A Referral-Based QoS Prediction Approach for Service-Based Systems

Feng-Jian Wang[1], Yen-Hao Chiu[1], Chia-Ching Wang[1], Kuo-Chan Huang[2*]

[1] Department of Computer Science, National Chiao-Tung University, Hsinchu, Taiwan.
[2] Department of Computer Science, National Taichung University of Education, Taichung, Taiwan.

* Corresponding author. Tel.: +886-4-22183813; email: kchuang@mail.ntcu.edu.tw
Manuscript submitted January 18, 2017; accepted May 2, 2017.
doi: 10.17706/jcp.13.2.176-186

Abstract: During development of service-based systems (SBS), the quality of services (QoS) plays an important role at helping select more suitable services. There have been several QoS prediction approaches proposed, however, their prediction accuracy is low when there are few historical records in the application environment. In this paper, we propose a new QoS prediction method based on a virtual platform methodology. The method first constructs a virtual platform based on Gaussian distribution regarding stability and performance of services. With the platform, a referral-based QoS prediction method has been developed to improve prediction accuracy. The experimental results indicate that our method outperforms previous approaches, achieving higher prediction accuracy, especially when there are few historical records available.

Key words: Service-based systems, quality of services, prediction, service selection, web service.

1. Introduction

In recent years, service-oriented computing (SOC) and cloud computing become more and more popular. In such an environment, everything can be offered as a service. There are more and more services and service-based systems (SBS) to be constructed by composing the services selected of functional equivalence specified in a workflow and selected from the public.

For the search and selection of appropriate services, the quality of services (QoS) may play a significant role. Different qualities of a service may be concerned from different aspects. Typical quality factors include cost effective, response time and throughput for performance, etc. There are several studies focused on the QoS prediction in recent years [1]-[5]. However, there is one common defect in previous studies: When the quantity of QoS records is low, the prediction result is inaccurate dramatically.

To improve the prediction accuracy, we propose an approach based on the concept that the more quantity of QoS records are adopted to predict, the more accuracy the prediction is. Generally, QoS records can be derived from other platforms [6]. However, it is hard to select the useful QoS records in other platforms. Thus, we propose a method to construct a virtual platform for taking advantage of the QoS records on other platforms based on Gaussian distribution. We also propose an algorithm using referral mechanism [6] to increase the density of the QoS matrix on current platform with the help of the virtual platform, in order to improve the prediction accuracy. The experiments indicate that our approach can get better accuracy than others according to MAE values [1].

2. Background

2.1. Service Prediction in SBS

Service-oriented architecture (SOA) is a software (architecture) design pattern based on discrete pieces of software providing application functionalities as services to other applications. Service-Oriented Computing (SOC) is a computing paradigm that utilizes services as the basic constructs and composes them in a rapid, low-cost and easy manner of distributed applications. SOC contains a framework to achieve data sharing and exchange based on SOA, and the systems developed based on SOA are also called service-based systems (SBS). In SBS development, the basic component is an individual service which contains certain functionality. The application is implemented by composing multiple individual services with a workflow.

Service recommendation is a hot topic in the development of SBS, and the software construction with such a recommendation can be done as followings:

1. Design a workflow to specify the application completely.
2. Generate the service set according to the processes in the workflow constructed, where each element in the set represents a distinct process in the workflow.
3. Determine all the required non-functional QoS values of each element in the service set.
4. Based on QoS values, select the most suitable concrete service for each element in the service set.
5. Compose the services to construct the new SBS.

In the past, there were several researches on SBS [7]-[9] focused on how to select most suitable services, and several others focused on improving SBS quality with model design [10]-[13]. Besides, many studies were done on the usage of QoS values for different users [1]-[5], [14]. The common premise of their researches is that the QoS values of services to target users are all known. However, in practice a user can hardly have ever invoked all services, meaning that the QoS values of the services that the user has not invoked before are unknown. Therefore, it is crucial to predict the QoS values of the services never invoked by the user before any QoS-based service selection method can be applied.

2.2. Collaborative Filtering

The collaborative filtering (CF) algorithm was originally developed to predict the utility of items to a particular user based on a database of user votes collected from a sample or population of other user databases [15]. It is a technique widely adopted by recommendation systems. The dataset used for the CF algorithm in SBS development can be described as a matrix, where each entry represents the QoS history (experience) of a service. Because many QoS values of services are unknown, most of the entries in this matrix are set as zero or null. This type of matrix is called sparse matrix. Applying sparse matrix in CF algorithms has a shortcoming: When the matrix density is low, the accuracy becomes low too. However, current approaches for QoS prediction do not concern how to improve the prediction accuracy with matrix density.

