Privacy-preserving Distributed Service Recommendation based on Locality-Sensitive Hashing

Lianyong Qi
School of Information Science and Engineering
Qufu Normal University
Email: lianyongqi@gmail.com

Haolong Xiang
State Key Laboratory for Novel Software Technology
Nanjing University
Email: hlx6700@gmail.com

Wanchun Dou
State Key Laboratory for Novel Software Technology
Nanjing University
Email: douwc@nju.edu.cn

Md Zakirul Alam Bhuiyan
Fordham University
Email: mbhuiyan3@fordham.edu

Yongrui Qin
University of Huddersfield
Email: y.qin2@hud.ac.uk

Xuyun Zhang
Department of Electrical & Computer Engineering
University of Auckland
Email: xuyun.zhang@auckland.ac.nz

***Abstract*—With the advent of IoT (Internet of Things) age, considerable web services are emerging rapidly in service communities, which places a heavy burden on the target users' service selection decisions. In this situation, various techniques, e.g., collaborative filtering (i.e., CF) is introduced in service recommendation to alleviate the service selection burden. However, traditional CF-based service recommendation approaches often assume that the historical user-service quality data is centralized, while neglect the distributed recommendation situation. Generally, distributed service recommendation involves inevitable message communication among different parties and hence, brings challenging efficiency and privacy concerns. In view of this challenge, a novel privacy-preserving distributed service recommendation approach based on Locality-Sensitive Hashing (LSH), i.e., *$DistSR_{LSH}$* is put forward in this paper. Through LSH, *$DistSR_{LSH}$* can achieve a good tradeoff among service recommendation accuracy, privacy-preservation and efficiency in distributed environment. Finally, through a set of experiments deployed on *WS-DREAM* dataset, we validate the feasibility of our proposal in handling distributed service recommendation problems.**

Keywords-distributed service recommendation; privacy; efficiency; locality-sensitive hashing; collaborative filtering

I. INTRODUCTION

With the advent of IoT (Internet of Things) age, a great number of web services are emerging rapidly in service communities [1]. The ever-increasing services available on the web, on one hand, provide abundant alternatives for target users' various service requirements, on the other hand, place a heavy burden on the target users' service selection decisions especially when many service candidates share same or similar functionalities [2-3].

In order to alleviate the service selection burden of target users, various service recommendation techniques are brought forth in the last decade, e.g., widely adopted collaborative filtering (i.e., CF)-based recommendation [4-7]. Through considering the historical user-service quality data, traditional CF-based recommendation approaches (including memory-based CF, model-based CF and hybrid CF) can predict target users' personalized preferences and further make accurate service recommendation.

However, existing CF-based service recommendation approaches often assume that the historical user-service quality data is centralized. Thus, the overall quality data generated from historical user-service invocations could be regarded as known already for subsequent service recommendation. While actually, the historical user-service quality data is sometimes not centralized, but distributed. For example, user *A* invoked web service *WS* from Amazon, while user *B* invoked *WS* from IBM.

In this distributed situation, two major challenges are raised. First, due to the privacy concern, neither Amazon nor IBM is willing to reveal the inner user-service quality data to each other, which makes it a difficult task to calculate the user similarity between *A* and *B* so as to make further recommendation. Second, due to the inevitable message communication between two distributed parties, i.e., Amazon and IBM, the service recommendation process if often time-consuming and cannot satisfy the target users' quick response requirements.

In view of the above two challenges, we introduce the Locality-Sensitive Hashing (LSH) technique in service recommendation, and further put forward a novel privacy-preserving distributed service recommendation approach based on LSH, i.e., *$DistSR_{LSH}$*. With the unique nature of LSH, *$DistSR_{LSH}$* can achieve a good tradeoff among service recommendation accuracy, privacy-preservation and efficiency in distributed environment.

Generally, the contributions of our paper are three-fold.

(1) To the best of our knowledge, few existing works considered the service recommendation problem in distributed environment. We recognize the substantial significance of distributed service recommendation and specify the problem formally.

(2) We employ Locality-Sensitive Hashing technique to aid the distributed service recommendation, so as to achieve a good tradeoff among recommendation accuracy, privacy-preservation and efficiency.

(3) A wide range of experiments are deployed on a real web service quality dataset *WS-DREAM* to validate the feasibility of our proposal. Experiment results indicate that *DistSR$_{LSH}$* achieves near-to-optimal recommendation accuracy but substantial improvements in privacy-preservation and efficiency.

The remainder of this paper is organized as follows. Related works are introduced In Section 2. In Section 3, we motivate our paper and in Section 4, we formalize the distributed service recommendation problem. In Section 5, Locality-Sensitive Hashing technique is introduced briefly, and afterwards, a novel approach named *DistSR$_{LSH}$* is put forward to deal with the privacy-preserving distributed service recommendation problem. A set of experiments are deployed in Section 6 to validate the feasibility of our proposal. And finally in Section 7, we summarize the paper and point out the future research directions.

II. RELATED WORKS

Existing research works associated with service recommendation could be generally divided into the following two major popular categories: content-based recommendation approach and CF-based recommendation approach.

