A Web Service Recommendation Approach based on Situation awareness

Chenguang Liu*, Huiping Lin, Yibing Xiong
School of Software and Microelectronics
Peking University
Beijing, China
Email: chenguangliu@pku.edu.cn, linhp@ss.pku.edu.cn, ybxiong@pku.edu.cn

***Abstract* — With the development of Service-Oriented Technologies, the amount of Web services grows rapidly. Situation awareness, as a very important computing paradigm which can provide more reasonable and complete representation of the user's environment, can be benefit to the discovering of the service user's requirement. Therefore in this paper, we explore to use the situational information to recommend services which satisfy the requirement of users better and meanwhile improve the QoS prediction accuracy. Both the convergences between users' service selections in different situations and their QoS experiences have been taken into account. Moreover, we investigate the potential relationships between the situations structure and the services selection. Experiment results indicate that our method achieves ideal performance.**

Keywords - Service Recommendation; Situation awareness; QoS Prediction

I. INTRODUCTION

Web services are considered as software functions provided at network addresses over the web or the cloud. They encapsulate application functionality and make them available through interfaces. In recent years, how to build the service recommendation that seamlessly satisfy service users' needs anywhere and anytime becomes a challenging research problem in both *Service-Oriented Computing* (SOC) and *Ubiquitous Computing* [1] fields.

Current service recommendation researches mainly focus on the consideration of the services' non-functional properties. The emergence of multiple Web services with overlapping functionality offers a set of alternatives for users to pick over based on their *Quality of Services* (QoS). QoS is a broad concept that encompasses a number of properties such as price, availability, reliability, and reputation [2]. There are several approaches [3, 4, 5] that use contextual information (e.g., location, time) for collaborative filtering to predict QoS values of Web services and have achieved accuracy improvement.

Situation-awareness, an emerging class in *Ubiquitous Computing* where situation is regarded as logically aggregated contexts, can provide more reasonable and complete representation of the user's environment. This computing paradigm offers a much better chance of providing high quality humanized services. However, utilizing the situation abstraction [6, 7] to recommend services still presents challenges. For instance, each service user in the same situation may have his/her respective functional inclination. The relations between situations should be taken into account when recommending a service. Service requirements of users in simple (low-level) situations present a convergence, but factors to consider are different when it comes to a complicated situation.

In this paper, we explore to use the situational information to recommend services which satisfy the requirement of users better and meanwhile improve the QoS prediction accuracy. Our work aims at addressing the following key issues:

1. Propose an approach to recommend services to users and predict QoS values through an enhanced collaborative filtering method based on both Situation-aware features and the convergence between service users in different situations.
2. Investigate the potential relationships between the situations perception structure and the services selection.

The rest of the paper is organized as followed: Section 2 highlights the key features in *Situation-aware Computing* as background knowledge. Section 3 illustrates a motivating scenario while the detailed approach of our service recommendation is introduced in Section 4. Section 5 presents experiment results and conclusions.

II. BACKGROUND: KEY FEATURES IN SITUATION-AWARE COMPUTING

Situation-aware Computing is a new information paradigm where people are empowered through a digital environment that is "aware" of their presence and situation, and is *sensitive*, *adaptive* and *responsive* to their needs. Therefore in order to provide better response to the users with seamlessly services anywhere and anytime, we briefly explain the relevant basics of *Situation-aware Computing* in this section.

The conception of "situation" in software engineering comes from Situation Theory [8], which defines the situation as a part of the way the world happens to be. "*In contrast with a 'world' which determines the value of every proposition, a situation corresponds to the limited parts of reality we perceive, reason about, and live in*". But it is not until the *Context-aware Computing* becomes one of the major research directions for Ubiquitous Computing that researchers think afresh the profound understanding of situation and focus on the utilization of its concept for better

978-0-7695-5026-8/13 $26.00 © 2013 IEEE
DOI 10.1109/SCC.2013.103

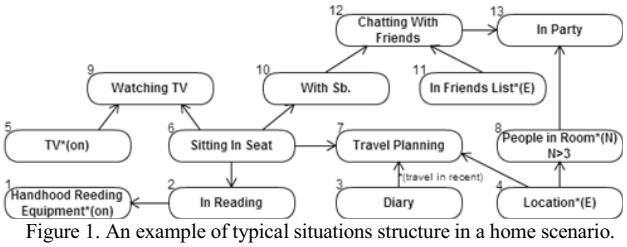

Figure 1. An example of typical situations structure in a home scenario.

experience.

