Amplified locality-sensitive hashing-based recommender systems with privacy protection

Xiaoxiao Chi[1], Chao Yan[2], Hao Wang [3], Wajid Rafique[4], Lianyong Qi[*]

1. School of Information Science and Engineering, Qufu Normal University, Rizhao 276800, China

2. School of Information Science and Engineering, Qufu Normal University, Rizhao 276800, China

3. Department of Computer Science, Norwegian University of Science and Technology, 2815 Gjøvik, Norway

4. State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, P. R. China

Department of Computer Science and Technology, Nanjing University, Nanjing, P. R. China

*. School of Information Science and Engineering, Qufu Normal University, Rizhao 276800, China

Abstract: With the advent of IoT (Internet of Things) age, the variety and volume of web services have been increasing at a fast speed. This often leads to users' selections for web services more complicated. Under the circumstance, a variety of methods such as Collaborative Filtering are adopted to deal with this challenging situation. While traditional Collaborative Filtering method has some shortcomings, one of which is that only centralized user-service data are considered while distributed quality data from multiple platform are ignored. Generally, service recommendation across different platforms often involves data communication among multiple platforms, during which user privacy may be disclosed and much computational time is required. Considering these challenges, a unique amplified LSH (Locality-Sensitive Hashing)-based service recommendation method, i.e., $SR_{Amplified\text{-}LSH}$, is proposed in the paper. $SR_{Amplified\text{-}LSH}$ can guarantee a good balance between accuracy and efficiency of recommendation and user privacy information. Finally, extensive experiments deployed on *WS-DREAM* dataset validate the feasibility of our proposed method.

Keywords: amplified LSH, collaborative filtering, privacy, distributed service recommendation, efficiency,

1. INTRODUCTION

With prosperity of internet, the volume and variety of service on internet are both increasing rapidly [1-2]. IoT (Internet of things) devices also generate huge amount of data which require rich resources to store and process these data [29, 31, 39, 35]. And the process of dealing with these IoT data also make large energy consumption as well as privacy leakage [24, 30]. Therefore, it is often a very exhausted and time-consuming job for users to find out the services they need from large volume of candidates. Besides, traditional manual search for finding appropriate services is lack of efficiency and accuracy [3-6]. In this situation, various lightweight service recommendation methods are adopted by researchers to address the abovementioned problems [32, 33]. For example, Collaborative Filtering (CF) is often recruited to analyze users' historical records on web services to make appropriate recommendation as well as to relief the heavy burden on users.

However, traditional Collaborative Filtering approaches often assume that the data are centralized from a single platform instead of multiple platforms [7-9]. For example, user *A* has invoked some web services on one platform *X* while user *B* has invoked other web services on another platform *Y*. So the data of users *A* and *B* are from two independent platforms, respectively. Generally, there are two main barriers in front of similarity calculation between *A* and *B*. First, due to probable privacy concerns, platform *X* and platform *Y* are often reluctant to share their data with

each other; in this situation, we cannot figure out the similarity between A and B through traditional CF-based approaches. Second, the volumes of data platform X and platform Y often grow rapidly with time elapsing, which bring additional communication cost and time delay between these two platforms; consequently, the efficiency of recommendation is decreased severely.

Considering these abovementioned challenges, we recruit Locality-Sensitive Hashing (LSH) [34] technique to get accurate results in the process of recommendation where users' privacy information can be preserved and efficiency of service recommendation in the distributed environment is also improved. However, LSH technique is a probability method, so during the process of recommending web services it is common phenomenon to generate "FP (False-positive)" and "FN (False-negative)" results by using the method. Concretely, we need to amply LSH technique at the am of reducing the probable "FP" and "FN" recommended results. Afterwards, by applying the amplified LSH technique to our approach, we propose a unique recommendation approach, i.e., $SR_{Amplified\text{-}LSH}$, which could be utilized in distributed environment. With the most essential characteristic of amplified LSH technique, $SR_{Amplified\text{-}LSH}$ performs better than traditional approaches in aspects of recommendation precision and time cost while ensuring that the sensitive user data are secure. At last, we validate the feasibility of $SR_{Amplified\text{-}LSH}$ through extensive real-world experiments.

In conclusion, our paper mainly consists of following contributions.

(1) We make brief introduction of LSH technique in the process of recommendation to protect users' privacy information as well as to improve the efficiency of recommendation.

(2) We recognize that LSH technique may lead to "FP" or "FN" recommended results as LSH is a probabilistic technique.

(3) We amplify traditional LSH to reduce the probability of generating "FP" or "FN" recommended results while guaranteeing high accuracy of recommended results.

