Amplified Locality-Sensitive Hashing for Privacy-Preserving Distributed Service Recommendation

Lianyong Qi[1,2](✉), Wanchun Dou[2], Xuyun Zhang[3], and Shui Yu[4]

[1] School of Information Science and Engineering, Qufu Normal University, Rizhao 276826, China
lianyongqi@gmail.com

[2] State Key Laboratory for Novel Software Technology, Department of Computer Science and Technology, Nanjing University, Nanjing 210023, China
douwc@nju.edu.cn

[3] Department of Electrical and Computer Engineering, University of Auckland, Auckland 1023, New Zealand
xuyun.zhang@auckland.ac.nz

[4] School of Information Technology, Deakin University, Melbourne 3125, Australia
syu@deakin.edu.au

Abstract. With the ever-increasing volume of services registered in various web communities, service recommendation techniques, e.g., Collaborative Filtering (i.e., CF) have provided a promising way to alleviate the heavy burden on the service selection decisions of target users. However, traditional CF-based service recommendation approaches often assume that the recommendation bases, i.e., historical service quality data are centralized, without considering the distributed service recommendation scenarios as well as the resulted privacy leakage risks. In view of this shortcoming, Locality-Sensitive Hashing (LSH) technique is recruited in this paper to protect the private information of users when distributed service recommendations are made. Furthermore, LSH is essentially a probability-based search technique and hence may generate "False-positive" or "False-negative" recommended results; therefore, we amplify LSH by AND/OR operations to improve the recommendation accuracy. Finally, through a set of experiments deployed on a real distributed service quality dataset, i.e., *WS-DREAM*, we validate the feasibility of our proposed recommendation approach named $DistSR_{Amplify\text{-}LSH}$ in terms of recommendation accuracy and efficiency while guaranteeing privacy-preservation in the distributed environment.

Keywords: Distributed service recommendation · Collaborative Filtering
Privacy-preservation · Recommendation accuracy
Amplified Locality-Sensitive Hashing

1 Introduction

With the ever-increasing volume of services registered in various web communities (e.g., *Amazon* and *IBM*), it is becoming a nontrivial task to find the web services that a target user is really interested in from massive candidates [1–3]. In this situation, various recommendation techniques, e.g., Collaborative Filtering (i.e., CF) are recruited to make service recommendation so as to reduce the heavy burden on the service selection decisions of target users [4–7].

However, traditional CF-based service recommendation approaches (e.g., user-based CF, item-based CF and hybrid CF) often assume that the service recommendation bases, i.e., historical service quality data are centralized, without considering the distributed service recommendation scenarios when historical service quality data are from multiple independent platforms (e.g., historical service quality data observed by user *A* and user *B* are recorded by *Amazon* and *IBM*, respectively). Generally, two challenges are present in the above distributed service recommendation problems. First, *IBM* is often not willing to share its data with *Amazon* due to privacy concerns, vice versa, which hampers the data collaboration between *Amazon* and *IBM* and consequently renders the distributed recommendation process infeasible. Second, the volume of service quality data recorded by *Amazon* and *IBM* may become increasingly huge with updates over time, which brings additional communication cost between these two platforms; as a consequence, the recommendation efficiency is reduced severely and cannot satisfy the quick response requirements from target users.

In view of these challenges, Locality-Sensitive Hashing (LSH) technique is recruited in this paper to achieve privacy-preserving and efficient service recommendation in the distributed environment. Furthermore, we amplify LSH so as to reduce the "False-positive" and "False-negative" recommended results. Finally, according to the amplified LSH, we proposal a novel distributed service recommendation approach, i.e., $DistSR_{Amplify\text{-}LSH}$. With the inherent nature of amplified LSH, $DistSR_{Amplify\text{-}LSH}$ can achieve a good recommendation performance in terms of recommendation accuracy and efficiency while guaranteeing privacy-preservation.

In summary, the contributions of our paper are three-fold.

(1) We introduce LSH technique into distributed service recommendation, so as to protect the private information of users and improve the recommendation efficiency. Meanwhile, we recognize the possible "False-positive" and "False-negative" recommended results produced by LSH-based recommendation approach.
(2) We amplify LSH by integrating the AND/OR operations, to reduce the "False-positive" and "False-negative" recommended results and improve the recommendation accuracy.
(3) Extensive experiments are conducted on a real distributed service quality dataset *WS-DREAM* to validate the feasibility of our proposal. Experiment results show that $DistSR_{Amplify\text{-}LSH}$ outperforms other state-of-the-art approaches in terms of recommendation accuracy and efficiency while guaranteeing privacy-preservation.

The rest of paper is structured as follows. We motivate our paper in Sect. 2 and formulate the distributed service recommendation problems in Sect. 3. In Sect. 4, LSH technique is introduced briefly and afterwards, an amplified LSH-based recommendation approach, i.e., $DistSR_{Amplify\text{-}LSH}$ is proposed to solve the privacy-preserving distributed service recommendation problems. In Sect. 5, a set of experiments are conducted to validate the feasibility of our proposal. Related work and comparison analyses are presented in Sect. 6. And finally, in Sect. 7, we conclude the whole paper and point out the future research directions.

2 Motivation

An example is presented in Fig. 1 to demonstrate the motivation of our paper. In Fig. 1, there are two platforms, i.e., *Amazon* and *IBM* which record the historical quality data of services $\{ws_1, \ldots, ws_n\}$ observed by target user u_{target} and user u_1, respectively. In this situation, according to the traditional CF recommendation approaches [7], it is necessary to calculate the similarity between u_{target} and u_1 (denoted by $sim(u_{target}, u_1)$). However, as Fig. 1 shows, the calculation of $sim(u_{target}, u_1)$ requires the collaboration between *Amazon* and *IBM* and often faces the following three challenges (see Fig. 1).

(1) Due to privacy concerns, *IBM* is often not willing to share its data with *Amazon*, which hampers the cross-platform collaboration between *Amazon* and *IBM* and consequently renders the calculation of $sim(u_{target}, u_1)$ infeasible.
(2) For both *Amazon* and *IBM*, their volume of recorded service quality data may become increasingly huge with updates over time; in this situation, the collaboration efficiency and scalability between *Amazon* and *IBM* may be decreased significantly and cannot satisfy the quick response requirements of target users.
(3) It is possible to generate the "False-positive" or "False-negative" recommended results; in this situation, user satisfaction is reduced significantly.

