JCSE Journal of Computing Science and Engineering

Regular Paper

Journal of Computing Science and Engineering,
Vol. 11, No. 3, September 2017, pp. 100-108

Response Time Prediction of IoT Service Based on Time Similarity

Huaizhou Yang* and Li Zhang
School of Computer Science, Xi'an Shiyou University, Xi'an, China
hzyang@xsyu.edu.cn, zhangli0223@163.com

Abstract
In the field of Internet of Things (IoT), smarter embedded devices offer functions via web services. The Quality-of-Service (QoS) prediction is a key measure that guarantees successful IoT service applications. In this study, a collaborative filtering method is presented for predicting response time of IoT service due to time-awareness characteristics of IoT. First, a calculation method of service response time similarity between different users is proposed. Then, to improve prediction accuracy, initial similarity values are adjusted and similar neighbors are selected by a similarity threshold. Finally, via a densified user-item matrix, service response time is predicted by collaborative filtering for current active users. The presented method is validated by experiments on a real web service QoS dataset. Experimental results indicate that better prediction accuracy can be achieved with the presented method.

Category: Ubiquitous computing

Keywords: Internet of Things; Web service; QoS prediction; Collaborative filtering

I. INTRODUCTION

With implementation and development of the Internet of Things (IoT), more IoT functions are provided via web services. This technology trend is mainly demonstrated in two aspects. First, in IoT environment, various embedded devices are deployed, forming IoT smart objects [1]. These smart objects have capability of data preprocessing, data storage, collaborative communication and context awareness. To shield complexity of knowledge representation and communication interface, web service standards are implemented on relatively resource-constrained devices by simplification and optimization. For example, device profiles for web services (DPWS) [2], as a subset of web service standards, is implemented in many embedded devices for seamless service interaction. Second, in the aspect of IoT infrastructure, Service-Oriented-Architecture (SOA) is widely accepted as IoT middleware solutions to solve issues of abstracting device functionalities and communication capabilities [3, 4]. Based on SOA, different IoT functions derived from smart objects, software entities and networks may be uniformly represented as web services, and new IoT applications may be constructed rapidly by service composition. The borderline between real and virtual world is blurred.

Extensive use of web services makes IoT QoS management extremely crucial that is the basic guarantee of successful service discovery, query, selection and on-demand provision. However, obtaining values of user-dependent QoS properties (e.g., response time and failure rate) is difficult, since these property values are different for different users and change dynamically in the real word. It is not an effective method to evaluate IoT services via real-word service invocations because this process is resource and time-consuming. Especially, when there are too many services in an IoT system, service users cannot select all suitable services for evaluation by direct service invocations. QoS prediction is propitious to solve these

Open Access http://dx.doi.org/10.5626/JCSE.2017.11.3.100 http://jcse.kiise.org

Received 23 August 2017; **Revised** 13 September 2017; **Accepted** 15 September 2017
*Corresponding Author

 pISSN: 1976-4677 eISSN: 2093-8020

issues above mentioned. Via QoS prediction results, a service user may select QoS-satisfied and optimal services before real service invocations, and meanwhile, the IoT system can actively recommend suitable services for current users according to their characteristics.

Various methods have been applied to predict QoS for service users. In a study of [5], runtime QoS of composite service is predicted by an autoregressive moving average model and QoS reduction rules. A similar study is conducted by [6]. They predict dynamic QoS of composite service by a probabilistic model and graph reduction algorithm. In addition, Markov model and Bayesian networks are used in the study of [7] for service modeling and response time prediction. However, these solutions are only suitable for QoS prediction of composite service but unsuitable for individual component service. Because QoS values usually change with environment factors (e.g., network delay, user location and service location), for predicting QoS of component services, many methods are proposed to improve QoS prediction accuracy by considering this environment information. In the study of [8], physical location of service provider and network status are considered for accurate QoS predictions. Geographical information of users is used to improve QoS prediction accuracy by [9]. Because these methods need special environment information, their universality is restricted.

In present studies, collaborative filtering technology is widely accepted for QoS prediction that has proven its validity for service recommendation in the traditional e-commerce field [10]. Methods of collaborative filtering can automatically predict personalized QoS values for current users by using information from similar users or service items [11]. Typical collaborative filtering methods include memory-based method and model-based method. Memory-based method can be divided as three types: user-based method, item-based method and their mixture [12]. By using the memory-based method, we can predict QoS values in an intuitive and comprehensible way. However, this kind of method has a drawback in that it is difficult to find similar users or service items when the user-item matrix is sparse. The model-based method usually predefines a domain model for a given system, and then trains the model by existing training datasets for further QoS prediction. This kind of method can incorporate special domain knowledge and multiform artificial intelligence techniques to achieve higher prediction accuracy [13, 14], but meanwhile has drawbacks of high complexity and limited universality.