Compared to *Context-aware Computing*, *Situation-aware Computing* encourages a deep separation of the different concerns. [6] decouples the decision making module and other concerns because of the complex mechanism on sensor readings. *Situation Programs* [7] take situation reasoning module to the next level and make the reasoning in this field more sound and solid (Fig. 1 shows a typical situations structure in a home scenario). There are also many researches related to the Ontology-based situation representation like [9].

Based on our previous researches, there are 3 major modules in a SA application: sensor-situation interpreting infrastructure, situation reasoning engine and service recommendation module (also called decision making module when it focus on offering local services). The sensor-situation interpreting infrastructure is responsible for situational context delivery and interprets it into a linguistic reasonable form (i.e. S_x :temperature*(*CE*, *TCE*>50) to S_x:temperature*(*CE*, '*TCE* is filled with hot water')). The situation reasoning engine starts with a low level trigger situation program, takes relations between situations to inference, seeks evidence spontaneously, and ends with a relatively complete situation as result. Relations between situations can be Containment, Logical equivalent, Compatibility/Incompatibility or Indistinguishability [10]. The service recommendation module gathers the perceived situation, user preference and other users' invoked records, and recommends a service to the service user. Fig. 2(a) shows the Situation-awareness model in our approach. We are inclined to confirm that these cognitive understanding of service user's environment should have meaningful impacts on his/her services selection. However, how to refine the existing Web services recommendation methods to suit Situation-aware features comes up with challenges.

III. MOTIVATING SCENARIO: THREE THINGS TO CONSIDER

There is a popular Situation-aware system called *Kinderszenen* in a digital future in which computation is embedded into the fabric of the world around us. People improve the convenience of life by using this SA system to a certain extent (Shown in Fig.2(b)). John and William work in the same building; they have similar choices of commuting behaviors, food preferences and magazine subscriptions. One day, John was perceived to be in the situation Sit_{c-1} - 'Ordering Takeaway'. The system soon found out William had ordered Pizza Hut twice through service S_1 in the same situation and sent a Pizza Hut takeaway menu with promotions to John by invoking S_1.

At the airport of the city, Edward and Marie were perceived to be in the situation Sit_{b-2}-*'Check In'*. Due to the location of Sit_{b-2} in the situations structure is comparatively simple, services other users had invoked were not diverse. *Kinderszenen* had to pick out one recommended service for each of them from: (1) Online Check In, (2) Cellphone Check In, and (3) Counter Check In. Edward is an undergraduate student from a local university and he travels often. Marie is in her sixties and was going to visit her daughter in another city. The system found some records of other students in Edwards' school and recommended the Online Check In service S_2 to him. Edward invoked the service on the road, dropped a luggage at airport and soon boarded. The amount time he took was shorter than the predicted time cost. As for Marie, the system sent her a concise guide through S_3 and she finished her check in procedure at the counter.

From these examples we can see, it is clearly helpful to recommend Web services with the consideration of the situational information and the convergence between the users' selections and requirements. Following three issues are addressed in the proposed service recommendation method in this paper:

(1) How to compute the convergence between users with the records in different situations and utilize it to satisfy the current service user's functional needs (Fig. 3 illustrates the way of using the convergences between users' service selections in different situations to recommend service);

(2) With a given candidate Web service and relevant invoke records of other users, how to employ the collaborative filtering method to predict its QoS performance;

(3) What factors have to be figured out in search of the potential interrelationships between the situations structure and the user's services selection.

