Abstract : Web services are integrated software components for the support of interoperable machine-to-machine interaction over a network. Web services have been widely employed for building service-oriented applications in both industry and academia in recent years. The number of publicly available Web services is steadily increasing on the Internet. However, this proliferation makes it hard for a user to select a proper Web service among a large amount of service candidates. An inappropriate service selection may cause many problems (e.g., ill-suited performance) to the resulting applications. In this paper, we propose a novel collaborative filtering-based Web service recommender system to help users select services with optimal Quality-of-Service (QoS) performance. Our recommender system employs the location information and QoS values to cluster users and services, and makes personalized service recommendation for users based on the clustering results. Compared with existing service recommendation methods, our approach achieves considerable improvement on the recommendation accuracy. Comprehensive experiments are conducted involving more than 1.5 million QoS records of real-world Web services to demonstrate the effectiveness of our approach.

I. INTRODUCTION
WEB services are software components designed to support interoperable machine-to-machine interaction over a network, usually the Internet. Web service employs WSDL (Web Service Description Language) for interface description and SOAP (Simple Object Access Protocol) for exchanging structured information. Benefiting from the cross-language and cross-platform characteristics, Web services have been widely employed by both enterprises and individual developers for building service-oriented applications. The adoption of Web services as a delivery model in business has fostered a paradigm shift from the development of monolithic applications to then dynamic set-up of business processes. When developing service-oriented applications, developers first design the business process according to requirements, and then try to find and reuse existing services to build the process. Currently, many developers search services through public sites like Google Developers (developers.google.com), Yahoo! Pipes (pipes.yahoo.com), programmableWeb (programmableweb.com), etc. However, none of them provide location-based QoS information for users. Such information is quite important for software deployment especially when trade compliance is concerned. Some Web services are only available in EU, thus software employing these services cannot be shipped to other countries. Without knowledge of these things, deployment of service-oriented software can be at great risk. Since selecting a high quality Web service among a large number of candidates is a non-trivial task, some developers choose to implement their own services instead of using publicly available ones, which incurs additional overhead in both time and resource. Using an inappropriate service, on the other hand, may add potential risk to the business process. Therefore, effective approaches to service selection and recommendation are in an urgent need, which can help service users reduce risk and deliver high-quality business processes. Quality-of-Service (QoS) is widely employed to represent the non-functional characteristics of Web services and has been considered as the key factor in service selection [33]. QoS is defined as a set of properties including response time, throughput, availability, reputation, etc. Among these QoS properties, values of some properties (e.g., response time, user-observed availability, etc.) need to be measured at the client-side [26]. It is impractical to acquire such QoS information from service providers, since these QoS values are susceptible to the uncertain Internet environment and user context (e.g., user location, user network condition, etc.). Therefore, different users may observe quite different QoS values of the same Web service. In other words, QoS values evaluated by one user cannot be employed directly by another for service selection. It is also impractical for users to acquire QoS information by evaluating all service candidates by themselves, since conducting real world Web service invocations is time consuming and resource-consuming. Moreover, some QoS properties (e.g., reliability) are difficult to be evaluated as long-
duration observation is required. To attack this challenge, this paper investigates personalized QoS value prediction for service users by employing the available past user experiences of Web services from different users. Our approach requires no additional Web service invocations. Based on the predicted QoS values of Web services, personalized QoS aware Web service recommendations can be produced to help users select the optimal service among the functionally equivalent ones. From a large number of real-world service QoS data collected from different locations, we find that the user-observed Web service QoS performance has strong correlation to the locations of users. Google Transparency Report1 has similar observation on Google services. To enhance the prediction accuracy, we propose a location-aware Web service recommender system (named LoRec), which employs both Web service QoS values and user locations for making personalized QoS prediction. Users of LoRec share their past usage experience of Web services, and in return, the system provides personalized service recommendations to them. LoRec first collects user-observed QoS records of different Web services and then groups users who have similar QoS observations together to generate recommendations. Location information is also considered when clustering users and services. The main contributions of this work are two-fold: First, we propose a novel location-aware Web service recommendation approach, which significantly improves the recommendation accuracy and time complexity compared with existing service recommendation algorithms. Second, we conduct comprehensive experiments to evaluate our approach by employing a real-world Web service QoS data set. More than 1.5 millions real-world Web service QoS records from more than 20 countries are engaged in our experiments. Comprehensive analysis on the impact of the algorithm parameters is also provided. The rest of this paper is organized as follows: Section 2 reviews related work of collaborative filtering and Web service recommendation. Section 3 presents the system architecture. Section 4 describes the proposed Web service recommendation algorithm. Section 5 shows our extensive experimental results, employing QoS values of real-world Web services, and Section 6 concludes the paper.