A. *Content-based Service Recommendation*

As an old but effective service recommendation manner, content-based recommendation approaches first analyze the similarity between different services, and then recommend the services that are similar (in terms of WSDL or tag description) to the target services (i.e., services invoked by a target user) to the target user. In [1], the authors study enhanced syntactical matching of web service descriptions and make further service recommendation. In [2], semantic aspect of web service description is discussed. In order to introduce service semantic into service recommendation applications, in [3], web service domain ontology is constructed by analyzing web service descriptions (e.g., WSDL and free text descriptors). Similarly, in [4], through analyzing the tree structure composed of synonyms and original meaning in semantic dictionaries such as HowNet and WordNet, the authors calculate the semantic distance between two services, and further obtain their semantic similarity for content-based service recommendation. In order to avoid the possible fake description of service tag, in [5], the authors leverage mashup descriptions and structures to discover important word features of services and bridge the vocabulary gap between mashup developers and service providers.

However, the above content-based recommendation methods often suffer from the over-specification problem [6]. Besides, it becomes a challenging task to automatically abstract the representative feature tags of web services, which blocks the automatic service recommendation heavily.

B. *CF-based Service Recommendation*

Different from the content-based recommendation, CF-based recommendation works based on the past user-service invocation records. Generally, two categories are available: memory-based CF and model-based CF.

(1) *memory-based CF*

A comprehensive service recommendation approach *WSRec* is put forward in [7], which combines user-based and item-based CF together. As service quality heavily depends on service invocation time, in order to make accurate service recommendation, a time-aware recommendation approach is brought forth in [8]. Similarly, location-aware service recommendation is performed in both [9] and [10], as geographically close users often experience similar service quality when they invoke an identical web service. Besides, different users hold distinct preferences, which also play an important role in service recommendation. Therefore, to make personalized recommendation, work [11] improves CF-based recommendation approach by integrating user preferences. Generally, the above memory-based CF recommendation approaches are easy-to-explain and effective when there is a great deal of available historical user-service quality data. However, memory-based CF approaches suffer from the scalability problem heavily, which means that the recommendation efficiency is often low when historical quality data is updated frequently. Beside, only few works (e.g., [12]) consider the privacy concern in recommendation. Third, the above approaches all assume that the historical user-service quality data is centralized, while neglect the distributed situations.

(2) *model-based CF*

Model-based CF approaches utilize the historical user-service quality data to build a recommendation model offline, and then make online recommendation based on the derived model. There are some classic model-based CF recommendation approaches, e.g., Matrix Factorization -based approaches [13], LDA-based approaches [14] and clustering-based approaches [15]. Generally, the above model-based CF approaches are efficient as the recommendation model could be trained offline. However, few works consider the privacy protection problem in recommendation process. Besides, similar to the memory-based CF approaches, the above model-based CF approaches are inappropriate to handle the distributed service recommendation.

With the above analyses, we can conclude that existing research works fall short in handling the distributed and privacy-preserving service recommendation problems. In view of the above shortcoming, a novel privacy-preserving distributed service recommendation approach named *DistSR$_{LSH}$* is put forward in this paper, which will be specified in more detail in the next section.

III. Motivation

Here, we utilize the example in Fig.1 to motivate our paper. Concretely, target user u_1 invokes web services $\{ws_1, \ldots, ws_{n1}\}$ from Amazon, user u_2 invokes web services $\{ws_1, \ldots, ws_{n2}\}$ from Microsoft, and user u_3 invokes web services $\{ws_1, \ldots, ws_{n3}\}$ from IBM. Then according to CF-based recommendation approach (e.g., user-based CF), the first step is to calculate the user similarity $sim(u_1, u_2)$ and $sim(u_1, u_3)$. However, the above similarity calculation process faces two major challenges:

(1) As the historical user-service quality data is distributed on different platforms, Microsoft and IBM are often not willing to open their observed quality data to Amazon (here, target user u_1 is on Amazon platform), due to the privacy concern.

(2) When the number of users or the number of services is large, the similarity calculation process may take a huge amount of time, as message communication is inevitable among Amazon, Microsoft and IBM; this means that the recommendation efficiency is often low and cannot satisfy the target users' quick response requirements.

In view of the above two challenges, an efficient and privacy-preserving distributed recommendation approach named $DistSR_{LSH}$ is introduced in the next section.

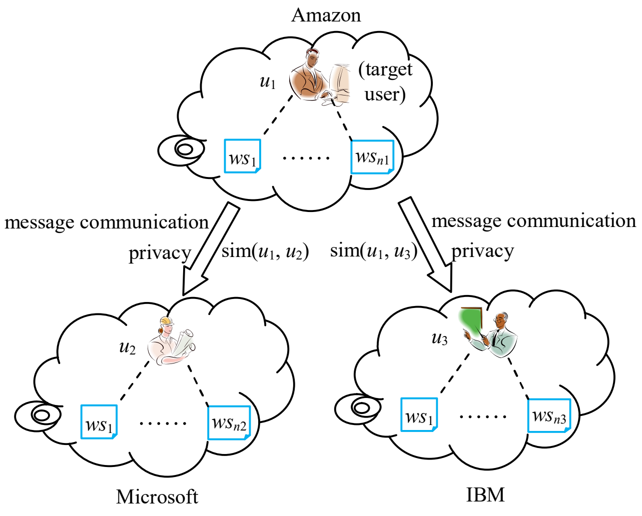


Figure 1. Distributed service recommendation: an example

IV. Problem Formalization

In this paper, we focus on the distributed service recommendation. To facilitate the following discussions, we first formalize the distributed service recommendation problem as a four-tuple $DistSR(PF, U, WS, q)$, where

(1) $PF = \{pf_1, \ldots, pf_z\}$: pf_k ($1 \le k \le z$) denotes k-th distributed service platform; e.g., $z = 3$ holds in Fig.1.

