Chapter 2
Neighborhood-Based QoS Prediction

Abstract With the increasing popularity of cloud computing as a solution for building high-quality applications on distributed components, efficiently evaluating user-side quality of cloud components becomes an urgent and crucial research problem. However, invoking all the available cloud components from user-side for evaluation purpose is expensive and impractical. To address this critical challenge, we propose a neighborhood-based approach, called CloudPred, for collaborative and personalized quality prediction of cloud components. CloudPred is enhanced by feature modeling on both users and components. Our approach CloudPred requires no additional invocation of cloud components on behalf of the cloud application designers. The extensive experimental results show that CloudPred achieves higher QoS prediction accuracy than other competing methods. We also publicly release our large-scale QoS dataset for future related research in cloud computing.

2.1 Overview

In the cloud environment, designers of cloud applications, denoted as component users, can choose from a broad pool of cloud components when creating cloud applications. These cloud components are usually invoked remotely through communication links. Quality of the cloud applications is greatly influenced by the quality of communication links and the distributed cloud components. To build a high-quality cloud application, non-functional Quality-of-Service (QoS) performance of cloud components becomes an important factor for application designers when making component selection [2]. Moreover, for the existing cloud applications, by replacing low-quality components with better ones, the overall quality of cloud applications can be improved.

Different from traditional component-based systems, cloud applications invoke components remotely by Internet connections. User-side QoS experiences of cloud components is thus greatly influenced by the unpredictable communication links. Personalized QoS evaluation is required for each user at the user-side. The most straightforward approach is to evaluate all the candidate components at the user-side. However, this approach is impractical in reality, since invocations of cloud

Y. Zhang and M.R. Lyu, *QoS Prediction in Cloud and Service Computing*,
SpringerBriefs in Computer Science, DOI 10.1007/978-981-10-5278-1_2

components may be charged. Even if the invocations are free, executing a large number of components invocations is time consuming and resource consuming.

Based on the above analysis, it is crucial for the cloud platform to deliver a personalized QoS information service to the application designers for cloud component evaluation. In order to provide personalized QoS values on m cloud components for n users by evaluation, at least $n \times m$ invocations need to be executed, which is almost impossible when n and m are very large. However, without sufficient and accurate personalized QoS values of cloud components, it is difficult for the application designers to select optimal cloud component for building high-quality cloud applications. It is an urgent task for the cloud platform providers to develop an efficient and personalized prediction approach for delivering the QoS information service to cloud application designers.

To address this critical challenge, we propose a neighborhood-based approach, called CloudPred, for personalized QoS prediction of cloud components. CloudPred is enhanced by feature modeling on both users and components. The idea of CloudPred is that users sharing similar characteristics (e.g., location, bandwidth) would receive similar QoS usage experiences on the same component. The QoS value of cloud component c observed by user u can be predicted by exploring the QoS experiences from similar users of u. A user is similar to u if they share similar characteristics. The characteristics of different users can be extracted from their QoS experiences on different components by performing nonnegative matrix factorization (NMF). By sharing local QoS experience among users, our approach CloudPred can effectively predict the QoS value of a cloud component c even if the current user u has never invoked the component c before. The experimental results show that compared with other well-known collaborative prediction approaches, CloudPred achieves higher QoS prediction accuracy of cloud components. Since CloudPred can precisely characterize users features (will be introduced in Sect. 2.3.2), even if some users have few local QoS information, CloudPred can still achieve high prediction accuracy.

In summary, this chapter makes the following contributions:

1. We formally identify the research problem of QoS value prediction in cloud computing and propose a novel neighborhood-based approach, named CloudPred, for personalized QoS value prediction of cloud components. CloudPred learns the characteristics of users by nonnegative matrix factorization (NMF) and explores QoS experiences from similar users to achieve high QoS value prediction accuracy. We consider CloudPred as the first QoS value prediction approach in cloud computing literature.
2. We conduct large-scale experiments to study the prediction accuracy of our CloudPred compared with other approaches. The experimental results show the effectiveness of our approach. Moreover, we also publicly release our large-scale QoS dataset for future research.

The remainder of this chapter is organized as follows: Sect. 2.2 describes the collaborative QoS framework in cloud environment. Section 2.3 presents our CloudPred approach in detail. Section 2.4 introduces the experimental results. Section 2.5 concludes the chapter.

2.2 Collaborative Framework in Cloud

Figure 2.1 shows the system architecture in cloud computing. In a cloud environment, the cloud provider holds a large number of distributed cloud components (e.g., databases, servers, Web services), which can be provided to designers for developing various cloud applications. The cloud application designers, called component users in this chapter, are located in different geographic and network environments. Since users invoke cloud components via different communication links, their usage experiences on cloud components are diverse in several QoS properties including response-time, throughput, etc. In order to provide personalized quality information of different components to application designers for optimal component selection, personalized QoS value prediction is an essential service of a cloud provider.

Within the cloud platform provided by a cloud provider, there are several modules implemented for managing the cloud components. Examples of management modules include *Task Scheduler*, which is responsible for task scheduling, *SLA Wrapper*, which is responsible for service-level negotiation between cloud provider and users, etc. In this chapter, we focus on the design of *QoS Monitor*, which is responsible for monitoring the QoS performance of cloud components from the users' perspective. The *QoS Monitor* consists of two subunits: *Collector*, which is used to collect QoS

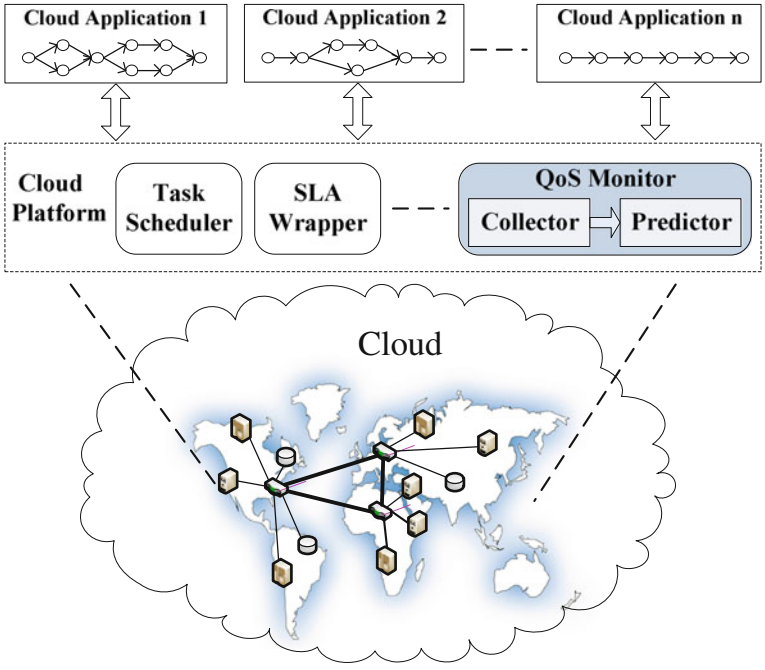


Fig. 2.1 System architecture. ©[2011] IEEE. Reprinted, with permission, from Ref. [8]

usage information from various component users, and *Predictor*, which is supposed to provide personalized QoS value prediction for different component users.

