AWSR: Active Web Service Recommendation Based on Usage History

Guosheng Kang1, JIANXUN LIU1, MINGDONG TANG1, XIAOQING (Frank) LIU2, BUQING CAO1, YU XU1

1Department of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan, China
2Department of Computer Science, Missouri University of Science & Technology, Rolla, USA
{guoshengkang, ljx529, tangmingdong}@gmail.com

Abstract—Web services are very prevalent nowadays. Recommending Web services that users are interested in becomes an interesting and challenging research problem. In this paper, we present AWSR (Active Web Service Recommendation), an effective Web service recommendation system based on users’ usage history to actively recommend Web services to users. AWSR extracts user’s functional interests and QoS preferences from his/her usage history. Similarity between user’s functional interests and a candidate Web service is calculated first. A hybrid new metric of similarity is developed to combine functional similarity measurement and nonfunctional similarity measurement based on comprehensive QoS of Web services. The AWSR ranks publicly available Web services based on values of the hybrid metric of similarity, so that a Top-K Web service recommendation list is created for a user. AWSR has been implemented and deployed on the Web. By conducting large-scale experiments based on a real-world Web services dataset, it is shown that our system effectively recommends Web services based on users functional interests and non-functional requirements with excellent performance.

Keywords—Web service, Web service recommendation, User interests, Usage history, TF/IDF, QoS

I. INTRODUCTION

As a new Web paradigm, Web services have been rapidly developed in recent years and played an increasingly important role in e-commerce, enterprise application integration, and other applications. As a result, the number of available Web services is increased drastically on the Internet. Users may find it hard to select Web services that meet their functional and non-functional requirements. Recently, Web service recommendation systems are attracting more and more attention since they enable users to find Web services with high QoS (Quality of Service) in an effective way. Most prior Web service recommendation approaches are based on collaborative filtering [1], which requires to compute similarity of users or services and can predict missing QoS values for users based on the QoS records of similar users or services. They are usually employed to find the Web service with the highest QoS scores in terms of response time, throughput, availability, reliability, or other quality factors. The main idea behind collaborative filtering is that if a user shares similar interest/tastes/opinions with other users on some items such as books, web pages, it is very likely that new services requested by the user will be similar to those of the similar users. Based on this rationale, the items selected by similar users will be recommended. However, existing collaborative filtering approaches take very little consideration on users’ own preferences and interests. Due to lack of a user’s preference, comprehensive QoS of Web services may not be accurately aggregated and computed in service recommendation. In addition, recommended Web services may not meet the user’s functional requirements at all since the existing recommendation approaches place an emphasis on Web services’ QoS of a list of quality factors and ignore users’ interests. Therefore, it is necessary to develop an active Web service recommendation method for recommending the best top-K Web services which meet both users’ functional and non-functional requirements based on their interests and usage history.

In this paper, we present AWSR (Active Web Service Recommendation), a novel Web service recommendation approach and framework based on both users’ interests and preferences according to usage history to recommend Web services actively. Usage history consists of used Web services and their corresponding preference records. AWSR extracts user interests and preferences from the user’s historical usage of Web services, based on the WSDL documents of the Web services invocated by the user, and the user’s preference records, respectively. The comprehensive experimental analysis shows that our recommendation system can recommend Web Services which are consistent with users’ interests and preferences from usage history and satisfy their functional and non-functional requirements. In particular, contributions of this paper are as follows:

- We present an innovative Web service recommendation approach based on user interests, usage history, and comprehensive QoS of the Web services.
- We implement the recommendation system prototype in our Web Services Supermarket [2]. We publicly released our large-scale real-world Web service WSDL documents and associated datasets, which cover 2000 Web Services.
- We conduct a large-scale experimental evaluation on real-world Web services. 10,000 Web services all over the world are gathered and used in the evaluation. And the experiments show that our recommendation system can effectively recommend suitable Web services to users consistent with users’ interests and preferences with high QoS values.

The rest of this paper is organized as follows: Section II introduces related works. Section III presents Web service recommendation scenarios, and system framework and architecture. Section IV discusses our active Web service recommendation approach in detail. Section V covers
implementation of the system and describes experimental results. At last, we draw conclusions and discuss our future work in Section VI.