(4) A raft of experiments are conducted on a real-world distributed QoS dataset *WS-DREAM* [1]to estimate the feasibility of our proposal.

The structure of the paper is described as follows. We make brief introduction of the motivation about our research in Section 2. In Section 3, we describe our problem with simple symbols to understand easily. We come up with a unique approach called $SR_{Amplified\text{-}LSH}$ in order to handle the recommendation problem in Section 4. A large amount of experiments are conducted in Section 5 to evaluate the feasibility of our proposal. Related works are briefly introduced in Section 6. And finally, in Section 7, we summarize the paper and point out the future research directions.

2. MOTIVATION

In Figure 1, there is an example illustrating the motivation of this paper. As you see in the picture, two users have both invoked large number of web services on platform X and platform Y. For example, assume that platform X and platform Y are Taobao and Jingdong respectively. There are so many customers buying products on the two giants. So the firms could produce a profile about customers based on users' activities such as what type of good user have purchased, which page user have gone through and so on. And firms can only hold their own data about customers. Note that in our paper the meanings of the word "user" and the "customer" are same. Concretely, user u_1 has invoked a lot of web services $\{ws_{1,1}, ws_{1,2}, ..., ws_{1,n}\}$ and users u_2 has also invoked

[1] https://wsdream.github.io/

many web services {$ws_{2,1}$, $ws_{2,2}$, ..., $ws_{2,n}$}. What we have to solve is calculating the similarity between user u_1 and user u_2. However, through using traditional CF-based methods there are some barriers existing in the process:

(1) Privacy concern. It is unsafe for different platforms to share their historical QoS data with each other. If they share their data without any protection measure, there will be large probability that partial private information of users is exposed.

(2) For both platforms of *X* and *Y*, their volume of historical QoS data would have a sharp growth and update with time. Under this circumstance, the efficiency and accuracy of traditional collaborative filtering methods would decrease and the requirement of short response time may not be satisfied.

(3) There is large probability to produce "FP" and "FN" recommended results. So the user satisfaction degree would be decreased significantly.

In view of these challenges, we come up with a unique method described as $SR_{Amplified\text{-}LSH}$ that is going to be introduced in the next part.

