computing services	Storage services	Network services
1	Amazon	EC2, EC2 Container Registry, EC2 Container Service, Lightsail, VPC, AWS Batch, AWS Elastic Beanstalk, AWS Lambda, Auto Scaling, Elastic Load Balancing	Simple Storage Service (S3), Elastic Block Storage (EBS), Elastic File System (EFS), Glacier, AWS Storage Gateway, AWS Snowball, AWS Snowball Edge, AWS Snowmobile	VPC, CloudFront, Route 53, AWS Direct Connect, Elastic Load Balancing
2	Google Cloud Platform	Compute Engine, App Engine, Container Engine, Cloud Functions	Cloud Storage, Cloud SQL, Cloud Bigtable, Cloud Spanner, Cloud Datastore, Persistent Disk	VPC, Load Balancing, Cloud CDN, Cloud Interconnect, Cloud DNS
3	Azure	Virtual Machine, VM Scale Sets, Container Service, Batch, Service Fabric	Blob, Queue, File, Data Lake Store, StorSimple,	Virtual Network, Load Balancer, Application Gateway, VPN Gateway, DNS, CDN, ExpressRoute
4	IBM	Bare Metal Servers, Cloud Virtual Servers, Container Registry, Cloud Foundry	Object Storage, File Storage, Block Storage, Content Delivery Network	Network Appliances, Direct Link, Domain Name Services, Load Balancer, Network Security

[6] https://www.rackspace.com/en-gb/legal/rackspace-cloud-private-edition-services-agreement-uk.

[7] https://github.com/wsdream/wsdream-dataset.

Table 3 Instances of computing service

Cloud service providers	Service name	Instance types	Models
Amazon[a]	EC2	T2, M3, M4	Nano, Micro, Small, Medium, Large, Xlarge, 2xlarge
Google Cloud[b]	Compute	Standard	n1_standard_1 to n1_standard_96
	Compute	Shared core	f1-micro, g1-small

[a]https://aws.amazon.com/ec2/instance-types/

[b]https://cloud.google.com/compute/pricing/

we can extract the service contexts using the instance types, models and features.

Filtering the service context With reference to Table 2, service instances and their values are to be filtered from the service vector (S_i). In this paper, we employ Bloom filters [30], a compact data structures for probabilistic representation of a set with user requirements. Consider the vector $S_i = \{a_1, a_2, \ldots, a_n\}$ of N elements. Bloom filters define membership information of S_i using a bit vector S_j of length M. For this, k hash functions, $h_1, h_2, \ldots, h_k$ with $h_i: X \rightarrow \{1..m\}$ are used. The following procedure builds an m bits Bloom filter, corresponding to the input vector S_i and using $h_1, h_2, \ldots, h_k$ hash functions. Therefore, if x is member of a set S_i, in the resulting Bloom filter S_j, all bits obtained corresponding to the hashed values of x are set to 1. Testing for membership of an element e is equivalent to testing that all corresponding bits of S_j are set.

```
Procedure BloomFilter (Vector Si, Hash_function hj, integer m, Vector Sj)
  filter = allocate m bits initialized to 0
        foreach x in Si:
              foreach Hash_function hj:
                if ([hj(x)] == 1)
                  Sj = filter[hj(x)]
              end foreach
        end foreach
return Sj
```

```
Procedure MembershipTest (Element e, Hash_function hj)
            foreach Hash_function hj:
              if filter [hj (e)]! = 1 return No
            end foreach
 return Yes
```

As a result, the Bloom filter guarantees no false negatives and uses limited memory while filtering the service context.

3.3.2 Service Similarity Computation

In common, similarities among the services are carried out by considering three important factors: (i) ratings of the service, (ii) user's past purchasing history and (iii) description about the services. In the first case, if any of the services are not rated earlier or if it is a new service, then it leads to the problem of cold start. In the second case, user's buying history is a dynamic one and their tastes may be varied with respect to time and service QoS factors. Therefore, we consider the third case to find out the similarities among services.

In existing approaches, two methods are used for similarity calculation among the items (here services), (i) Pearson correlation coefficient (PCC) [33–35] and (ii) cosine (COS) [36] methods. Both are memory-based models and used to find the item similarity. In PCC, the similarity between two items i and j is calculated as follows:

$$Sim(i,j) = \frac{\sum_{u \in U} (r_{u,i} - r'_i)(r_{u,j} - r'_j)}{\sqrt{\sum_{u \in U} (r_{u,i} - r'_i)^2} \sqrt{\sum_{u \in U} (r_{u,j} - r'_j)^2}} \quad (1)$$

Here, r' is the average QoS value when the user invokes them.