II. RELATED WORK

With an increasing number of available Web services on the Internet, it can be problematic to find a suitable Web service which meets user’s requirements both functionally and non-functionally. QoS-based approaches for Web service selection have been investigated in a number of recent literatures [3-8], intending to identify optimal Web services from a set of candidates according to users’ requests considering both functional and non-functional requirements. These approaches are somewhat passive in discovering Web services for users since they highly depend on users’ requests. Previous Web service recommendation approaches are typically based on collaborative filtering (CF) [9]. Specifically, Ref. [10] proposes a user-based CF algorithm using PCC (Pearson Correlation Coefficient) to compute similarity between users. Users who have similar historical QoS experiences on Web services are deemed to be similar. For any active user, the missing QoS values of a Web service can be predicted by considering the corresponding QoS values of services used by his/her similar users. Finally, Web services with high predicted QoS values are recommended. Ref. [11] proposes a novel hybrid collaborative filtering algorithm for Web service QoS prediction by systematically combining both item-based PCC (IPCC) and user-based PCC (UPCC). Adapted from [11], Ref. [1] presents an improved similarity measurement for users and Web services, which takes the personal characteristics of users and Web services into account when calculating similarity using PCC. Experimental results in [1] show that the adaptation can make the QoS value prediction more accurate. The authors of Ref. [12] also recognize the influence of the characteristics of Web services’ QoS. According to their observation, QoS parameters of Web services (such as responding time, reliability), regarded as a set of user-perceived properties, are highly related to users’ physical locations. Therefore, different from the aforementioned work, [12] proposes an scalable hybrid CF algorithm, which incorporates users’ locations to help identify similar users. Experimental results show the efficiency and accuracy of QoS prediction are both improved compared to prior recommendation algorithms.

However, previous QoS-based Web service recommendation approaches usually aim to predict the values of QoS attributes of Web services and consider little about Web services’ usage history to extract users’ interests or preferences. As a consequence, they cannot be directly employed in a real Web service recommendation system to recommend services actively without users’ queries. However, these approaches are very useful to filter and rank Web services on non-functional requirements in Web service selection. Based on the recorded user invocation and query history to improve the accuracy of the user similarity calculation, ref [13] proposes a Web service selection system which combines QoS-based matching score and the collaborative filtering based score. Actually, employing users’ interests to actively recommend products or information has been widely studied in lots of literatures [14-17] and adopted by many recommendation systems in e-commerce applications, such as, Amazon, eBay, CDNOW, FAB, Music.Yahoo.com, etc., which only consider user interests, since QoS is not a main factor to them when recommending. However, to the best of our knowledge, no such studies appear in the field of Web service recommendation. Our work fills the gap by proposing a practical Web service recommendation framework based on user’s interests and preferences from service usage history. Different from the existing approaches, we extract user interests according to the WSDL documents based on the historically invoked Web services of the user. We also extract user’s potential preference on QoS according to his/her historical QoS preference records. With the extracted user interests and potential QoS preferences, we can successfully recommend suitable Web services to users with desired functionalities and QoS values even without users’ queries.

III. PRELIMINARIES

A. A Motivating Example

Figure 1 shows a common Web service recommendation scenario, where weight $i$ presents a QoS preference vector which was given by a user when using the $i^{th}$ historical Web service in the past. In this example, a user used 5 Web services, and his used Web services and preferences are recorded in the system. The user, of course, hope that the system can recommend and provide personalized Web services to him. In a dynamic service environment, when new Web services with high QoS and user-interested functionalities emerge on the Internet, the user hopes to be informed through recommendation, so that the user can narrow down the scope of search of services to reduce search time. As illustrated in Figure 1, the user used two Web services to book tickets first and three ones to book hotels later. Here, according to the history of his used Web services, it is reasonable to infer that the user is likely to travel in the future. When new Web services about travel or tourism, including services of booking flight tickets and booking hotel, emerge, they should be recommended to this user. In addition, the user hopes that the QoS of recommended services should satisfy his preferences.
Table I shows usage history examples. The used Web services and their QoS preference records are listed in the table. Considering that WSDL documents are used for describing Web services and can be obtained easily by Web service search engines, such as Binding Point, Grand Central, Seekda, Web Service List and Web Services Supermarket [2]. We will analyze the WSDL documents of used Web services and the preference records so that the potential user interests and preferences can be obtained. Then the system transforms the user interests and preferences into the user’s potential request, where the user interests corresponds to the user’s functional requirements and preferences corresponds to the user’s non-functional requirements, respectively.