In view of these challenges, we amplify LSH and further propose an amplified LSH-based service recommendation approach, i.e., $DistSR_{Amplify\text{-}LSH}$, so as to achieve privacy-preserving and efficient service recommendation in the distributed environment.

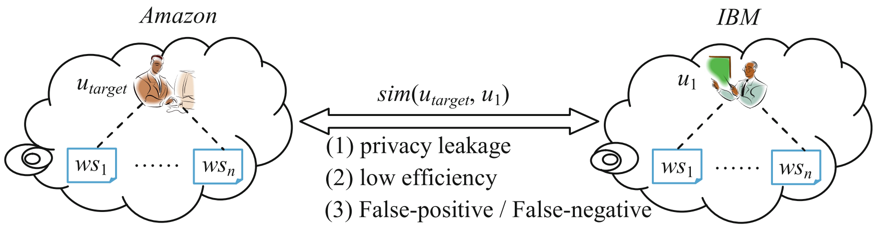


Fig. 1. Distributed service recommendation and its challenges: an example

3 Problem Formulation

Generally, the distributed service recommendation problems involving multiple platforms can be formulated as a five-tuple $Dist\_Ser\_Rec(PF, U, u_{target}, WS, q)$, where

(1) $PF = \{pf_1, \ldots, pf_z\}$: the set of platforms that record historical service quality data observed by users; e.g., $z = 2$ holds in Fig. 1.
(2) $U = \{u_1, \ldots, u_m\}$: the set of users. For each user, his/her observed service quality data are recorded by a platform in set PF.
(3) u_{target}: a target user to whom a recommender system intends to recommend services. Here, $u_{target} \in U$ holds.

(4) $WS = \{ws_1, \ldots, ws_n\}$: the set of candidate web services. For simplicity, we assume that the services held by different platforms $pf_1, \ldots, pf_z$ are the same. Specifically, if user u has never invoked web service ws, then the quality data of ws observed by u is denoted by 0.
(5) q is a quality dimension of web services, e.g., *response time*. For simplicity, only a quality dimension is considered in the following discussions.

With the above formulation, we can specify the privacy-preserving distributed service recommendation problems more formally as below: according to the historical quality data (over dimension q) of services ($\in WS$) observed by users ($\in U$) in multiple platforms ($\in PF$), select an optimal service ($\in WS$, never invoked by u_{target}) and recommend it to u_{target}, during which the private information of users (e.g., *the service quality data observed by a user*, *the service set invoked by a user*) should not be exposed to other users. To achieve this goal, a novel distributed service recommendation approach $DistSR_{Amplify\text{-}LSH}$ is introduced in the next section.

4 A Privacy-Preserving Distributed Service Recommendation Approach: $DistSR_{Amplify\text{-}LSH}$

We firstly introduce the Locality-Sensitive Hashing technique briefly in Subsect. 4.1; afterwards, in Subsect. 4.2, we introduce the details of our proposed amplified LSH-based service recommendation approach $DistSR_{Amplify\text{-}LSH}$.

4.1 Locality-Sensitive Hashing

The main idea behind LSH [8] is: select a specific hash function (or a hash function family) so that (1) for two neighboring points in original data space, they are still neighbors after hash with large probability (2) for two non-neighboring points in original data space, they are still non-neighbors after hash with large probability. More formally, a hash function $h(.)$ is called a LSH function iff conditions (1) and (2) hold, where x and y are two points in original data space, $d(x, y)$ denotes the distance between points x and y, $h(x)$ is the hash value of point x, $P(A)$ represents the probability that event A holds, $\{d_1, d_2, p_1, p_2\}$ are a set of thresholds.

$$\text{If } d(x,y) \leq d_1, \text{then } P(h(x) = h(y)) \geq p_1 \tag{1}$$

$$\text{If } d(x,y) \geq d_2, \text{then } P(h(x) = h(y)) \leq p_2 \tag{2}$$

The rationale of LSH-based neighbor search can be explained intuitively in Fig. 2, where $h(.)$ is a LSH function and $\{A, B, C, D\}$ are four points in original data space. As points A and B are neighbors, they are projected into the same bucket of the LSH table, i.e., $h(A) = h(B)$ with large probability; while points C and D are non-neighbors, therefore, they are projected into different buckets, i.e., $h(C) \neq h(D)$ with large probability. Conversely, if $h(A) = h(B)$ holds, then points A and B are neighbors in original data space with large probability; if $h(C) \neq h(D)$ holds, then points C and D are non-neighbors in original data space with large probability.

Assume X is a new point and we hope to find its similar neighbors in a privacy-preserving way (i.e., without revealing the inner details of point X), then we can first calculate point X's hash value, i.e., $h(X)$. Next, if $h(X) = h(A) = h(B)$ holds, then points A and B are point X's neighbors; while if $h(X) = h(C)$ holds, then point C is the only neighbor of point X. This is the main rationale of LSH-based neighbor search. As the LSH table can be built offline, the neighbor search efficiency can be improved significantly. Besides, for a point (e.g., point A in Fig. 2), only its hash value with little privacy is used in LSH-based neighbor search; as a consequence, the private information of this point can be protected.

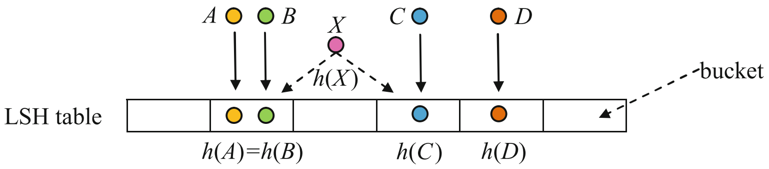


Fig. 2. Rationale of LSH-based neighbor search

4.2 *DistSR*

The amplified LSH-based service recommendation approach, i.e., $DistSR_{Amplify\text{-}LSH}$ is essentially a variant of user-based CF approach, which mainly consists of the four steps in Fig. 3.

Step 1: Build user indexes offline based on LSH. Choose a LSH function to project the users into corresponding buckets offline, based on the users' observed service quality data. Then the bucket No. can be regarded as the indexes for users.