2.3. Inheritance and Referral Mechanisms

Nguyen *et al*. [6] used three auxiliary mechanisms to solve trust bootstrapping with three typical cases for web services. The mechanisms adopted in their work include:

- The inheritance mechanism allows a web service to carry some degree of trust from its service provider.
- The referral mechanism gives a new web service a trust score based on referrals from other communities.
- The guarantee mechanism assigns a web service a temporary trust score through guarantee conditions.

Each environment has its own characteristics. For example, the platform of early version may have worse

performance than the platform of current version. Due to the difference of environment characteristics, a service may be given different trust score when it is applied in a different environment. Thus, the QoS values cannot be used for prediction directly.

3. Referral-Based QoS Prediction

3.1. Overview of Our Approach to Predict QoS

As shown in Fig. 1, our approach contains three steps to make QoS prediction:

- Step 1 creates a virtual platform and the corresponding QoS records by adopting the QoS records on another reference platform.
- Step 2 generates a matrix containing modified QoS records according to the QoS records in current and virtual platforms.
- Step 3 calculates the predicted QoS values for a designated user.

To simplify the description of our approach, first we introduce the data structures adopted:

- A User-Service matrix (in short as US matrix) contains the QoS records in current platform.
- A Referral User-Service matrix (in short as RUS matrix) contains the QoS records in the virtual platform.
- A Modified User-Service matrix (in short as MUS matrix) contains the modified QoS records.
- A service vector for a designated user (in short as SVDU) contains the predicted QoS values for the designated user.

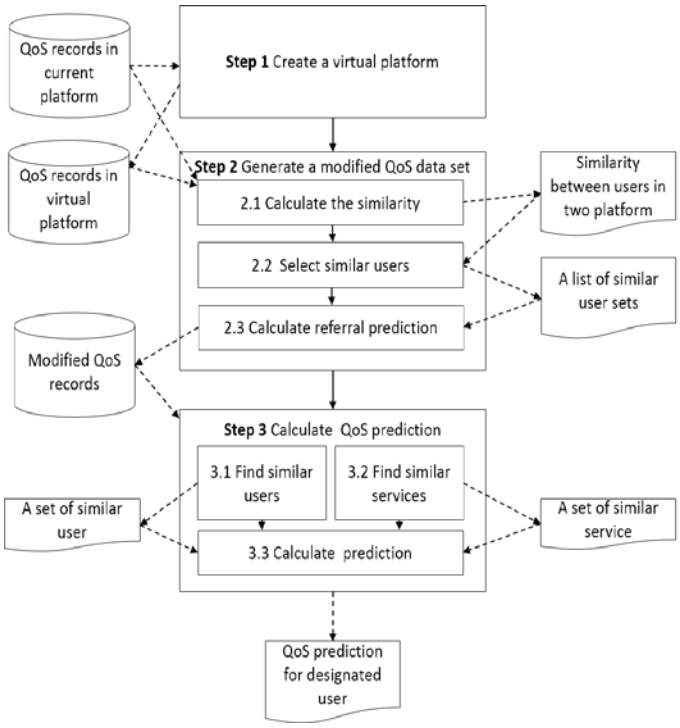

Fig. 1. A workflow for referral-based QoS prediction.

The characteristic of a platform may be described using the values of services' quality in some emphasized aspects. In this paper, our work is emphasized on the performance and system stability, which are adopted to evaluate the system mostly. The construction of our virtual platform is based on these two quality factors. In the platform, a large number of statistically independent QoS values are treated falling in Gaussian distribution. The invalid values can be easily filtered out according to a distinct pair of

parameters (performance, stability) and Gaussian distribution.

A US matrix, e.g. Fig. 2, represents the relationship (QoS values) between user i and service j. A white entry indicates that there is a null relationship between the corresponding pair of user and service, while a gray one means not. A RUS matrix, e.g. Fig. 3, is a US matrix which is recorded associated with the virtual platform. Similar to US matrix, the white entry in a RUS matrix indicates null, and the gray not.

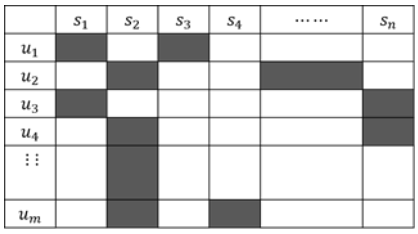

Fig. 2. An example for user-service matrix.