The idea of our approach is to share local cloud component usage experience from different component users, to combine this local information to get a global QoS information of all components, and to make personalized QoS value prediction based on both global and local information. As shown in Fig. 2.1, each component user keeps local records of QoS usage experiences on cloud components. Since cloud applications are running on an identical cloud platform, QoS information can be collected by an identical interface on the platform side. If a component user would like to get personalized QoS information service from the cloud provider, authorization should be given to *Collector* for accessing its local QoS records. *Collector* then collects those local QoS records from different component users. Based on the collected QoS information, *Predictor* can perform personalized QoS value prediction and forward the prediction results to component users for optimizing the design of cloud applications. The detailed collaborative prediction approach will be presented in Sect. 2.3.

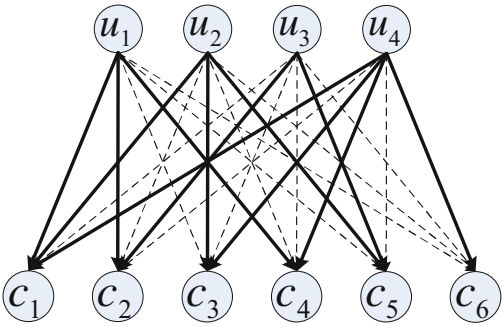
2.3 Collaborative QoS Prediction

We first formally describe the QoS value prediction problem on cloud components in Sect. 2.3.1. Then, we learn the user-specific and component-specific features by running latent features learning algorithm in Sect. 2.3.2. Based on the latent features, similarities between users and components are calculated in Sect. 2.3.3. Finally, the missing QoS values are predicted by applying the proposed algorithm CloudPred in Sect. 2.3.4.

2.3.1 Problem Description

Let us first consider a typical toy example in Fig. 2.2a. In this bipartite graph $G = (U \cup C, E)$, its vertices are divided into two disjoint sets U and C such that each edge in E connects a vertex in U and one in C. Let $U = \{u_1, u_2, \ldots, u_4\}$ be the set of component users, $C = \{c_1, c_2, \ldots, c_6\}$ denote the set of cloud components, and E (solid lines) represent the set of invocations between U and C. This bipartite graph G is modeled as a weighted directed graph. Given a pair (i, j), $u_i \in U$, and $c_j \in C$, edge e_{ij} is included in E if user u_i has invoked component c_j before. The weight w_{ij} on edge e_{ij} corresponds to the QoS value (e.g., response-time in this example) of that invocation. Given the set E, our task is to effectively predict the weight of potential invocations (the broken lines).

The process of cloud component QoS value prediction is illustrated by a user-component matrix as shown in Fig. 2.2b, in which each entry denotes an observed



(a) User-Component Invocation Graph

	c_1	c_2	c_3	c_4	c_5	c_6
u_1	0.98	0.23		0.22		
u_2	0.13		0.27		0.25	
u_3		0.37			0.36	
u_4	0.69		0.22	0.22		0.34

(b) User-Component Matrix

	c_1	c_2	c_3	c_4	c_5	c_6
u_1	0.98	0.23	**0.24**	0.22	**0.22**	**0.53**
u_2	0.13	**0.26**	0.27	**0.25**	0.25	**0.43**
u_3	**0.85**	0.37	**0.38**	**0.36**	0.36	**0.12**
u_4	0.69	**0.23**	0.22	0.22	**0.22**	0.34

(c) Predicted User-Component Matrix

Fig. 2.2 Toy example for QoS prediction. ©[2011] IEEE. Reprinted, with permission, from Ref. [8]

weight in Fig. 2.2a. The problem we study in this chapter is then how to precisely predict the missing entries in the user-component matrix based on the existing entries. Once the missing entries are accurately predicted, we can provide users with personalized QoS information, which is valuable for automatic component ranking, component selection, task scheduling, etc.

We observe that although about half of the entries are already known in Fig. 2.2b, every pair of users still have very few commonly invoked components (e.g., u_1 and u_2 only invoke c_1 in common, u_3 and u_4 have no commonly invoked components even if together they invoke all the six components). Since the similarity between two users are calculated by comparing their obtained QoS values on common components, the problem of few common components observed above makes it extremely difficult to precisely calculate similarity between users. Motivated by latent factor model [6], we therefore first factorize the sparse user-component matrix and then use $V^T H$ to approximate the original matrix, where the low-dimensional matrix V denotes the user latent feature space, and the low-dimensional matrix H represents the low-dimensional item latent feature space. The rows in V and H represent different features. Each column in V represents an user, and each column in H denotes a component. The value of a entry in the matrices indicates how the associated feature applies to the corresponding user or component. In this example, we use four dimensions to perform the matrix factorization and obtain:

$$V = \begin{bmatrix} 0.32 & 0.15 & 0.31 & 0.33 \\ 0.23 & 0.15 & 0.26 & 0.28 \\ 0.30 & 0.20 & 0.24 & 0.34 \\ 0.47 & 0.23 & 0.59 & 0.21 \end{bmatrix},$$

$$H = \begin{bmatrix} 0.73 & 0.35 & 0.31 & 0.26 & 0.32 & 0.42 \\ 0.60 & 0.31 & 0.27 & 0.22 & 0.28 & 0.36 \\ 0.69 & 0.37 & 0.32 & 0.27 & 0.33 & 0.45 \\ 0.95 & 0.46 & 0.42 & 0.35 & 0.41 & 0.54 \end{bmatrix},$$

where columns in V and H denote the latent feature vectors of users and components, respectively.