II. RELATED WORK

2.1 Collaborative Filtering

Collaborative Filtering (CF) is widely employed in commercial recommender systems, such as Netflix and Amazon.com [4], [18], [19], [22]. The basic idea of CF is to predict and recommend potential favorite items for a particular user employing rating data collected from other users. CF is based on processing the user-item matrix. Breese et al. [3] divide the CF algorithms into two broad classes: memory-based algorithms and model-based algorithms. The most analyzed examples of memory-based collaborative filtering include user-based approaches [3], [11], [15], item-based approaches [9], [18], [23], and their fusion [27]. User-based approaches predict the ratings of users based on the ratings of their similar users, and item-based approaches predict the ratings of users based on the information of item similarity. Memory-based algorithms are easy to implement, require little or no training cost, and can easily take ratings of new users into account. However, memory-based algorithms do not scale well to a large number of users and items due to the high computation complexity. Model-based CF algorithms, on the other hand, learn a model from the rating data using statistical and machine learning techniques. Examples include clustering models [30], latent semantic models [12], [13], latent factor models [5], and so on. These algorithms can quickly generate recommendations and achieve good online performance. However, these models must be rebuilt when new users or items are added to the system.

2.2 Service Selection and Recommendation

Service selection and recommendation have been extensively studied to facilitate Web service composition in recent years. Wang et al. [28] present a Web service selection method by QoS prediction with mixed integer program. Zhang et al. [34] provide a fine-grained reputation system for QoS-based service selection in P2P system. Zheng et al. [37] provide a QoS-based ranking system for cloud service selection. Zhu et al. [38] employ clustering techniques to their QoS monitoring agents and provide Web service recommendations based on the distance between each user and their agents. El-Hadad et al. [10] propose a selection method considering both the transactional properties and QoS characteristics of a Web service. Hwang et al. [14] use a finite state machine to model the permitted invocation sequences of Web service operations, and propose two strategies to select Web services that are likely to successfully complete the execution of a given sequence of operations. Kang et al. [16] propose AWS-R system to recommend services based on users’ historical functional interests and QoS preferences. Barakat et al. [2] model the quality dependencies among services and proposes a Web service selection method for Web service composition. Airifai and Risse [1] propose a method to meet users’ end-to-end QoS requirements employing integer programming (MIP) to find the optimal decomposition of global QoS constraints into local constraints. A certain amount of work has been done to apply CF to Web service recommendation. Shao et al. [24] employ a user-based CF algorithm to predict QoS values. Works in [17], [25] apply the idea of CF in their systems, and use Movie Lens data for experimental analysis. Combination tasks of different types of CF algorithms are also engaged in Web service recommendation. Zheng et al. [36] combine user-based and item-based CF algorithms to recommend Web services. They also integrate Neighborhood approach with Matrix
Factorization in their work [35], Yu [32] presents an approach that integrates matrix factorization with decision tree learning to bootstrap service recommender systems. Meanwhile, several tasks employ location information to Web service recommendation [7].