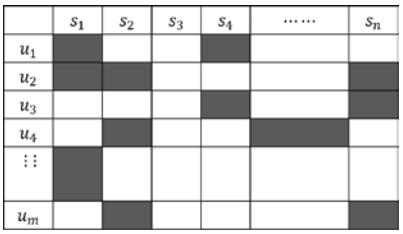

Fig. 3. An example for referral user-service matrix.

A MUS matrix, e.g. Fig. 4, is a US matrix which is derived by computing the US matrix of current platform and a RUS matrix based on the CF algorithm [3]. Each entry is white, gray, or black. The coloring for each entry is similar to US matrix except the black ones, each of which indicates that the corresponding entry value changes from null after the computation based on the CF algorithm. A designated user is a user who needs to predict QoS values with the MUS matrix based on the CF algorithm. The SUDV of a designated user, as in Fig. 5, is a vector which contains the QoS values of the services for the designated user. The color of each entry in an SUDV is gray or black, where the former indicates that its content has not change, and the latter indicates that the corresponding entry in the US is white and its content is re-calculated by applying the CF algorithm with the MUS matrix.

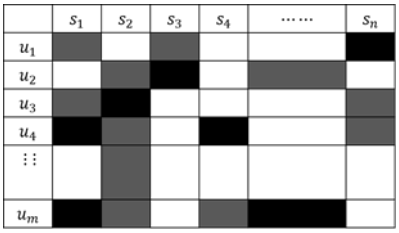

Fig. 4. Modified user-service matrix.

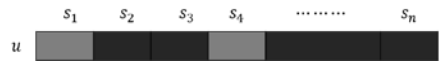

Fig. 5. service vector for designated user.

3.2. Creation of Virtual Platform

The first step for integrating the QoS records on a platform into the virtual platform is to modify the US matrix with the Gaussian method [16].

$$rus_{i,j}^{k} = \begin{cases} 0.5 + \frac{US_{i,j}^{k} - \overline{US^{k}}}{2\times 3\sigma_k} & ,0 \le 0.5 + \frac{US_{i,j}^{k} - \overline{US^{k}}}{2\times 3\sigma_k} \le 1 \\ 0 & ,0.5 + \frac{US_{i,j}^{k} - \overline{US^{k}}}{2\times 3\sigma_k} < 0 \\ 1 & ,0.5 + \frac{US_{i,j}^{k} - \overline{US^{k}}}{2\times 3\sigma_k} > 1 \end{cases} \quad (1)$$

In equation (1), $rus_{i,j}^{k}$ is the k^{th} QoS value for user *i* and service *j* derived after normalization in the virtual platform. $US_{i,j}^{k}$ represents the k^{th} QoS value for user *i* and service *j* in current platform. $\overline{US^{k}}$ is the average of the k^{th} QoS values in current platform. σk denotes the standard deviation for the k^{th} QoS value in current platform. By using the 3–σ rule, equation (1) can map approximately 99% data to the [0, 1]

interval directly. When the value $(US_{i,j}^k)$ is out of range [0, 1], the target value $(rus_{i,j}^k)$ is assigned as 0 if it is smaller than 0, or 1 otherwise.

The next step is to construct the corresponding QoS records in the virtual platform by transforming the data generated from the first step with equation (2) below:

$$RUS_{i,j}^k = 3\sigma_k' \times (2 \times rus_{i,j}^k - 1) + \overline{RUS^k} \tag{2}$$

In equation (2), $rus_{i,j}^k$ is the normalized k^{th} QoS value for user *i* and service *j* in the virtual platform. RUS^k is the mean of the k^{th} QoS values in the virtual platform. σ_k' denotes the standard deviation for the k^{th} QoS values in the virtual platform. Through these parameters, $RUS_{i,j}^k$ which is the k^{th} QoS value for user *i* and service *j* in the virtual platform is derived. The virtual platform is constructed by changing the standard deviation and the average values of the k^{th} QoS values, $\forall k, 1 \le k \le$ (*the number of qualities concerned*).

3.3. Generating an MUS Matrix

All of the similarity calculations in our approach are made according to equation (3), which computes the similarity between a pair of users (*i, j*) where user *i* in the US matrix for current platform *a* and user *j* in the RUS matrix for the virtual platform *b*. In equation (3), $S(u_{i,a})$ is the set of services which have ever been invoked by user *i* on platform *a*, and $S(u_{j,b})$ is the set of services which have been invoked by user *j* on platform *b*.