Note that V and H are dense matrices with all entries available. Then, we calculate the similarity between users and components using four-dimensional matrices V and H, respectively. Therefore, all the missing values can be predicted by employing neighborhood-based collaborative method, as shown in Fig. 2.2c.

Now, we formally define the problem of cloud component QoS value prediction as follows: Given a set of users and a set of components, predict the missing QoS value of components when invoked by users based on existing QoS values. More precisely:

> Let U be the set of m users and C be the set of n components. A QoS element is a triplet (i, j, q_{ij}) representing the observed quality of component c_j by user u_i, where $i \in \{1, \ldots, m\}$, $j \in \{1, \ldots, n\}$ and $q_{ij} \in \mathbb{R}^k$ is a k-dimensional vector representing the QoS values of k^{th} criteria. Let Ω be the set of all pairs $\{i, j\}$ and Λ be the set of all known pairs (i, j) in Ω. Consider a matrix $W \in \mathbb{R}^{m \times n}$ with each entry w_{ij} representing the observed k^{th} criterion value of component c_j by user u_i. Then, the missing entries $\{w_{ij} | (i, j) \in \Omega - \Lambda\}$ should be predicted based on the existing entries $\{w_{ij} | (i, j) \in \Lambda\}$.

Typically the QoS values can be integers from a given range (e.g., $\{0, 1, 2, 3\}$) or real numbers of a close interval (e.g., $[-20, 20]$). Without loss of generality, we can map the QoS values to the interval $[0, 1]$ using the function $f(x) = (x - w_{min})/(w_{max} - w_{min})$, where w_{max} and w_{min} are the maximum and minimum QoS values, respectively.

2.3.2 *Latent Features Learning*

In order to learn the features of the users and components, we employ matrix factorization to fit a factor model to the user-component matrix. This method focuses on filtering the user-component QoS value matrix using low-rank approximation. In other words, we factorize the QoS matrix into two low-rank matrices V and H. The idea behind the factor model is to derive a high-quality low-dimensional feature representation of users and components based on analyzing the user-component matrix. The premise behind a low-dimensional factor model is that there is only a small number of factors influencing QoS usage experiences and that a user's QoS

usage experience vector is determined by how each factor applies to that user and the items.

Consider the matrix $W \in \mathbb{R}^{m \times n}$ consisting of m users and n components. Let $V \in \mathbb{R}^{l \times m}$ and $H \in \mathbb{R}^{l \times n}$ be the latent user and component feature matrices. Each column in V represents the l-dimensional user-specific latent feature vector of a user, and each column in H represents the l-dimensional component-specific latent feature vector of a component. We employ an approximating matrix $\tilde{W} = V^T H$ to fit the user-item matrix W:

$$w_{ij} \approx \tilde{w}_{ij} = \sum_{k=1}^{l} v_{ki} h_{kj}, \tag{2.1}$$

The rank l of the factorization is generally chosen so that $(m + n)l < mn$, since V and H are low-rank feature representations [3]. The product $V^T H$ can be regarded as a compressed form of the data in W.

Note that the low-dimensional matrices V and H are unknown and need to be learned from the obtained QoS values in user-component matrix W. In order to optimize the matrix factorization, we first construct a cost function to evaluate the quality of approximation. The distance between two nonnegative matrices is usually employed to define the cost function. One useful measure of the matrices' distance is the Euclidean distance:

$$F(W, \tilde{W}) = \|W - \tilde{W}\|_F^2 = \sum_{ij} (w_{ij} - \tilde{w}_{ij})^2, \tag{2.2}$$

where $\|\cdot\|_F^2$ denotes the Frobenius norm.

In this chapter, we conduct matrix factorization as solving an optimization problem by employing the optimized objective function in [3]:

$$\begin{aligned} \min_{V,H} \quad & f(V, H) = \sum_{(i,j) \in \Lambda} [\tilde{w}_{ij} - w_{ij} \log \tilde{w}_{ij}], \\ s.t. \quad & \tilde{w}_{i,j} = \sum_{k=1}^{l} v_{ki} h_{kj}, \\ & V \geq 0, \\ & H \geq 0. \end{aligned} \tag{2.3}$$

where $V, H \geq 0$ is the nonnegativity constraints leading to allow only additive combination of features.

In order to minimize the objective function in Eq. (2.3), we apply incremental gradient descent method to find a local minimum of $f(V, H)$, where one gradient step intends to decrease the square of prediction error of only one rating, that is, $\tilde{w}_{ij} - w_{ij} \log \tilde{w}_{ij}$. We update the V and H in the direction opposite of the gradient in each iteration:

$$v_{ij} = v_{ij} \sum_k \frac{w_{ik}}{\tilde{w}_{ik}} h_{jk}, \tag{2.4}$$

$$h_{ij} = h_{ij} \sum_k \frac{w_{ik}}{\tilde{w}_{ik}} v_{jk}, \tag{2.5}$$

$$v_{ij} = \frac{v_{ij}}{\sum_k v_{kj}}, \tag{2.6}$$

$$h_{ij} = \frac{h_{ij}}{\sum_k h_{kj}}. \tag{2.7}$$

Algorithm 1 shows the iterative process for latent feature learning. We first initialize matrices V and H with small random nonnegative values. Iteration of the above update rules converges to a local minimum of the objective function given in Eq. (2.3).

Algorithm 1: Latent Features Learning Algorithm

Input: W, l
Output: V, H
Initialize $V \in \mathbb{R}^{l \times m}$ and $H \in \mathbb{R}^{l \times n}$ with small random numbers;
repeat
  for *all* $(i, j) \in \Lambda$ **do**
    $\tilde{w}_{ij} = \sum_k v_{ki} h_{kj}$;
  end
  for *all* $(i, j) \in \Lambda$ **do**
    $v_{ij} \leftarrow v_{ij} \sum_k \frac{w_{ik}}{\tilde{w}_{ik}} h_{jk}$;
    $h_{ij} \leftarrow h_{ij} \sum_k \frac{w_{ik}}{\tilde{w}_{ik}} v_{jk}$;
    $v_{ij} = \frac{v_{ij}}{\sum_k v_{kj}}$;
    $h_{ij} = \frac{h_{ij}}{\sum_k h_{kj}}$;
  end
  for *all* $(i, j) \in \Lambda$ **do**
    $\tilde{w}_{ij} = \sum_k v_{ki} h_{kj}$;
  end
until *Converge*;