Similarly, the cosine method is used to find the item similarity in many works and is shown in Eq. (2):

$$Sim(i,j) = \frac{\sum_{u \in U} (r_{u,i} r_{u,j})}{\sqrt{\sum_{u \in U} (r_{u,i}^2)} \sqrt{\sum_{u \in U} (r_{u,j}^2)}} \quad (2)$$

In some circumstances, the similarities of items are not predicted correctly, and hence, the existing methods yield incorrect result, because PCC does not consider the differences of QoS attribute values. In the case of cosine method, the angles of the vectors are measured by neglecting the length of the vectors. To overcome these shortcomings, the contextual information of the services is used for calculating their similarities. In our work, similarity of services can be evaluated by extracting the service QoS values from the cloud service providers. Normally, the similarity computation is based on user–service matrix. It is composed of m users and n available services.

Construction of service-to-service matrix with same service context is a tedious one. An alternate to this approach is to scan the service instances with its features (Table 4) and then identify the related services by computing their similarities.

As per Table 4, we obtain the context value C for the services $S_1, S_2, \ldots, S_i$ of the providers P_1, P_2, P_n, respectively. Suppose that service context $C_{1,1}$ is similar to $C_{3,3}$, and then, they are grouped together. Indirectly, the context

Table 4 An example of provider–service–context matrix

Provider/service/context	S_1	S_2	S_3	S_4	…	S_{i-1}	S_i	…
P_1	$C_{1,1}$	$C_{1,2}$	$C_{1,3}$	$C_{1,4}$	…	$C_{1,i-1}$	$C_{1,i}$	…
P_2	$C_{2,1}$	$C_{2,2}$	$C_{2,3}$	$C_{2,4}$	…	$C_{2,i-1}$	$C_{2,i}$	…
P_3	$C_{3,1}$	$C_{3,2}$	$C_{3,3}$	$C_{3,4}$	…	$C_{3,i-1}$	$C_{3,i}$	…
P_4	$C_{4,1}$	$C_{4,2}$	$C_{4,3}$	$C_{4,4}$	…	$C_{4,i-1}$	$C_{4,i}$	…
…	…	…	…	…	…	…	…	…
P_{n-1}	$C_{n-1,1}$	$C_{n-1,2}$	$C_{n-1,3}$	$C_{n-1,4}$	…	$C_{n-1,i-1}$	$C_{n-1,i}$	…
P_n	$C_{n,1}$	$C_{n,2}$	$C_{n,3}$	$C_{n,4}$	…	$C_{n,i-1}$	$C_{n,i}$	…

values may give QoS about services that can perform the similarity among the services. The QoS is numerical in nature, and their similarity represents the degree of service consistency. We measure the service instances characteristics to determine the similarity. Our method to measure the similarity between the services S_i and S_j by their context is:

$$Sim(S_i, S_j) = \frac{\sum_{s=1}^{n} val(Context(S_i)) \cap val(Context(S_j))}{|S|} \quad (3)$$

The following algorithm calculates service similarity using context values. The user requirements, service repository and the service matrix with the extracted services details have been taken as input for the algorithm. By considering the input values, the service context is evaluated and extracted by applying the Bloom filter. Then, the filtered services (S_j) with its service instances such as compute, storage and network are analyzed. Now, the context similarity (by Eq. 3) is checked, and then, the service instances are arranged in an order with the service matrix (S_m).

Algorithm 1: Service Similarity Calculation among the Extracted Services

Input: (1) User Requirements U_{Rec}
(2) Service Repository S_{Reg}
(3) Service Matrix S_m

Output: Matrix RS

```
For each service Si in SReg do
If Si contains description of services then
   Invoke BloomFilter (Vector Si, Hash_function hj , integer m, Vector Sj)
   MembershipTest (Element e, Hash_function hj)
Else
   Error ("No Description available")
End if
For each service instances Si in Sj do
If speed && storage && bandwidth >= URec then
Sort (Si)
Replace the top most services into Recommended-Service-Repository RS
Output RS
```

3.3.3 QoS Prediction Using Service Context Matrix Factorization

In our earlier work [11], we approached the service prediction by using MapReduce framework. By constructing the appropriate mappers and reducers, we have processed the quality of services to ensure about its trust level before they are selected. If the user's responses are increased, then it becomes difficult to predict the QoS. To overcome, we proposed a model-based approach with matrix factorization concept which in turn uses the services context.

Matrix factorization technique is more efficient than the item-based collaborative filtering [3, 37–42] and MapReduce [11, 14, 43]. This method predicts the missing values by factorizing the user–service details. With this, the underlying details of the services are discovered and termed as

'service context.' In addition, the predicted context values influence the user's choice while posting their requirements. Hence, we proposed a matrix factorization-based prediction method to improve the recommendation process. Our predicting process defines the function:

$$f(Cloud\_User) \cap f(Cloud\_Service) \cap f(Service\_Context) \Rightarrow QoS \tag{4}$$