(2) $U = \{U_1, \ldots, U_z\}$: U_k ($1 \le k \le z$) denotes the user set corresponding to distributed platform pf_k. Concretely, $U_k = \{u_{k\text{-}1}, \ldots, u_{k\text{-}m}\}$: $u_{k\text{-}i}$ ($1 \le i \le m$) denotes i-th user on platform pf_k .

(3) $WS = \{ ws_1, \ldots, ws_n\}$: ws_j ($1 \le j \le n$) denotes the j-th web service. Here, to ease the following discussions, we assume that the services on every distributed platform $pf_1, \ldots, pf_z$ are the same. For example, $n_1 = n_2 = n_3$ holds in Fig.1. Note that if a user did not invoke a service, then the corresponding user-service quality data is null.

(4) q is a user-concerned quality dimension of web services, e.g., *response time*. For simplicity, we only consider a quality dimension in the following discussions.

V. A Privacy-preserving Distributed Service Recommendation Approach: $DistSR_{LSH}$

In this section, we introduce the details of our proposed distributed service recommendation approach $DistSR_{LSH}$. Concretely, we introduce the Locality-Sensitive Hashing technique briefly in subsection 5.A; afterwards, in subsection 5.B, we introduce the concrete steps of LSH-based recommendation approach $DistSR_{LSH}$.

A. Locality-Sensitive Hashing

Locality-Sensitive Hashing, i.e., LSH was put forward by Alex Andoni in 1999 [16] and has been proven to be an effective technique to deal with many distributed applications, e.g., distributed information retrieval. Next, we introduce the unique property of LSH.

The main idea of LSH is: select a specific hashing function (or a hashing function family) so that (1) for two neighboring data points in original data space, they are still neighbors after hashing with a large probability (2) for two non-neighboring data points in original data space, they are still non-neighboring after hashing with a large probability.

A hashing function that satisfies the above two conditions are called a LSH function. More formally, a hashing function $h(\ )$ is a LSH function iff the following conditions (1) and (2) hold, where x and y are two data points in original data space, $d\ (x, y)$ denotes the distance between x and y, $h(x)$ is the hashing value of x after projection based on hashing function $h(\ )$, $P(X)$ represents the probability that condition X holds, $\{d_1, d_2, p_1, p_2\}$ are a set of thresholds. If condition (1) and (2) hold simultaneously, then hashing function $h(x)$ is a qualified LSH function and called (d_1, d_2, p_1, p_2)-sensitive.

$$\text{If } d\ (x, y) \le d_1, \text{ then } P(h(x) = h(y)) \ge p_1 \qquad \textbf{(1)}$$

$$\text{If } d\ (x, y) \ge d_2, \text{ then } P(h(x) = h(y)) \le p_2 \qquad \textbf{(2)}$$

Then through a LSH function $h(\ )$ (or a LSH function family, see Fig.2), all the L data points $\{x_1, \ldots, x_L\}$ in original data space could be projected into a set of buckets $b_1, \ldots, b_t$, where each bucket b_i ($1 \le i \le t$) only contains l_i ($l_i << L$) neighboring data points. Thus, if a target user wants to find the similar neighbors of original input X, we can calculate $h(X)$ and further find the bucket (assume b_i) corresponding to $h(X)$. Then according to the unique property of LSH, all the l_i data points in bucket b_i are similar neighbors of X with a large probability. Thus, the searching space is reduced from L to l_i; as $l_i << L$ holds, the searching efficiency is improved significantly. Besides, through hashing projection, the privacy information of data points in original data space is transparent to the target users; for example, the target user in Fig.2 only know the hashing value $h(x)$, but does not know the original value x. In this way, the data privacy is protected. This is the reason why LSH could be recruited for efficient and privacy-preserving distributed business applications. To facilitate the understanding of readers, the symbols recruited in this paper are specified in Table 1.

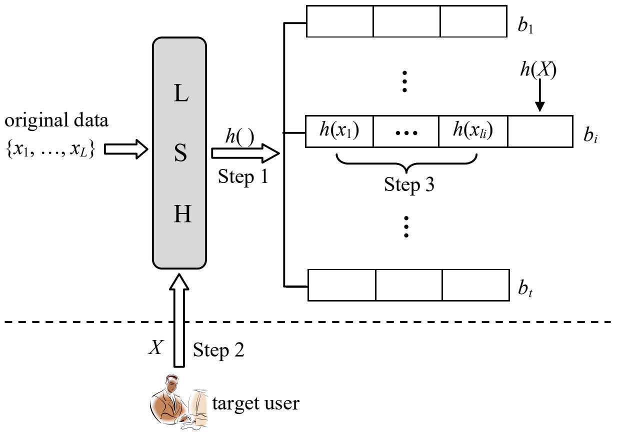


Figure 2. LSH-based service recommendation process

B. *DistSR_{LSH}*: *LSH-based Distributed Service Recommendation*

Generally, our proposed distributed web service recommendation approach *DistSR*$_{LSH}$ (essentially a kind of improved user-based CF) consists of three steps (see Fig.2), each of which is generalized in Fig.3.