In Table II we list three possible recommended Web services which are related to travel. The recommended Web services are listed in a top-down order. According to the Table 1, it can be known that the user considers the first QoS attribute more importantly than the second one. Though $W_2$ and $W_1$ are pretty close in terms of QoS, the comprehensive QoS of $W_2$ is better than that of $W_2$ for this user.

### TABLE I  USAGE HISTORY EXAMPLES

<table>
<thead>
<tr>
<th>Usage History</th>
<th>WSDL</th>
<th>Weight($w_1$, $w_2$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>record_1</td>
<td><a href="http://services.laureloak.it/TicketSystem.asmx?WSDL">http://services.laureloak.it/TicketSystem.asmx?WSDL</a> (0.9, 0.1)</td>
<td></td>
</tr>
<tr>
<td>record_2</td>
<td><a href="http://degel.ise.bgu.ac.il/iSpaceBook/Service.asmx?WSDL">http://degel.ise.bgu.ac.il/iSpaceBook/Service.asmx?WSDL</a> (0.7, 0.3)</td>
<td></td>
</tr>
<tr>
<td>record_3</td>
<td><a href="http://web.newhotel.com/WSNewHotelSrvWSNewHotel.asmx?WSDL">http://web.newhotel.com/WSNewHotelSrvWSNewHotel.asmx?WSDL</a> (0.6, 0.4)</td>
<td></td>
</tr>
<tr>
<td>record_4</td>
<td><a href="http://www.allysoft.ru/XML/Connector.asmx?WSDL">http://www.allysoft.ru/XML/Connector.asmx?WSDL</a> (0.8, 0.2)</td>
<td></td>
</tr>
<tr>
<td>record_5</td>
<td><a href="http://ws1.touricoholidays.com/HotelsService.asmx?wsdl">http://ws1.touricoholidays.com/HotelsService.asmx?wsdl</a> (0.7, 0.3)</td>
<td></td>
</tr>
</tbody>
</table>

### TABLE II  TOP-3 WEB SERVICES RECOMMENDATION LIST

<table>
<thead>
<tr>
<th>Top-3</th>
<th>WSDL</th>
<th>QoS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$W_3$</td>
<td><a href="http://ws.henoo.net/travelsearch.asmx?WSDL">http://ws.henoo.net/travelsearch.asmx?WSDL</a> (0.8, 0.7)</td>
<td></td>
</tr>
<tr>
<td>$W_2$</td>
<td><a href="http://puturist.com/Admin/Travel/WebServices/Travel/TravelService.asmx?wsdl">http://puturist.com/Admin/Travel/WebServices/Travel/TravelService.asmx?wsdl</a> (0.7, 0.8)</td>
<td></td>
</tr>
<tr>
<td>$W_3$</td>
<td><a href="http://ts.afnt.co.uk/travelSearch.asmx?WSDL">http://ts.afnt.co.uk/travelSearch.asmx?WSDL</a> (0.7, 0.6)</td>
<td></td>
</tr>
</tbody>
</table>

### B. System Framework and Architecture

Now we describe the system framework of our Active Web Service Recommendation system. As shown in Figure 2, AWSR recommends a Top-K Web service recommendation list for a user according to the usage history and Web services which are obtained by our Web service search engine Web Services Supermarket from the Internet.

The AWSR is the core of the framework. From Figure 2, it can be seen that the AWSR must be provided with the user’s usage history and Web services from the Internet. AWSR consists of three components: functional evaluation, non-functional evaluation and QoS-aware Web service ranking, as shown in Figure 3. There are two phases in the functional evaluation component. In phase 1, AWSR obtains user interests as the user’s functional requirements. AWSR constructs user interests according to the WSDL documents in the user’s historical record. In phase 2, AWSR evaluates the Web service candidates’ functional features. These features are described by similarities between the user interests obtained in phase 1 and WSDL documents of Web services in our Web Services Supermarket. The nonfunctional evaluation component also has two phases. In phase 1, AWSR obtains user preferences as user’s non-functional requirements. In phase 2, AWSR computes the QoS utilities of different Web services according to the preferences obtained in phase 1. Finally, AWSR combines both functional and non-functional features of Web services for ranking in the QoS-aware Web service ranking component. A Top-K Web service recommendation list is then generated and provided based on the user’s interests and preferences from the usage history.