These factors represent the domains of users, services, service details and QoS, respectively. For indexing the above such factors, we define the following notations: u for cloud users, s for cloud services and c for service contexts. The notation P_{sc} indicates an available quality score of the cloud service s under the context c, and the notation P'_{sc} represents the predicted value of P_{sc}. The available QoS values (P_{sc}) are depending upon the provider's service details, whereas the prediction is based on the service context such as speed, storage and network access. By improving Eq. (3), the prediction rule is defined as:

$$P'_{sc} = \left(U_{vector} + \sum_{context=1}^{n} C_{vector,context}\right)\left(S_{vector} + \sum_{context=1}^{n} S_{vector,context}\right) \tag{5}$$

where U_{vector} and S_{vector} are real-valued vectors which represent the cloud_user 'u' and cloud_ service 's.' $C_{vector,context}$ and $S_{vector,context}$ represent the contextual values of user and services, respectively. Even though the prediction rule includes the user context, our focus is to observe the service context only. Therefore, we give consideration only to service context and the user contextual values are optional here. Through this, we define the learning procedure as follows:

$$L = \frac{1}{2}\sum_{(u,s,c)}\left[\left(P_{sc} - \left(U_{vector} + \sum_{context=1}^{n} C_{vector,context}\right)\left(S_{vector} + \sum_{context=1}^{n} S_{vector,context}\right)\right)^2 + \lambda_s\left(||S_{vector}||^2 + \sum_{context=1}^{n} ||S_{vector,context}||^2\right) + \lambda_u\left(||U_{vector}||^2 + \sum_{context=1}^{n} ||C_{vector,context}||^2\right)\right] \tag{6}$$

Here, the terms (u, s, c) symbolize to find vectors of user, service, context, respectively. Followed by that, we have given the regularization terms for adjusting the loss value because of missing QoS values in the input training dataset. Also, λ_s and λ_u denote the latent factors of service and user context, respectively, for controlling the regularization terms. The regularization process with respect to user and service is shown as follows:

$$U_{loss} = \sum_{i=1}^{n}\left(\frac{1}{2} * \lambda_u * (U_{vector})^2\right) \tag{7}$$

$$S_{loss} = \sum_{i=1}^{n}\left(\frac{1}{2} * \lambda_s * (S_{vector})^2\right) \tag{8}$$

We employ gradient descent algorithm to perform the update operations with the specified parameter settings to derive the solutions for user and services. The update rules for user (U_{vector}) and service (S_{vector}) are as follows:

$$C_{vector1,context} = \sum_{i=1}^{n} C_{vector,context-eta} * U_{vector},\ S_{vector1,context} = \sum_{i=1}^{n} S_{vector,context-eta} * S_{vector} \tag{9}$$

After obtaining the updated information with a predefined iteration (eta) and initial learning rate (λ), the unknown QoS values are predicted. By changing the values of matrix dimension, performance of the prediction process is improved and the loss values are reduced.

Algorithm 2 comprises the steps involved in prediction of better QoS values. Here, the user and service vectors are considered to construct the user matrix and service matrix. The obtained QoS values are normalized according to their service context, and then, prediction rule (Eq. 5) is applied. Further, the regularization process (Eqs. 7, 8) is performed to calculate the loss value. Finally, the update rule (Eq. 9) is applied to ensure the better QoS value collection.

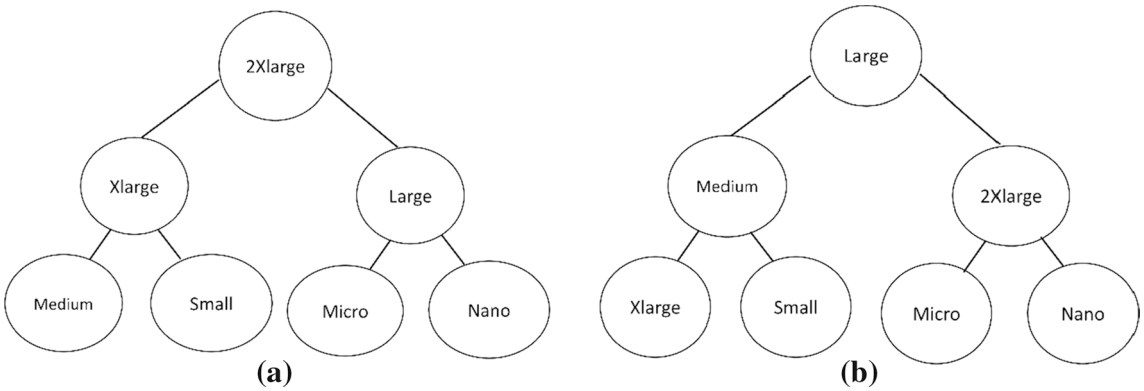

Fig. 3 Instances order **a** before QoS Prediction and **b** after QoS prediction