Step1: building user index offline. Select a LSH function family to project the users on distributed platforms into corresponding buckets (the bucket No. is the user index) offline, based on the historical user-service quality data.

Step2: online neighbor finding. According to the selected LSH function family, target user is projected into a bucket; then all the users in the bucket are considered as the target user's similar neighbors (with a large probability).

Step3: Top-K service recommendation. According to the target user's neighbors derived in Step 2, predict target user's quality over never-invoked services, and return the Top-K services.

Figure 3. Three steps of distributed service recommendation approach *DistSR*$_{LSH}$

Table 1. Symbol specifications

symbol	specification
z	number of distributed service platforms
m	number of users in each service platform
n	number of web services (we assume services in different platforms are the same)
q	a quality dimension of web services
$d\,(\,,)$	distance between two points
$P(\,)$	probability
d_1, d_2, p_1, p_2	threshold
$h(\,)$	a LSH function
$H(\,)$	LSH function family
L	number of data points (or users); $L = z*m$ holds here
$b_1, \ldots, b_t$	buckets in a hashing table
l_i	number of data points (or users) in bucket b_i
X	input or profile of a target user
T	number of LSH tables
r	number of LSH functions in each LSH table

Step 1: building user index offline.

In this step, we select a LSH function $h(u)$ or a LSH function family $H(u) = \{ h_1(u), \ldots, h_r(u)\}$ to build index for all the users u distributed on different platforms. The selection of LSH functions depends on the adopted "distance" (see condition (1)-(2) in subsection 5.A) formula. As Pearson Correlation Coefficient (PCC) [17] is widely adopted as the similarity measurement or distance measurement in service recommendation, we can adopt the LSH functions corresponding to PCC for index building here.

Concretely, for a user u, we can specify her/his historical quality information over n web services $ws_1, \ldots, ws_n$ with an n-dimensional vector $\vec{u} = (ws_1.q, \ldots, ws_n.q)$, where q is a target user-concerned quality dimension of web services and $ws_j.q = 0$ if user u did not invoke service ws_j before. Then according to [18], for the above n-dimensional vector $\vec{u}$, its LSH function $h(\vec{u})$ is represented by (3). Here, $\vec{v}$ is an n-dimensional vector $(v_1, \ldots, v_n)$ where v_j $(1 \leq j \leq n)$ is a random value in range [-1, 1]; symbol "$\circ$" denotes the dot product between two vectors. To ease the readers' understanding, we give an intuitive understanding of LSH as follows: take vector $\vec{v}$ as a hyper plane, if two vectors $\vec{u_1}$ and $\vec{u_2}$ are located on the same side of hyper plane $\vec{v}$ (i.e., both $\vec{u_1} \circ \vec{v} > 0$ and $\vec{u_2} \circ \vec{v} > 0$ hold, or, both $\vec{u_1} \circ \vec{v} \leq 0$ and $\vec{u_2} \circ \vec{v} \leq 0$ hold), then $\vec{u_1}$ and $\vec{u_2}$ are similar to some extent.

$$h(\vec{u}) = \begin{cases} 1 & \text{if } \vec{u} \circ \vec{v} > 0 \\ 0 & \text{if } \vec{u} \circ \vec{v} \leq 0 \end{cases} \qquad \textbf{(3)}$$

Thus through LSH function in (3), user u is hashed into a binary value of 0 or 1. As indicated in (1) and (2), LSH is essentially a probability-based approach; therefore, the more hashing functions or hashing tables are recruited, the more accurate similarity is obtained. So in order to make accurate service recommendation, multiple LSH functions or tables are necessary. Concretely, assume there are T LSH tables, each of which consists of r LSH functions. Then for each LSH table, an r-dimensional vector (i.e., user index in the table) $H(\vec{u}) = (h_1(\vec{u}), \ldots, h_r(\vec{u}))$ is obtained for user u after LSH, and each element in $H(\vec{u})$ is equal to 0 or 1. Furthermore, two users u_1 and u_2 are hashed into an identical bucket after LSH, iff condition in (4) holds. Namely, if there is at least one LSH table (totally T tables) where the indexes of u_1 and u_2 are equal, then u_1 and u_2 could be regarded as similar neighbors. In this way, we can build index for any user distributed on different platforms, in an offline manner.

$$\exists x, \text{ satisfy } H_x(\vec{u_1}) = H_x(\vec{u_2}) \ (x \in \{1, \ldots, T\}) \qquad \textbf{(4)}$$

Step 2: online neighbor finding.

In Step 1, we have built index for each user offline based on LSH. Next, for a target user u_{target}, we can find his/her approximate neighbors online, whose major process is as follows: first, calculate index for u_{target} based on the adopted LSH function family; second, find the bucket corresponding to the index for u_{target} ; third, all the users in the bucket are regarded as similar neighbors of u_{target} with a large probability.

Step 3: Top-K service recommendation.

Next, we utilize the similar neighbors (derived in Step 2) of target user u_{target} to make service recommendation. Concretely, for each service ws never invoked by u_{target} , we predict its quality q by u_{target}, i.e., $ws.q_{target}$ based on (5), where set NB denotes u_{target}'s neighbors derived in Step 2, $ws.q_i$ denotes ws' quality over q observed by u_i. Finally, we rank all the services (never invoked by u_{target}) by the predicted quality in (5) and select the Top-K services as the final recommendation results.

$$ws.q_{target} = \frac{1}{|NB|} * \sum_{u_i \in NB} ws.\, q_i \qquad \textbf{(5)}$$

Through the above three steps, we can finish the distributed service recommendation process and finally recommend K services to the target user. More formally, our proposal could be specified by pseudo code as below.