Step 2: Define "neighbor" relationship between users based on amplified LSH. In order to reduce the "False-positive" and "False-negative" search results, we amplify LSH to define a novel "neighbor" relationship between two users based on the user indexes derived in Step 1.

Step 3: Online neighbor finding for u_{target}. According to the "neighbor" relationship defined in Step 2, find the similar neighbors of u_{target}.

Step 4: Service recommendation. According to the historical service quality data observed by neighbors (derived in Step 3) of u_{target}, predict the quality of services never invoked by u_{target}, and return the quality-optimal service to u_{target}.

Fig. 3. Four steps of amplified LSH-based service recommendation approach $DistSR_{Amplify\text{-}LSH}$

Step 1: Build User Indexes Offline Based on LSH

In this step, we choose a LSH function $h(.)$ to build indexes for all the m users in distributed platforms. The selection of LSH function $h(.)$ depends on the adopted "distance" type of d (x, y) (see conditions (1)–(2) in Subsect. 4.1). As Pearson Correlation Coefficient (PCC) [9] is often taken as the similarity measurement or distance type in service recommendation, we adopt the LSH function $h(.)$ corresponding to PCC distance for user indexes building.

Concretely, for a user u, we can model her/his historical quality data over n services, i.e., ws_1, …, ws_n with an n-dimensional vector $\overrightarrow{u} = (ws_1.q, \ldots, ws_n.q)$, where q is a quality dimension of services and $ws_j.q$ denotes service ws_j's quality over dimension q observed by user u. Specifically, $ws_j.q = 0$ if user u has never invoked web service ws_j before. Then for vector $\overrightarrow{u}$, its LSH function $h(\vec{u})$ can be represented by (3) [10]. Here, $\overrightarrow{v}$ 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 understanding of readers, we explain the physical meaning of (3) as follows: take vector $\overrightarrow{v}$ as a hyper plane for space partition; if two vectors $\overrightarrow{u_a}$ and $\overrightarrow{u_b}(1 \leq a, b \leq m$ and $a \neq b)$ are located on the same side of $\overrightarrow{v}$(i.e., both $\overrightarrow{u_a} \circ \overrightarrow{v} > 0$ and $\overrightarrow{u_b} \circ \overrightarrow{v} > 0$ hold, or, both $\overrightarrow{u_a} \circ \overrightarrow{v} \leq 0$ and $\overrightarrow{u_b} \circ \overrightarrow{v} \leq 0$ hold), then $\overrightarrow{u_a}$ and $\overrightarrow{u_b}$ can be regarded as similar (with high probability).

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

Thus through the LSH function $h(.)$ in (3), each user $u(u \in U)$ is converted to a binary value $h(\vec{u})(\in \{0, 1\})$ However, a single LSH function often falls short in profiling the service quality data observed by a user. Therefore, we choose r LSH functions $h_1(.)$, …, $h_r(.)$ to generate an r-dimensional vector $H(\vec{u}) = (h_1(\vec{u}), \ldots, h_r(\vec{u}))$ offline for user u. Then $H(\vec{u})$ can be regarded as the index for user u.

Step 2: Defining "Neighbor" Relationship Between Users Based on Amplified LSH

According to the LSH theory, two users u_a and u_b are similar neighbors iff they fall into the same bucket, i.e., $H(\overrightarrow{u_a}) = H(\overrightarrow{u_b})$, vice versa. However, as formulas (1) and (2) indicate (see Subsect. 4.1), LSH is essentially a probability-based search technique; therefore, it is inevitable to produce unsatisfactory search results. In other words, LSH may generate "False-positive" (i.e., dissimilar users of the target user are regarded as similar) or "False-negative" (i.e., similar neighbors of the target user are regarded as dissimilar) neighbor search results, which reduces the service recommendation accuracy severely.

To overcome this shortcoming, we amplify LSH by adopting the AND/OR operations over multiple LSH functions or LSH tables. Concretely, the following two strategies are taken to amplify LSH.

Strategy-1*: OR operation over r LSH functions so as to reduce the "False-negative" search results.*

In Step 1, r LSH functions $h_1(.)$, …, $h_r(.)$ are recruited to build index for user u, i.e., $H(\vec{u}) = (h_1(\vec{u}), \ldots, h_r(\vec{u}))$. Here, in order to reduce the "False-negative" search results, OR operation is taken over these r LSH functions. More concretely, two users u_a and u_b are regarded as similar iff the condition in (4) holds. Here, we utilize equation $H(\overrightarrow{u_a}) \overset{OR}{=} H(\overrightarrow{u_b})$ to represent the "similarity" relationship defined in condition (4).

$$\exists j, \text{satisfy}\, h_j(\overrightarrow{u_a}) = h_j(\overrightarrow{u_b})(1 \le j \le r) \tag{4}$$

Strategy-2: *AND operation over T LSH tables so as to reduce the "False-positive" search results.*

In order to reduce the "False-positive" search results, we repeat Step 1 T times to generate T hash tables. Next, AND operation is taken over these T LSH tables. More concretely, two users u_a and u_b are regarded as similar neighbors iff the condition in (5) holds. Here, $H_x(.)$ denotes the LSH function family (see *Strategy-1*) recruited in x-th LSH table; we utilize $u_a \overset{sim}{\leftrightarrow} u_b$ to represent the "similarity" relationship defined in condition (5).

$$\forall x \in \{1, \ldots, T\}, \text{satisfy}\, H_x(\overrightarrow{u_a}) \overset{OR}{=} H_x(\overrightarrow{u_b}) \tag{5}$$

Thus through *Strategy-1* and *Strategy-2*, we amplify LSH and define a novel "neighbor" relationship between two users u_a and u_b, i.e., $u_a \overset{sim}{\leftrightarrow} u_b$, so as to reduce the "False-negative" and "False-positive" search results and improve the recommendation accuracy.

Step 3: Online Neighbor Finding for u_{target}

The index for u_{target}, i.e., $H(\overrightarrow{u_{target}})$ can be calculated based on the LSH function family $\{h_1(.), \ldots, h_r(.)\}$ chosen in Step 1. Afterwards, if $u_{target} \overset{sim}{\leftrightarrow} u_a$ holds according to (4) and (5), then u_a can be regarded as a similar neighbor of u_{target} and is put into set *NB_Set*, where *NB_Set* denotes the neighbor set of u_{target}.