Being different from the related studies, our study focuses on response time prediction of IoT service due to significance in many QoS properties. Unlike traditional service composition systems, most IoT systems are time-sensitive or even real-time systems [15]. Distributed real-time interaction between services and users is the main characteristic of IoT system [16]. Therefore, accurate service response time prediction becomes a major challenge for successful IoT service applications. To avoid expensive real service invocations, we use historical service response time records to make predictions for current users by collaborative filtering. In contrast to classical prediction methods, a novel method of response time prediction is presented in this study that is more suitable for response time prediction due to great change of response time values in the real world. To depress the negative influence of dissimilar neighbors, we adjust the initial calculation result of similarity, and select satisfied similar neighbors by a threshold. Focusing on the issue of sparse matrix, we densify the user-item matrix by missing value predictions of matrix elements. Validity of our prediction method is proven by multiple experiments on a real QoS dataset.

The main contribution of this study is twofold:

- We combine user-based and item-based methods of collaborative filtering technology to make service response time prediction, and meanwhile, adaptively balance prediction dependence degree on these two methods according to time similarity values.
- Focusing on the characteristic that the IoT service response time values change greatly, we propose multiple prediction accuracy improvement methods, including similarity adjustment, threshold filtering and user-item matrix densification.

II. TIME SIMILARITY CALCULATION OF IOT SERVICE RESPONSE TIME

A. The Construction of User-Item Matrix

Historical data of IoT service response time can be obtained through many ways, such as user logs or service invocation records provided by corresponding IoT middlewares. Using these historical data, we can construct an $M \times N$ user-item matrix $A = (a_{ij})$. This matrix consists of M service users $U = \{u_1, u_2, \ldots u_M\}$ and N service items $S = \{s_1, s_2, \ldots s_N\}$. The matrix element a_{ij} represents the response time value of service s_j observed by user u_i. When the user u_i never used service s_j before $a_{ij} = null$.

User-item matrix is the data source of further response time prediction by collaborative filtering. The more historical service response time data is collected from different users, the more accurate prediction results can be obtained. Fortunately, if there is an IoT service user wants to yield better prediction results, he or she needs to provide more past service usage data as much as possible. Therefore, with accumulation of historical data provided by various users, service response time prediction will become more accurate.

B. The Calculation of Time Similarity

Collaborative filtering technology usually uses user-based and item-based methods to make service predictions

and service recommendations. Through these two methods, historical information of similar users or similar service items is used to predict unknown information for current active users. In this study, we select Pearson's correlation coefficient to calculate similarity between users or service items. Similarity between user u and v is calculated as follows.

$$Sim(u, v) = \frac{\sum_{s \in S'}(a_{us} - \overline{a}_u)(a_{vs} - \overline{a}_v)}{\sqrt{\sum_{s \in S'}(a_{us} - \overline{a}_u)^2}\sqrt{\sum_{s \in S'}(a_{vs} - \overline{a}_v)^2}} \quad (1)$$

In formula (1), a_{us} and a_{vs} are the response time values of service item s observed by user u and v, respectively. $\overline{a}_u$ and $\overline{a}_v$ are the average response time values of all services observed by user u and v, respectively. $S' = S_u \cap S_v \subseteq S$ is a set of service items that have been invoked by user u and v previously. When $S' = \varnothing$, i.e., when there are no same service items observed by user u and v, $Sim(u, v) = null$. The similarity between service item x and y is calculated as follows.

$$Sim(x, y) = \frac{\sum_{u \in U'}(a_{ux} - \overline{a}_x)(a_{uy} - \overline{a}_y)}{\sqrt{\sum_{u \in U'}(a_{ux} - \overline{a}_x)^2}\sqrt{\sum_{u \in U'}(a_{uy} - \overline{a}_y)^2}} \quad (2)$$

In formula (2), a_{ux} and a_{uy} are the response time values of service item x and y, respectively, observed by user u. $\overline{a}_x$ and $\overline{a}_y$ are average response time values of service item x and y, respectively, observed by all users. $U' = U_x \cap U_y \subseteq U$ is a set of users that have invoked both service items x and y previously. When $U' = \varnothing$, i.e., when not any of users has invoked service item x and y previously, $Sim(x, y) = null$.

It is suitable to calculate time similarity by Pearson's correlation coefficient, because this method is not sensitive to change of absolute value between two vectors. For example, similarity between vector (2,4,3,9) and (4,8,6,18) is 1 through calculation of Pearson's correlation coefficient, although values of vector one are only half of vector two. IoT service response time values often change greatly due to many factors, such as communication bandwidth and user location. Therefore, response time values observed by different users on the same service item usually differ largely. But through time similarity, we can find potential and valuable information to make service response time predictions.