III. METHODOLOGY

Values of some QoS properties (e.g., response time) on the same Web service vary quite differently from user to user. Through the analysis of a real-world Web service QoS data set2 (see Section 5 for details), which contains 1.5 millions service invocation records evaluated by users from more than twenty countries, we find that some QoS properties highly relate to the physical locations of users. For example, the response time of a service observed by closely located users usually fluctuates mildly around a certain value. On the other hand, the response time observed by users who are far away from each other sometimes varies significantly. Based on this finding, our recommendation algorithm takes location information into consideration to improve the recommendation accuracy. Our recommendation algorithm is designed as a three-phase process, i.e., 1) user region creation, 2) service region creation, and 3) QoS prediction & recommendation, which will be presented in Section 4.1 to Section 4.3, respectively.

3.1 Phase 1: User Region Creation

In this phase, users will be clustered into different regions according to their locations and historical QoS records. At the beginning, we retrieve users’ approximate locations by their IP addresses. The location information reveals a user’s country, city, latitude/longitude, ISP and domain name. Then users from the same city will be grouped together to form initial regions. These small regions will be aggregated into large ones with a bottom-up hierarchical clustering method [20]. The clustering method has two parts: initialization and aggregation. In the initialization part, we select nonsensitive user regions for aggregation, and compute the similarity between each region pair with Eq. (5). To aggregate regions, 1. Select the most similar region pair (regioni, regionj), merge the two regions to regioni if their similarity exceeds the similarity threshold \(_u\), otherwise stop this region aggregation process. To merge the two regions,

a. Compute the sensitivity and region center of \(_V\) this newly merged region regioni. Remove this region from aggregation process if it becomes a sensitive one.

b. Remove similarities between regionj and other existing regions.

c. Update similarities between regioni and other existing regions.

3.2. Repeat the above step.

Threshold \(_u\) is a tunable parameter that can be adjusted to trade off accuracy for time and space requirements. \(_u\)‘s impact on prediction accuracy will be addressed in Appendix A.

3.3. Phase 2: Service Region Creation

Normally, each user only uses a limited amount of Web services. Compared with the large number of services on the Internet, the number of services with user submitted QoS records is relatively small. Thus, it is difficult to find similar users, and predicting missing QoS values only from user’s perspective is not enough. Clustering Web services can help LoRec find potential similar services. Different from retrieving user location from an IP address, LoRec directly clusters Web services based on their QoS similarity. This is because some companies regard the physical location of data center as a secret and use IP address to hide the real locations. Take Google for example. It has data centers located in Asia, Europe, America, etc. but physical locations retrieved from Google’s IP addresses used in different country-specific versions of Google Search are all listed to Mountain View, California. Another reason is due to the use of the distributed system architecture. To enhance user interaction and to minimize delay, service providers will route user requests to different servers according to user locations or application types. Usually the server that processes requests is different from the one that responds to the users. Thus, retrieving a service location from an IP address does not prove much value. In LoRec, Web services are aggregated with a bottom-up hierarchical clustering algorithm. We use median vector rather than mean vector as the cluster center to minimize the impact of outliers. The similarity between two clusters is defined as the similarity of their centers. Each Web service is regarded as a cluster at the outset. The algorithm aggregates the pairs of the most similar clusters until none of the pairs’ similarities exceeds threshold \(_u\)

IV. TIME COMPLEXITY ANALYSIS

We discuss the worst case time complexity of LoRec recommendation algorithm. We analyze the clustering phase and QoS value prediction phase in Sections 4.5.1 and 4.5.2 respectively. We assume the input is a full matrix with m users and n Web services.
4.1. Time Complexity of Clustering