Algorithm 2: Service Context Matrix Factorization	
Input:	Preprocessed *dataset*, User *U*, Service *S*, Dimension *D*, Learning rate *L*, Regularization-parameter *Lambda*
Output:	Predicted QoS values
	U_{vector} = createMatrix (U, D)
	S_{vector} = createMatrix (S, D)
1	Obtain the QoS data about the services
2	Repeat
3	For each *service*, *user* do
4	Apply the prediction rule in Equation 5
5	Define the learning procedure with regularization parameter using Equation 6
6	Update the loss values of user $U_{loss} = \sum_{i=1}^{n} (\frac{1}{2} * \lambda_u * (U_{vector})^2)$
7	Update the loss values of service $S_{loss} = \sum_{i=1}^{n} (\frac{1}{2} * \lambda_s * (S_{vector})^2)$
8	Until forever
9	Apply Equation 9 for the QoS updates
10	Output predicted QoS values

3.4 Service Recommendation

After the QoS prediction process, the services are listed according to their highest values for the recommendation purpose. From the listing, the user may opt the services as per their wish by considering the factors such as price, response time, availability and throughput. Instead of higher-level service recommendation, the user can avail service pack according to their budgetary constraints. Hence, the proposed system saves the user time and allows them to utilize the second level of services.

Our intention toward the design of the recommender system is to find the relevant services with respect to the user's requirements (Table 1). Hence, the related services are grouped together after their QoS prediction. By considering the training dataset (Table 2), the proposed work prepares the selected service instances with the newly predicted QoS values and ranks them as per the user's requirements. In simple, the highest QoS values are accumulated in an array (max-heap) to prepare the ranked list of instances for the user's choice. By considering the computing instances of Amazon (Table 3), the max-heap is constructed for the recommendation and shown in Fig. 3.

4 Experimental Evaluation

To demonstrate the effectiveness of the proposed work, we conducted an experiment with WS-DREAM [44, 45] dataset. In this section, we have described the evaluation

Table 5 Accuracy comparison for the response time dataset

Methods	Matrix density = 0.05		Matrix density = 0.10		Matrix density = 0.15		Matrix density = 0.20		Matrix density = 0.25	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
Biased MF	0.5934	1.3821	0.5135	1.2625	0.478	1.2088	0.4569	1.1786	0.4396	1.1561
LN LFM	0.5602	1.3965	0.5007	1.2884	0.4712	1.2818	0.4512	1.1541	0.4429	1.1390
PMF	0.5690	1.5371	0.4866	1.3163	0.4522	1.2205	0.4308	1.1690	0.4181	1.1395
NMF	0.5456	1.4727	0.4775	1.2824	0.4465	1.2019	0.4291	1.1616	0.4161	1.1374
Proposed	0.5451	1.4714	0.4726	1.2456	0.4214	1.1634	0.4012	1.1511	0.3910	1.1123

Table 6 Accuracy comparison for the throughput dataset

Methods	Matrix density = 0.05		Matrix density = 0.10		Matrix density = 0.15		Matrix density = 0.20		Matrix density = 0.25	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
Biased MF	21.836	56.857	17.852	48.327	15.937	44.303	14.922	42.140	14.124	40.622
LN LFM	20.401	52.420	17.727	46.971	16.765	44.987	16.342	44.143	16.145	43.711
PMF	19.094	57.907	15.974	48.039	14.657	44.007	13.908	41.727	13.407	40.315
NMF	18.882	57.517	15.571	47.790	14.254	43.864	13.525	41.648	13.084	40.318
Proposed	18.024	56.533	15.128	45.989	13.564	42.321	12.212	40.214	11.431	39.001

methodology and parameter settings and compared the results with the existing methods.

4.1 Dataset Description

We use WS-DREAM dataset for our QoS prediction process. This dataset describes real-world QoS evaluation results from 339 users on 5825 Web services. The contents of the dataset include the following text data. (i) userlist.txt—which contains information of 339 service users, and (ii) wslist.txt—contains information of the 5825 Web services. In addition, two major data files rtMatrix.txt (339 * 5825 user–item matrix of response time) and tpMatrix.txt (339 * 5825 user–item matrix for throughput) are also used for our experimental work.

4.2 Feature Engineering

In our experiments, we have used service details (wslist) of dataset1 from WS-DREAM that contains the details of services such as service id, service description, provider details and IP number. The above fields are considered in our works and processed through Algorithm 1 for extracting the required details, namely context values of services. The resulted data are treated as training set for our experimental purpose toward the prediction of QoS values. In addition to that, the response time and throughput values as per the considered data of service details are evaluated through benchmarking evaluation metrics, which are detailed in the following section.

4.3 Evaluation Metrics

In our work, we use two well-known statistical accuracy metrics: mean absolute error (MAE) and root mean squared error (RMSE), for evaluating the predicted accuracy with respect to response time and throughput.

MAE measures the average magnitude of the errors from the predicted values. For all predicted services, MAE is calculated as:

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} |p_{sc} - p'_{sc}| \tag{10}$$

RMSE measures the differences between actual and predicted values, expressed as:

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (p_{sc} - p'_{sc})^2} \tag{11}$$

For both cases, P_{sc} denotes the original QoS value of the service i and P'_{sc} denotes the predicted value for the service. Both MAE and RMSE express average model prediction error and range from 0 to ∞. The smaller value of MAE indicates better QoS from the predicted values, whereas RMSE should be more useful when large errors are particularly undesirable.