Step 4: Service Recommendation

If $NB\_Set = \emptyset$, then no similar neighbors of u_{target} are returned and the subsequent service recommendation fails accordingly. Otherwise, we utilize the similar neighbors in *NB_Set* to make a service recommendation to u_{target}. Concretely, for each service ws ($\in WS$) never invoked by u_{target}, we can predict its quality over dimension q observed by u_{target}, denoted by $ws.q_{target}$, based on the equation in (6). Here, $ws.q_a$ denotes ws' quality over dimension q observed by user u_a; $NB\_Set^{\#}$ represents the set of u_{target}'s neighbors who have invoked service ws before, i.e., $NB\_Set^{\#} = \{ u_a \mid u_a \in NB\_Set, ws.q_a \neq 0\}$. Finally, we select a service with the optimal quality predicted by (6) and recommend it to u_{target}.

$$ws.q_{target} = \frac{1}{|NB\_Set^{\#}|} * \sum_{u_a \in NB\_Set^{\#}} ws.\, q_a \tag{6}$$

Thus through the above four steps of $DistSR_{Amplify\text{-}LSH}$, we can recommend the quality-optimal service to the target user, so as to finish the privacy-preserving distributed service recommendation process. More formally, our proposal can be specified by the following pseudo code.

```
Algorithm: $DistSR_{Amplify\text{-}LSH}(PF, U, WS, u_{target}, q)$

Inputs: (1) $PF = \{pf_1, ..., pf_z\}$: platform set;
        (2) $U = \{u_1, ..., u_m\}$: user set;
        (3) $WS = \{ws_1, ..., ws_n\}$: web service set;
        (4) $user_{target}$: a target user;
        (5) $q$: a quality dimension of web services.

Output: $ws_{optimal}$: a candidate service with optimal predicted quality

/*  Step 1: Build user indexes offline based on LSH */
1  for $j$ = 1 to $r$ do    // $r$ LSH functions in each LSH table
2      for $k$ = 1 to $n$ do // $n$-dimensional vector depicting a user
3          $v_{jk}$ = random [-1, 1]
4      end for
5      for $a$ = 1 to $m$ do    // $m$ users
6          calculate $h_j(\overrightarrow{u_a})$based on (3)
7      end for
8  end for
9  for $a$ = 1 to $m$ do
10     $H(\overrightarrow{u_a}) = (h_1(\overrightarrow{u_a}), \ldots, h_r(\overrightarrow{u_a}))$
11 end for
12 repeat line 1-11 $T$ times to generate $T$ LSH tables offline
/*  Step 2 - Step 3: Define "neighbor" relationship between users based on amplified LSH
and online neighbor finding for $u_{target}$    */
13  count = 0 // number of LSH tables in which the condition in (4) holds
14  for $a$ = 1 to $m$ do
15      for $x$ = 1 to $T$ do    // $T$ LSH tables
16          if $H_x(\overrightarrow{u_a}) \overset{OR}{=} H_x(\overrightarrow{u_{target}})$ holds based on (4)
17          then count + +
18              continue
19          else break
20          end if
21      end for
22      if count = $T$
23      then put $u_a$ into $NB\_Set$
24      end if
25  end for
/*  Step 4: Service recommendation    */
26  for $i$ = 1 to $n$ do      // $n$ candidate web services
27      if $ws_i.q_{target} = 0$    // $u_{target}$ has never invoked $ws_i$ before
28      then COUNT = 0    // size of set $NB\_Set^{\#}$ in (6)
29          for $K$ = 1 to | $NB\_Set$ |    do
```

```
                if $ws_i.q_K \neq 0$
                then COUNT + +
                    $ws_i.q_{target} = ws_i.q_{target} + ws_i.q_K$
                end if
            end for
            $ws_i.q_{target} = ws_i.q_{target}$ / COUNT
        end if
    end for
    $ws_{optimal} = \{ws_i \mid ws_i.q_{target} = \text{OPTIMAL}\{ws_i.q_{target}\}\}$
    return $ws_{optimal}$ to $u_{target}$
```

5 Experiments

5.1 Experiment Dataset and Deployment

In this section, a set of experiments are conducted to validate the feasibility of our proposed service recommendation approach *DistSR*$_{Amplify\text{-}LSH}$. The experiments are based on a real distributed service quality dataset *WS-DREAM* [11] which collects the historical quality data of 5825 web services (from different countries) observed by 339 users. In our experiments, each country that hosts a set of web services is regarded as a distributed platform, so as to simulate the distributed service recommendation scenario. Besides, only a quality dimension of services, i.e., *response time* is considered; furthermore, some entries in the user-service quality matrix are removed randomly to simulate the missing quality data.

To demonstrate the feasibility and advantages of our proposed *DistSR*$_{Amplify\text{-}LSH}$ approach, we compare our proposal with three state-of-the-art approaches: *UPCC* [12], *P-UIPCC* [13] and *PPICF* [14]. Concretely, the following two evaluation measures are examined and compared, respectively (as user privacy can be protected well by the intrinsic nature of LSH technique, we will not evaluate the capability of privacy-preservation of our proposal here).

(1) *time cost*: consumed time for generating the final recommended results, through which we can test the recommendation efficiency.
(2) *MAE* (Mean Absolute Error): average difference between predicted quality and real quality of recommended services, through which we can test the recommendation accuracy (the smaller the better).

The experiments are conducted on a Lenovo computer with 2.40 GHz processors and 12.0 GB RAM. The machine runs under Windows 10, JAVA 8 and MySQL 5.7. Each experiment is carried out 10 times and the average experiment results are reported finally.