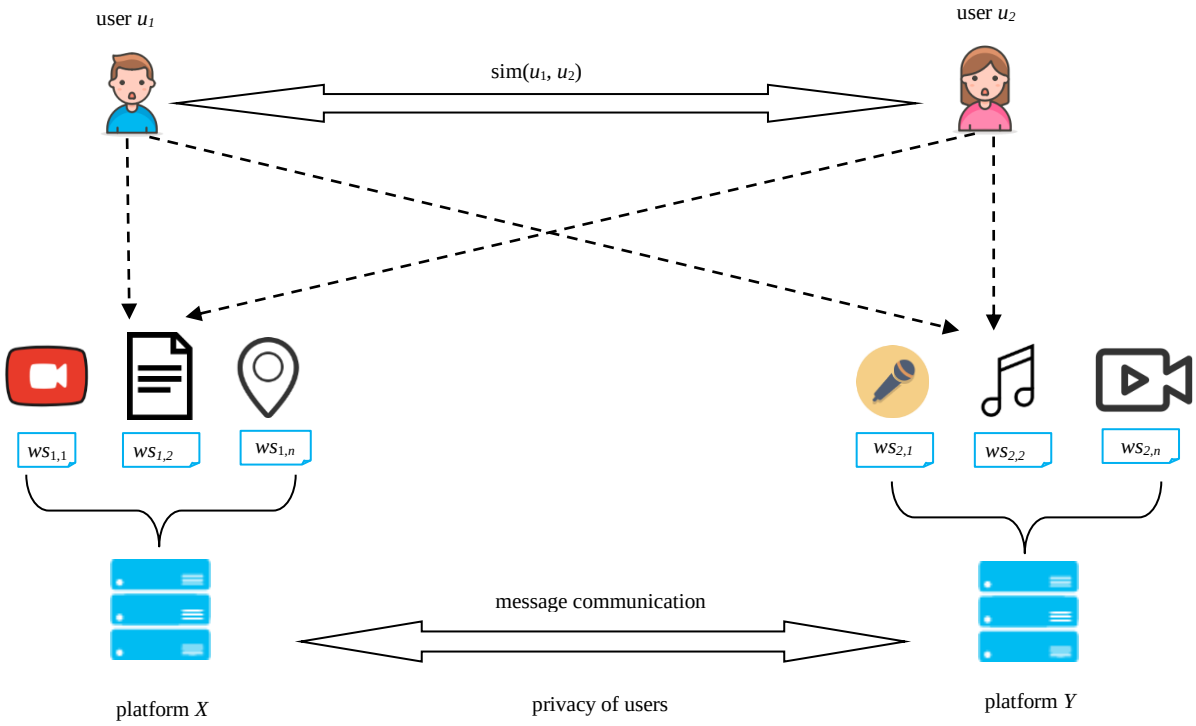


Figure 1. Distributed service recommendation: an example

3. PROBLEM FORMULATION

In the paper, what we focus on is the research of recommendation on web services in distributed environment. To clarify the following discussions and make the content of research easier to understand, we first simply describe the problem as a three-tuple **SR(*PF*, *U*, *WS*)**, where

1. ***PF*** = {pf_1,⋯, pf_z}: pf_k ($1 \leqslant k \leqslant z$) denotes *k*-th distributed service platform.
2. ***U*** = {U_1, ⋯, U_z}: U_k ($1 \leqslant k \leqslant z$) indicates the collection of users corresponding to the pf_k.

More concrete, U_k = {$u_{k\text{-}1}$, ⋯, $u_{k\text{-}m}$}: $u_{k\text{-}i}$ ($1 \leqslant i \leqslant m$) indicates *i*-th user on pf_k .

3. $\boldsymbol{WS}$ = {ws_1, ⋯, ws_n}. Here, ws_j ($1 \leqslant j \leqslant n$) denotes all services contained in the j-th platform pf_j. And ws_j = {ws_{j-1}, ws_{j-2}, ⋯, ws_{j-q}} where ws_{j-p} ($1 \leqslant p \leqslant q$) denotes the p-th web service in the j-th platform. Note that users are supposed to invoke these web services, thus the user-service quality data will be recorded. If a user doesn't use a service, the QoS data is null.

4. A RECOMMENDATION SOLUTION BASED ON AMPLIFIED LOCALITY-SENSITIVE HASHING: $SR_{Amplified\_LSH}$

In this section, we will introduce our proposed method, $SR_{Amplified\text{-}LSH}$, in order to improve the accuracy of recommendation results and the efficiency of recommendation. The following parts will described the method in detail.

4.1 LSH Family

Before making introduction of amplified locality-sensitive hashing, we first clarify the basic idea of locality-sensitive hashing. LSH is a kind of hash technique that is aiming to find the most appropriate neighbors in an efficient way and meanwhile protects the privacy of users or items. Generally, the main idea about LSH is going to be described as follows:

Suppose that A and B are two data objects distributed in space and $d_{(A,B)}$ is represented as the distance between point A and point B. If the two points are very close in distance, they will be hashed in same bucket via hash function $f(.)$ with large probability. Otherwise, if they are far away from each other in distance, they will be hashed in different buckets instead of same bucket via hash function $f(.)$.

More formally, the expression $f(A) = f(B)$ is used to indicate that item A has been hashed in the same buckets with item B. And the expression $f(A) \neq f(B)$ is applied to indicate that the bucket where A has been hashed is different from B. Generally, a hash function f (.) is regarded as a qualified hash function if the conditions (1) and (2) are satisfied. Thus, a set of functions of this form will be called a family of (d_1, d_2, p_1, p_2)-sensitive hash functions where d_1 is lower than d_2.

$$\text{If } d(x, y) \leq d_1, \text{ then the probability that } f(x) = f(y) \text{ is at least } p_1 \quad (1)$$

$$\text{If } d(x, y) \geq d_2, \text{ then the probability that } f(x) = f(y) \text{ is at most } p_2 \quad (2)$$

However, LSH is a probabilistic method. Thus, during the process of calculating, there would be generating "FP" or "FN" results. In this way, the recommended results would be inaccurate and unsatisfied. So we propose our amplified LSH method to decrease the "FP" and "FN" results in order to enhance the accuracy of recommended results and improve the degree of satisfactory of users.

4.2 Amplify Locality-sensitive Hashing Family

In this part, we will amplify the original hash family F to generate a new hash family named F' through using AND/OR-construction [36]. According to original hash family, the process of

building amplified hash family through "OR" construction is described as follows: Suppose that the original hash family called *F* also contains *s* hash functions. Under this circumstance, several hash functions will be selected randomly to compose a new hash function of the amplified hash family. If the process of selection is done for several times, the amplified hash family will contain several new hash functions. For example, if the process of selection is done for 10 times, then the amplified hash family will contain 10 new hash functions. So we will build target user's index with the amplified hash family. So "OR" construction is that a certain user *u* falls into the same side with the target user after being hashed by at least one new hash function. "And" construction is that a certain user *u* falls into the same side with the target user after being hashed only by all new hash functions.