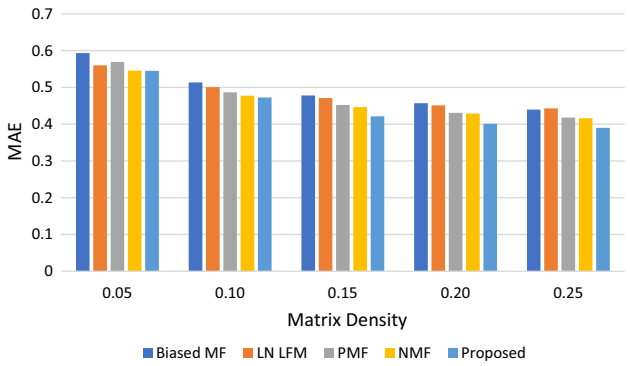

Fig. 4 Response time error rates for MAE

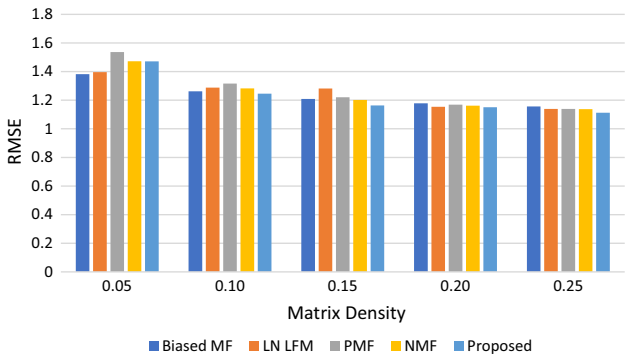

Fig. 5 Response time error rates for RMSE

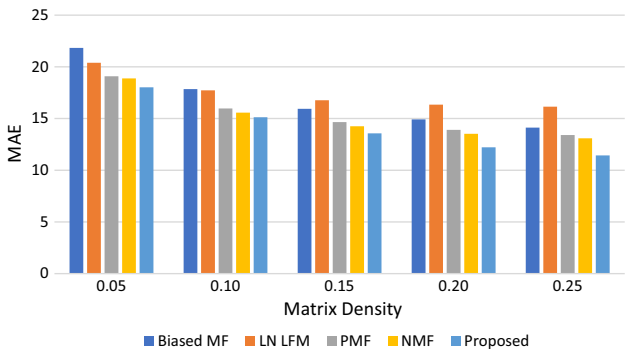

Fig. 6 Throughput error rates for MAE

4.4 Parameter Setting

As per the model-based approaches, the default parameter setting for our proposed work is as follows: The range for the matrix density starts from 5 to 25 with the incremental value of 5. We have used the default dimension value as 10. The initial learning rates are 0.01 and 0.001 for runtime and throughput process, respectively. We have carried out 20 rounds with the maximum of 300 iterations for both of the