Algorithm: *DistSR*$_{LSH}$

Inputs: $PF = \{pf_1, \ldots, pf_z\}$: distributed platforms
$U_I = \{u_{I\text{-}1}, \ldots, u_{I\text{-}m}\}$: user set for platform pf_I
$WS = \{ ws_1, \ldots, ws_n\}$: web service set
q: a quality dimension of web services
u_{target}: target user requesting recommended services

Output: *Rec_Ser_Set*: service set recommended to u_{target}

```
/* Step 1: building user index offline*/
1 for $i$ = 1 to $T$  do  // $T$ LSH tables
2    for $I$ = 1 to $z$  do
3       for $J$ = 1 to $m$  do
4          $H_i(u_{I\text{-}J}) = (h_{i\text{-}1}(\overrightarrow{u_{I-J}}), \ldots, h_{i\text{-}r}(\overrightarrow{u_{I-J}}))$
5          for $j$ = 1 to $r$  do   // $r$ LSH functions
6             for $k$ = 1 to $n$  do //$n$-dimensional hashing vector
7             $h_{ijk}$ = random [-1, 1]
8             end for
9             if $\overrightarrow{u_{I-J}} \circ \overrightarrow{h_{ij}} > 0$  // dot product
10               then $h_{i\text{-}j}(\overrightarrow{u_{I-J}}) = 1$
11               else $h_{i\text{-}j}(\overrightarrow{u_{I-J}}) = 0$
12            end if
13         end for
14      end for
15 end for
/*  Step 2: online neighbor finding */
16 set $NB = \varnothing$  // neighbor set of $u_{target}$
17  for $i$ = 1 to $T$  do
18     $H_i(u_{target}) = (h_{i\text{-}1}(\overrightarrow{u_{target}}), \ldots, h_{i\text{-}r}(\overrightarrow{u_{target}}))$
19     for $j$ = 1 to $r$  do
20        if $\overrightarrow{u_{target}} \circ \overrightarrow{h_{ij}} > 0$
21        then $h_{i\text{-}j}(\overrightarrow{u_{target}}) = 1$
22        else $h_{i\text{-}j}(\overrightarrow{u_{target}}) = 0$
23        end if
24     end for
25     find bucket with index $H_i(u_{target})$ and put its users in $NB$
26  end for
/*  Step 3: Top-K service recommendation */
27 for $j$ = 1 to $n$  do
28    if $ws_j.q_{target} = 0$  // $u_{target}$ never invoked $ws_j$ before
29       then count = 0
30            for $i$ = 1 to $|NB|$  do  // all neighbors of $u_{target}$
31               if $ws_j.q_i \neq 0$  // $i$-th neighbor invoked $ws_j$
32               then count ++
33                     $ws_j.q_{target} = ws_j.q_{target} + ws_j.q_i$
34               end if
35            end for
36            $ws_j.q_{target} = ws_j.q_{target}$ / count
37    end if
38 end for
39 put Top-K services with highest $q$ into set Rec_Ser_Set
40 return Rec_Ser_Set to $u_{target}$
```

VI. Experiment

In this section, a set of experiments are conducted to validate the feasibility of our proposed *DistSR*$_{LSH}$ approach, when dealing with the privacy-preserving distributed web service recommendation problems. The experiments are based on a real web service quality dataset *WS-DREAM* [19] which describes real-world QoS evaluation results from 339 users on 5825 Web services. We randomly divide the 339 users into 10 parts so as to simulate the distributed service recommendation scenario (in this paper, we mainly focus on the service recommendation efficiency and privacy concerns, so the user division manner does not affect the final experiment results). Besides, we consider the user-service response-time dimension, and 90% of the existing response-time data in *WS-DREAM* is removed so that we can predict the missing response-time data and make further recommendation. In each recommendation process, only Top-3 services are generated.

In order to validate the feasibility of our proposal in terms of distributed recommendation efficiency, accuracy and privacy-preservation, we test the following three criteria, respectively:

(1) *time cost*: consumed time for generating service recommendation results.

(2) *MAE* (Mean Absolute Error): average difference between predicted quality and real quality of recommended services.

(3) *privacy-preservation*: confusion degree between real user-service quality and its index after LSH hashing.

Besides, we compare our proposed $DistSR_{LSH}$ approach with four related approaches: *UPCC* [20], *IPCC* [21], *TLACF* [9] (clustering-based approach, parameter *flag* = 0 and *d* = 0) and *WSWalker* [22] (random walk-based approach, parameter ε = 0.0001, *max-depth* = 6). The experiments were conducted on a Dell laptop with 2.80 GHz processors and 2.0 GB RAM. The machine is running under Windows XP and JAVA 1.5. Each experiment was carried out 10 times and the average experiment results were adopted finally.