2.3.3 Similarity Computation

Given the latent user and component feature matrices V and H, we can calculate the neighborhood similarities between different users and components by employing Pearson correlation coefficient (PCC) [5]. PCC is widely used in memory-based recommendation systems for similarity computation. Due to the high accuracy, we

adopt PCC in this chapter for the neighborhood similarity computation on both sets of users and components. The similarity between two users u_i and u_j is defined by performing PCC computation on their l-dimensional latent feature vectors V_i and V_j with the following equation:

$$S(u_i, u_j) = \frac{\sum_{k=1}^{l}(v_{ik} - \overline{v}_i)(v_{jk} - \overline{v}_j)}{\sqrt{\sum_{k=1}^{l}(v_{ik} - \overline{v}_i)^2}\sqrt{\sum_{k=1}^{l}(v_{jk} - \overline{v}_j)^2}}, \tag{2.8}$$

where $v_i = (v_{i1}, v_{i2}, \ldots, v_{il})$ is the latent feature vector of user u_i and v_{ik} is the weight on the kth feature. $\overline{v}_i$ is the average weight on l-dimensional latent features for user u_i. The similarity between two users $S(i, j)$ falls into the interval $[-1, 1]$, where a larger value indicates higher similarity.

Similar to the user similarity computation, we also employ PCC to compute the similarity between component c_i and item c_j as following:

$$S(c_i, c_j) = \frac{\sum_{k=1}^{l}(h_{ik} - \overline{h}_i)(h_{jk} - \overline{h}_j)}{\sqrt{\sum_{k=1}^{l}(h_{ik} - \overline{h}_i)^2}\sqrt{\sum_{k=1}^{l}(h_{jk} - \overline{h}_j)^2}}, \tag{2.9}$$

where $h_i = (h_{i1}, h_{i2}, \ldots, h_{il})$ is the latent feature vector of component c_i and h_{ik} is the weights on the k^{th} feature. $\overline{h}_i$ is the average weight on l-dimensional latent features for component c_i.

2.3.4 Missing QoS Value Prediction

After computing the similarities between users, we can identify similar neighbors to the current user by ordering similarity values. Note that PCC value falls into the interval $[-1, 1]$, where a positive value means similar and a negative value denotes dissimilar. In practice, QoS usage experience of less similar or dissimilar users may greatly decrease the prediction accuracy. In this chapter, we exclude those users with negative PCC values from the similar neighbor set and only employ the QoS usage experiences of users with Top-K largest PCC values for predicting QoS value of the current user. We refer to the set of Top-K similar users for user u_i as Ψ_i, which is defined as:

$$\Psi_i = \{u_k | S(u_i, u_k) > 0, rank_i(k) \leq K, k \neq i\}, \tag{2.10}$$

where $rank_i(k)$ is the ranking position of user u_k in the similarity list of user u_i, and K denotes the size of set Ψ_i.

Similarly, a set of Top-K similar components for component c_j can be denote as Φ_j by:

$$\Phi_j = \{c_k | S(c_j, c_k) > 0, rank_p(k) \leq K, k \neq j\}, \tag{2.11}$$

where $rank_j(k)$ is the ranking position of component c_k in the similarity list of component c_j, and K denotes the size of set Φ_j.

To predict the missing entry w_{ij} in the user-component matrix, user-based approaches employ the values of entries from Top-K similar users as follows:

$$w_{ij} = \overline{w}_i + \sum_{k \in \Psi_i} \frac{S(u_i, u_k)}{\sum_{a \in \Psi_i} S(u_i, u_a)} (w_{kj} - \overline{w}_k), \tag{2.12}$$

where $\overline{w}_i$ and $\overline{w}_k$ are the average observed QoS values of different components by users u_i and u_k, respectively.

For component-based approaches, entry values of Top-K similar components are employed for predicting the missing entry w_{ij} in the similar way:

$$w_{ij} = \overline{w}_j + \sum_{k \in \Phi_j} \frac{S(i_j, i_k)}{\sum_{a \in \Phi_j} S(i_j, i_a)} (w_{ik} - \overline{w}_k), \tag{2.13}$$

where $\overline{w}_j$ and $\overline{w}_k$ are the average available QoS values of component c_j and c_k by different users, respectively.

In user-component-based approaches, the predicted values in Eqs. (2.12) and (2.13) are both employed for more precise prediction in the following equation:

$$w^*_{ij} = \lambda \times w^u_{ij} + (1 - \lambda) \times w^c_{ij}, \tag{2.14}$$

where w^u_{ij} denotes the predicted value by user-based approach and w^c_{ij} denotes the predicted value by component-based approach. The parameter λ controls how much the hybrid prediction results rely on user-based approach or component-based approach. The proper value of λ can be trained on a small sample dataset extracted from the original one. We summarize the proposed algorithm in Algorithm 2.

2.4 Experiments

In this section, in order to show the prediction quality improvements of our proposed approach, we conduct several experiments to compare our approach with several state-of-the-art collaborative filtering prediction methods.

In the following, Sect. 2.4.1 gives the description of our experimental dataset, Sect. 2.4.2 defines the evaluation metrics, Sect. 2.4.3 compares the prediction quality of our approach with some other methods, and Sects. 2.4.4, 2.4.5, and 2.4.6 study the impact of training data density, Top-K, and dimensionality, respectively.