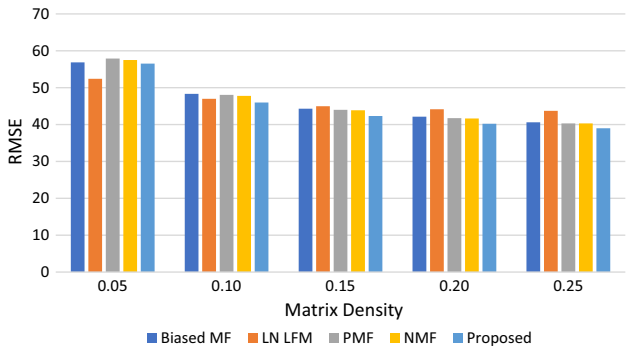

Fig. 7 Throughput error rates for RMSE

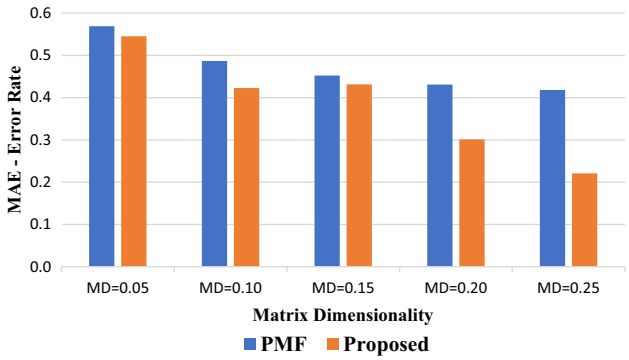

Fig. 8 Error rates of MAE for response time dataset

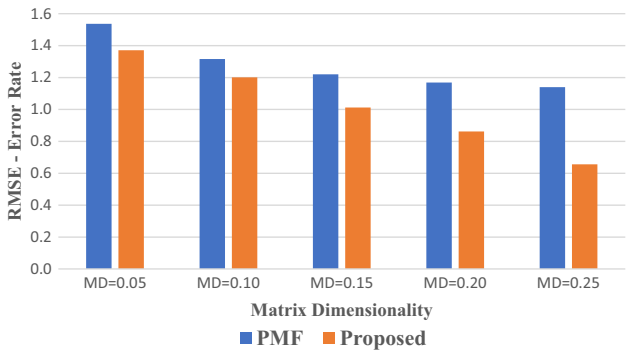

Fig. 9 Error rates of RMSE for response time dataset

QoS factors (response time and throughput). The regularization parameter is varied from 40 to 800 in order to calculate the MAE and RMSE values.

4.5 Accuracy Comparison

Our proposed approach is compared with well-known model-based approaches:

Biased MF—this approach [3] uses the average QoS value as the predicted value. It is a simple QoS prediction method without any optimization involved.
LN LFM—latent neighbor–latent factor model [46], which combines the LFM model and latent neighbor model to improve the QoS value prediction accuracy.
NMF—uses nonnegative matrix factorization [47] to factorize the QoS matrix into two matrices X and Y, with the property that all three matrices have no negative elements.
PMF—applies the probabilistic matrix factorization method on the user–item matrix to generate recommendations [48].

4.6 Experimental Results and Analysis

By considering the parameter settings in Sect. 4.4, we have predicted the QoS values and it is presented in Table 5. We observed that our service context-based approach obtains smaller MAE and RMSE values for the response time dataset. In addition, Table 6 shows the predicted values for the throughput dataset. Here, the values are improved than the other methods.

Since our proposed method is based on model-based approach, the existing methods with their parameter settings have produced different results. In contrast, our proposed method, namely service context-aware matrix factorization (SCMF), preprocesses the services by considering their context. Hence, the volume of the dataset is reduced before they are processed for the QoS prediction in terms of response time and throughput. When service is sparse, SCMF improves the response time and throughput prediction. Obviously, the model is more robust when the matrix dimensionality is increased. Figures show the experimental results for the response time and the throughput dataset. Figures 4 and 5 show the MAE and RMSE results of response time dataset.

Consider the metric MAE, the proposed method produces less error rates in the beginning and sometimes produces the higher error rate in the end. When the matrix dimensionality got increased, the error rates reduced automatically. Due to the impact of service context-based QoS prediction, the proposed method slightly improves the error rates of the taken metrics for the throughput dataset. Figures 6 and 7 show the predicted error values of throughput dataset.

4.7 Impact of Service Context in Error Rate Reduction

Service context determines how the preprocessing of services reduces the sparse dataset before the QoS prediction. To examine, the broker performs the service similarity based on its context values (Algorithm 1). For the preprocessed dataset, proposed method predicts the missing QoS values by Algorithm 2 and their results for the response time dataset are shown in Figs. 8 and 9.

To achieve the better prediction accuracy, the values of matrix dimensionality are extended to some higher range. However, it leads to higher computation cost. Hence, the changes in the dimension values are avoided.

5 Conclusion and Future Works

Recommendation of cloud services with its QoS values is an immediate need of cloud computing system. This paper proposes a service context-aware cloud broker framework for the infrastructure type of cloud service recommendation. The proposed broker extracts the service details based on their contextual information. The challenges addressed in the calculation of service similarity with the existing methods have been addressed by proposing the context-based service similarity checking algorithm. In addition, the broker performs better QoS prediction by improving the PMF model with service context-based matrix factorization approaches. Hence, we predicted the missing QoS values for the services with its specified contextual information. Through our experiments, we proved that our proposed broker-based service–context-aware recommendation system outperforms a better result than the other model-based approaches. The results of the proposed work further explore various promising future research avenues. For example, proposing a service recommendation system by incorporating the contextual information of users such as location, interest and financial status would be an interesting one. Further, addition of intelligence aspects to the broker in performing recommendation would also be an important research avenue for further exploration.

Compliance with Ethical Standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Adomavicius, G.; Tuzhilin, A.: Context-Aware Recommender Systems. Recommender Systems Handbook. Springer, Berlin (2011)
2. Dourish, P.: What we talk about when we talk about context. Pers. Ubiquitous Comput. **8**(1), 19–30 (2004)
3. Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 426–434. ACM (2008)
4. Gao, X.; Yu, C.: A fuzzy-based recommendation system for cloud accounting service. In: 13th IEEE International Conference on