C. The Adjustment of Time Similarity

The overestimation problem often occurs in traditional similarity calculation process. When the amount of the same services invoked by two users is much less than the total amount of services invoked by these two users, similarity is occasional, untrusted, and even totally dissimilar. To depress influence of the overestimation issue and improve prediction accuracy, we adjust initial similarity calculation results of formulas (1) and (2) as follows.

$$Sim'(u, v) = \frac{|S_u \cap S_v|}{|S_u| + |S_v| - |S_u \cap S_v|} \times Sim(u, v) \quad (3)$$

$$Sim'(x, y) = \frac{|U_x \cap U_y|}{|U_x| + |U_y| - |U_x \cap U_y|} \times Sim(x, y) \quad (4)$$

In the above two formulas, $|S_u|$ and $|S_v|$ are quantities of services invoked by user u and v, respectively. $|S_u \cap S_v|$ is the quantity of same services invoked by user u and v. $|U_x|$ and $|U_y|$ are the quantities of users that invoked service x and y, respectively. $|U_x \cap U_y|$ is the quantity of users that invoked service x and y. $Sim'(u, v)$ and $Sim'(x, y)$ are the new similarity values. Through similarity adjustment, initial similarity values are decreased when $|S_u \cap S_v|$ and $|U_x \cap U_y|$ are small. The range of values of $Sim'(u, v)$ and $Sim'(x, y)$ is the same as that of $Sim(u, v)$ and $Sim(x, y)$, which is [−1, 1].

III. THE RESPONSE TIME PREDICTION OF IOT SERVICE

A. The Selection of Similar Neighbors

The Top-K algorithm is often used to select similar neighbors in collaborative filtering method, i.e., the top k most similar users or items are selected for further prediction. Some users or items only have few similar neighbors, and sometimes even have none. In this situation, normal Top-K algorithm will introduce dissimilar neighbors for calculation and highly decrease prediction accuracy. To address this issue, we improve the normal Top-K algorithm by adding a similarity threshold. Similar neighbors are selected by using the following formulas.

$$SN_u = \{v \mid v \in TopK_u \wedge Sim'(u, v) > \delta \wedge v \neq u\}, 0 < \delta < 1 \quad (5)$$

$$SN_s = \{s' \mid s' \in TopK_s \wedge Sim'(s, s') > \delta \wedge s' \neq s\}, 0 < \delta < 1 \quad (6)$$

In the above two formulas, SN_u is a set of similar users of user u and SN_s is a set of similar service items of service s. $TopK_u$ and $TopK_s$ are the sets of similar users and service items selected by normal Top-K algorithm, respectively. δ is a similarity threshold. Because negative correlation between similar neighbors has no practical meaning in prediction of IoT service response time in general, we set the range of values of δ as (0,1) to exclude negative dissimilar neighbors with negative or smaller similarity. Suitable value of δ depends on the specific user-item matrix.

B. The Densification of User-Item Matrix

Prediction accuracy will become low when the user-item matrix is sparse, called the sparse matrix problem. In this situation, we can predict missing values of user-item matrix to make it denser. Through densification of user-item matrix, prediction accuracy of collaborative filtering can be greatly improved [12]. Given a missing value $a_{us} = null$, when it has similar users but no similar services, i.e., $SN_u \neq \varnothing \wedge SNs = \varnothing$, a_{us} is predicted as follows.

$$MVU(a_{us}) = \overline{u} + \frac{\sum_{v \in SN_u} Sim'(u,v)(a_{vs} - \overline{v})\frac{\overline{u}}{\overline{v}}}{\sum_{v \in SNu} Sim'(u,v)}, a_{vs} \neq null \tag{7}$$

$MVU(a_{us})$ is the response time prediction value of a_{us}. $\overline{u}$ and $\overline{v}$ are the average response time of services invoked by user u and his similar user v, respectively. When a_{us} only has similar services but no similar users, i.e., $SN_u = \varnothing \wedge SNs \neq \varnothing$, a_{us} is predicted as follow.

$$MVS(a_{us}) = \overline{s} + \frac{\sum_{s' \in SNs} Sim'(s,s')(a_{us'} - \overline{s'})\frac{\overline{s}}{\overline{s'}}}{\sum_{s' \in SNs} Sim'(s,s')}, a_{us'} \neq null \tag{8}$$

$MVU(a_{us})$ is the response time prediction value of a_{us}. $\overline{s}$ and $\overline{s'}$ are the average response time of service s and its similar service s', respectively, invoked by all users.

Being different from the normal calculation method of collaborative filtering, in formulas (7) and (8), $\overline{u}/\overline{v}$ and $\overline{s}/\overline{s'}$ are added, that are amplitude ratios of average service response time between two similar neighbors. We provide an example to explain the reason. Suppose there are four service items and a user u with response time vector (2,4,3,9). We make an assumption that we do not know response time value 9 of the fourth service item, i.e., we treat it as a missing value and the new experimental response time vector of u is (2,4,3, null). Then, we predict this missing value by similar users of u. We first give three prediction examples as follows, and then make an explanation.