4.3 $SR_{Amplified\text{-}LSH}$: Amplified LSH method for Service Recommendation

The basic idea about amplified LSH is five aspects: first, according to original hash family *F*, new amplified LSH function Family *F'* is constructed. Second, based on new amplified hash family, we need to build users' indices. Third, it's important to define a kind of "similarity" relationship between users based on amplified LSH technique. Forth, we need to find neighbors of target users based on users' indices. And finally, optimal web service with optimal value will be recommended to target user.

In summary, the $SR_{Amplified\text{-}LSH}$ approach mainly includes the five steps, as shown in Figure. 2. And in table 1, we give the symbols corresponding definition clearly.

Step1: Construct new amplified LSH Family *F'*. According to original LSH family, we built new amplified LSH family in this step.

Step2: Build users' indices offline according to new amplified LSH family. Fully utilizing users' historical QoS data, we generate users' indices contained less private information.

Step3: Define similarity relationship between users based on AND-OR-AND strategy. According to two kinds of different standards described in 4.2, we need to define relationship between users based on the three stages.

Step4: Finding similar users online for u_{target}. According to step 3, we need to find the neighbors of target users.

Step5: Web service recommendation. According to step 4, we need to recommend web services to target user.

Figure 2. Concrete process of $SR_{Amplified\text{-}LSH}$

(1) Step1: Construct new amplified LSH family *F'*

Before constructing new amplified LSH family *F'*, we need to construct original hash family *F*.

According to part A, for different two points, we need a certain distance measure to evaluate the distance between them. Actually, there are several kinds of distance measures to be adopted. Here, we select Pearson Distance to calculate the distance between the two points. According to the type of distance being adopted, there is corresponding hash functions named *f*(.). More concretely, for a user *u*, the recommender system can extract the historical QoS data on *n* web services to build an *n*-dimensional vector $\vec{u}$ = (r_1, r_2, ..., r_n). And each LSH function can be represented as an *n*-dimensional random vector. In equation (3), vector $\vec{v}$ = (v_1, v_2, …, v_n) denotes an *n*-dimensional vector where v_j ($1 \leqslant j \leqslant n$) is a random value ranging from -1 to 1; the mathematic symbol "$\circ$" represents the dot product of two vectors. If result of dot product between vector $\vec{u}$ and vector $\vec{v}$ is greater than zero, the value of function will be equal to binary value 1. Otherwise, the value of function will be equal to binary value 0. What above mentioned is the process of constructing original LSH family.

We need to build new amplified LSH family of hash functions based on abovementioned original hash family. Assume that *F* is the original hash family and *F'* is the new hash family. What we need to do firstly is randomly choosing hash functions in family *F* for fixed *t*. And a new hash function represented as a single member of *F'* is composed of fixed *t* functions. Next, the process of randomly choosing functions from original family *F* needs to be achieved for *k* times in order to generate all new hash functions in *F'* .Therefore, we will get the new amplified hash family which have *k* members.

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

(1) Step2: Build users' indices offline based on new amplified hash family.

Due to privacy concern, it's necessary to protect users' sensitive QoS data. So we use amplified LSH technique to transform users' QoS data to insensitive users' indices in order to protect users' privacy.

According to the step 1, we have built the new amplified family. So we are aiming to build users' indices in this step. For a user's QoS data vector $\vec{u}$, it has to be calculated with each member of *F'*. Each member in new amplified hash family *F'* is composed of fixed *t* ($1 < t < s$) hash functions which are randomly chosen from original hash family *F*. The vector of each user is converted to a binary value $f(\vec{u})$ ($\in\{0,1\}$) through dot product with one hash function in original hash family *F*. So under the circumstance, we can obtain binary values for fixed *t*. That is h_i = (f_1, f_2, ..., f_t) ($1 \leqslant i \leqslant k$) and the value of each element of h_i is either 0 or 1. For simplicity, it's necessary to convert the sequence of binary values into a decimal number which is called X. However, even a single member of new amplified family may not fully describe the user's preferences about web services. So we need to repeat the process with all remaining members of new hash family *F'*. Finally, we can get decimal values for fixed *k*. The user indices can be represented as $H(\vec{u}) = (H_1(\vec{u}), H_2(\vec{u}), ..., H_k(\vec{u}))$ and $H_i(\vec{u})$ is a decimal number.

In this way, we have built users' indices offline and the indices of users have less private information about users. Thus, in the process of building users' indices, users' privacy have been protected very well under the distributed platforms.