### IV. ACTIVE WEB SERVICE RECOMMENDATION APPROACH

#### A. Similarity Model

Now we describe a model for computing similarities between a user functional interests and a candidate Web service. All the Web services discovered by the search
engine are looked upon as the corpus and then we use TF/IDF (Term Frequency/Inverse Document Frequency) [18] algorithm to weight the importance of terms in the corpus. TF/IDF is a statistical measure used to evaluate how important a word is to a document in a corpus. The importance increases proportionally to the number of times a word appearing in the document but is offset by frequency of the word in the corpus. Variations of the TF/IDF weighting scheme are often used by search engines as a central tool in scoring and ranking a document’s relevance given a user query [19]. In particular, it analyzes the most common terms appearing in each Web service document and appearing less frequently in other documents.

We extract the meaningful words from the WSDL documents to form the corpus. WSDL is the Web service description document which conforms to the rules of XML document. There may be a lot of words or tokens which need to be preprocessed. There are mainly three ways of WSDL Preprocessing as follows:

1. **Normalization**
   As for the misspelled and abbreviated words in real-word WSDL documents, they need to be replaced by normalized forms.

2. **Word Stemming**
   A stem is the basic part of a word that never changes even morphologically infected. As for these words with prefix (e.g., -in, -un, -dis, -non, et al) or suffixes (e.g., -s, -es, -ed, -er, or,-ing, -ion, et al), they must be performed word stemming to eliminate the difference among inflectional morphemes.

3. **Tokens and Stop-words Removing**
   As for the tokens and stop-words with little substantive meaning in real-word WSDL documents, they must be removed when preprocessing.

The term count in the given document is simply the number of times a given term appears in that document. This count is usually normalized to prevent a bias towards longer documents (which may have a higher term count regardless of the actual importance of that term in the document) to give a measure of the importance of the term $t_{i,j}$ within a particular document $WSDL_i$. Thus we have the term frequency $tf(t_{i,j}, WSDL_i)$, defined in the simplest case as the occurrence count of a term in a document. After WSDL documents are preprocessed, we get valid terms and then term frequencies are calculated according to formula (1).

$$tf(t_{i,j}) = \frac{freq(t_{i,j}, WSDL_i)}{|WSDL_i|}$$

Where
- $tf(t_{i,j})$ is $j^{th}$ term frequency of $i^{th}$ WSDL document;
- $t_{i,j}$ is $j^{th}$ term in $i^{th}$ WSDL document;
- $WSDL_i$ is the WSDL document of $i^{th}$ Web service;
- $freq(t_{i,j}, WSDL_i)$ is the occurrence number of $t_{i,j}$ in $WSDL_i$;
- $|WSDL_i|$ is the number of terms in $i^{th}$ WSDL document.

The inverse document frequency is a measure of the general importance of the term (obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient), which is illustrated in formula (2).

$$idf(t_{i,j}) = \log \frac{|WSDL|}{|\{WSDL_i : t_{i,j} \in WSDL_i\}|}$$

Where
- $|WSDL|$ is the cardinality of WSDL, or the total number of documents in the corpus;
- $|\{WSDL_i : t_{i,j} \in WSDL_i\}|$ is the number of documents where the term $t_{i,j}$ appears (i.e., $tf(t_{i,j}) \neq 0$). If the term is not in the corpus, this will lead to a division-by-zero. It is therefore common to adjust the formula to $1 + |\{WSDL_i : t_{i,j} \in WSDL_i\}|$.

Due to that WSDL documents are generally short, we chose to give higher weight to the IDF value to normalize the inherent bias as formula (3), while the common implementation of TF/IDF gives equal weights to term frequency and inverse document frequency (i.e., $\omega = tf \ast idf$). The reason behind this modification is to normalize the inherent bias of TF measure in short documents [20].

$$\omega_{i,j} = tf(t_{i,j}) \ast idf^2(t_{i,j})$$

A high weight in TF/IDF is reached by a high term frequency (in the given document) and a low document frequency of the term in the whole collection of documents. The weights hence tend to filter out common terms.