In our approach, constructing an MUS contains three steps in sequence:

1. *Generate similar user sets.* This step computes the similarity between each user in the US matrix for platform *a* and each user in the RUS matrix for the virtual platform *b* by using equation (3), a modified PCC.
2. *Select k most similar user in similar user set.* This step first derives the value k according to [17], and then constructs a set of most similar users, $SR(u_i)$, for each user u_i. $SR(u_i)$ is constructed by selecting k most similar users among all users.
3. *Calculates referral prediction.* This step calculates the QoS values of each user with equation (4). As in [6], a *threshold* value *T* between 1 and *k* is defined to represent the number of similar users and a *balance* value is defined with standard deviations on both platforms as follows. $c(u_i, s)$ is the number of invocations for service *s* by the users in $SR(u_i)$, $\frac{\sigma}{\sigma'}$ is the *balance* value where σ is the standard deviation in current platform, and σ' is the standard deviation in the virtual platform.

$$sim(u_{i,a}, u_{j,b}) = \frac{\sum_{s \in S(u_{i,a}) \cap S(u_{j,b})} (US_{u_{i,a},s} - \overline{US_{u_{i,a}}})^T (US_{u_{j,b},s} - \overline{US_{u_{j,b}}})}{\sqrt{\sum_{s \in S(u_{i,a}) \cap S(u_{j,b})} (US_{u_{i,a},s} - \overline{US_{u_{i,a}}})^2} \sqrt{\sum_{s \in S(u_{i,a}) \cap S(u_{j,b})} (US_{u_{j,b},s} - \overline{US_{u_{j,b}}})^2}} \tag{3}$$

$$MUS_{u_i,s} = \begin{cases} \overline{US_{u_{i,a}}} + \dfrac{\sum_{u_j \in SR(u_{i,a})} sim(u_{i,a}, u_{j,b}) \left(US_{u_{j,b},s} - \overline{US_{u_{j,b}}}\right)}{\sum_{i=1} sim(u_{i,a}, u_{j,b})} \times \dfrac{\sigma}{\sigma'}, & c(u_i, s) \ge T \\ no\ prediction, & c(u_i, s) < T \end{cases} \tag{4}$$

3.4. Calculating QoS Prediction

The calculation of predicting QoS values consists of three steps. Step 1 finds the similar-users set for the

designated user with the modified QoS records. Step 2 uses the modified QoS records to find the similar services for each service which is in current platform but has never been invoked by the designated user. Step 3 calculates the predicted QoS values of specific services for the designated user with the QoS values provided by the set of similar users/services derived in the previous two steps.

Step 1 can be further divided into two parts:

- Compare the similarity between each pair of users (*i, j*) in the same platform.
- Generate the similar set *Sim(u)*, which represents the set of all similar users to the designated user *u*. Then, the *k* most similar users are selected from *Sim(u)* and put into a set *SR(u)* to be used in the following steps. If *k* > |*Sim(u)*| or 2), all users in *Sim(u)* are put into *SR(u)*.

Step 2 also contains two parts:

- Calculate the similarity between each pair of services.
- Generate the similar set Sim(*s*), which represents the set of all similar services to service s which was never invoked by designated user. Then, the k most similar services are selected from Sim(*s*) and put into a set SR(*s*) to be used in the following steps. If k > |Sim(*s*)|, all users in Sim(*s*) are put into SR(*s*).

Step 3 calculates the QoS prediction values by applying equations (5) and (6) defined below:

$$pu(u,s) = \overline{US_u} + \frac{\sum_{u_j \in SR(u)} simu(u,u_j)(US_{u_j,s} - \overline{US_{u_j}})}{\sum_{u_j \in SR(u)} simu(u,u_j)} \tag{5}$$

$$ps(u,s) = \overline{US_s} + \frac{\sum_{s_j \in SR(s)} sims(s,s_j)(US_{u,s_j} - \overline{US_{s_j}})}{\sum_{s_j \in SR(s)} sims(s,s_j)} \tag{6}$$

Equations (5) and (6) represent a user-based approach [3] and an item-based approach, respectively. Existing methods usually adopt one of these two approaches. However, we found that if QoS prediction is done by one of them only, the predicted value may not be accurate enough. Here we present a method to achieve better prediction accuracy by integrating both of them. To make the integration, we first define two confidence weights for the user-based and item-based approaches respectively. The calculation of these two weights is defined in equations (7) and (8).

$$con_u = \sum_{u_i \in SR(u)} \frac{simu(u,u_i)}{\sum_{u_i \in SR(u)} simu(u,u_i)} \times simu(u,u_i) \tag{7}$$

$$con_s = \sum_{s_i \in SR(s)} \frac{sims(s,s_i)}{\sum_{s_i \in SR(s)} sims(s,s_i)} \times sims(s,s_i) \tag{8}$$