5.2 Experiment Results and Analyses

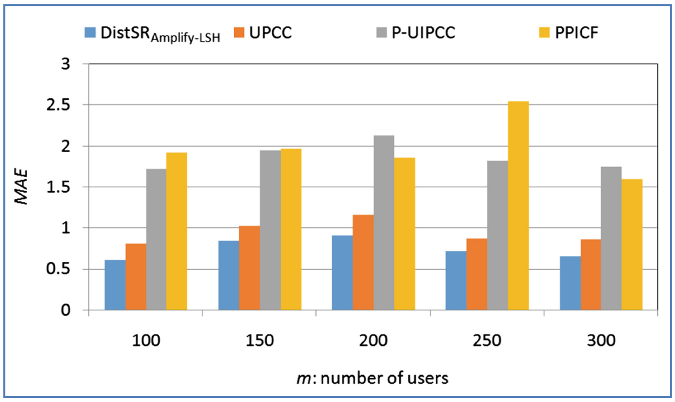
Concretely, the following five profiles are tested and compared in our experiments. Here, *m* and *n* denote the number of users and number of web services, respectively;

T and *r* denote the number of LSH tables and number of hash functions in each LSH table, respectively.

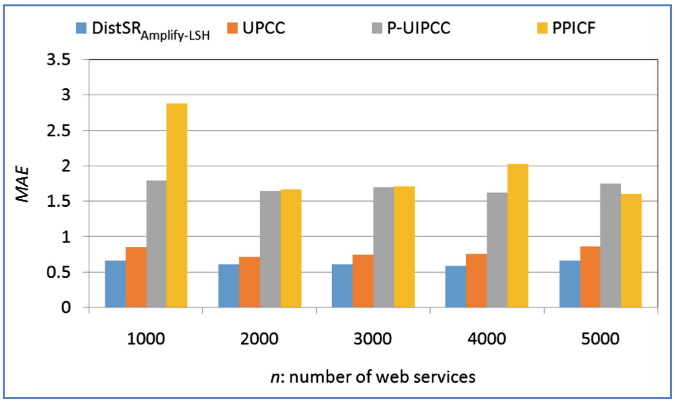
***Profile* 1: Recommendation Accuracy Comparison**

In this profile, we test the recommendation accuracy (i.e., *MAE*, the smaller the better) of four approaches. The experiment parameters are set as follows: *m* is varied from 100 to 300, *n* is varied from 1000 to 5000, and $T = r = 10$ holds. The experiment results are shown in Fig. 4.

In Fig. 4(a), $n = 5000$ holds. The experiment results show that the recommendation accuracy values of *P-UIPCC* and *PPICF* approaches are often low (i.e., *MAE* values are high). This is because several approximation strategies (e.g., data obfuscation, divide-merge) are adopted in these two approaches so as to protect the user privacy, while the approximation strategies reduce the recommendation accuracy significantly. No data approximation strategy is recruited in *UPCC* approach; therefore, the recommendation accuracy of *UPCC* is higher than *P-UIPCC* and *PPICF* approaches. While our proposed $DistSR_{Amplify\text{-}LSH}$ approach outperforms the other three ones in



(a) $n = 5000$



(b) $m = 300$

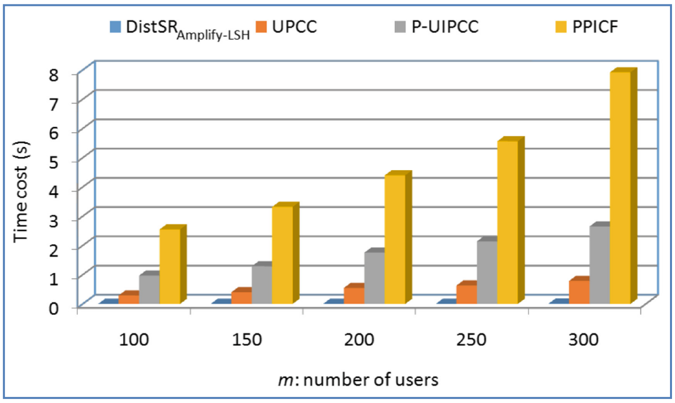
Fig. 4. Recommendation accuracy comparison of four approaches

terms of recommendation accuracy, which is due to the following two reasons. First, according to the inherent nature of LSH, only the "most similar" neighbors of a target user can be returned for subsequent service recommendation in our $DistSR_{Amplify\text{-}LSH}$ approach; as a consequence, high recommendation accuracy is often guaranteed. Second, the AND/OR operations are adopted in our approach to amplify LSH, through which the "False-positive" and "False-negative" search results are reduced; accordingly, the recommendation accuracy is improved.

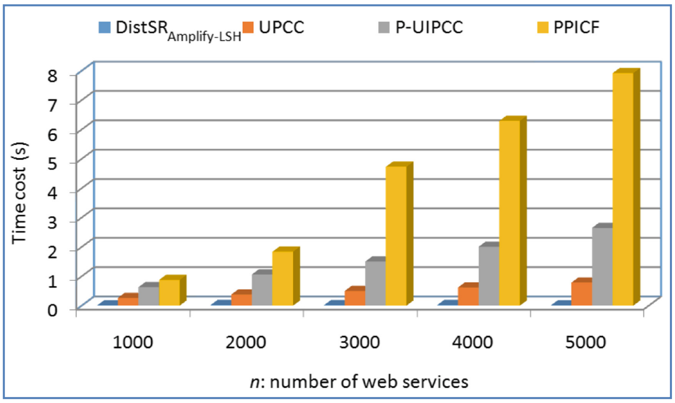
Similar experiment results can be observed from Fig. 4(b) where $m = 300$ holds and n is varied from 1000 to 5000. The reasons are the same as those in Fig. 4(a) and are not repeated here.

***Profile* 2: Recommendation Efficiency Comparison**

In this profile, we test the recommendation efficiency of four approaches. The recruited experiment parameters are set as below: m is varied from 100 to 300, n is varied from 1000 to 5000, $T = r = 10$ holds. The concrete experiment results are shown in Fig. 5.