(2) Step3: Define similarity relationship between users based on AND-OR-AND strategy

According to the LSH theory, if user u_a has fallen into the same bucket with user u_b, they will be considered as similar neighbors with large probability. However, in essence LSH is a search technique based on probability: therefore, it is of large probability to generate unsatisfactory recommended results. In other words, LSH may generate "FP" or "FN" results, which reduces the accuracy of recommendation about web services to a large extent.

To put up with this shortcoming, we utilize AND/OR strategies in different stages to improve the accuracy of the recommendation and satisfy users' needs sufficiently.

Strategy-1: "AND" operation over *t* functions of a member of new LSH family to reduce "FP" results.

In step 1, we have built a new amplified LSH family where each member of the family includes fixed *t* original hash functions. In order to make the "false-positive" results decrease, "AND" operation is adopted over these *t* hash functions.

Suppose that *t* is equal to 5, so a member of new amplified hash family includes 5 hash functions selected from original LSH family. For two users u_a and u_b, "AND" operation over these 5 hash functions commands that values of the five hash functions should be equal correspondingly.

$$\forall i,\ satisfy\ f_i(u_a) = f_i(u_b),\ 1 \le i \le t \tag{4}$$

Strategy-2: "OR" operation over *k* members of new LSH family for reducing the "FN" results.

In step (1), we have built a new amplified hash family where fixed *k* hash functions are included in the family. And each member consists of original hash functions for fixed *t*. At the aim of reducing the "FN" results, we apply "OR" operation to these *k* hash functions. More specifically, if condition (5) holds, the two users will be considered as similar users. Here, we adopt the expression $H(\overrightarrow{u_a})\ \underline{OR}\ H(\overrightarrow{u_b})$ to express the "similarity" relationship between u_a and u_b.

$$\exists\, i,\ satisfy\ H_i(\overrightarrow{u_a}) = H_i(\overrightarrow{u_b}),\ 1 \le i \le k \tag{5}$$

Strategy-3: "AND" operation over *T* hash tables in at the aim of reducing the "FN" results.

At the aim of improving accuracy of recommended results, we need to repeat Step (1) and Step (2) to generate fixed *T* hash tables. we adopt the expression, $u_a \xleftrightarrow{sim} u_b$, to represent relationship defined in (6).

$$\forall\, i,\ satisfy\ H_x(\overrightarrow{u_a})\ \underline{OR}\ H_x(\overrightarrow{u_b}),\ 1 \le i \le T \tag{6}$$

In conclusion, we use strategy-1, strategy-2 and strategy-3 to define a unique relationship between u_a and u_b i.e., $u_a \xleftrightarrow{sim} u_b$ in order to improve the accuracy of services recommendation.

(4) Step4: Finding similar users online for u_{target}

The index for u_{target}, i.e. $H(\overrightarrow{u_{target}})$ can be calculated based on step (1) and step (2). If the condition $H(\overrightarrow{u_a}) = H(\overrightarrow{u_{target}})$ holds, user *a* will be considered as similar neighbor of target user. Thus, we will put them into same bucket called *Neighbor*. Thus, all elements of *Neighbor* have same index. And users in the same bucket with the target user are the neighbors of the target user. However, we will generate *T* hash tables, we need to search through all hash tables to find all neighbors of target users and put them into a set called $Neighbor^{\#}$. In this paper, for simplicity, we do not consider the number of appearances of neighbors.

(5) Step5: Web service recommendation

In this step, we need to recommend the web services which have never been invoked by target user u_{target}. For each new web services ws_j never rated by u_{target} before, it's QoS data will be predicted based on u_{target}'s neighbors recorded in $Neighbor^{\#}$. Specifically, the prediction process is specified in (7) where $|\ Neighbor^{\#}\ |$ indicates the size of set $Neighbor^{\#}$. Next, we should calculate all data of web services which target user has never invoked and determine an optimal web services as the recommendation result and recommend it to u_{target}.

$$predict\_data = \frac{1}{|\ Neighbor^{\#}\ |} * \sum_{u_a \in Neighbor^{\#}} r_a \tag{7}$$

Algorithm: $SR_{Amplified\text{-}LSH}$

Inputs: (1) $PF = \{pf_1, pf_2, \ldots, pf_z\}$: platform set;
(2) $U = \{u_1, u_2, \ldots, u_m\}$: user set;
(3) $WS = \{ws_1, ws_2, \ldots, ws_n\}$: web service set;
(4) U_{target} : a target user