Service Systems and Service Management (ICSSSM), pp. 1–6 (2016)
5. Mezni, H.; Abdeljaoued, T.: A cloud services recommendation system based on fuzzy formal concept analysis. Data Knowl. Eng. **116**, 100–123 (2018)
6. Afify, Y.M.; Moawad, I.F.; Badr, N.L.; Tolba, M.F.: A personalized recommender system for SaaS services. Concurr. Comput. Pract. Exp. **29**(4), e3877 (2017)
7. Hawalah, A.; Fasli, M.: Utilizing contextual ontological user profiles for personalized recommendations. Expert Syst. Appl. **41**(10), 4777–4797 (2014)
8. Baltrunas, L.; Ludwig, B,; Ricci, F.: Matrix factorization techniques for context aware recommendation. In: Proceedings of the Fifth ACM Conference on Recommender Systems, pp. 301–304. ACM (2011)
9. Zhu, J.; He, P.; Zheng, Z.; Lyu, M.R.: Online QoS prediction for runtime service adaptation via adaptive matrix factorization. IEEE Trans. Parallel Distrib. Syst. **28**(10), 2911–2924 (2017)
10. Rajganesh, N.; Ramkumar, T.: A review on broker based cloud service model. CIT J. Comput. Inf. Technol. **24**(3), 283–292 (2016)
11. Nagarajan, R.; Thirunavukarasu, R.; Shanmugam, S.: A fuzzy-based intelligent cloud broker with MapReduce framework to evaluate the trust level of cloud services using customer feedback. Int. J. Fuzzy Syst. **20**(1), 339–347 (2018)
12. Nagarajan, R.; Thirunavukarasu, R.: A review on intelligent cloud broker for effective service provisioning in cloud. In: 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS), pp. 519–524. IEEE (2018)
13. Osadchiy, T.; Poliakov, I.; Olivier, P.; Rowland, M.; Foster, E.: Recommender system based on pairwise association rules. Expert Syst. Appl. **115**, 535–542 (2019)
14. Meng. S.; Tao, X.; Dou, W.: A preference-aware service recommendation method on Map-Reduce. In: 16th IEEE International Conference on Computational Science and Engineering (CSE), pp. 846–853 (2013)
15. Xu, Y.; Yin, J.; Deng, S.; Xiong, N.N.; Huang, J.: Context-aware QoS prediction for web service recommendation and selection. Expert Syst. Appl. **53**, 75–86 (2016)
16. Qi, L.; Wang, R.; Hu, C.; Li, S.; He, Q.; Xu, X.: Time-aware distributed service recommendation with privacy-preservation. Inf. Sci. **480**, 354–364 (2019)
17. Ding, S.; Li, Y.; Wu, D.; Zhang, Y.; Yang, S.: Time-aware cloud service recommendation using similarity-enhanced collaborative filtering and ARIMA model. Decis. Support Syst. **107**, 103–115 (2018)
18. Ma, H.; Hu, Z.; Li, K.; Zhu, H.: Variation-aware cloud service selection via collaborative QoS prediction. IEEE Trans. Serv. Comput. (2019). https://doi.org/10.1109/TSC.2019.2895784
19. Liu, J.; Chen, Y.: A personalized clustering-based and reliable trust-aware QoS prediction approach for cloud service recommendation in cloud manufacturing. Knowl. Based Syst. **174**, 43–56 (2019)
20. Wang, F.F.; Chen, F.Z.; Li, M.Q.: A collaborative filtering method for cloud service recommendation via exploring usage history. In: Proceeding of the 24th International Conference on Industrial Engineering and Engineering Management 2018, pp. 716–725. Springer (2019)
21. Fan, X.; Hu, Y.; Li, J.; Wang, C.: Context-aware ubiquitous web services recommendation based on user location update. In: IEEE International Conference on Cloud Computing and Big Data (CCBD), pp. 111–118 (2015)
22. Kuang, L.; Xia, Y.; Mao, Y.: Personalized services recommendation based on context-aware QoS prediction. In: 19th IEEE International Conference on Web Services (ICWS), pp. 400–406 (2012)
23. Yu, Z.; Wong, R.; Chi, C.H.: Efficient role mining for context-aware service recommendation using a high-performance cluster. IEEE Trans. Serv. Comput. **10**, 914–926 (2015)
24. Xue, X.; Hongfang, H.; Wang, S.; Qin, C.: Computational experiment-based evaluation on context-aware O2O service recommendation. IEEE Trans. Serv. Comput. (2016). https://doi.org/10.1109/TSC.2016.2638083
25. Jiang, Z.; Zhou, A.; Wang, S.; Sun, Q.; Lin, R.; Yang, F.: Personalized service recommendation for collaborative tagging systems with social relations and temporal influences. In: IEEE International Conference on Services Computing (SCC), pp. 786–789 (2016)
26. Colombo-Mendoza, L.O.; Valencia-Garcia, R.; Rodriguez-Gonzalez, A.; Alor-Hernandez, G.; Samper-Zapater, J.J.: RecomMetz: a context-aware knowledge-based mobile recommender system for movie showtimes. Expert Syst. Appl. **42**(3), 1202–1222 (2015)
27. Li, S.; Wen, J.; Luo, F.; Gao, M.; Zeng, J.; Dong, Z.: A new QoS-aware web service recommendation system based on contextual feature recognition at server-side. IEEE Trans. Netw. Serv. Manag. **14**(2), 332–342 (2017)
28. Sundermann, C.V.; Domingues, M.A.; Da Silva Conrado, M.; Rezende, S.O.: Privileged contextual information for context-aware recommender systems. Expert Syst. Appl. **57**, 139–158 (2016)
29. Jiang, Y.; Tao, D.; Liu, Y.; Sun, J.; Ling, H.: Cloud service recommendation based on unstructured textual information. Future Gener. Comput. Syst. **97**, 387–396 (2019)
30. Wu, H.; Yue, K.; Li, B.; Zhang, B.; Hsu, C.H.: Collaborative QoS prediction with context-sensitive matrix factorization. Future Gener. Comput. Syst. **82**, 669–678 (2017)
31. Nagarajan, R.; Selvamuthukumaran, S.; Thirunavukarasu, R.: A fuzzy logic based trust evaluation model for the selection of cloud services. In: IEEE International Conference on Computer Communication and Informatics (ICCCI), pp. 1–5 (2017)
32. Nagarajan, R.; Thirunavukarasu, R.: A fuzzy-based decision-making broker for effective identification and selection of cloud infrastructure services. Soft Comput. **23**(19), 9669–9683 (2019)
33. Gong, M.; Xu, Z.; Xu, L.; Li, Y.; Chen, L.: Recommending web service based on user relationships and preferences. In: 20th IEEE International Conference on Web Services (ICWS), pp. 380–386 (2013)
34. Yao, L.; Sheng, Q.; Ngu, A.; Yu, J.; Segev, A.: Unified collaborative and content-based web service recommendation. IEEE Trans. Serv. Comput. **8**(3), 453–466 (2014)
35. Shardanand, U.; Maes, P.: Social information filtering: algorithms for automating word of mouth. In: Proceedings of SIGCHI Conference on Human Factors in Computing Systems, pp. 210–217 (1995)
36. Sarwar, B.; Karypis, G.; Konstan, J.; Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: Proceedings of International Conference on World Wide Web (WWW'01), pp. 285–295 (2001)
37. Birtolo, C.; Ronca, D.; Armenise, R.: Improving accuracy of recommendation system by means of item-based fuzzy clustering collaborative filtering. In: 11th IEEE International Conference on Intelligent Systems Design and Applications (ISDA), pp. 100–106 (2011)
38. Wei, J.; He, J.; Chen, K.; Zhou, Y.; Tang, Z.: Collaborative filtering and deep learning based recommendation system for cold start items. Expert Syst. Appl. **69**, 29–39 (2017)
39. Zhang, Y.; Song, W.: A collaborative filtering recommendation algorithm based on item genre and rating similarity. In: IEEE International Conference on Computational Intelligence and Natural Computing, pp. 72–75 (2009)
40. Sun, T.; Wang, L.; Guo, Q.: A collaborative filtering recommendation algorithm based on item similarity of user preference. In:

Second International Workshop on Knowledge Discovery and Data Mining, pp. 60–63 (2009)
41. Qin, J.; Cao, L.; Peng, H.: Collaborative filtering recommendation algorithm based on weighted item category. In: Control and Decision Conference (CCDC), pp. 2782–2786 (2016)
42. B, Juan.: Collaborative filtering recommendation algorithm based on semantic similarity of item. In: IEEE Fifth International Conference on Advanced Computational Intelligence (ICACI), pp. 452–454 (2012)
43. Meng, S.; Dou, W.; Zhang, X.; Chen, J.: KASR: a keyword-aware service recommendation method on mapreduce for big data applications. IEEE Trans. Parallel Distrib. Syst. **25**(12), 3221–3231 (2014)
44. Zheng, Z.; Zhang, Y.; Lyu, M.R.: Investigating QoS of real-world web services. IEEE Trans. Serv. Comput. **7**(1), 32–39 (2014)
45. Zheng, Z.; Zhang, Y.; Lyu, M.R.: Distributed QoS evaluation for real-world web services. In: Proceedings of the 8th International Conference on Web Services (ICWS'10), Miami, Florida, pp. 83–90 (2010)
46. Yu, D.; Liu, Y.; Xu, Y.; Yin, Y.: Personalized QoS prediction for web services using latent factor models. In: IEEE International Conference on Services Computing (SCC), pp. 107–114 (2014)
47. Lee, D.D.; Seung, H.S.: Algorithms for non-negative matrix factorization. In: Leen, T.K., Dietterich, T.G., Tresp, V. (eds.) Advances in Neural Information Processing Systems, pp. 556–562. MIT Press, US (2001)
48. Zheng, Z.; Ma, H.; Lyu, M.R.; King, I.: Collaborative web service QoS prediction via neighborhood integrated matrix factorization. IEEE Trans. Serv. Comput. **6**(3), 289–299 (2013)