Algorithm 2: CloudPred Prediction Algorithm

Input: W, l, λ
Output: W^*

Learn V and H by applying Algorithm 1 on W;
for *all* $(u_i, u_j) \in U \times U$ **do**
  calculate the similarity $S(u_i, u_j)$ by Eq. (2.8);
end
for *all* $(c_i, c_j) \in C \times C$ **do**
  calculate the similarity $S(c_i, c_j)$ by Eq. (2.9);
end
for *all* $(i, j) \in \Lambda$ **do**
  construct similar user set Ψ_i by Eq. (2.10);
  construct similar component set Φ_j by Eq. (2.11);
end
for *all* $(i, j) \in \Omega - \Lambda$ **do**
  calculate w_{ij}^u by Eq. (2.12);
  calculate w_{ij}^i by Eq. (2.13);
  $w_{ij}^* = \lambda \times w_{ij}^u + (1 - \lambda) \times w_{ij}^c$;
end

2.4.1 Dataset Description

In real world, invoking thousands of commercial cloud components for large-scale experiments is very expensive. In order to evaluate the prediction quality of our proposed approach, we conduct experiments on our Web service QoS dataset [9]. Web service, a kind of cloud component, can be integrated into cloud applications for accessing information or computing service from a remote system. The Web service QoS dataset includes QoS performance of 5825 openly accessible real-world Web services from 73 countries. The QoS values are observed by 339 distributed computers located in 30 countries from PlanetLab, which is a distributed test bed consisting of hundreds of computers all over the world. In our experiment, each of the 339 computers keeps invocation records of all the 5825 Web services by sending null operating requests to capture the characteristics of communication links. Totally 1,974,675 QoS performance results are collected. Each invocation record is a k-dimensional vector representing the QoS values of k criteria. We then extract a set of 339×5825 user-component matrices, each of which stands for a particular QoS property, from the QoS invocation records. For simplicity, we use two matrices, which represent response-time and throughput QoS criteria, respectively, for experimental evaluation in this chapter. Without loss of generality, our approach can be easily extended to include more QoS criteria.

The statistics of Web service QoS dataset are summarized in Table 2.1. Response-time and throughput are within the range 0–20 s and 0–1000 kbps, respectively. The

Table 2.1 Statistics of WS QoS dataset. ©[2011] IEEE. Reprinted, with permission, from Ref. [8]

Statistics	Response-time	Throughput
Scale	0–20 s	0–1000 kbps
Mean	0.910 s	47.386 kbps
Num. of users	339	339
Num. of web services	5828	5828
Num. of records	1,974,675	1,974,675

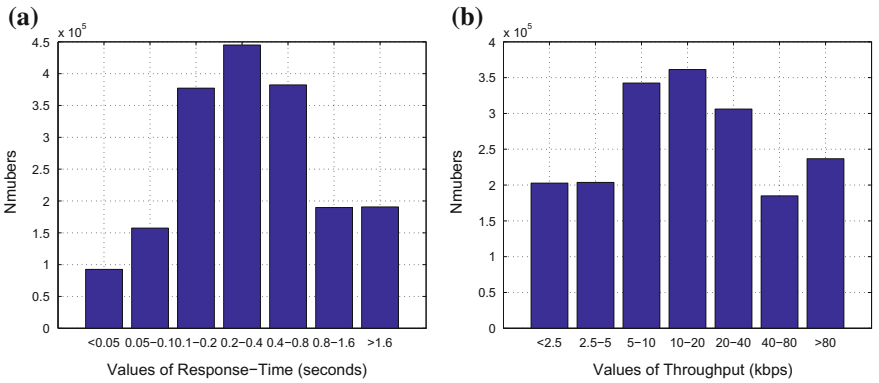


Fig. 2.3 Value distributions. ©[2011] IEEE. Reprinted, with permission, from Ref. [8]

means of response-time and throughput are 0.910 s and 47.386 kbps, respectively. Figure 2.3 shows the distributions of response-time and throughput. Most of the response-time values are between 0.1–0.8 s, and most of the throughput values are between 5–40 kbps.

2.4.2 Metrics

We assess the prediction quality of our proposed approach in comparison with other methods by computing mean absolute error (MAE) and root-mean-squared error (RMSE). The metric MAE is defined as:

$$MAE = \frac{\sum_{i,j} |w_{ij} - w^*_{ij}|}{N}, \tag{2.15}$$

and RMSE is defined as:

$$RMSE = \sqrt{\frac{\sum_{i,j} (w_{ij} - w^*_{ij})^2}{N}}, \tag{2.16}$$

where w_{ij} is the QoS value of Web service c_j observed by user u_i, w^*_{ij} denotes the QoS value of Web service c_j would be observed by user u_i as predicted by a method, and N is the number of predicted QoS values.

2.4.3 Performance Comparison

In this section, we compare the prediction accuracy of our proposed approach CloudPred with some state-of-the-art approaches:

1. UPCC (User-based collaborative filtering method using Pearson correlation coefficient): this method employs PCC to calculate similarities between users and predicts QoS value based on similar users [1, 7].
2. IPCC (Item-based collaborative filtering method using Pearson correlation coefficient): this method employs PCC to calculate similarities between Web services and predicts QoS value based on similar items (item refers to component in this chapter) [5].
3. UIPCC (User-item-based collaborative filtering method using Pearson correlation coefficient): this method is proposed by Ma et al. in [4]. It combines UPCC and IPCC approaches and predicts QoS value based on both similar users and similar Web services.
4. NMF (Nonnegative Matrix Factorization): This method is proposed by Lee and Seung in [3]. It applies nonnegative matrix factorization on user-item matrix for missing value prediction.

In this chapter, in order to evaluate the performance of different approaches in reality, we randomly remove some entries from the matrices and compare the values predicted by a method with the original ones. The matrices with missing values are in different sparsity. For example, 10% means that we randomly remove 90% entries from the original matrix and use the remaining 10% entries to predict the removed entries. The prediction accuracy is evaluated using Eqs. (2.15) and (2.16) by comparing the original value and the predicted value of each removed entry. Our proposed approach CloudPred performs matrix factorization in Sect. 2.3.2 and employs both similar users and similar Web services for predicting the removed entries. The parameter settings of our approach CloudPred are Top-K = 10, dimensionality = 20, and $\lambda = 0.5$ in the experiments. Detailed impact of parameters will be studied in Sects. 2.4.4, 2.4.5 and 2.4.6.