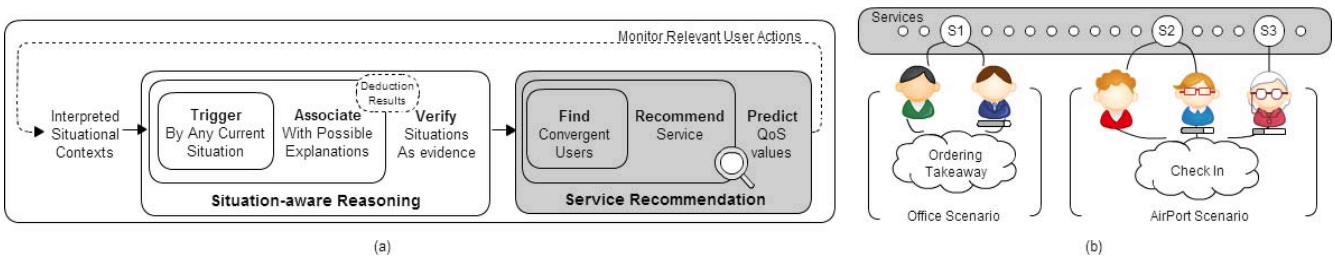

Figure 2. (a) The Situation-awareness model in our approach; (b) A Motivating Scenario.

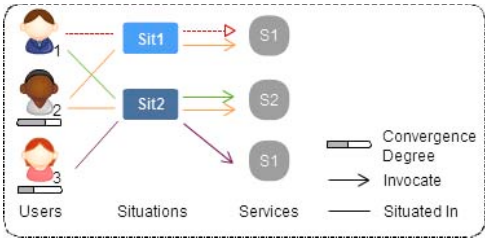

Figure 3. Use convergence in service recommendation.

IV. SERVICE RECOMMENDATION BASED ON SITUATION-AWARENESS

A. Problem Definition

To simplify the explanation of our approach, we formalize the terms as followed.

1) $M = (\mathcal{S}, \mathcal{T}, c, \mathcal{S}_t, R_c)$ is the situation structure. As the core component of a SA application, this quintuple is defined based on the *Kripke* structure for perception [10] where $\mathcal{S}$ is a universal set of situation programs in our system; $\mathcal{T}$ is a serial, transitive relation for abductive reasoning; $c \subseteq (\mathcal{S}_t \times \mathcal{S})$ is a logical equation relation where $\mathcal{S}_t$ is a sub-set of $\mathcal{S}$; The corner mark t infers any element in $\mathcal{S}_t$ is transparent to the other modules (e.g. service recommendation module) in the system. R_c is the retrieve consumption(RC) of element in $\mathcal{S}/\mathcal{S}_t$. e.g. the time to query a projector's state is much shorter than the time to make the noise dosimeter monitor the noise level which may last a few minutes. Note that as we take equivalent representation (c) of situations into account of our approach to solve the multiple recognition ways problem [11], the containing relationship no longer follows its original definition. Fig. 4 shows an example of the situation structure containing logical equivalent relationship. We have to restrict its transitive relationship within the $\mathcal{S}/\mathcal{S}_t$ range and when it comes to a transparent situation, no bridgeable containment can get over. Here we are not going to discuss about situation reasoning further but focusing on the basic relations needed later in our recommendation module.

2) $S_m = \{s_1, \ldots s_m\}$ is the set of candidate services, where each $s_i | 1 \le i \le m$ denotes one service.

3) $U_n = \{u_1, \ldots u_n\}$ is the set of users in our system served as service consumers when the situation reasoning module learn their situational information, where each $u_i | 1 \le i \le n$ denotes one user.

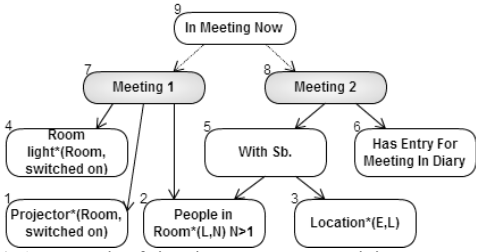

Figure 4. An example of situations structure containing two recognition ways of Sit_9.

4) $r_{i,\varphi,s}$ is the vector of QoS values that the user u_i provides as a feedback after invoking the service s_j in situation φ.

5) $\bar{r}(U_n, j, s) = avg\{r_{i,\varphi,s} | u_i \in U_n \wedge i \neq j \wedge Sit_\varphi \in \mathcal{S} \wedge r_{i,\varphi,s} \neq \emptyset\}$ is a vector of average of QoS values of service s observed by the other users in U_n except u_j. We can get the average of QoS values of the same service observed by all the users in U_n by specifying j to 0.