Considering that user’s interests may change with time, to obtain user’s latest interests accurately, we construct a user interests vector by integrating WSDL documents of recently used Web services into one Big WSDL document. By using TF/IDF, the Big WSDL document can be transformed into a term vector and each WSDL document of Web services can be transformed into a term vector as well, which are defined as follows:

$$<User _ Interests> = \{(t_1, \omega_{1,1}), (t_2, \omega_{1,2}), \ldots, (t_m, \omega_{1,m})\}$$

$$<Web _ Service>_i := \{(t_1, \omega_{i,1}), (t_2, \omega_{i,2}), \ldots, (t_m, \omega_{i,m})\}$$

Here, if $t_k$ does not appear in Big WSDL document or WSDL document of $WS_i$, then $\omega_{i,k} = 0$ or $\omega_{i,k} = 0$. Finally, we measure the cosine similarity $S_i$ between user interests and $WS_i$ as formula (4).

$$S_i = \frac{\sum_{j} (\omega_{i,j} \times \omega_{j})}{\sqrt{\sum_{i} (\omega_{i,j} \times \omega_{j})}}$$

### B. QoS Model

By assuming $m$ criteria are used for assessing quality of a Web service, we can describe the service quality using a QoS vector $(q_{i,1}, q_{i,2}, \ldots, q_{i,m})$, where $q_{i,j}$ represents value of the $j^{th}$ quality attribute of Web service $i$. There are two types of QoS attributes. If the higher the value, the lower the quality, this QoS attribute would be considered as a negative criterion (e.g., Response Time and Price). On the other hand, if the higher the value, the higher the quality, this QoS attribute would be considered as a positive criterion (e.g., Availability and Reliability). Each QoS
criterion value should be normalized to achieve uniform measurement. In this section, we transform each criterion value to a real value between 0 and 1 by comparing it with the maximum and minimum values of that particular criterion among all available Web service candidates. Concretely, for a negative criterion, the normalized value of \( q_{i,j} \) would be scaled by \( q'_{i,j} \) according to formula (5), and for a positive criterion, \( q_{i,j} \) would be scaled by \( q''_{i,j} \) according to formula (6) which are defined as follows.

\[
q'_{i,j} = \begin{cases} 
\frac{Q_{\text{max}}(j) - q_{i,j}}{Q_{\text{max}}(j) - Q_{\text{min}}(j)}, & \text{if } Q_{\text{max}}(j) \neq Q_{\text{min}}(j) \\
1, & \text{if } Q_{\text{max}}(j) = Q_{\text{min}}(j)
\end{cases} \tag{5}
\]

\[
q''_{i,j} = \begin{cases} 
\frac{q_{i,j} - Q_{\text{min}}(j)}{Q_{\text{max}}(j) - Q_{\text{min}}(j)}, & \text{if } Q_{\text{max}}(j) \neq Q_{\text{min}}(j) \\
1, & \text{if } Q_{\text{max}}(j) = Q_{\text{min}}(j)
\end{cases} \tag{6}
\]

Where the maximum value \( Q_{\text{max}}(j) \) and minimum value \( Q_{\text{min}}(j) \) of the criterion are computed as follows:

\[
Q_{\text{max}}(j) = \max_{k \in [1,n]} q_{k,j},
\]

\[
Q_{\text{min}}(j) = \min_{k \in [1,n]} q_{k,j}
\]

A weight vector \( W = (w_1, w_2, ..., w_m) \) is used to represent user’s preferences given to different criterions with \( w_i \in \mathbb{R}^+ \) and \( \sum_{j=1}^{m} w_j = 1 \). The following formula is used to compute the utility of \( WS_i \)

\[
U_i = \sum_{j=1}^{m} q''_{i,j} \times w_j \tag{7}
\]

In the AWSR, we derive potential preferences from the historical preferences of the user. The following formula is used to compute the potential preferences of the user, where \( W_i \) is the preference when using record, and \( n \) is the total number of Web services used by the user in the past. Take Table 1 as an example, \((0.9 + 0.7 + 0.6 + 0.8 + 0.7)/5 = 0.74 \) and \((0.1 + 0.3 + 0.4 + 0.2 + 0.3)/5 = 0.26 \), the potential preference of the user would be \((0.74, 0.26)\).

\[
W = \sum_{i=1}^{n} W_i / n \tag{8}
\]

C. QoS-Aware Web Service Recommendation

After obtaining user interests and the potential user preferences discussed respectively in part A and B in this section, the functional component computes the similarity \( S_i \) between user functional interests and \( WS_i \), and its non-functional component of AWSR computes QoS utility of each \( WS_i \). A final rating score \( R_t \) is used to evaluate the conformity of each \( WS_i \) to achieve the recommendation goal, which is shown as follows:

\[
R_t = \lambda \times \frac{1}{\log_2 (P_{S_i} + 1)} + (1 - \lambda) \times \frac{1}{\log_2 (P_{U_t} + 1)} \tag{9}
\]