(1) Suppose user u only has one similar user v with response time vector (4,8,6,18). Every value of (4,8,6,18) is double of that of (2,4,3,9). Similarity between (2,4,3,9) and (4,8,6,18) is 1 and similarity between (2,4,3, null) and (4,8,6,18) is 0.478 by the calculation of formula (1). $\overline{u} = (2+4+3)/3 = 3$ and $\overline{v} = (4+8+6+18)/4 = 9$. We can get $MVU(a_{us4}) = 3 + 0.478 \times (18 - 9) \times (3/9)/0.478 = 6$ by formula (7). Because the actual value is 9, the prediction accuracy is $1 - |6-9|/9 \approx 66.67\%$. If we predict the missing value based on the normal calculation formula without amplitude ratio, shown in following formula (9), we can get $MVU(a_{us4}) = 3 + 0.478 \times (18 - 9)/0.478 = 12$. Prediction accuracy is $1 - |12-9|/9 \approx 66.67\%$.

$$MVU(a_{us}) = \overline{u} + \frac{\sum_{v \in SN_u} Sim'(u,v)(a_{vs} - \overline{v})}{\sum_{v \in SNu} Sim'(u,v)}, a_{vs} \neq null \tag{9}$$

(2) Suppose user u only has one similar user w with response time vector (6,12,9,27). Every value of (6,12,9,27) is triple of that of (2,4,3,9). Similarity between (2,4,3,9) and (6,12,9,27) is still 1 and similarity between (2,4,3, null) and (6,12,9,27) is still 0.478 by the calculation of formula (1). $\overline{u} = (2+4+3)/3 = 3$ and $\overline{w} = (6+12+9+27)/4 = 13.5$. We can get $MVU(a_{us4}) = 3 + 0.478 \times (27 - 13.5) \times (3/13.5)/0.478 = 6$ by formula (7). Prediction accuracy is 66.67%. But through the formula (9), the prediction result is $MVU(a_{us4}) = 3 + 0.478 \times (27 - 13.5)/0.478 = 16.5$, much greater than the actual value 9. Prediction accuracy is only $1 - |16.5-9|/9 \approx 16.67\%$.

(3) Suppose user u has two similar users v and w. We can get $MVU(a_{us4}) = 3 + [0.478 \times (18 - 9) \times (3/9) + 0.478 \times (27 - 13.5) \times (3/13.5)]/(0.478 + 0.478) = 6$ by formula (7). Prediction accuracy is 66.67%. But through the formula (9), the prediction result is $MVU(a_{us4}) = 3 + [0.478 \times (18 - 9) + 0.478 \times (27 - 13.5)]/(0.478 + 0.478) = 14.25$. Prediction accuracy is only $1 - |14.25-9|/9 \approx 41.67\%$.

The three prediction examples above indicate that our method is more suitable to make response time predictions. A more stable and better prediction accuracy can be obtained by considering response time amplitude ratios between similar neighbors. The main reason is that negative impact of the great change in response time values on prediction results is depressed by amplitude ratio, and every similar neighbor can provide a more reasonable improvement value for prediction calculation process.

When $SN_u \neq \varnothing \wedge SNs \neq \varnothing$, to use the information of similar neighbors as much as possible, we combine the user-based and item-based methods to predict matrix missing values. a_{us} is predicted as follows.

$$MV(a_{us}) = \omega(a_{us}) \times MVU(a_{us}) + (1 - \omega(a_{us})) \times MVS(a_{us}),\ 0 < \omega(a_{us}) < 1 \tag{10}$$

$\omega(a_{us})$ is a weight related to a_{us}, used to determine dependence degree of missing value prediction to user-based and item-based prediction results. $MV(a_{us})$ is the composite prediction value of a_{us}. $\omega(a_{us})$ is related to the similarity values of similar users and similar service items. For example, suppose a_{us} has two similar users with similarity (0.28, 0.31), and meanwhile, has two similar service items with similarity (0.73, 0.85). Because similarity of similar service items is much greater than that of similar users, in the composite prediction result, the weight of $MVS(a_{us})$ should be much higher than that of $MVU(a_{us})$. $\omega(a_{us})$ is calculated as follows.

$$AveSimU(a_{us}) = \frac{\sum_{v \in SN_u} Sim'(u,v)}{|SN_u|} \quad (11)$$

$$AveSimS(a_{us}) = \frac{\sum_{s' \in SN_s} Sim'(s,s')}{|SN_s|} \quad (12)$$

$$\omega(a_{us}) = \frac{AveSimU(a_{us})}{AveSimU(a_{us}) + AveSimS(a_{us})} \quad (13)$$

$AveSimU(a_{us})$ and $AveSimS(a_{us})$ are average similarity values of similar users and service items of a_{us}, respectively. When $SN_u = \varnothing \wedge SNs = \varnothing$, i.e., a_{us} has not any similar neighbors, we do not make missing value prediction to avoid influence of negative prediction results on prediction accuracy.