The time complexity of calculating the median and MAD of each service is $O(m \log m)$. For $n$ services, the time complexity is $O(mn \log m)$. With MAD and median, we identify the region-sensitive services from the service perspective. Since there are at most $m$ records for each service, the time complexity of each service is $O(m)$ using Definition 1. Therefore, the total time complexity of region-sensitive service identification is $O(mn \log m + nm) \frac{1}{4} O(mn \log m)$. In terms of the user region aggregation part, we assume there are $10$ user regions in the beginning. Since there are at most $n$ services used by both regions, the time complexity of the region similarity is $O(n)$ using Eq. (5). We use a matrix to store the similarity between each two regions, and the complexity for computing similarity matrix is $O(10n)$. The aggregation of two user regions will be executed at most $10 \_ 1$ times, in case that all regions are non-sensitive, extremely correlate to each other and finally aggregate into one region. In each iteration, we first compare at most $10 \_ 1$ headers of the priority queues to find the most similar pairs. Since the number of user regions that can be aggregated decreases with each iteration, the real search time will be less than $10 \_ 1$ in the following iterations. For the selected pair of user regions, we calculate the new center and update their similar user regions. Because the number of users involved in the two user regions is uncertain, we use the number of all users as the upper bound and the complexity is $O(mn \log m)$. We employ the priority queue to sort similar user regions, and the insertion and deletion of a similar region is $O(\log m)$. Thus, the time complexity is $O(10n \_ 1 \log m) \frac{1}{4} O(mn \log m)$. As the above steps are linearly combined, the total time complexity of user clustering is $O(mn \log m)$. In the phase of service region creation, there are $n$ services at the beginning. The aggregation of two service regions will be executed at most $n \_ 1$ times, in case that all services are merged into one cluster. In each iteration, we first compare at most $n \_ 1$ heads of the priority queues to find the most similar pairs. Since the number of clusters that can be aggregated decreases with each iteration, the real search time will be less than $n \_ 1$ in the following iterations. For the selected pair, we calculate the new center and update their similar clusters. Because the number of services involved in two clusters is uncertain, we use the number of all services as the upper bound and the complexity is $O(mn \log n)$. The insertion and deletion of a similar region is $O(\log n)$, since we employ the priority queue to sort similar regions. Thus, the time complexity is $O(n \_ 1 \log n) \frac{1}{4} O(mn \log n \log n)$. As the above steps are linearly combined, the total time complexity of service clustering is $O(mn \log m)$. In the phase of service region creation, there are $n$ services at the beginning. The aggregation of two service regions will be executed at most $n \_ 1$ times, in case that all services are merged into one cluster. In each iteration, we first compare at most $n \_ 1$ heads of the priority queues to find the most similar pairs. Since the number of clusters that can be aggregated decreases with each iteration, the real search time will be less than $n \_ 1$ in the following iterations. For the selected pair, we calculate the new center and update their similar clusters. Because the number of services involved in two clusters is uncertain, we use the number of all services as the upper bound and the complexity is $O(mn \log n)$. The insertion and deletion of a similar region is $O(\log n)$, since we employ the priority queue to sort similar regions. Thus, the time complexity is $O(n \_ 1 \log n) \frac{1}{4} O(mn \log n \log n)$.

4.2 DISADVANTAGES

However, three unsolved problems of the previous work affect the performance of current service recommender systems:

1. The first problem is that the existing approaches fail to recognize the QoS variation with users’ physical locations.
2. The second problem is the online time complexity of memory based CF recommender systems. The increasing number of web services and users will pose a great challenge to current systems.
3. The last problem is that current web service recommender systems are all black boxes, providing a list of ranked web services with no transparency into the reasoning behind the recommendation results. It is less likely for users to trust a recommendation when they have no knowledge of the underlying rationale.

4.3 ADVANTAGES

The main advantages of our proposed system: First, we combine the model based and memory based CF algorithms for web service recommendation, which...
significantly improves the recommendation accuracy and time complexity compared with previous service recommendation algorithms. Second, we design a visually rich interface to browse the recommended web services, which enables a better understanding of the service performance.

V. CONCLUSION

This paper presents a QoS-aware Web service recommendation approach. The basic idea is to predict Web service QoS values and recommend the best one for active users based on historical Web service QoS records. We combine prediction results generated from service regions and user regions, which achieves better results than existing approaches. We also find that the combination result is much better than the result from any single method, either the prediction generated from user regions or the one generated from Web service regions. This is because these two methods analyze the problem from different aspects and the combination of them counteracts the error of individual methods.

REFERENCES