Where \( P_{S_i} \) is the functional rank position and \( P_{U_t} \) is the non-functional rank position of \( WS_i \) among all Web service candidates. Since the absolute values of similarity and service quality indicate different features of Web services and include different units and ranges, rank positions rather than absolute values are a better choice to indicate the appropriateness of all candidates [21]. The \( 1/\log_2 (p + 1) \) calculates the appropriateness value of a candidate in position \( p \) for a user’s potential requirements. \( \lambda \in [0,1] \) defines how much the functionality factor is more important than the non-functionality factor in the final recommendation. Here, \( \lambda \) can be a constant to allocate a fixed percentage of the two parts’ contributions to the final rating score \( R_t \). However, it is more reasonable if \( \lambda \) is expressed as a monotone increasing function with \( P_{S_i} \), which is illustrated as follows:

\[
\lambda = f(P_{S_i})
\]

\( \lambda \) is larger when the position in the similarity rank is lower. This means a Web service is inappropriate if it cannot provide the required functionality to the user no matter how high the comprehensive QoS is. The relationship between recommendation accuracy and the formula of \( \lambda \) will be identified to extend the AWSR prototype in our future work.

V. EXPERIMENTS AND IMPLEMENTATION

In this section, firstly we will present the implementation of our system. Secondly, we will discuss experiments to study the performance of our approach compared with the QoS-based only approach and Interest-based only approach.

A. Implementation

We have developed AWSR in our Web Services Supermarket. Figure 4.a shows the login page of our Web Services Supermarket. Figure 4.b shows the page after clicking to check Top-10 Web service recommendation list generated and provided for the user in the previous example mentioned in section 3.1, and the first two Web services about travel can be seen here as expected. The user can search its needed Web services by developing a query or probably by viewing the Top-10 Web service recommendation list by a simple click below, which offers the user much convenience. As the user uses more and more Web services, our system can recommend more accurate Web services that the user is really interested in. Most importantly, in some cases, the user may prefer the recommended services rather than that searched by a query. In other cases, after browsing the recommended services, the user would be able to form a query better.
B. Evaluate Top-K Web service Recommendation List

Now we discuss the experiments in detail. We conduct large-scale real-world experiments to study the performance of the Top-K Web services recommended by our recommendation system. To obtain real-world WSDL documents, we developed a Web Services Supermarket with a Distributed Web Service Search Engine Based on Map/Reduce to crawl WSDL links from different Web resources (e.g., UDDI, Web service portal, and Web service search engine). We obtain totally more than 10,000 Web services from the Internet. To measure the non-functional attributes of the available Web services, we developed a Distributed Testing Platform for Web Services to test QoS of Web services continuously. In our experiments, we choose 2000 Web services as a dataset and assume that the user has randomly used 10 Web services in the dataset.

In most of the searching scenarios, users tend to look at the top items of the returned result list. The items in higher position, especially the first position, are more important than the items in lower positions in the returned result list. It is the same as our recommendation system. To evaluate the quality of Top-K Web service recommendation list, we use the well-know DCG (Discounted Cumulative Gain) [22] approach as performance evaluation metric. DCG value can be calculated by the following formula:

\[
DCG_k = \sum \frac{(2^{U_i} - 1)}{\log_2(1 + p_i)}
\]

where \(U_i\) is the \(WS_i\) QoS utility value, \(p_i\) is the rank position of \(WS_i\) in Top-K Web service recommendation list, and \(DCG_k\) is the discounted cumulative gain of Top-K QoS utilities in a Web service recommendation list. The gain is accumulated starting at the top of ranking and discounted at lower ranks. A large \(DCG_k\) value means a high QoS utility of the Top-K recommended Web services. This experiment is to evaluate the QoS of recommended Web services.
In Figure 5 shows the DCG values of Top-K recommended Web services. (a) and (b) show the Top-K DCG values of our AWSR system are a little lower than QoS-based approach (when $\lambda = 0.5$, 35.6320 of AWSR compared with 53.0293 of QoS-Based for Top-5 and 42.4148 of AWSR compared with 60.0535 of QoS-Based for Top-10). QoS-based approach recommends Web services with high comprehensive QoS no matter whether the user is interested in the Web services or not. In contrast, our AWSR recommends Web services which the user is interested in and have high comprehensive QoS values as well. So it is normal that the Top-K DCG values of AWSR are a little lower than QoS-based approach, but the values of the two are very close, which is more apparent when $\lambda = 0.4$. (c) and (d) show Top-K DCG values of our AWSR system are considerably much higher than Interest-based approach (i.e., 35.6230 of AWSR compared with 7.6224 of Interest-Based for Top-5 and 42.4148 of AWSR compared with 12.2775 of Interest-Based for Top-10, when $\lambda = 0.5$). With the increase of $K$, the disparity of the two becomes increasingly larger. Even when $\lambda$ becomes smaller, the trend still does not change. Overall, AWSR can recommend high quality Web services based on usage history in the first position.