Output: $ws_{optimal}$: an optimal candidate web service being recommended to the target user

```
1  /* Step 1- Step 2: Building user indices according to new amplified LSH family */
2   for i = 1 to s do
3    for j = 1 to n do
4        v_ij = random [-1, 1]
5    end for
6   end for
7   for i = 1 to k do // k LSH functions in each new LSH family
8       h_i(.) = randomly choose fixed t functions from original family
9       for j =1 to t do
10           calculate h_i(u_a)  based on (4)
11      end for
12      transform h_i(u_a)  into a decimal value X
13      H_i(u_a) = X
14  end for
15  H(u_a) = (H_1(u_a), H_2(u_a), ..., H_k(u_a))
16  Repeating line 1-15 T times to produce T hash tables
```

```
17  /* Step 3 - Step 4: Defining "friend" of users according to amplified LSH and online friend
    search for u_target */
18  for i = 1 to m do
19     for x = 1 to T do
20        if H_x(u_i) OR H_x(u_target)  holds according to (5) and  u_a <-sim-> u_b  holds based on (6)
21        then put u_i into Neighbor
22        end if
23     end for
24  end for
```

```
25  /* Step 5: Movie Recommendation */
26  for i = 1 to n do // n candidate movies
27       if ws_target,i = 0 // u_target has never seen the movie
28        then for j =1 to | Neighbor |  do
29                 if ws_j,i ≠ 0
30                  then COUNT++
31                 ws_target,i = ws_target,i + ws_j,i
32                 end if
33              end for
34              ws_target, i = ws_target,i / COUNT
35       end if
36  end for
37  ws_optimal = max{ws_target,1,...,ws_target,n}
38  return ws_optimal to target user u_target
```

5. EVALUATION

In this part, we are going to discuss the results of our proposed methods compared with other methods.

5.1 Experimental Settings

Here, our estimation are mainly based on a raft of experiments using the *WS-DREAM* dataset [30]. And the dataset records the throughput values of 5825 services monitored by 339 users around the world. To ensure the quality of the prediction, 99% of entries are removed randomly from the dataset, i.e. the density of the throughput matrix is 1%. In addition, 10 original hash functions are generated randomly, i.e. s = 10; the number of vectors in each new hash function, t = 2, 3, 4, 5; the number of hash tables, T = 4, 6, 8, 10; the number of new hash functions, k = 4, 6, 8, 10.

To estimate the feasibility of our proposed method in aspects of recommendation accuracy and efficiency, we compare the following four criteria between different methods, respectively:

(1) MAE: indicates the average distance between the predicted QoS and actual QoS.

(2) RMSE: reflects the degree of difference between the predicted value and the real value of web services.

(3) Number of similar users: we use this criteria to evaluate the number of neighbors returned by different approaches. Usually, we consider that there is an association between the number of neighbors of target user and MAE.

(4) Time costs: we use this criteria to evaluate the efficiency of the three approaches.

In addition, we would compare our proposed approach $SR_{Amplified\text{-}LSH}$ with two advanced approaches, *UPCC* [21] which is benchmark method and $SerRec_{distri\text{-}LSH}$ [16] which recruited LSH technique to improve accuracy of recommended results, to show the effectiveness and efficiency of our solution. Each experiment was run on a Dell computer with 2.80 GHz CPU and 4.0 GB memory. Software configurations include Windows 7 and Python 3. Each test is repeated 100 times and their average values are registered finally.

5.2 Evaluation Results.

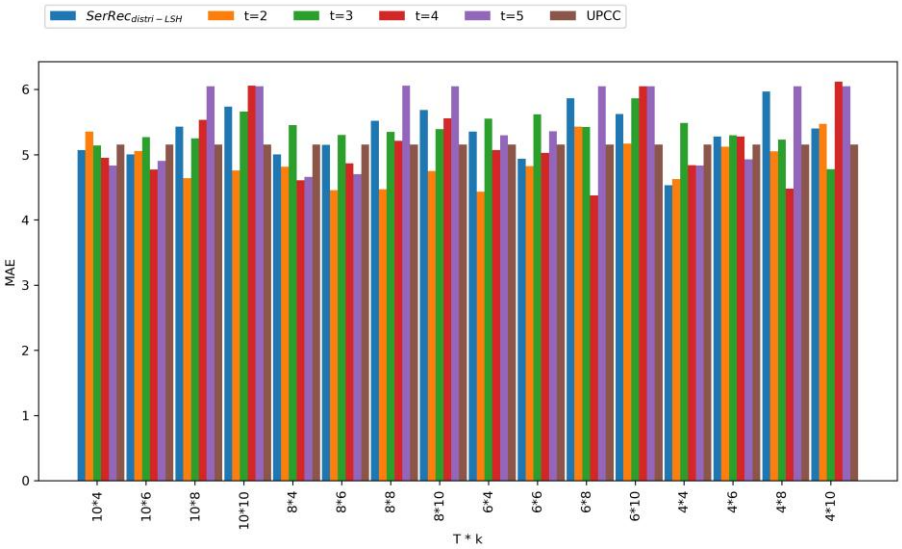
We evaluate the performance of our solution using the following four profiles.