(a) $n = 5000$

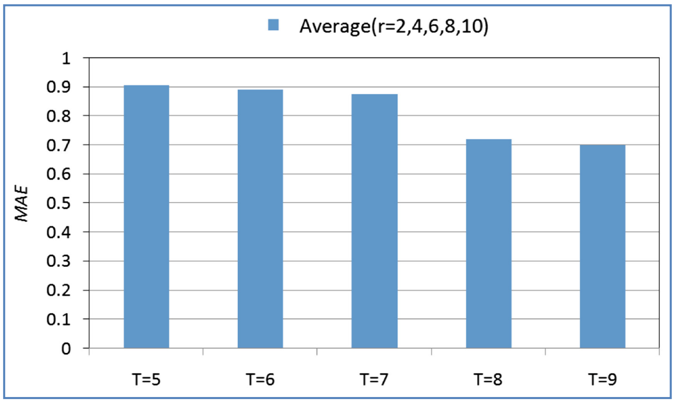


(b) $m = 300$

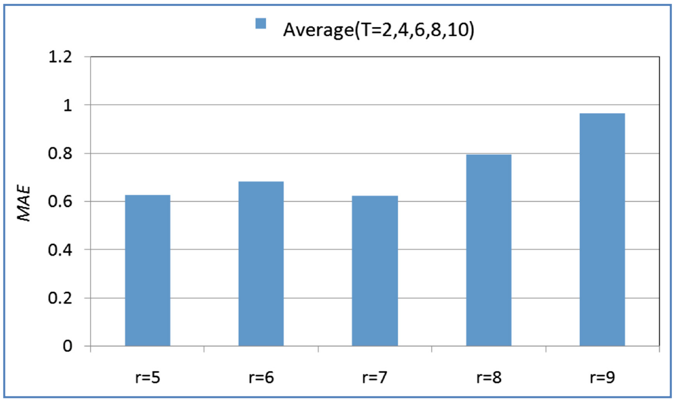
Fig. 5. Recommendation efficiency comparison of four approaches

In Fig. 5(a), $n = 5000$ holds. The experiment results indicate that the time costs of *P-UIPCC* and *PPICF* approaches both increase with the growth of m and are often high because of the recruited additional privacy-preservation operations (e.g., data obfuscation, divide-merge). Similar experiment result can be observed from the *UPCC* approach in Fig. 5; however, *UPCC* performs better than *P-UIPCC* and *PPICF* in recommendation efficiency as no additional privacy-preservation operations are adopted. While our $DistSR_{Amplify\text{-}LSH}$ approach outperforms the other three approaches in terms of recommendation efficiency, because the recruited LSH tables can be built offline and the rest step of online neighbor finding is very efficient (in the best cases, the time complexity of online neighbor finding is O(1) [15]). Similar experiment results can be observed from Fig. 5(b), which are not analyzed repeatedly.

Profile* 3: Recommendation Accuracy of $DistSR_{Amplify\text{-}LSH}$ with Respect to *T and r
The number of LSH tables (i.e., T) and the number of LSH functions in each LSH table (i.e., r) are two key parameters in our proposal. Therefore, in this profile, we test the



(a) *MAE* w.r.t. *T*



(b) *MAE* w.r.t. *r*

Fig. 6. Recommendation accuracy of $DistSR_{Amplify\text{-}LSH}$ w.r.t. T and r

recommendation accuracy of $DistSR_{Amplify\text{-}LSH}$ with respect to T and r. The parameters are set as follows: $m = 200$, $n = 3000$. The concrete experiment results are shown in Fig. 6.

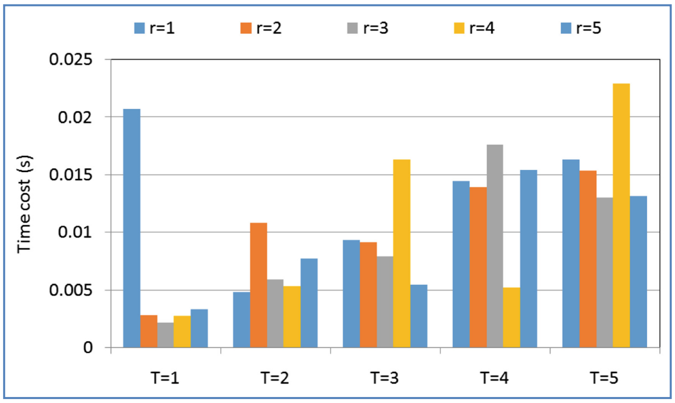
In Fig. 6(a), $T = \{5, 6, 7, 8, 9\}$, $r = \{2, 4, 6, 8, 10\}$. For each T value, we test the *MAE* values corresponding to different r values, and finally the average *MAE* value is adopted. The experiment results indicate that the recommendation accuracy increases (i.e., *MAE* decreases) slightly when T grows. This is because in *Strategy-2*, the AND operation is taken over the generated T LSH tables; therefore, more LSH tables often mean a stricter search condition for neighbors as well as higher service recommendation accuracy.

In Fig. 6(b), $T = \{2, 4, 6, 8, 10\}$, $r = \{5, 6, 7, 8, 9\}$. For each r value, we test the *MAE* values corresponding to different T values, and finally the average *MAE* value is adopted. The experiment results show that the recommendation accuracy decreases (i.e., *MAE* increases) with the growth of r approximately. This is because in *Strategy-1*, the OR operation is taken over the chosen r LSH functions; as a consequence, more LSH functions often mean looser search condition for neighbors as well as lower recommendation accuracy.

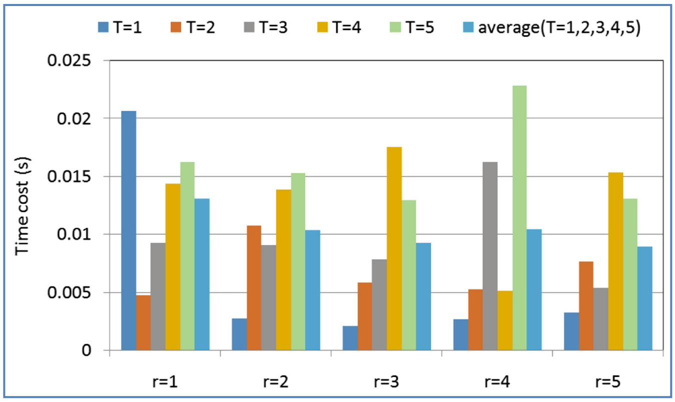
Profile* 4: Recommendation Efficiency of $DistSR_{Amplify\text{-}LSH}$ with Respect to *T and r In this profile, we test the efficiency of $DistSR_{Amplify\text{-}LSH}$ with respect to T and r. Here, $m = 200$, $n = 3000$, T is varied from 1 to 5, r is varied from 1 to 5. The experiment results are shown in Fig. 7. As Fig. 7(a) shows, the time cost of our proposal increases approximately with the growth of T because of the AND operation (over T LSH tables) adopted in *Strategy-2*. While as can be seen from Fig. 7(b), the time cost of our proposal does not exhibit a very regular variation tendency when r grows. Typically, the average time cost (i.e., the blue column) stays approximately the same with the growth of r because of the OR operation (over r LSH functions) adopted in *Strategy-1*.