6) $P(i, \varphi)$ is the set of services that user u_i invoked in situation φ, where $u_i \in U_n$ and $Sit_\varphi \in \mathcal{S}$.

7) $P(i, j, \varphi) = P(i, \varphi) \cap P(j, \varphi)$ is the set of service items that are invoked both by user i and user j in situation φ.

B. Service Recommendation

With the given user u_i, the situation inference result Sit_ε and a reasonable amount of invoke records, we firstly find a set of similar users (*convergent set*) for him/her.

This convergence is affected both by the other users' service choices they invoked in the similar situations and the shared QoS experiences. From the researches in recent years, *Pearson Correlation Coefficient* (PCC) was introduced commonly in recommendation systems for similarity computation. Based on PCC, some collaborative filtering methods like user-based PCC (UPCC), item-based PCC (IPCC) or hybrid method based on UPCC and IPCC were applied in Web service recommendation and achieved significant accuracy improvement [5, 12, 13, 14]. To find the convergent set for u_i, a Situation-aware PCC is employed to compute the degree of experience similarity between u_i and other users using the following formula:

$$\mathrm{sim}_e(\mathrm{i},\mathrm{j}) = \frac{\sum_{\varphi\in\hat{\mathcal{S}}}\sum_{s\in \mathrm{P}(i,j,\varphi)}(r_{i,\varphi,s}-\bar{r}_i)(r_{j,\varphi,s}-\bar{r}_j)}{\sqrt{\sum_{\varphi\in\hat{\mathcal{S}}}\sum_{s\in \mathrm{P}(i,\varphi)}(r_{i,\varphi,s}-\bar{r}_i)^2}\sqrt{\sum_{\varphi\in\hat{\mathcal{S}}}\sum_{s\in \mathrm{P}(j,\varphi)}(r_{j,\varphi,s}-\bar{r}_j)^2}} \quad (1)$$

where $\varphi \in \hat{\mathcal{S}}$ ($\hat{\mathcal{S}} = \mathcal{S}/\mathcal{S}_\mathrm{t}$) is the non-transparent iterator situation, P is the set of Web services u_i and u_j invoked in φ, $\bar{r}_i$ and $\bar{r}_j$ represent the average QoS values of them respectively. Note that it can be seen from the above formula that the degree of experience is in the interval of [-1, 1]. The larger a value is, the more similar in experience two users probably are.

After computing the experience similarities between u_i and other service users, we integrate them with the preference similarities to define the *convergence* between those users. By comparing the Web services choices they commonly invoked in each situation, $\mathrm{con}(\mathrm{i},\mathrm{j})$ denotes not only the QoS experience similarity between u_i and u_j but also suggests their tendency of choosing Web services in certain situations. Unlike the *similarity weight* in [13] which the author employed to solve the overestimating problem of PCC [15], our *convergent weight* also has implications on the clustering relations between Web services and the situations structure. Hence, the convergence between u_i and u_j is:

$$\mathrm{con}(\mathrm{i},\mathrm{j}) = \mathrm{sim}_e(i,j)\sum_{\varphi\in\hat{\mathcal{S}}}\frac{2\times|\mathrm{P}(i,j,\varphi)|}{|\mathrm{P}(i,\varphi)|+|\mathrm{P}(j,\varphi)|} \quad (2)$$

where $|\mathrm{P}(i,j,\varphi)|$ is the number of service items that are invoked both by user i and user j in situation φ, $|\mathrm{P}(i,\varphi)|$ and $|\mathrm{P}(j,\varphi)|$ are the numbers of services invoked by user i and user j respectively. However, due to the relative location of Sit_φ in the situations structure and its diverse relationships with other situations, factors affecting the functional requirement of service user may vary. An ideal solution to this problem is to add a *structure weight* w_φ which balances the influence of situations complexity of Sit_φ. An improved convergence computation of different users is defined as:

$$\mathrm{con}_f(\mathrm{i},\mathrm{j}) = \mathrm{sim}_e(i,j)\sum_{\varphi\in\mathcal{S}/\mathcal{S}_\mathrm{t}}\frac{2\times w_\varphi\times|\mathrm{P}(i,j,\varphi)|}{|\mathrm{P}(i,\varphi)|+|\mathrm{P}(j,\varphi)|} \qquad (3)$$

where w_φ represents the *structure weight* of Sit_φ and it is calculated through the followed algorithm (to assign a *structure weight* for each situation in $\mathcal{S}$ based on its specific abductive relations in $\mathcal{T}$).