The experimental results are shown in Table 2.2. For each row in the table, we highlight the best performer among all methods. From Table 2.2, we can observe that our approach CloudPred obtains better prediction accuracy (smaller MAE and RMSE values) than other methods for both response-time and throughput under different matrix densities. The MAE and RMSE values of dense matrices (e.g., matrix density is 80 or 90%) are smaller than those of sparse matrices (e.g., matrix density is 10 or 20%), since a denser matrix provides more information for predicting the missing values. In general, the MAE and RMSE values of throughput are larger than those

Table 2.2 Performance comparisons (A smaller MAE or RMSE value means a better performance). ©[2011] IEEE. Reprinted, with permission, from Ref. [8]

Matrix density (%)	Metrics	Response-time (seconds)				
		IPCC	UPCC	UIPCC	NMF	CloudPred
10	MAE	0.7596	0.5655	0.5654	0.6754	**0.5306**
	RMSE	1.6133	1.3326	1.3309	1.5354	**1.2904**
20	MAE	0.7624	0.5516	0.5053	0.6771	**0.4745**
	RMSE	1.6257	1.3114	1.2486	1.5241	**1.1973**
80	MAE	0.6703	0.4442	0.3873	0.3740	**0.3704**
	RMSE	1.4102	1.1514	1.0785	1.1242	**1.0597**
90	MAE	0.6687	0.4331	0.3793	0.3649	**0.3638**
	RMSE	1.4173	1.1264	1.0592	1.1121	**1.0359**
Matrix density (%)	Metrics	Throughput (kbps)				
		IPCC	UPCC	UIPCC	NMF	CloudPred
10	MAE	31.6722	26.2015	22.6567	19.7700	**19.0009**
	RMSE	65.5220	61.9658	57.4653	57.3767	**51.8236**
20	MAE	35.1780	21.9313	18.1230	15.7794	**15.4203**
	RMSE	66.6028	56.5441	50.0435	50.1402	**44.8975**
80	MAE	29.9146	14.5497	12.4880	12.5107	**10.7881**
	RMSE	64.3079	44.3738	39.6017	39.2029	**36.8506**
90	MAE	29.9404	13.8761	12.0662	11.6960	**10.4722**
	RMSE	63.7149	42.5534	38.0763	36.7555	**35.9225**

of response-time because the scale of throughput is 0–1000 kbps, while the scale of response-time is 0–20 s. Compared with other methods, the improvements of our approach CloudPred are significant, which demonstrates that the idea of combining global and local information for QoS prediction is realistic and reasonable.

2.4.4 Impact of Matrix Density

In Fig. 2.4, we compare the prediction accuracy of all the methods under different matrix densities. We change the matrix density from 10 to 90% with a step value of 10%. The parameter settings in this experiment are Top-K = 10, dimensionality = 20, and $\lambda = 0.5$.

Figure 2.4a, b shows the experimental results of response-time, while Fig. 2.4c, d shows the experimental results of throughput. The experimental results show that our approach CloudPred achieves higher prediction accuracy than other competing methods under different matrix density. In general, when the matrix density is

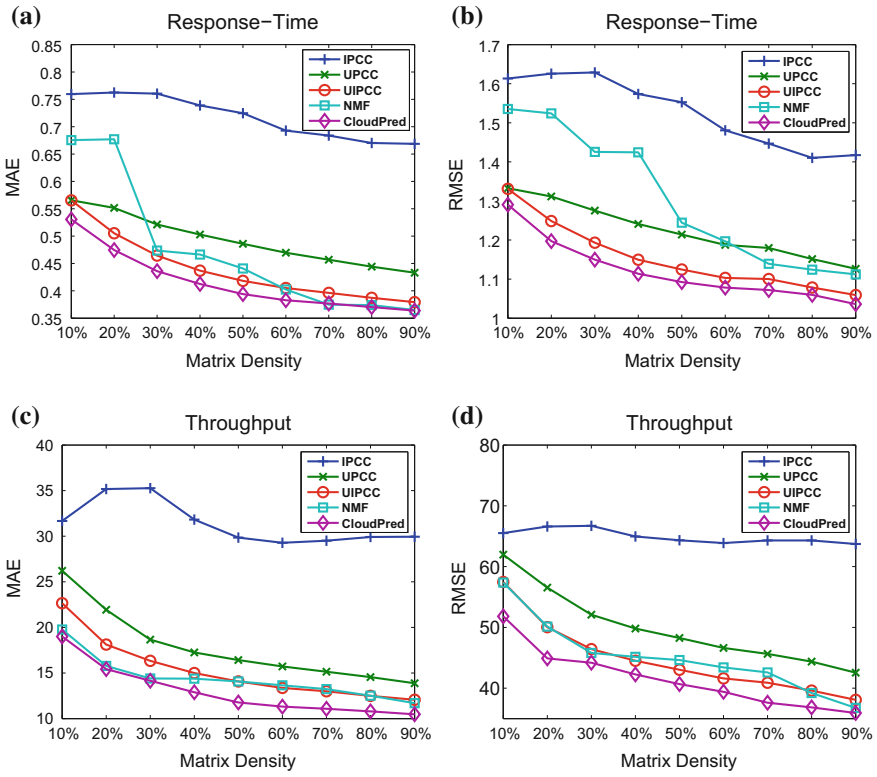


Fig. 2.4 Impact of matrix density. ©[2011] IEEE. Reprinted, with permission, from Ref. [8]

increased from 10 to 30%, the prediction accuracy of our approach CloudPred is significantly enhanced. When the matrix density is further increased from 30 to 90%, the enhancement of prediction accuracy is quite limited. This observation indicates that when the matrix is very sparse, collecting more QoS information will greatly enhance the prediction accuracy, which further demonstrates that sharing local QoS information among cloud component users could effectively provide personalized QoS estimation.

In the experimental results, we observe that the performance of IPCC is much worse than that of other methods. The reason is that in our Web service dataset, the number of users, which is 339, is much smaller than the number of components, which is 5258. When some entries are removed from the user-component matrices, the number of common users between two components, on average, is very small, which would greatly impact the accuracy of common user-based similarity computation between components. Therefore, the prediction accuracy of similar item-based method IPCC is greatly decreased by the inaccuracy similarity computation between components.

2.4.5 *Impact of Top-K*

The parameter Top-K determines the size of similar user and similar component sets. In Fig. 2.5, we study the impact of parameter Top-K by varying the values of Top-K from 10 to 50 with a step value of 10. Other parameter settings are dimensionality = 10 and $\lambda = 0.5$.

Figure 2.5a, b shows the MAE and RMSE results of response-time, respectively, while Fig. 2.5c, d shows the MAE and RMSE results of throughput, respectively. The experimental results show that our approach CloudPred achieves best prediction accuracy(smallest MAE and RMSE values) when Top-K is set around 10. Under both sparse matrix, whose density is 10%, and dense matrix, whose density is 90%, all the prediction accuracies decrease when we decrease the Top-K value from 10 to 2 or increase from 10 to 18. This is because too small Top-K value will exclude useful information from some similar users and similar components, while too large Top-K value will introduce noise from dissimilar users and dissimilar components, which will impact the prediction accuracy.