Profile* 5: Recommendation Failure Rate of $DistSR_{Amplify\text{-}LSH}$ with Respect to *T The number of LSH tables, i.e., T plays an important role in the successful service recommendation because of the adopted AND operation in *Strategy-2*. Generally, for a target user, more LSH tables mean a stricter search condition for similar neighbors. Therefore, if T is large enough, the search condition may become too strict to find a qualified neighbor of the target user; in this situation, service recommendation is failed. In this profile, we test the failure rate of $DistSR_{Amplify\text{-}LSH}$ with respect to T. The parameters are set as follows: $m = 200$, $n = 3000$, $r = 1$, T is varied from 5 to 15. Concrete experiment results are presented in Fig. 8.

As Fig. 8 shows, the failure rate of our $DistSR_{Amplify\text{-}LSH}$ approach increases with the growth of T approximately; this is because more LSH tables (i.e., a larger T value) often mean a stricter search condition for similar neighbors and hence may lead to a recommendation failure. While when T is small (e.g., when $T = 5$ or $T = 6$), the failure rate of $DistSR_{Amplify\text{-}LSH}$ drops to a small value even 0. Therefore, through tuning the parameter value of T, a low recommendation failure rate can be guaranteed.



(a) Time cost w.r.t. *T*



(b) Time cost w.r.t. *r*

Fig. 7. Recommendation efficiency of $DistSR_{Amplify\text{-}LSH}$ w.r.t. *T* and *r* (Color figure online)

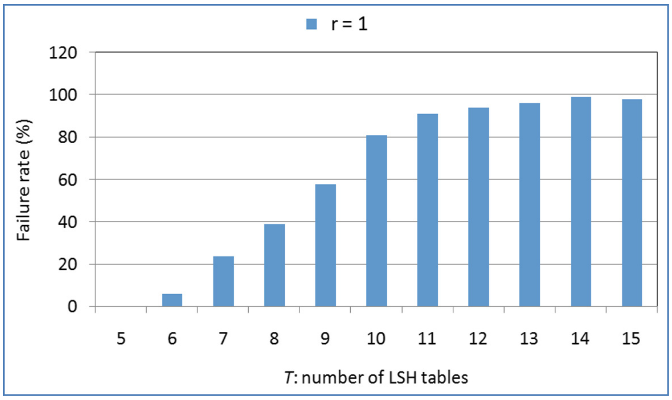


Fig. 8. Recommendation failure rate of $DistSR_{Amplify\text{-}LSH}$ w.r.t. *T*

6 Related Work and Comparison Analyses

In this section, we compare our proposal with existing CF-based recommendation approaches. Generally, CF-based service recommendation approaches can be divided into two categories: model-based CF and memory-based CF.

6.1 Model-Based CF

Model-based CF approaches recruit the historical service quality data observed by users to build a user-service recommendation model offline, and then make online recommendation based on the obtained model. Several model-based CF approaches are introduced for service recommendation, e.g., *Matrix Factorization* [16], *Latent Dirichlet Allocation* [17] and *clustering* [18]. Generally, the recommendation efficiency of these model-based CF approaches is high as the recommendation model can be trained offline. However, these approaches have two shortcomings. First, they seldom consider the privacy leakage risks in service recommendation process. Second, they often assume that the historical service quality data used for service recommendation are centralized, without considering the distributed service recommendation scenarios where historical service quality data are from multiple independent platforms; therefore, these approaches fall short in handling the distributed service recommendation problems.

6.2 Memory-Based CF

Memory-based CF recommendation approaches first employ the historical service quality data to find similar users or similar services, and then utilize them to make service recommendations. User-based CF and item-based CF are employed for service recommendation in [19] and [20], respectively. In order to combine the advantages of user-based CF and item-based CF, a hybrid service recommendation approach named *WSRec* is proposed in [21]. Experiment results show that the recommendation performance is improved. As the running quality of a web service often depends on the service invocation context (e.g., *service invocation time*, *service location*, *user location*), time-aware recommendation and location-aware recommendation are put forward by [22] and [23], respectively. Besides, user preferences also play an important role in the service selection decisions of a target user; in view of this observation, preference-aware CF recommendation approach is introduced in [24], where the users with similar preferences are recruited to make service recommendations.

However, the above approaches do not consider the privacy leakage risks in service recommendation process. In view of this shortcoming, in [25], users are suggested to release only a small portion of their observed service quality data to the public so that the remaining majority of data are still secure. However, the released small portion of data can still reveal part of a user's private information. In order to protect the private information of users better, data obfuscation strategy is adopted in [13] to hide the real service quality data. However, as the service quality data used to make service recommendations have been obfuscated, the recommendation accuracy is reduced accordingly. In view of this drawback, a "divide-merge" strategy is put forward in [14] where each piece of service quality data is divided into several segments with little private information, and then the segments are recruited for user similarity calculation and service recommendations. However, two shortcomings are present in [14]. First, the recommendation efficiency is decreased severely as the adopted "divide-merge" operations bring additional time cost. Besides, some private information of users cannot be protected well, e.g., *the service intersection commonly invoked by two users*.

In view of the shortcomings of existing research work, a novel amplified LSH-based recommendation approach named $DistSR_{Amplify\text{-}LSH}$ is proposed in this paper, to solve the privacy-preserving distributed service recommendation problems in the distributed environment. Through the extensive experiments conducted on a real distributed service quality dataset *WS-DREAM*, we validate the feasibility and advantages of our proposal in terms of recommendation accuracy and efficiency while guaranteeing privacy-preservation.