The computation of the final QoS prediction values consists of the following two activities based on the integration of both user-based and item-based prediction results:

1) Compute the confidence weights con_u and con_s according to equations (7) and (8).
2) Predict QoS values by using equation (9) with λ, con_u, con_s:

$$p(u,s) = \alpha \times pu(u,s) + \beta \times ps(u,s) \tag{9}$$

where,

1) $\alpha = \frac{\lambda \times con_u}{\lambda \times con_u + (1-\lambda) \times con_s}$

2) $\beta = \frac{(1-\lambda) \times con_s}{\lambda \times con_u + (1-\lambda) \times con_s}$

3) $\alpha + \beta = 1$

4) $0 \leq \alpha, \beta, \lambda \leq 1$

4. Experiment

4.1. Approaches to Be Compared and Metric

In the following experiments, we compare our method with several previous approaches, including user-based CF algorithm using PCC (UPCC) [3], item-based CF algorithm using PCC (IPCC) [17], user-mean (UserMean), item-mean (ItemMean), hybrid CF algorithm using PCC (HPCC) [18], and matrix factorization (MF) [19]. To compare different methods, we use a well-known metric, Mean Absolute Error (MAE) [1], for measuring the prediction accuracy of each method. MAE indicates the average of all distances between predicted values and real QoS values.

4.2. Experimental Results

To conduct our experiments, a real world web service QoS dataset [20] is adopted. This dataset is published at the website (http://www.wsdream.net). It contains about 1.5 million records of web service invocations for 100 web services located in more than 20 countries. The records are monitored by 150 distributed user nodes from the Planet-Lab [21] located in 20 countries, and each node is asked to invoke one service 100 times to get the average QoS record for each (node, service) pair. The QoS records can be arranged inside a 150×100 matrix, the User-Service (US) Matrix defined in the previous section. In our experiments, each entry in the US Matrix contains a three-tuple, each representing one of the following QoS values: reliability, response times, and throughput.

The experiments were done by the following steps:

1) Divide the US matrix into two parts. The top 100 rows are for the training matrix and the rest is for the designated matrix. The training matrix contains historical QoS records, and each row i in the designated matrix contains the service vector for designated user i.
2) Remove some entries from the training matrix randomly to create a sparse training matrix, used for simulating the real-world environment. For example, if the expected density of the sparse matrix is 5%, 95% of the entries in the training matrix need be removed randomly.
3) Construct a RUS matrix and an MUS matrix according to method described in the previous section, and remove some items from the MUS matrix randomly to create a sparse matrix. The density of the resultant matrix is named as a referral density.
4) Get a random number G between 1 and 100, which is the number of services described in the service vector.
5) Repeat the following work 50 times and get the average of the results to compute the average MAE. The work includes: 1) select a row *i* from the designated matrix, which represents the service vector for designated user *i*, 2) get a sample of G entries from the service vector randomly, and 3) calculate the result according to the method described in the previous section.

Table 1. Comparison of Prediction Performance

Metrics	Method	Reliability			Response time			Throughput		
		Density 5%	Density 10%	Density 15%	Density 5%	Density 10%	Density 15%	Density 5%	Density 10%	Density 15%
MAE	UserMean	0.0701	0.0755	0.067	1504.90	1545.41	1419.11	2557.50	2561.75	2548.39
	ItemMean	0.0877	0.0863	0.091	1665.00	1705.83	1758.32	2690.27	2637.06	2732.41
	UPCC	0.0658	0.0604	0.0579	1042.25	965.13	869.10	1519.09	1305.27	1206.35
	IPCC	0.0676	0.0591	0.0541	1188.02	941.97	873.12	1373.97	1208.62	1058.05
	HPCC	0.0581	0.0538	0.0456	1063.22	754.22	721.997	973.22	797.09	772.516
	MF	0.4899	0.4293	0.3886	1721.41	1487.87	1426.77	1813.59	333.91	222.95
	Our method	0.0482	0.0427	0.0356	940.07	737.12	653.41	857.39	706.94	633.48

The parameters used in our method, as introduced in the previous section, or for the experiments are set as follows: the density of the sparse training matrix are set as 5%, 10%, and 15%, and $\lambda = 0.5$, K=15, referral density=5%, and T=5.

The experimental results in Table I show that our method achieves better prediction accuracy than others for two quality factors of services: reliability and response time. For throughput of services, our method leads to better prediction accuracy than others only when the density is 5%.