ALGORITHM 1: CALCULATE STRUCTURE WEIGHTS
Input: M %situation structure
Output: W %*sw* of situations in $\mathcal{S}$

Begin
  $\widetilde{H} = \{\varphi \in \mathcal{S}, \nexists(\varphi,*) \in \mathcal{T}\}$ %find top-level situations
  For φ: $\widetilde{H}$ **Do** set $\mathrm{w}_\varphi = 1$
  While $\widetilde{H} \neq \emptyset$ **Do** $\forall\varphi$: $\widetilde{H}$
    For o: $(\mathrm{o},\varphi) \in \mathcal{T}$ **Do** $\widetilde{H} = \widetilde{H} \cup \{o\}$
      If $\mathrm{o} \in \mathcal{S}_t$ $\mathrm{w_o} = \mathrm{w}_\varphi$ **else** $\mathrm{w_o} = \theta \cdot \mathrm{w}_\varphi$
    $\widetilde{H}$ removes φ %recursive traversal
  End
  Return W
End.

We select users with *Top-N* convergences to form a set $\widetilde{\mathrm{U}}_i$. Inspired by [13], a convergent user u_e will be removed from the convergent users set of u_i if its *convergent weight* is equal to or smaller than 0 since the vector $\mathrm{V}_{conf(i)}$ selects top n users and not all users have enough of convergent users in practice.

$$\widetilde{\mathrm{U}}_i = \{\mathrm{u}_e | \mathrm{u}_e \in \widetilde{\mathrm{U}}_i, \mathrm{con}_f(\mathrm{i},\mathrm{e}) > 0\} \qquad (4)$$

Let $\mathrm{V}_{conf(i)}$ be the vector of convergence values of these n services users. Then we retrieve the invoke records of these n users in situation Sit_ε. The relationship between them and the K Web services is denoted by an $K \times N$ matrix, called the situation-service matrix of situation Sit_ε with respect to the service user u_i. The entry in this matrix $m_{k,n}$ represents truth value of whether u_n has invoked Web service s_k when he/she is in Sit_ε. Since a user may have invoked more than one Web services in a particular situation through different times, entry values in a column may be several non-zero values. And if the user n has not invoked any Web service in that situation before, column n will be all zero values.

Once the $\mathrm{V}_{conf(i)}$ and its corresponding situation-service matrix are generated, they are used for computing the vector of *Situation-aware recommendation degrees* of the Web services for u_i. By multiplying them, we are able to obtain these recommendation degrees which integrate both potential users' preference and their QoS experience. Therefore the vector of *Situation-aware recommendation degrees* is defined as following equation: (Fig. 5 shows an example of the multiplying process)

$$\mathrm{V}_{r(i,\varepsilon)} = \mathrm{M}_{i,\varepsilon} \times \mathrm{V}_{con_f(i)} \qquad (5)$$

$$\mathrm{V}_{r(5,14)} = \begin{matrix} s_7 \\ s_{23} \\ s_{25} \\ s_{92} \end{matrix}\begin{bmatrix}1 & 0 & 1\\ 0 & 0 & 1\\ 1 & 1 & 0\\ 0 & 1 & 1\end{bmatrix} \times \begin{bmatrix}0.3711\\ 0.2302\\ 0.2141\end{bmatrix} = \begin{bmatrix}0.5852\\ 0.2141\\ 0.6013\\ 0.4443\end{bmatrix}$$

Figure 5. An example of multiplying process.

where the element of the highest value indicates the most recommended Web service for u_i in that situation Sit_ε (e.g. s_{25} to u_5 in Sit_{14}).

We have identified the recommended service. Thus another kind of prediction, to predict the QoS values of that service, can be calculated from the active service user's historical records and similar users of him/her.