***Profile*1**: The accuracy of recommendation results based on the three approaches

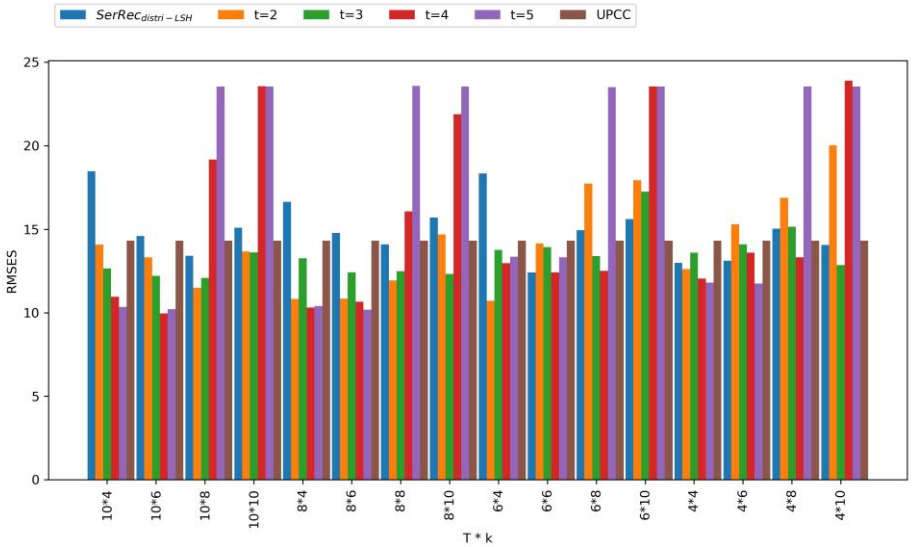
Accuracy is an important criteria to evaluate the performance of recommendation approaches and the degree of satisfaction of the users. Here, we evaluate the accuracy of our proposed method $SR_{Amplified\text{-}LSH}$ through the *MAE* and *RMSE* metrics (a smaller value is better) and compare them with those of $SerRec_{distri\text{-}LSH}$ and *UPCC* methods. In both the $SR_{Amplified\text{-}LSH}$ and $SerRec_{distri\text{-}LSH}$ approaches, the parameters T and k both vary from 4 to 10; in the $SR_{Amplified\text{-}LSH}$ approach, parameter t = 2, 3, 4, 5. The experimental results are presented in Figure. 3.

As Figure 3 shows, the *MAE* and *RMSE* of the *UPCC* approach remain unchanged though T-k pairs are changed, as the T and k parameters are not used in the *UPCC* approach. Though privacy-protection solutions are employed in the former two approaches to protect sensitive information of users, $SR_{Amplified\text{-}LSH}$ and $SerRec_{distri\text{-}LSH}$ generally perform better than *UPCC* in terms of accuracy with appropriate parameters. However, our $SR_{Amplified\text{-}LSH}$ approach often performs better than $SerRec_{distri\text{-}LSH}$ in terms of both *MAE* and *RMSE*, and its performance is better

than that of the benchmark *UPCC* approach if appropriate parameters (e.g., *T*, *k* and *t*) are selected. This is because we enhance the capability of LSH in searching for the neighboring users by selecting *t* vectors each time for hashing (Step 1). Therefore, according to the amplified LSH technique, $SR_{Amplified\text{-}LSH}$ an often identify the "most similar" neighbors of a target user and then produce accurate recommendation results accordingly.