Five parameters, which are involved in our method or the experimental setting, could affect the experimental results, including λ, number of services, referral density, K, and T. In the following, we present a series of experimets to evaluate the influence of each parameter. In the experiments, the values of the Normalized Mean Absolute Error (NMAE) [4] of our method are used to draw the line graphs for presenting the impact of each parameter.

To observe the influence of number of services, its value is set from 5 to 50, +5 per step. The rest parameters are set as follows: density=20%, λ=0.5, referral density=10%, K=15, and T=10. The trend of NMAE for number of services, shown in Fig. 6, indicates that: NMAE declines when the number of services increases. In other words, increasing the number of services can effectively improve prediction accuracy. To observe the influence of K, i.e. the size of similar set, its value is set from 6 to 20, +2 per step. The rest parameters are set as follows: density=20%, λ=0.5, referral density=10%, number of services=20, and T=5. The trend of NMAE with K, shown in Fig. 7, indicates that the trends of the three QoS attributes are similar. For response time and throughput, NMAE decreases when the number of K grows in the range [6], [14]. On the other hand, NMAE increases when K grows in the range from 14 to 20. For reliability, NMAE decreases when K grows in the range from 6 to 16, but increases when K grows in the range from 16 to 20. The experimental results show that it is important to choose an appropriate K value. However, further research is required to study how to determine an appropriate K value.

To observe the influence of λ, i.e. the weight for the user-based approach, its value is set from 0 to 1, +0.1 per step. The rest parameters are set as follows: density=10%, K=15, referral density=10%, number of services=20, and T=10. The trend of NMAE with λ, shown in Fig. 8, indicates that the smallest NMAE value occurs when λ =0.1. When $\lambda > 0.1$, the larger λ is, the bigger NMAE is. When $\lambda < 0.1$, the less λ is, the bigger NMAE is. This result shows that for a hybrid approach, the item-based approach needs be given more weight than the user-based approach.

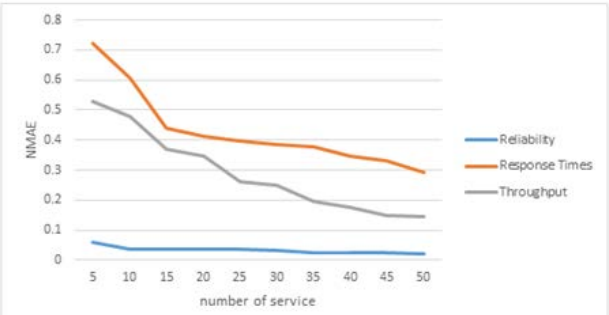

Fig. 6. Influence of number of services.

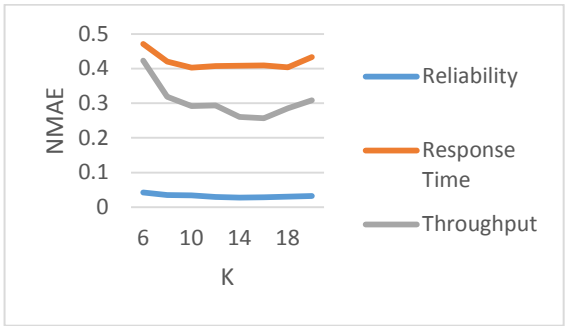

Fig. 7. Influence of K.

To observe the influence of referral density, its value is set from 5% to 50%, +5% per step. The rest parameters are set as follows: density=20%, K=15, number of services=20, λ=0.5, and T=10. The trend of NMAE with referral density, shown in Fig. 9, indicates that the trends of the three QoS attributes are similar although the fluctuation for reliability is relatively smaller. In general, as the referral density increases, NMAE decreases, implying better prediction accuracy. However, there is a turning point appearing at referral density=25%, where NMAE for response time and throughput increases suddenly.

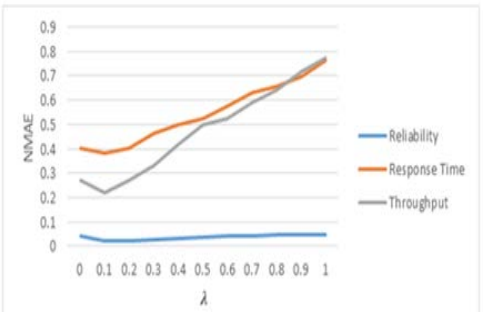

Fig. 8. Influence of λ.