7 Conclusions

In this paper, we put forward a novel privacy-preserving service recommendation approach based on amplified Locality-Sensitive Hashing, i.e., $DistSR_{Amplify\text{-}LSH}$, to handle the distributed service recommendation problems. Through Locality-Sensitive Hashing, user indexes can be built offline; as a consequence, the neighbor search efficiency and service recommendation efficiency are improved significantly. Besides, due to the inherent nature of LSH, user privacy can be protected during the distributed service recommendation process. Moreover, we amplify LSH by integrating the AND/OR operations over multiple LSH tables or LSH functions, through which the "False-positive" and "False-negative" recommended results are reduced; as a result, the service recommendation accuracy is improved significantly. Finally, through a set of experiments deployed on the distributed service quality dataset *WS-DREAM*, we validate the feasibility of our proposed $DistSR_{Amplify\text{-}LSH}$ approach. Experiment results demonstrate that our proposal can achieve a good recommendation performance in terms of recommendation accuracy and efficiency while guaranteeing privacy-preservation.

In the future, we will further investigate the distributed service recommendation problems with multiple quality dimensions. Besides, the running qualities of web services are often not static, but dynamic; therefore, we will study the dynamic quality-aware distributed service recommendation problems in the future.

Acknowledgements. This paper is partially supported by the National Key Research and Development Program of China (No. 2017YFB1001800), Natural Science Foundation of China (Nos. 61402258, 61672276), UoA Faculty Research Development Fund (No. 3714668), Open Project of State Key Laboratory for Novel Software Technology (No. KFKT2016B22).

References

1. Blake, M.B., Saleh, I., Wei, Y., Schlesinger, I.D., Yale-Loehr, A., Liu, X.: Shared service recommendations from requirement specifications: a hybrid syntactic and semantic toolkit. Inf. Softw. Technol. **57**, 392–404 (2015)

2. Al-Hassan, M., Haiyan, L., Jie, L.: A semantic enhanced hybrid recommendation approach: a case study of e-Government tourism service recommendation system. Decis. Support Syst. **72**, 97–109 (2015)
3. Segev, A., Sheng, Q.: Bootstrapping ontologies for web services. IEEE Trans. Serv. Comput. **5**(1), 33–44 (2012)
4. Cao, G., Kuang, L.: Identifying core users based on trust relationships and interest similarity in recommender system. In: IEEE International Conference on Web Services, pp. 284–291 (2016)
5. Zhong, Y., Fan, Y., Tan, W., Zhang, J.: Web service recommendation with reconstructed profile from mashup descriptions. IEEE Trans. Autom. Sci. Eng. (2016)
6. Mashal, I., Chung, T.-Y., Osama, O.: Toward service recommendation in internet of things. In: IEEE International Conference on Ubiquitous and Future Networks, pp. 328–331 (2015)
7. Zheng, Z., Ma, H., Lyu, M.R., King, I.: QoS-aware web service recommendation by collaborative filtering. IEEE Trans. Serv. Comput. **4**(2), 140–152 (2011)
8. Gionis, A., Indyk, P., Motwani, R.: Similarity search in high dimensions via hashing. VLDB **99**(6), 518–529 (1999)
9. Lee Rodgers, J., Nicewander, W.A.: Thirteen ways to look at the correlation coefficient. Am. Stat. **42**(1), 59–66 (1988)
10. Data Mining and Query Log Analysis for Scalable Temporal and Continuous Query Answering (2015). http://www.optique-project.eu/
11. Zheng, Z., Zhang, Y., Lyu, M.R.: Investigating QoS of real world web services. IEEE Trans. Serv. Comput. **7**(1), 32–39 (2014)
12. Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: International Conference on Uncertainty in Artificial Intelligence, pp. 43–52 (1998)
13. Zhu, J., He, P., Zheng, Z., Lyu, M.R.: A privacy-preserving QoS prediction framework for web service recommendation. In: IEEE International Conference on Web Services, pp. 241–248 (2015)
14. Li, D., Chen, C., Lv, Q., Shang, L., Zhao, Y., Lu, T., Gu, N.: An algorithm for efficient privacy-preserving item-based collaborative filtering. Future Gener. Comput. Syst. **55**, 311–320 (2016)
15. Slaney, M., Casey, M.: Locality-sensitive hashing for finding nearest neighbors. IEEE Sig. Process. Mag. **25**(2), 128–131 (2008)
16. Yao, L., Sheng, Q.Z., Qin, Y., Wang, X., Shemshadi, A., He, Q.: Context-aware point-of-interest recommendation using tensor factorization with social regularization. In: International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1007–1010 (2015)
17. Zhong, Y., Fan, Y., Huang, K., Tan, W., Zhang, J.: Time-aware service recommendation for mashup creation in an evolving service ecosystem. In: IEEE International Conference on Web Services, pp. 25–32 (2014)
18. Wu, C., Qiu, W., Zheng, A., Wang, X., Yang, X.: QoS prediction of web services based on two-phase k-means clustering. In: IEEE International Conference on Web Services, pp. 161–168 (2015)
19. Rong, H., Huo, S., Hu, C., Mo, J.: User similarity-based collaborative filtering recommendation algorithm. J. Commun. **35**(2), 16–24 (2014)
20. Chung, K.-Y., Lee, D., Kim, K.J.: Categorization for grouping associative items using data mining in item-based collaborative filtering. Multimed. Tools Appl. **71**(2), 889–904 (2014)
21. Jiang, C., Duan, R., Jain, H.K., Liu, S., Liang, K.: Hybrid collaborative filtering for high-involvement products: a solution to opinion sparsity and dynamics. Decis. Support Syst. **79**, 195–208 (2015)

22. Wang, X., Zhu, J., Zheng, Z., Song, W., Shen, Y., Lyu, M.R.: A spatial-temporal QoS prediction approach for time-aware web service recommendation. ACM Trans. Web **10**(1), 7 (2016)
23. Yu, C., Huang, L.: A web service QoS prediction approach based on time- and location-aware collaborative filtering. Serv. Oriented Comput. Appl. **10**(2), 135–149 (2016)
24. Fletcher, K.K., Liu, X.F.: A collaborative filtering method for personalized preference-based service recommendation. In: IEEE International Conference on Web Services, pp. 400–407 (2015)
25. Dou, W., Zhang, X., Liu, J., Chen, J.: HireSomeII: towards privacy-aware cross-cloud service composition for big data applications. IEEE Trans. Parallel Distrib. Syst. **26**(2), 455–466 (2015)