To predict the QoS values for the service *s*, our approach utilize users in above filtered set $\widetilde{\mathrm{U}}_i$ and their QoS experience by employing the followed equation:

$$\mathrm{Pre}_{r_{i,s}} = \overline{u}_i + \frac{\sum_{u_a\in\widetilde{\mathrm{U}}_i}\mathrm{con}_f(\mathrm{a},\mathrm{i})(r_{a,s}-\overline{u}_a)}{\sum_{u_a\in\widetilde{\mathrm{U}}_i}\mathrm{con}_f(\mathrm{a},\mathrm{i})} \qquad (6)$$

where $\overline{u}_a$, $\overline{u}_i$ are the vectors of average QoS values different Web services observed by the user u_a and u_i respectively, and $\mathrm{con}_f(\mathrm{a},\mathrm{i})$ is the convergent weight between u_i and u_a.

V. EXPERIMENT & CONCLUSION

A. Experimental Setup & Data collection

To verify the prediction accuracy of our proposed approach, we implemented a prototype Situation-aware Web service invoke system, which stimulates situation reasoning result from a pre-defined situations structure for volunteers from different areas to select Web services to invoke. Our services list contains 111 public available *handpicked* Web services. Services with *WSDL* file or address are invoked through *SOAP* based on the classes generated with java. The others are *RESTful* Web services and are invoked in *REST* way [16]. These Web services are invoked through a uniform interface and auto-adjust to its appropriate invoke method via an *xml* configuration file. The situations structure we defined has 25 situations in 3 different scenarios (Home, Office, and Cinema). In each scenario, the test person labels her activity and situation mapped to the situations structure. We assign the probability of appearance of each situation to the Web service users and they choose the wanted service under the stimulated environment as close to real as possible. A simple record is a quintuple in the following form:

< User Number, Situation Number, Selected Service Number, Response Status, QoS values (Error code) >

Response Status identifies the invoke process successful or not. In the dataset we collected, we found failure circumstances to be rare and most of the failure cases are mainly because a few particular service providers (e.g., *imdb.com - Internet Movie Database*) are intercepted by network gateway. The fifth element is the Error Code if Response Status value of the same row is in the range of failure, otherwise it will be QoS values like Response Time (the time duration between a service user sending a request and receiving the corresponding response). Our dataset contains 600+ real Web services choices, relevant situational information and the executed results. Since this experiment

focuses on utilizing Situation-aware features to recommend Web services and investigating the potential interrelationship between the situation abstraction and services selection, both functional satisfaction and missing value prediction accuracy should be evaluated.

B. Measuring Functional Satisfaction

Hitherto, we have introduced the implementation details and gathered enough data to evaluate our approach. In simpler terms, a *hit rate* is a term used to describe the success rate of a recommendation effort. It is practical to measure our recommendation method through the actually invoked services from the records against the service in $V_{r(i,\varepsilon)}$ with highest *recommendation degree*. We measure the functional satisfaction from the aspect of discovering the relationship between the convergent size of $\widetilde{U}_i$ and the hit rates they brought by.

In this part, there are 30 randomly selected training users and the rest 8 test users are divided into 4 test groups. For each test groups, we change the size of $\widetilde{U}_i$ during the recommendation processes. Fig. 6 demonstrates the *hit rate* analysis result by using different convergent set size of $\widetilde{U}_i$. The abscissa axis means the number of similar users in $\widetilde{U}_i$, and the vertical axis represents the average recommendation hit rate of randomly selected situations of all users in each test group (marked as *TG1~4*). The figure shows that there are rises of average hit rates when the convergent set size becomes larger. And they approached to flats after the sizes come to 5 until reaching 8. Note that the reasonable choice of *Top-N* users is related to the scale of the dataset and if we change the density of the training records the hit rates might decrease when the convergent size continually grows up. In general, either size = 5 or 6 is the suitable choice of our experiments.