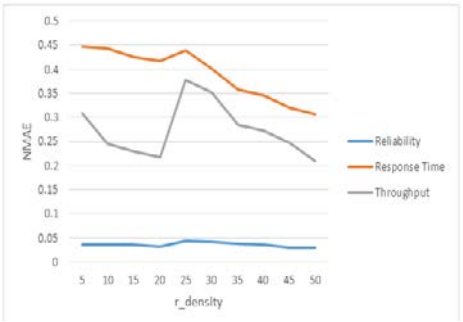

Fig. 9. Influence of referral density.

To observe the influence of the threshold T, its value is set from 2 to 20, +2 per step. The rest parameters are set as follows: density=20%, K=20, referral density=10%, λ=0.5, and number of services =20. The trend of NMAE with T is shown in Fig. 10. The experimental result shows that NMAE achieves its smallest value when T =14 which is a turning point. When T >14, the larger T is, the bigger NMAE is. On the other hand, when T<14, the less T is, the bigger NMAE is. This result indicates that careful selection of the T value is important to achieve goo prediction accuracy. However, further research is needed to study how to choose an appropriate T value under different configurations.

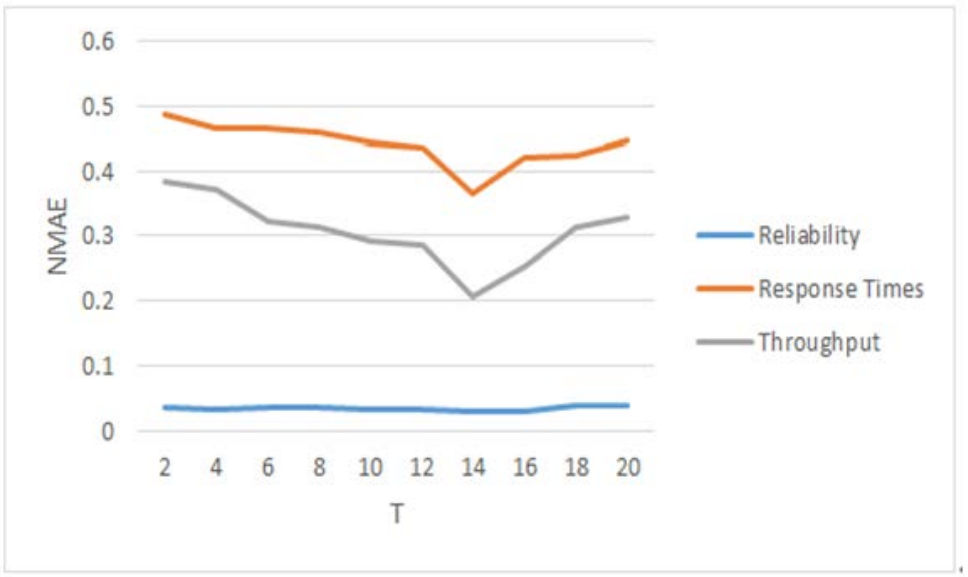

Fig. 10. Influence of T.

5. Conclusion and Future Works

In the paper, we propose an innovative approach to predict the QoS values in service-based systems. Different from previous studies, our approach constructs a virtual platform based on Gaussian distribution and applies the user experience in the virtual platform to increase the density of the sparse matrix in current platform, in order to achieve better prediction accuracy. Our approach improves the prediction accuracy of a sparse matrix based on a method modified from the collaborative filtering technique [2]. The experimental results show that our approach can deliver more accurate QoS prediction than previous methods in terms of MAE. In this paper, our approach works based on a single virtual platform. A promising future research direction is to study the possibility of improving the prediction accuracy further by constructing and referring to the QoS records on more than one virtual platform.

References

[1] Chen, X., Zheng, Z., Liu, X., Huang, Z., & Sun, H. (2013). Personalized QoS-aware web service recommendation and visualization. *IEEE Transactions on Services Computing, 6(1),* 35–47.

[2] Qiu, W., Zheng, Z., Wang, X., Yang, X., & Lyu, M. (2013). Reputation-aware QoS value prediction of web services. *Proceedings of IEEE International Conference on Services Computing*. (pp. 41–48).

[3] Shao, L., Zhang, J., Wei, Y., Zhao, J., Xie, B., & Mei, H. (2007). Personalized QoS prediction forweb services via collaborative filtering. *Proceedings of IEEE International Conference on Web Services* (pp. 439–446).

[4] Wu, J., Chen, L., Feng, Y., Zheng, Z., Zhou, M. C., & Wu, Z. (2013). Predicting quality of service for selection by neighborhood-based collaborative filtering. *IEEE Transactions on Systems, Man, and Cybernetics*, *43(2),* 428–439.

[5] Zheng, Z., Ma, H., Lyu, M., & King, I. (2011). QoS-aware web service recommendation by collaborative filtering. *IEEE Transactions on Services Computing*, *4(2),* 140–152.