C. The Response Time Prediction Algorithm for Current Active Users

Via the algorithm shown in Fig. 1, current active users can get response time prediction values of services they did not use before. The prediction process is based on densified user-item matrix, like the matrix missing value prediction. The only difference is that we predict response time by following formula (14) when $SN_u = \varnothing \wedge SNs = \varnothing$.

The formula (14) represents the traditional user-mean and item-mean methods. $\bar{u}$ is the average response time value of the services invoked by the current active user u. $\bar{s}$ is the average value of response times observed by different users on the same service item s.

$$MV(a_{us}) = \begin{cases} (\bar{u} + \bar{s})/2, & \bar{u} \neq null \wedge \bar{s} \neq null \\ \bar{u}, & \bar{u} \neq null \wedge \bar{s} = null \\ \bar{s}, & \bar{u} = null \wedge \bar{s} \neq null \\ null, & \bar{u} = null \wedge \bar{s} = null \end{cases} \quad (14)$$

IV. The Validation of Prediction Accuracy

A. The Experiment Design

In this study, we use WS-DREAM dataset [14] to validate our prediction method. This dataset is composed of real historical QoS records of 100 distributed web service items invoked by 150 service users in different places. Every user invoked a service 100 times. The total number of records is 1.5×10^6. This dataset is suitable for the study of IoT services due to the reason that more IoT functions are provided through web services.

We exclude 11 service items unsuccessfully invoked by all users, and then construct a 150×89 user-item matrix as the experiment dataset. The matrix element is the average response time of 100 times service invocations. From the experiment dataset, we select 90% users (i.e., 135 users) to construct a 135×89 user-item matrix, and then select other 10% users (i.e., 15 users) as current active users. To simulate the sparse matrix, we randomly remove a number of elements of the user-item matrix, and then acquire matrixes with specific densities. In addition, we randomly remove some service response time values of current active users, and then use remaining values to make predictions. The number of the remaining values is set as an experiment parameter called service invocation number, representing quantities of response time values provided by current active users. In the experiments, we use different methods to predict response time missing values (i.e., those randomly removed values) for current active users, and then compare prediction results with removed real values to yield prediction accuracy of different methods. Prediction accuracy is calculated as follows.

$$PreAcc = \left[1 - \frac{\sum_{u,s}(|a_{us} - MV(a_{us})|/a_{us})}{N}\right] \times 100\% \quad (15)$$

Input: 1. A user-item matrix A
2. The historical service usage information of current active users U
Output: The response time prediction values of services unused by U
Begin
Construct a new densified user-item matrix A' based on initial matrix A
For each $u \in U$
Add the service response time vector of user u to A', and form a new matrix A''
For each $a_{us} = null$ in A''
If $SN_u \neq \varnothing \wedge SNs = \varnothing$ then predict a_{us} by formula (7)
If $SN_u = \varnothing \wedge SNs \neq \varnothing$ then predict a_{us} by formula (8)
If $SN_u \neq \varnothing \wedge SNs \neq \varnothing$ then predict a_{us} by formula (10)
If $SN_u = \varnothing \wedge SNs = \varnothing$ then predict a_{us} by formula (14)
End For
End For
End

Fig. 1. the response time prediction algorithm for current active users.

PreAcc is prediction accuracy. a_{us} is the real response time value of service item s observed by current active user u. $MV(a_{us})$ is the prediction value. N is the quantity of prediction values.

B. Prediction Accuracy Comparisons

We compare our composite prediction method (named as CP) with four common prediction methods and a typical prediction method, that are user-mean (UM), item-mean (IM), user-based (UB), item-based (IB) and WSRec [12]. Comparison results are shown in Table 1.

UM uses average value of the known service item response times of a current active user to predict other unknown service item response times for this user, while IM uses average response time of a service item observed by other users to predict response time of this service item for the current active user. UB only uses similar users to make predictions and IB only uses similar service items to make predictions. WSRec is like CP that also synthetically uses user-based and item-based collaborative filtering methods to make predictions. But there are some significant differences between them. WSRec does not consider similarity that is less than or equal to 0, while CP uses more flexible similarity threshold to exclude dissimilar neighbors. In addition, WSRec uses the method of setting parameters to balance the prediction dependence degree on user-based and item-based methods, while CP determines dependence degree by the method of adaptive weight. Especially, in contrast with WSRec, amplitude ratios of average response time between similar neighbors are considered in CP, making prediction results more accurate.