C. QoS Value Prediction Accuracy Evaluation

Another result of our approach is the QoS prediction of the recommended service. To evaluate this kind of accuracy performance, we make use of *Mean Absolute Error* (MAE) to measure the accuracies and compare them with other commonly used QoS prediction approaches. MAE is a statistical accuracy metric which is widely used to measure the prediction quality in collaborative filtering methods [4, 5, 12]. It is defined as the average absolute deviation of the predictions and is computing using the following equation:

$$\mathrm{MAE} = \frac{\sum_{i,j} |\mathrm{Pre}_{r_{i,j}} - r_{i,j}|}{N} \quad (7)$$

where $\mathrm{Pre}_{r_{i,j}}$ denotes the predicted QoS values of Web service s_j to user u_i computed through the method in section 4.B, $r_{i,j}$ denotes the executed result one, and N denotes the number of the records involved in. Smaller MAE values represent higher prediction accuracy.

This time we use the correctly recommended records from the test users' invoke histories. Similar to comparisons in other literature, we compare our method with common prediction methods like use-based algorithm using PCC (UPCC), item-based algorithm using PCC (IPCC), user-mean (UMEAN) and item-mean (IMEAN) to evaluate its prediction performance. Algorithm details of UPCC, IPCC, UMEAN and IMEAN can be found in [3, 4, 5, 13], and we do not repeat here. We vary the number of convergent users as 1, 3, 5 and 8 at the beginning of the recommendation process to study the prediction performance. By contrast, we also give the same parameters to the compared methods (e.g. vary the Top-K users/items selection). We take *Round-trip Time* (RTT) as the main QoS value since our hand-picked services obtain pretty low failure rates. Each experiment is performed 10 times and their average values are taken as results.

The experiment results in Fig.7 show that user-based methods (UPCC and UMEAN) work inferior to the service item-based methods (IPCC and IMEAN). This phenomenon is common in other literatures and may be prompted by the distribution of the *Autonomous Systems* (ASs) where the service users located. It can be seen from the table that our Situation-aware Recommendation method obtains smaller MAE values. The MAEs share obvious decreasing tendency with the increase number of convergent users (from 1 to 5), and almost all level off when the given number continues to increase. This shared feature suggests that both goals of our method: 1)meet user's functional requirements and 2)predict the QoS experience, are fulfilled well in a similar trend.

Finally, as we add the *structure weight* w_φ in filtering process to balance the differences between the services selections in different situations, we conduct experiments 10 times by varying θ (in Algorithm 1) to study its impacts. As shown in Table I, θ=1 indicates the results of our method

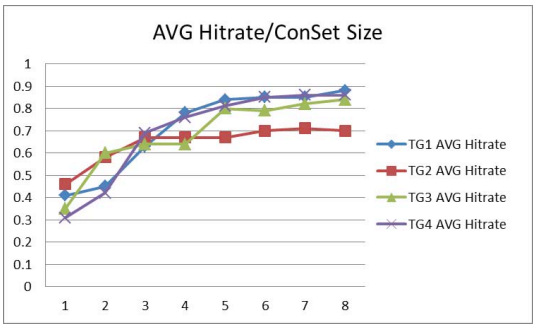

Figure 6. Impact of the convergent size to recommendation hit rates.

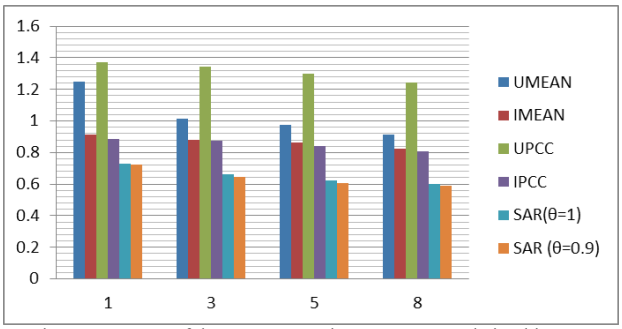

Figure 7. Impact of the convergent size to recommendation hit rates.

TABLE I. MAE PERFORMANCE COMPARISON

Given	1	3	5	8
UMEAN	1.249	1.016	0.974	0.913
IMEAN	0.916	0.881	0.864	0.823
UPCC	1.369	1.343	1.297	1.244
IPCC	0.884	0.875	0.841	0.806
SAR(θ=1)	0.728	0.663	0.623	0.602
SAR (θ=0.9)	0.721	0.648	0.605	0.591

without employing *structure weight*. The last row shows the results with θ=0.9 and the average w_{φ} in our situations structure 0.87. The experiments with an appropriate θ further improve the prediction accuracies, and the overall predicted result is in accordance with Situation-aware features.