[6] Nguyen, H. T., Yang, J., & Zhao, W. (2012). Bootstrapping trust and reputation for web services. *Proceedings of IEEE 14th International Conference on Commerce And Enterprise Computing* (pp. 41–48).

[7] Hwang, S. Y., Lim, E. P., Lee, C. H., & Chen, C. H. (2008). Dynamic web service selection for reliable web service composition. *IEEE Transactions on Services Computing*, *1(2),* 104–116.

[8] Mohebi, A., Ding, C., & Chi, C. H. (2012). Efficient QoS-based service selection with consideration of user requirements. *Proceedings of IEEE 16th International Enterprise Distributed Object Computing Conferenc.* (pp. 183–190).

[9] Xiong, P., Fan, Y., & Zhou, M. (2008). QoS-aware web service configuration. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, *38(4),* 888–895.

[10] Lin, J., Jiang, C., Hu, H., Cai, K. Y., Yau, S., & Huang, D. (2012). A reputation system for trustworthy QoS information in service-based systems. *Proceedings of COMPSACW-2012* (pp. 176–181).

[11] Qi, S., Li, B., Liu, C., Wu, X., & Song, R. (2012). A trust impact analysis model for composite service evolution. *Proceedings of the 19th Asia-pacific Software Engineering Conference: vol. 1.* (pp. 73–78).

[12] Yau, S., & An, H. (2011). Anonymous service usage and payment in service-based systems. *Proceedings of IEEE 13th International Conference on High Performance Computing And Communications* (pp. 714–720).

[13] Yau, S., & Huang, D. (2011). Distributed monitoring and adaptation of multiple QoS in service-based systems. *Proceedings of COMPSACW-2011* (pp. 31–36).

[14] Cao, B., Liu, J., Tang, M., Zheng, Z., & Wang, G. (2013). Mashup service recommendation based on user interest and social network. *Proceedings of IEEE 20th International Conference on Web Services* (pp. 99–106).

[15] Breese, J. S., Heckerman, D., & Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence* (pp. 43–52).

[16] Casella, G., & Berger, R. (2002). *Statistical Inference* (2nd ed.).

[17] Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. *Proceedings of the 10th International Conference on World Wide Web* (pp. 285–295).

[18] Zheng, Z., Ma, H., Lyu, M. R., & King, I. (2009). Wsrec: A collaborative filtering based web service recommender system. *Proceedings of the 7th International Conference on Web Services* (pp. 437-444).

[19] Lee, D., & Seung, H. (1999). Learning the parts of objects by nonnegative matrix factorization. *Nature*, *401(6755),* 788–791.

[20] Zheng, Z., & Lyu, M. R. (2010). Collaborative reliability prediction of service-oriented systems. *Proceedings of the 32th ACM/IEEE International Conference on Software Engineering: Vol. 1.* (pp. 35–44).
[21] Chen, Y. H., & George, E. I. (2002). *A Bayesian Model for Collaborative Filtering* (Report No. TX 78712). Department of MSIS, University of Texas at Austin.

Feng-Jian Wang completed his Ph.D. program in the Dept. of E.E.C.S., Northwestern University in 1988. Since then, he worked in National Chiao-Tung University, Taiwan. During his Ph.D. program, he worked on incremental analysis of data flow. Thereafter, he worked on the development, reuse, and data analysis of object-oriented programming language. Since 1995, he worked on the analysis and design of workflow programs and his laboratory constructed a workflow management system named Agentflow, where he studied a series of supporting analysis and tools associated with editing activities. Currently, he is focused on workflow, SOA and cloud computing techniques, and developing a rich interface pattern for virtual reality.

Yen-Hao Chiu received his M.S. degrees in computer science from National Chiao-Tung University, Taiwan, in 2014. His main research interests include software engineering and services computing.

Chia-Ching Wang received his M.S. degrees in computer science from National Chiao-Tung University, Taiwan, in 2015. His main research interests include distributed computing, software engineering, and web services.

Kuo-Chan Huang received his B.S. and Ph.D. degrees in the Department of Computer Science and Information Engineering, National Chiao-Tung University, Taiwan, in 1993 and 1998, respectively. He is currently an associate professor in Computer Science Department and the director of Computer and Network Center at National Taichung University of Education, Taiwan. He is a member of ACM and IEEE Computer Society. He has served as workshop chair, publicity chair, and program committee member for several international conferences. His research areas include parallel processing, cluster, grid, and cloud computing, workflow computing, and services computing.