According to service invocation number, i.e., quantities of response time values provided by every current active user, we divide the experiment into three groups (10, 20, and 30). D10, D15, …, D30 represent that the densities of user-item matrix are 10%, 15%, …, 30%, respectively. Every data in Table 1 is an average value of 30 times experiment results. Some conclusions can be made from experiment results. (1) In contrast with other methods, CP obtains the best prediction accuracies under all experiment conditions. (2) Focusing on the dataset used in this experiment, prediction accuracy order of the six methods from low to high is IM, IB, UM, UB, WSRec and CP. It is worth noting that, in general, prediction accuracy of IB is higher than that of UB in e-business applications. But being different from user scores of commodities (e.g., user scores of movies), IoT service response time values usually change greatly. Therefore, in this situation, prediction accuracy of IB is lower than that of UB. (3) Prediction accuracy of UM slowly increases with increase in service invocation number, while it is irrelevant to density of the

Table 1. The prediction accuracy (PreAcc) comparison

Service invocation number	Prediction method	PreAcc (%)				
		D10	D15	D20	D25	D30
10	UM	69.56	69.41	69.61	69.44	69.68
	IM	54.55	54.86	55.39	56.21	56.73
	UB	71.58	72.23	72.64	73.12	74.47
	IB	57.60	57.77	58.39	59.17	60.05
	WSRec	69.39	72.05	72.87	73.90	74.72
	CP	73.82	75.17	76.70	76.98	78.43
20	UM	70.62	71.57	71.28	71.50	71.41
	IM	54.52	54.81	55.68	56.17	56.83
	UB	72.09	72.80	73.61	74.42	75.28
	IB	58.61	60.02	59.93	60.34	61.12
	WSRec	69.97	71.89	73.23	74.75	75.44
	CP	74.16	75.85	76.37	78.79	79.82
30	UM	71.53	71.77	71.88	72.07	71.86
	IM	54.72	54.88	55.63	56.02	56.81
	UB	72.72	73.42	74.17	74.87	75.90
	IB	59.61	59.93	60.64	61.29	61.96
	WSRec	69.95	72.23	73.86	74.68	76.24
	CP	74.54	75.97	78.04	79.92	80.66

user-item matrix. Prediction accuracy of IM slowly increases with increase in density of user-item matrix, while it is irrelevant to the service invocation number. Being different from UM and IM, because information of similar users and similar items is considered, prediction accuracies of UB and IB gradually increase with the increase in service invocation number and matrix density. (4) Because WSRec and CP overcome the defect that existing information cannot be fully used by other four common prediction methods, prediction accuracies are continually improved with the increase of information provided by user-item matrix and current active users.

C. The Impact of Similarity Adjustment

To depress influence of the overestimation issue, initial similarities are adjusted in CP method. We compare the difference between CP with similarity adjustment and CP without similarity adjustment under different service invocation numbers. Density of user-item matrix is set as 20%. The comparison result is shown in Fig. 2.

In all situations of service invocation numbers, CP with similarity adjustment has much better prediction accuracy than CP without similarity adjustment. With increase of service invocation number, more response time values of current active users are provided. Therefore, more similar neighbors can be found, which makes prediction accuracies of both methods increase gradually. The improvement effect of similarity adjustment becomes more obvious when the service invocation number increases. This is due to increase of similar neighbors and accumulation of improvement effects. When using the dataset in this study, the improvement amplitude of prediction accuracy is limited through similarity adjustment. But in the situation of large dataset with more overestimation issues, the impact of similarity adjustment will become obvious.

D. The Impact of Improved Top-K Algorithm

The improved Top-K algorithm excludes similar neighbors with negative similarities by adding a similarity threshold δ. Too large value of δ will exclude some valuable similar neighbors, while too small value of δ will include dissimilar neighbors and decrease prediction accuracy obviously. Meanwhile, too large or too small value of Top-K will also impact prediction accuracy. Suitable values of Top-K and δ depend on specific dataset. We set δ = 0.14 and Top-K=9 in our experiments after comparing many experiment results of different Top-K and δ values. Then, we compare the improved Top-K algorithm with the normal Top-K algorithm. The experiment result is shown in Fig. 3. In this experiment, density of user-item matrix is set at 15%.

In contrast with the normal Top-K algorithm, improved Top-K algorithm has much better prediction accuracy. With increase of service invocation number, similar neighbors with negative similarities are naturally excluded by Top-K value. Therefore, improved Top-K algorithm is more suitable to the situation than information current active user is lacking.

E. The Impact of Matrix Densification

To depress influence of sparse matrix, we densify the user-item matrix before making predictions. Differences of prediction accuracies between CP with matrix densification and CP without matrix densification are compared under different matrix densities. Comparison results are shown in Fig. 4. In this experiment, service invocation number is set at 10.