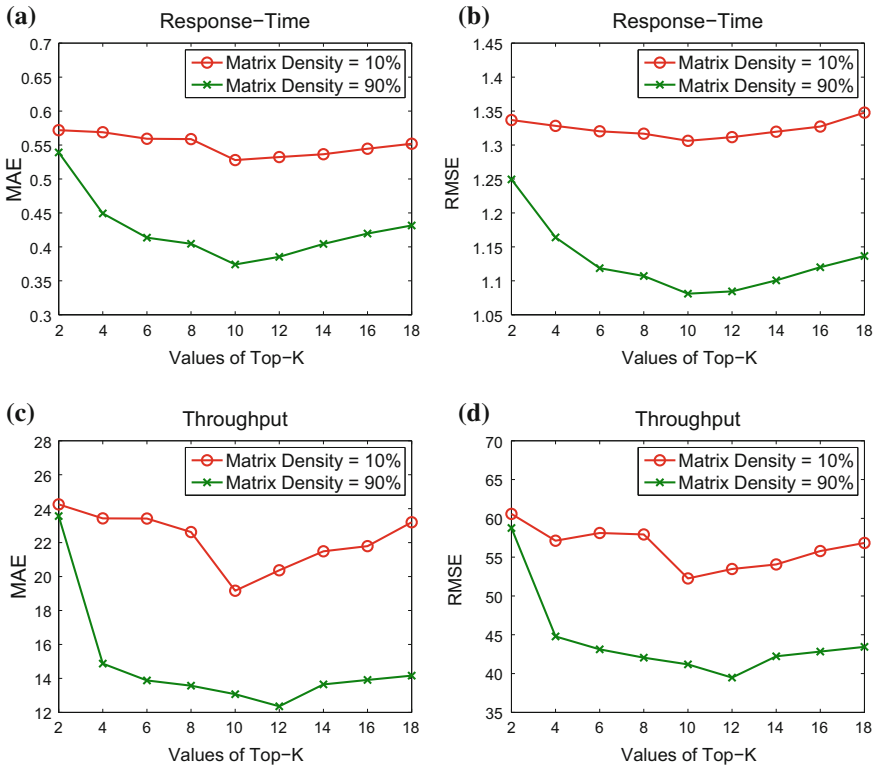


Fig. 2.5 Impact of Top-K. ©[2011] IEEE. Reprinted, with permission, from Ref. [8]

2.4.6 Impact of Dimensionality

The parameter dimensionality determines the number of latent features used to characterize user and cloud component. In Fig. 2.6, we study the impact of parameter dimensionality by varying the values of dimensionality from 10 to 50 with a step value of 10. Other parameter settings are Top-K = 10 and $\lambda = 0.5$.

Figure 2.6e, f shows the MAE and RMSE values of response-time, while Fig. 2.6g, h shows the MAE and RMSE values of throughput. When the matrix density is 90%, we observe that our approach CloudPred achieves the best performance when the value of dimensionality is 30, while smaller values like 10 or larger values like 50 can potentially hurt the prediction accuracy. This observation indicates that when the user-component matrices are dense, 10 latent factors is not enough to characterize the features of user and component which are mined from the rich QoS information, while 50 latent factors is too many since it will cause overfitting problem. When the matrix density is 10%, we observed that the prediction accuracy of our approach CloudPred decreases (MAE and RMSE increase) when the value of dimensionality is increased from 10 to 50. This observation indicates that when the user-component matrices are sparse, 10 latent factors is already enough to characterize the features of user and component which are mined from the limited user-component QoS information, while other larger values of dimensionality will cause the overfitting problem.

2.4.7 Impact of λ

The parameter λ determines how much the final prediction results rely on user-based approach or component-based approach. A larger value of λ means user-based approach contributes more to the hybrid prediction. A smaller value of λ means component-based approach contributes more to the hybrid prediction. In Fig. 2.7, we study the impact of parameter λ by varying the values of λ from 0 to 1 with a step value of 0.1. Other parameter settings are dimensionality = 10 and Top-K = 10.

Figure 2.7a, b shows the MAE and RMSE results of response-time, respectively. The experimental results show that the value of λ impacts the recommendation results significantly, which demonstrates that combining the user-based approach and component-based approach improves the recommendation accuracy. The prediction accuracies increase when we increase the value of λ at first. But when λ surpasses a certain threshold, the prediction accuracy decreases with further increase of the value of λ. This phenomenon coincides with the intuition that purely using the user-based approach or purely using the component-based approach cannot generate better results than the hybrid approach. From Fig. 2.7, we observed that when $\lambda \in [0.4, 0.7]$, CloudPred achieves the best performance, while a smaller value or a larger value can potentially degrade the prediction performance. Moreover, the insensitivity of the optimal value of λ shows that the parameter of CloudPred is easy to train.