Similarly, to evaluate the user interests matching, we also use DCG which is illustrated in formula (11) to compare it with the QoS-based only approach and Interest-based only approach.

$$DCG_k = \sum \frac{(2^{S_i} − 1)}{\log_2(1 + p_i)}$$

where $S_i$ is the similarity between user functional interests and $WS_i$, $p_i$ is the rank position of $WS_i$ in Top-K Web service recommendation list, and $DCG_k$ is the discounted cumulative gain of Top-K user interest similarities in Web service recommendation list. The gain is accumulated starting at the top of the ranking and discounted at lower ranks. A large $DCG_k$ value means high user interest similarities of the Top-K recommended Web services. This experiment aims to evaluate the user interest relevance of recommended Web services.

In Figure 6, (a) and (b) show the Top-K DCG values of our AWSR system are considerably much higher than QoS-based approach (i.e., 1.2724 of AWSR compared with 0.0130 of QoS-Based for Top-5 and 1.5898 of AWSR compared with 0.0188 of QoS-Based for Top-10, when $\lambda = 0.5$). As can be seen from (a) and (b) in Figure 6, Top-K DCG values of our AWSR system are constantly much higher than QoS-based approach and with the increase of $K$, the disparity of the two becomes increasingly larger. Even when $\lambda$ becomes smaller, the trend still does not change. Though, in Figure 5, the Top-K DCG values of AWSR are a little lower than QoS-based approach, the recommended Web services by QoS-based approach are hardly related to user’s interests according to Figure 6. Therefore, Web services recommended by QoS-based approach may be meaningless to the user. In contrast, much more relevant Web services are recommended in high positions by our approach. (c) and (d) show the Top-K DCG values of our AWSR system are lower than Interest-based approach (when $\lambda = 0.5$, 1.2724 of...
AWSR compared with 1.6514 of Interest-Based for Top-5 and 1.5898 of AWSR compared with 2.1562 of Interest-Based for Top-10. Interest-based approach recommends Web services with high interest relevance no matter whether the comprehensive QoS of Web services is good or not. In contrast, our AWSR recommends Web services which have high comprehensive QoS values and the user is interested in as well. So it is normal that the Top-K DCG values of AWSR are comparatively lower than QoS-based approach. Overall, AWSR is able to recommend the interest-relevant Web services with high comprehensive QoS values.

VI. CONCLUSION

In this paper we present an active Web service recommendation approach to find the desired Web services for users actively based on their usage history. Both functional (user interests) and non-functional (user preference) characteristics of Web services are incorporated in our approach. We extract user’s functional interests and QoS preferences from their usage history. Then, TF/IDF is applied to calculate the similarity between the user’s functional interests and Web services. Combining the functional similarity and non-functional similarity based on comprehensive QoS of Web services, AWSR ranks the publicly available Web services so that a Top-K Web service recommendation list is generated for a user.

Large-scale experiments with real-world data in a distributed environment are conducted to study the performance of our AWSR prototype. The results show that our approach can recommend Web services with desired user’s functional interests and non-functional requirement satisfaction effectively with excellent performance.

In future work, we will conduct data mining in our dataset to investigate formulas of $A$ in order to achieve optimized performance and make more experiments to evaluate the usefulness of our approach. Clustering algorithms for similarity computation and preference computation will be designed for improving the accuracy of assessment of user interests and preferences. More QoS attributes will be included in our study too.

ACKNOWLEDGMENT

The work was supported by the National Natural Science Foundation of China under grant No.90818004 and No.61100054, Program for New Century Excellent Talents in University under grant No.NCET-10-0140, Excellent Youth Foundation of Hunan Scientific Committee No.11JJ1011, and Scientific Research Fund of Hunan University under grant No.09K085 and No.11B048.

REFERENCES