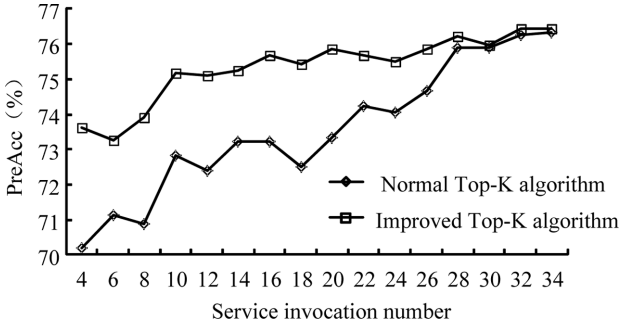

Fig. 3. The impact of improved Top-K algorithm.

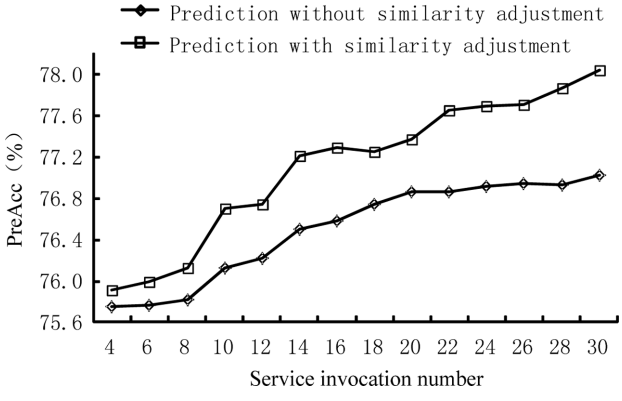

Fig. 2. The impact of similarity adjustment.

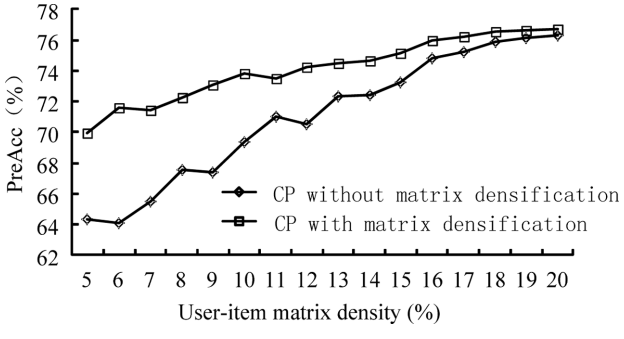

Fig. 4. The impact of matrix densification.

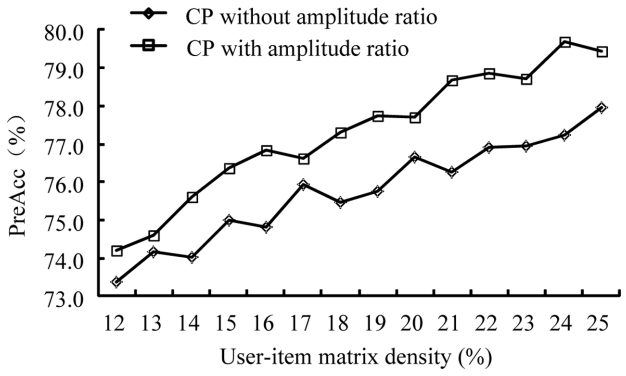

Fig. 5. The impact of amplitude ratio.

After applying user-item matrix densification, CP can yield much better prediction accuracy. With increase of matrix density, prediction accuracies of CP with matrix densification and CP without matrix densification increase gradually due to increased information for prediction. In the situation of low matrix density, improvement effect of matrix densification is more obvious. From experiment results, we also know that matrix densification can decrease volatility of prediction results and increase its stability, especially in the situation of low matrix density.

F. The Impact of Amplitude Ratio

To adapt to the great change of IoT service response time values, we add response time amplitude ratio in prediction process. Prediction accuracies of CP with amplitude ratio and CP without it are compared under different matrix densities. Comparison results are shown in Fig. 5. Service invocation number is set at 25 in this experiment.

The experiment result indicates that considering response time amplitude ratio in prediction process can increase prediction accuracy under different matrix densities. In this experiment dataset, quantity of effective similar neighbors is limited. When facing a large user-item matrix, the impact of amplitude ratio will become more obvious.

V. CONCLUSION

Service response time is a typical user-dependent QoS attribute, difficult to be predicted for IoT users. We combine user-based and item-based collaborative filtering methods to make service response time predictions. Many approaches are applied to improve prediction accuracy, including similarity adjustment, Top-K algorithm improvement, matrix densification and adaptive weight calculation. Especially, the response time amplitude ratio is considered in our study, suitable to address the issue that the response time of the same service often changes greatly to different users in complex IoT environment. Effectiveness of our prediction method is validated by multiple experiments on real dataset. By using prediction results, the IoT system can recommend a set of services with optimal response time for current active users that decreases time consumption of service search and selection, and meanwhile, increases the ability to provide on-demand services.