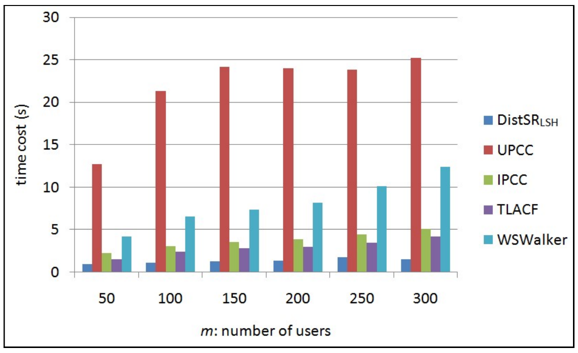
A. Experiment Results and Analyses

Concretely, the following five profiles are tested and compared in our experiments. Here, as Table 1 indicates, *L* and *n* denote the number of users and number of web services, respectively, *T* and *r* denote the number of LSH tables and number of hashing functions in each LSH table, respectively.

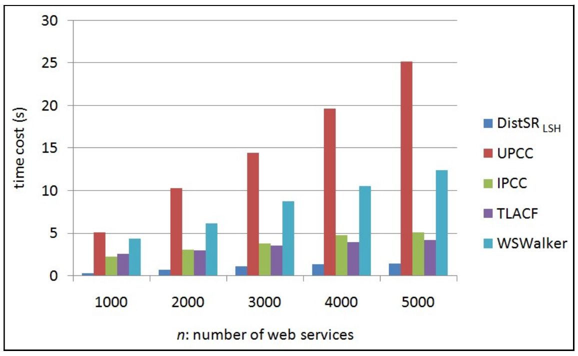
***Profile*1: efficiency comparison among five approaches**

In this profile, we test the recommendation efficiency of five approaches. The recruited parameters are set as below: *L* is varied from 50 to 300, *n* is varied from 1000 to 5000, *T* = *r* = 10 holds. The concrete experiment results are demonstrated in Fig.4.

In Fig.4(a), *n* = 5000 holds. The experiment results indicate that the time costs of five approaches all increase with the growth of *m*, while our proposed $DistSR_{LSH}$ approach outperforms the other four ones as most work (i.e., user index building) in $DistSR_{LSH}$ could be done offline, and the candidate space for similar neighbors of a target user is reduced significantly with the help of unique nature of LSH. Thus, the recommendation efficiency is improved significantly. Similar results could be observed from Fig.4(b), which is not explained repeatedly.

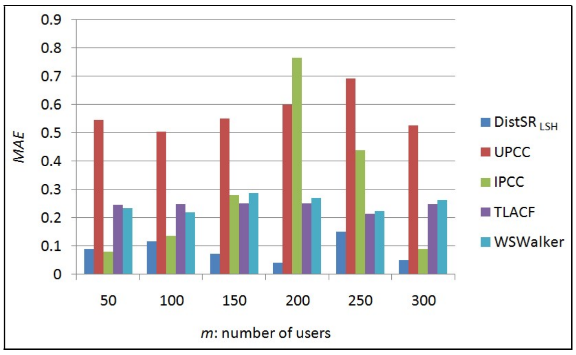


(a) *n* = 5000

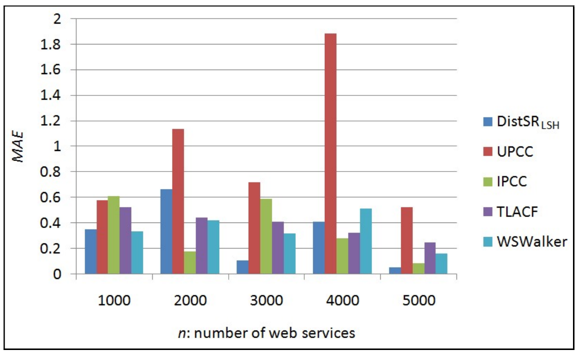


(b) *m* = 300

Figure 4. Recommendation efficiency comparison



(a) *n* = 5000



(b) *m* = 300

Figure 5. Recommendation accuracy comparison

***Profile*2: accuracy comparison among five approaches**

In this profile, we test the accuracy (i.e., *MAE*, the smaller the better) of five recommendation approaches. The experiment parameters are set as follows: *L* is varied from 50 to 300, *n* is varied from 1000 to 5000, and $T = r = 10$ holds. The experiment results are shown in Fig.5.

Concretely, in Fig.5(a), $n = 5000$ holds; and the experiment results demonstrate that the recommendation accuracies of five approaches do not change significantly with the growth of *m* (except for *IPCC* approach), as only Top-3 services are recommended to the target user in each recommendation approach; besides, our proposed $DistSR_{LSH}$ approach outperforms the rest four ones in terms of recommendation accuracy, which is due to the inherent nature of our adopted LSH technique (i.e., two neighboring users are projected into the same bucket after LSH hashing with a large probability).

In Fig.5(b), $m = 300$ holds; and the experiment results show that the recommendation accuracy of our $DistSR_{LSH}$ approach sometimes outperforms the other four ones (e.g., when $n = 3000$ or 5000); while when $n = 1000$, 2000 or 4000, $DistSR_{LSH}$ does not perform very well in terms of recommendation accuracy, which is due to the fact that our adopted LSH technique is actually a probability-based technique and hence, cannot always guarantee to achieve an optimal service recommendation result. However, as Fig.5(b) shows, the accuracy of $DistSR_{LSH}$ approach is still acceptable in most cases.