D. Conclusions and Future work

This paper presents a service recommendation approach based on Situation-awareness. We explored a way of using situational information to recommend service which better satisfy the requirement of users, meanwhile, improve the QoS prediction accuracy. Through experiments on a prototype system we collected real Web services choices in situations labeled environment, and investigated the relationships between the situations structure and the services selection. By comparison with other existing methods, our method achieves higher performance. The experiment result of validating the situations structure's impact is in accordance with our expectation. In this study, we merely consider a few relationships between situations to assist the recommendation process. There are still many features of *Situation-aware Computing* can be utilized in the future. Our future work will mainly focus on the integration of *Situation-aware Computing* and *Service-Oriented Computing* to provide high quality services in sensor networks.

ACKNOWLEDGMENT

The work is supported in part by the following funds: Genie Project of the Danish Council for Strategic Research (Grant#2106-080046); National Natural Science Foundation of China Project (Grant#61033005).

REFERENCES

[1] M. Weiser, "Some computer science issues in ubiquitous computing," Mobile Computing and Communications Review, no.3, 1999, 12.

[2] J. O'Sullivan, D. Edmond, and A.t. Hofstede, "What's in a Service?" Distributed and Parallel Databases, vol. 12, Sept. 2002, nos. 2-3, pp. 117-133.

[3] M. Zhang, X. Liu, R. Zhang, H. Sun, "A Web Service Recommendation Approach Based on QoS Prediction Using Fuzzy Clustering," Proceedings of International Conference on Services Computing, 2012, pp. 138-145.

[4] L. Kuang, Y. Xia, Y. Mao, "Personalized Services Recommendation Based on Context-Aware QoS Prediction," Proceedings of International Conference on Web Services, 2012, pp. 400-406.

[5] M. Tang, Y. Jiang, J. Liu, X. Liu, "Location-Aware Collaborative Filtering for QoS-Based Service Recommendation," Proceedings of International Conference on Web Services, 2012, pp. 202-209.

[6] AK. Dey, "Providing architectural support for building context-aware applications," PhD thesis, Georgia Institute of Technology, 2000.

[7] SW. Loke, "Representing and reasoning with situations for context-aware pervasive computing: a logic programming perspective," Knowl Eng Rev Vol. 19, 2004, No. 3. pp. 213-233, doi: 10.1017/S0269888905000263.

[8] J. Barwise, J. Perry "Situations and Attitudes, " MIT Press, Cambridge, MA, USA, 1983.

[9] T. Gu, HK. Pung, D. Zhang, "A service-oriented middleware for building context-aware services," Network and Computer Applications, 28, 2005 (1), pp. 1-18.

[10] Y. Shoham, AD. Val, "A Logic for Perception and Belief," Department of Computer Science, Stanford University, 1991.

[11] SW. Loke, "On representing situations for context-aware pervasive computing: six ways to tell if you are in a meeting" . In 'PerCom Workshops', IEEE Computer Society, 2006, pp 35-39.

[12] M. Deshpande, G. Karypis, "Item-based top-n recommendation algorithms," ACM Transactions on Information Systems TOIS 22(1), 2004, pp. 143-177.

[13] Z. Zheng, H. Ma, M. Lyu, I. King, "WSRec: A Collaborative Filtering Based Web Service Recommender System," Proceedings of International Conference on Web Services, 2009, pp. 437-444, doi: 10.1109/ICWS.2009.30.

[14] W. Lo, J. Yin, S. Deng, Y. Li, Z. Wu, "An Extended Matrix Factorization Approach for QoS Prediction in Service Selection", Proceedings of International Conference on Services Computing, 2012, pp. 162-169, doi: 10.1109/SCC.2012.36.

[15] M. McLaughlin, J. Herlocker, "A collaborative filtering algorithm and evaluation metric that accurately model the user experience," 27th annual international ACM SIGIR conference on Research and development in information retrieval, 2004, pp. 329-336, doi: 10.1145/1008992.1009050.

[16] http://www.ibm.com/developerworks/webservices/library/ws-restful