REFERENCES

1. F. Giancarlo, "Agents meet the IoT: toward ecosystems of networked smart objects," *IEEE Systems, Man, and Cybernetics Magazine*, vol. 2, no. 2, pp. 43-47, 2016.
2. S. N. Han, G. M. Lee, N. Crespi, V. L. Nguyen, H. Kyoungwoo, B. Mihaela, and G. Patrick, "DPWSim: a devices profile for web services (DPWS) simulator," *IEEE Internet of Things Journal*, vol. 2, no. 3, pp. 221-229, 2015.
3. B. Cheng, D. Zhu, S. Zhao, and J. L. Chen, "Situation-aware IoT service coordination using the event-driven SOA paradigm," *IEEE Transactions on Network and Service Management*, vol. 13, no. 2, pp. 349-361, 2016.
4. F. Wang, L. Hu, J. Zhou, and K. Zhao, "A data processing middleware based on SOA for the internet of things," *Journal of Sensors*, vol. 2015, no. 4, pp. 1-8, 2015.
5. J. Li, X. Luo, Y. N. Xia, Y. K. Han, and Q. S. Zhu, "A time series and reduction-based model for modeling and QoS prediction of service compositions," *Concurrency and Computation: Practice and Experience*, vol. 27, no. 1, pp. 146-163, 2014.
6. Y. Ngoko, C. Cerin, and A. Goldman, "Graph reuction for QoS prediction of cloud-service compositions," *Business Process Integration and Management*, vol. 7, no. 2, pp. 89-102, 2014.
7. V.V. Atluri, and H. Mohanty, "Web service response time prediction using HMM and bayesian network," in *Proceedings of the International Conference on Intelligent Computing, Communication and Devices*, Bhubaneswar, India, 2014, pp. 327-335.
8. D. J. Yu, Y. Liu, Y. S. Xu, and Y. Yin, "Personalized QoS prediction for web services using latent factor models," in *Proceedings of the 2014 IEEE International Conference on Services Computing*, Anchorage, AK, 2014, pp. 107-114.
9. Z. Chen, L.M. Shen, and F. Li, "Exploiting web service geographical neighborhood for collaborative QoS prediction," *Future Generation Computer Systems*, vol. 68, no. 3, pp. 248-259, 2017.
10. V. Narayan, R. K. Mehta, M. Rai, A. Gupta, M. Singh, S. Verma, A. Patel, and S. Yadav, "E-Commerce recommendation method based on collaborative filtering technology," *International Journal of Current Engineering and Technology*, vol. 7, no. 3, pp. 974-982, 2017.
11. Z. B. Zheng, and M. R. Lyu, "Collaborative reliability prediction for service-oriented systems," in *Proceedings of the ACM/IEEE 32nd International Conference on Software Engineering*, Cape Town, South Africa, 2010, pp. 35-44.
12. Z. B. Zheng, H. Ma, M. R. Lyu, and I. King, "QoS-aware web service recommendation by collaborative filtering," *IEEE*

Transactions on Services Computing, vol. 4, no. 2, pp. 140-152, 2011.
13. W. Lo, J. W. Yin, Y. Li, and Z. H. Wu, "Efficient web service QoS prediction using local neighborhood matrix factorization," *Engineering Applications of Artificial Intelligence*, vol. 38, pp. 14-23, 2015.
14. Z. B. Zheng, H. Ma, M. R. Lyu, and I. King, "Collaborative web service QoS prediction via neighborhood integrated matrix factorization," *IEEE Transactions on Services Computing*, vol. 6, no. 3, pp. 289-299, 2013.
15. M. Munoz-Organero, G.A. Ramirez-Gonzalez, P. J. Munoz-Merino, and C. D. Kloos, "Recommender system based on space-time similarities," *IEEE Pervasive Computing*, vol. 9, no. 3, pp. 81-87, 2010.
16. M. Chen, J. F. Wan, and F. Li, "Machine-to-machine communications: architectures, standards and applications," *KSII Transactions on Internet and Information Systems*, vol. 6, no. 2, pp. 480-497, 2012.

Huaizhou Yang

Huaizhou Yang, PhD, is working as an Associate Professor in the College of Computer Sciences, Xi'an Shiyou University, China. His current research interests focus on service computing, Internet of Things, cloud computing and data mining. In these areas, he has published over 20 papers in journals, book chapters, and international conferences and workshops.

Li Zhang

Li Zhang obtained her bachelor's degree in Computer Science from Liaocheng University, China, in 2015. Currently, she is a master student in Computer Science at Xi'an Shiyou University, China. Her current research interests include Internet of Things, service recommendation and data mining.