Profile*3: privacy-preservation of $DistSR_{LSH}$ w.r.t. *T and r

The four related recommendation approaches, i.e., *UPCC*, *IPCC*, *TLACF* and *WSWalker* do not consider the privacy protection; therefore, in this profile, we only test the privacy-preservation effect of our proposed $DistSR_{LSH}$ approach. According to $DistSR_{LSH}$, the original user-service quality data are hashed into different buckets based on the calculated LSH index. Therefore, in this profile, we utilize the bucket density (i.e., the number of derived similar neighbors) to approximately represent the privacy-preservation effect (a larger density often means better privacy-preservation effect). The experiment parameters are set as follows: $L = 300$, $n = 5000$, *T* is varied from 10 to 20, *r* is varied from 2 to 12. The experiment results are shown in Fig.6.

As Fig.6 shows, when the number of hashing functions in each LSH table, i.e., *r* is small, many users are projected into the same bucket as the target user, namely many candidate neighbors are obtained. While with the growth of *r*, the number of neighbors is reduced significantly, as the neighbor "criteria" becomes stricter.

Profile*4: accuracy of $DistSR_{LSH}$ w.r.t. *T and r

In this profile, we test the service recommendation accuracy of $DistSR_{LSH}$ with respect to *T* and *r*. The experiment parameters are set as follows: $L = 300$, $n = 5000$, *T* is varied from 10 to 20, *r* is varied from 2 to 12. The experiment results are shown in Fig.7.

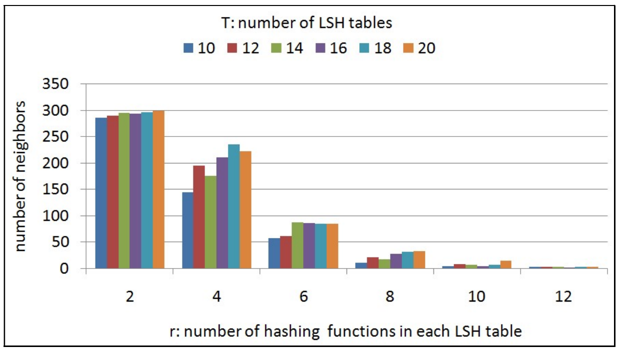


Figure 6. Privacy-preservation w.r.t. T and r ($DistSR_{LSH}$)

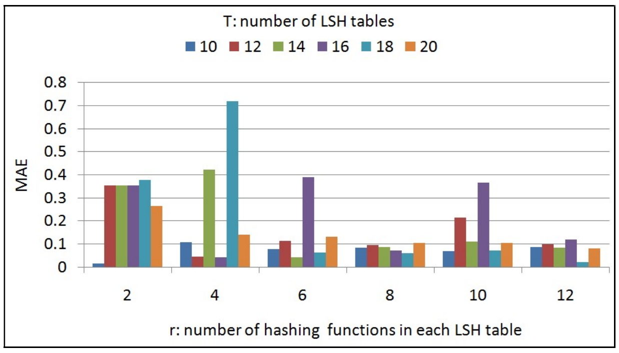


Figure 7. MAE w.r.t. T and r ($DistSR_{LSH}$)

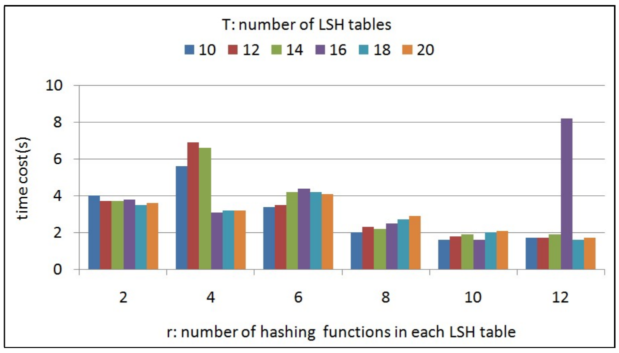


Figure 8. Time cost w.r.t. T and r ($DistSR_{LSH}$)

As Fig.7 shows, the MAE values of $DistSR_{LSH}$ decrease approximately with the growth of *r*; this is because a larger *r* means stricter filtering condition for similar neighbors and hence, those "really similar" neighbors of a target user could be obtained finally.

Profile*5: efficiency of $DistSR_{LSH}$ w.r.t. *T and r

In this profile, we test the efficiency of $DistSR_{LSH}$ with respect to *T* and *r*. Here, $L = 300$, $n = 5000$, *T* is varied from 10 to 20, *r* is varied from 2 to 12. The experiment results are shown in Fig.8. As Fig.8 shows, the time costs of $DistSR_{LSH}$ decrease with the growth of *r*; this is because when *r* grows, the filtering condition becomes stricter and only a few "really similar" neighbors are returned and hence, the recommendation efficiency is improved.

VII. Conclusions

In this paper, we put forward a novel privacy-preserving service recommendation approach based on Locality-Sensitive Hashing, i.e., $DistSR_{LSH}$, to handle the distributed recommendation problem. Through Locality-Sensitive Hashing, candidate space of similar neighbors of a target user could be reduced significantly, so that the recommendation efficiency is improved. Besides, for distributed users, their invoked service quality information is transparent to the target users through hashing technique, which protects user privacy very well. Finally, we validate the feasibility of our proposal through a set of experiments deployed on well-known *WS-DREAM* dataset. The experiment results demonstrate that our proposed $DistSR_{LSH}$ approach can achieve a good tradeoff among service recommendation accuracy, privacy-preservation and efficiency in distributed environment.