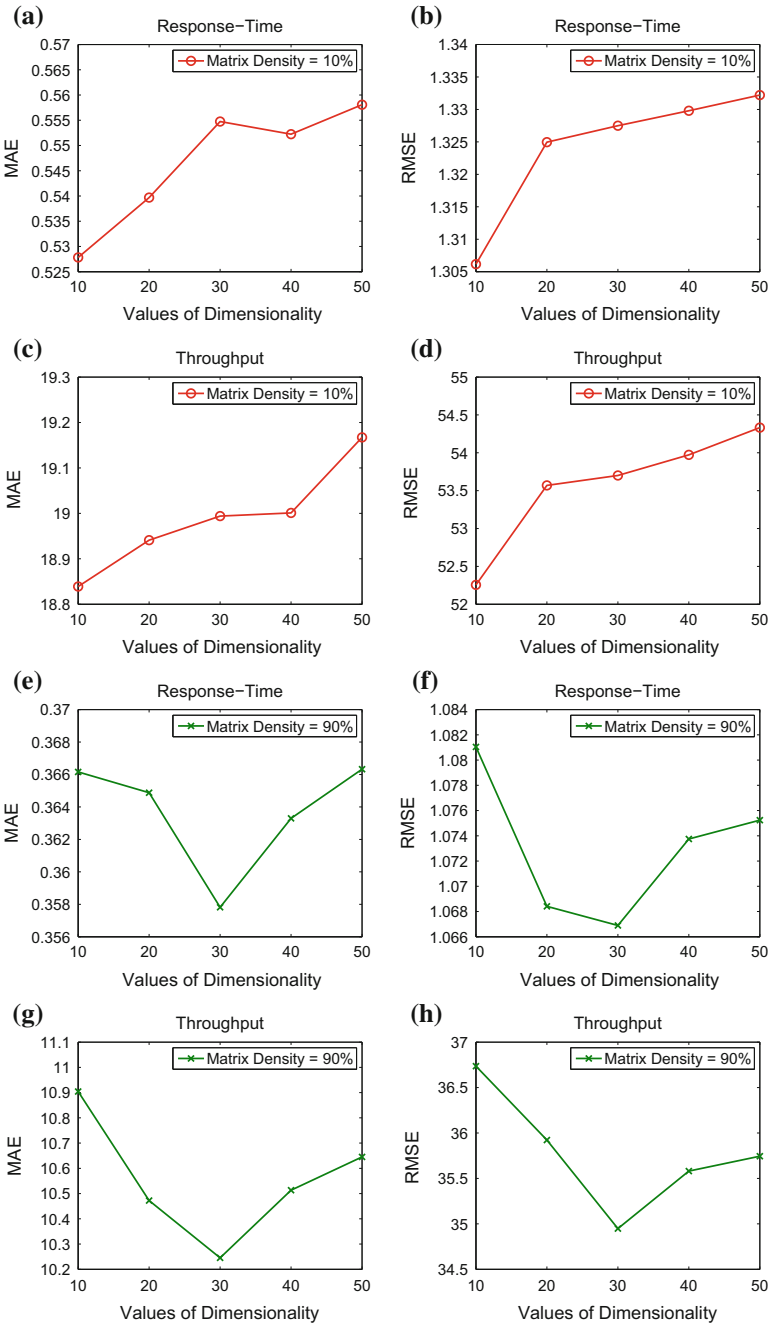


Fig. 2.6 Impact of dimensionality. ©[2011] IEEE. Reprinted, with permission, from Ref. [8]

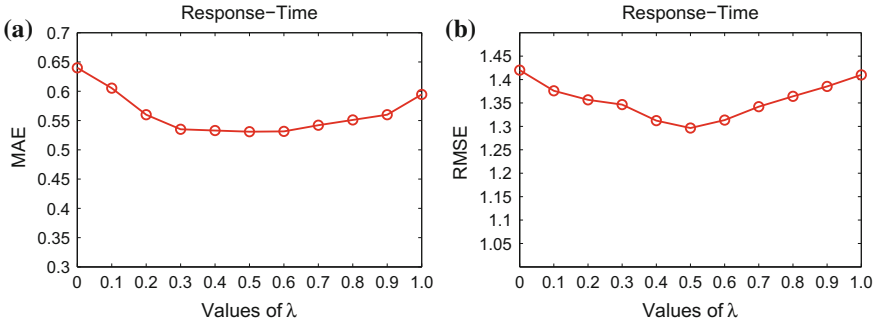


Fig. 2.7 Impact of λ. ©[2011] IEEE. Reprinted, with permission, from Ref. [8]

2.5 Summary

Based on the intuition that a user's cloud component QoS usage experiences can be predicted by exploring the past usage experience from both the user and its similar users, we propose a novel neighborhood-based approach, which is enhanced by feature modeling on both user and component, called CloudPred, for collaborative and personalized QoS value prediction on cloud components. Requiring no additional invocation of cloud components, CloudPred makes the QoS value prediction by taking advantage of both local usage information from similar users and similar components and global invocation information shared by all the users. The extensive experimental results show that our approach CloudPred achieves higher prediction accuracy than other competing methods.

Since the Internet environment is highly dynamic, the QoS performances of a cloud component may be variable against time (e.g., due to the network traffic, server workload). In our current approach, the QoS values are observed over a long period, which represent the average QoS performance of cloud components. Since the average QoS performance of cloud components is relatively stable, the predicted QoS values provide valuable information of unused cloud components for the users. In our future work, we will explore an online prediction algorithm to handle the dynamically changing QoS values by fusing with the time information.

Currently, we are collecting QoS information of Web service, which is a kind of cloud component. In the future, we will conduct more experiments to evaluate our approach in commercial clouds which contain multiple kinds of cloud components. For future work, we will investigate more techniques for improving the similarity computation (e.g., clustering models, latent factor models, data smoothing). We will also conduct more investigations on the correlations and combinations of different QoS properties.

References

1. J. Breese, D. Heckerman, C. Kadie, Empirical analysis of predictive algorithms for collaborative filtering, in *Proceedings of UAI'98* (1998), pp. 43–52
2. B. Hayes, Cloud computing. Commun. ACM **51**(7), 9–11 (2008)
3. D. Lee, H. Seung, Learning the parts of objects by non-negative matrix factorization. Nature **401**(6755), 788–791 (1999)
4. H. Ma, I. King, M. Lyu, Effective missing data prediction for collaborative filtering, in *Proceeding of SIGIR'07* (2007), pp. 39–46
5. P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, J. Riedl, GroupLens: an open architecture for collaborative filtering of netnews, in *Proceeding of CSCW'94* (1994), pp. 175–186
6. R. Salakhutdinov, A. Mnih, Probabilistic matrix factorization. Adv. Neural Inf. Process. Syst. (NIPS) **20**, 1257–1264 (2008)
7. L. Shao, J. Zhang, Y. Wei, J. Zhao, B. Xie, H. Mei, Personalized QoS prediction for web services via collaborative filtering, in *Proceeding of ICWS'07* (2007), pp. 439–446
8. Y. Zhang, Z. Zheng, M.R. Lyu, Exploring latent features for memory-based QoS prediction in cloud computing, in *IEEE Symposium on Reliable Distributed Systems (SRDS)* (IEEE, 2011), pp. 1–10
9. Z. Zheng, Y. Zhang, M. Lyu, Distributed QoS evaluation for real-world web services, in *Proceeding of ICWS'10* (2010), pp. 83–90

http://www.springer.com/978-981-10-5277-4

QoS Prediction in Cloud and Service Computing
Approaches and Applications
Zhang, Y.; Lyu, M.R.
2017, XI, 122 p. 41 illus., 12 illus. in color., Softcover
ISBN: 978-981-10-5277-4