In the future, we will investigate the distributed service recommendation problems with multiple quality dimensions. Besides, service quality is not stable, but dynamic; therefore, we will study the dynamic quality-aware distributed service recommendation problems in the future.

Acknowledgment

This paper is partially supported by Natural Science Foundation of China (No. 61402258, No. 61672276), Open Project of State Key Laboratory for Novel Software Technology (No. KFKT2016B22).

References

[1] M. Brian Blake, Iman Saleh, Yi Wei, Ian D. Schlesinger, Alexander Yale-Loehr, and Xuanzhe Liu. Shared Service Recommendations from Requirement Specifications: A Hybrid Syntactic and Semantic Toolkit. *Information and Software Technology*, 57: 392-404, 2015.

[2] Malak Al-Hassan, Haiyan Lu, and Jie Lu. A Semantic Enhanced Hybrid Recommendation Approach: A Case Study of E-Government Tourism Service Recommendation System. *Decision Support Systems*, 72: 97-109, 2015.

[3] Aviv Segev, and Quanzheng Sheng. Bootstrapping Ontologies for Web Services. *IEEE Transactions on Services Computing*, 5(1): 33-44, 2012.

[4] Gaofeng Cao, and Li Kuang. Identifying Core Users based on Trust Relationships and Interest Similarity in Recommender System, *IEEE International Conference on Web Services*, pp. 284-291, 2016.

[5] Yang Zhong, Yushun Fan, Wei Tan, and Jia Zhang. Web Service Recommendation With Reconstructed Profile From Mashup Descriptions. *IEEE Transactions on Automation Science and Engineering*, 2016.

[6] Ibrahim Mashal, Tein-Yaw Chung, and Alsaryrah Osama. Toward Service Recommendation in Internet of Things. *IEEE International Conference on Ubiquitous and Future Networks*, pp. 328-331, 2015.

[7] Zibin Zheng, Ma Hao, R. Lyu Michael, and Irwin King. Qos-aware Web Service Recommendation by Collaborative Filtering. *IEEE Transactions on Services Computing*, 4(2): 140-152, 2011.

[8] Xinyu Wang, Jianke Zhu, Zibin Zheng, Wenjie Song, Yuanhong Shen, and Michael R. Lyu. A Spatial-Temporal QoS Prediction Approach for Time-aware Web Service Recommendation, *ACM Transactions on the Web*, 10(1): 7, 2016.

[9] Chengyuan Yu, and Linpeng Huang. A Web Service QoS Prediction Approach based on Time- and Location-aware Collaborative Filtering. *Service Oriented Computing and Applications*, 10(2): 135-149, 2016.

[10] Mingdong Tang, Yechun Jiang, Jianxun Liu, and Xiaoqing Liu. Location-aware Collaborative Filtering for QoS-based Service Recommendation, *IEEE International Conference on Web Services*, pp. 202-209, 2012.

[11] Kenneth K. Fletcher, and Xiaoqing Frank Liu. A Collaborative Filtering Method for Personalized Preference-based Service Recommendation. *IEEE International Conference on Web Services*, pp. 400-407, 2015.

[12] Jieming Zhu, Pinjia He, Zibin Zheng, and Michael R. Lyu. A Privacy-preserving QoS Prediction Framework for Web Service Recommendation. *IEEE International Conference on Web Services*, pp. 241-248, 2015.

[13] Yao, Lina, Quan Z. Sheng, Yongrui Qin, Xianzhi Wang, Ali Shemshadi, and Qi He. Context-aware Point-of-Interest Recommendation Using Tensor Factorization with Social Regularization, *International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1007-1010, 2015.

[14] Yang Zhong, Yushun Fan, Keman Huang, Wei Tan, and Jia Zhang. Time-aware Service Recommendation for Mashup Creation in An Evolving Service Ecosystem. *IEEE International Conference on Web Services*, pp. 25-32, 2014.

[15] Chen Wu, Weiwei Qiu, Zibin Zheng, Xinyu Wang, and Xiaohu Yang. QoS Prediction of Web Services based on Two-phase K-means Clustering. *IEEE International Conference on Web Services*, pp. 161-168, 2015.

[16] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity Search in High Dimensions via Hashing. *VLDB*, 99(6): 518-529, 1999.

[17] Lee Rodgers, Joseph, and W. Alan Nicewander. Thirteen Ways to Look at the Correlation Coefficient. *The American Statistician*, 42(1): 59-66, 1988.

[18] Data Mining and Query Log Analysis for Scalable Temporal and Continuous Query Answering, *http://www.optique-project.eu/*, 2015.

[19] Z. Zheng, Y. Zhang, and M. R. Lyu. Investigating QoS of Real World Web Services. *IEEE Transactions on Services Computing*, 7(1): 32–39, 2014.

[20] Breese JS, Heckerman D, Kadie C. Empirical Analysis of Predictive Algorithms for Collaborative Filtering, 1998. John S. Breese, David Heckerman, and Carl Kadie. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. *International Conference on Uncertainty in Artificial Intelligence*, pp. 43-52, 1998.

[21] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. *ACM conference on Computer Supported Cooperative Work*, pp. 175-186, 1994.

[22] Mingdong Tang, Xiaoling Dai, Buqing Cao, and Jianxun Liu. WSWalker: A Random Walk Method for QoS-aware Web Service Recommendation. *IEEE International Conference on Web Services*, pp. 591-598, 2015.