(a) *MAE*



(b) *RMSE*

Figure 3. Comparison of accuracy of three approaches with MAE and RMSE

***Profile*2**: The number of neighbors of u_{target} returned by the three approaches

In user-based CF methods, the recommended item list to u_{target} is based on the neighbors of u_{target}. Therefore, the size of set $Neighbor^{\#}$ can also influence the final recommended results as well as the user satisfaction degree. Considering this, we compare the number of neighbors of u_{target} returned by the three approaches. In both the $SR_{Amplified\text{-}LSH}$ and $SerRec_{distri\text{-}LSH}$ approaches, parameters T and k both vary from 4 to 10; in the $SR_{Amplified\text{-}LSH}$ approach, parameter t = 2, 3, 4, 5. The experimental reports are presented in Figure. 4. Figure. 4 reveals that the size of the neighbor set of u_{target} in the *UPCC* approach stays stable relative to the *T*-*k* pairs, as the parameters T and k are not used in the *UPCC* approach. In addition, in the $SR_{Amplified\text{-}LSH}$ and $SerRec_{distri\text{-}LSH}$ approaches, the neighbors of u_{target} decrease when T increases or when k decreases. The reason is that a larger T or a smaller k often indicates a tighter filtering condition for the neighbor search; as a consequence, fewer neighbors of u_{target} would be returned. Our proposed $SR_{Amplified\text{-}LSH}$ approach returns fewer neighbors of u_{target} than the $SerRec_{distri\text{-}LSH}$ approach as the amplified LSH technique used in $SR_{Amplified\text{-}LSH}$ can narrow the neighbor search condition compared to the traditional LSH technique employed in $SerRec_{distri\text{-}LSH}$. Furthermore, in $SR_{Amplified\text{-}LSH}$, a larger t often indicates a narrower neighbor filtering condition and as a result, outputs fewer neighbors of u_{target}.

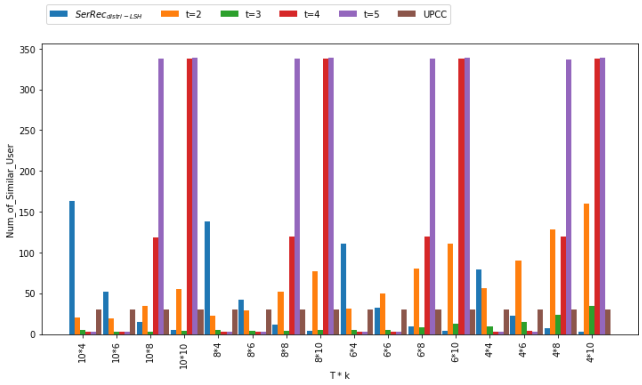
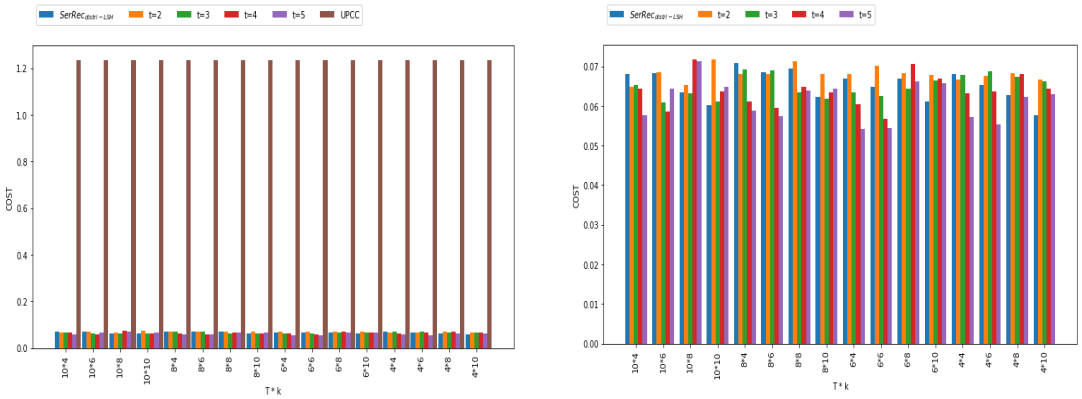


Figure 4.Comparison of number of similar users on three approaches

***Profile*3**: Time costs of the three approaches

We measure the efficiency of the three approaches in this profile. In $SR_{Amplified\text{-}LSH}$ and $SerRec_{distri\text{-}LSH}$, the parameters T and k both vary from 4 to 10; in $SR_{Amplified\text{-}LSH}$, parameter t = 2, 3, 4, 5. The experimental results are presented in Figure 5. As Figure 5 (a) shows, the time cost of *UPCC* remains stable though the *T*-*k* pair changes; $SR_{Amplified\text{-}LSH}$ and $SerRec_{distri\text{-}LSH}$ both achieve better efficiency than *UPCC* as the user indices can be generated offline in these two LSH-based approaches. In addition, less computational time is needed in $SR_{Amplified\text{-}LSH}$ than in $SerRec_{distri\text{-}LSH}$, as the amplified LSH technique adopted in $SR_{Amplified\text{-}LSH}$ often guarantees the return of fewer neighbors of u_{target} to use in the subsequent recommendation process. Furthermore, as presented in Figure 5 (b), in $SR_{Amplified\text{-}LSH}$, when T increases or k decreases, fewer neighbors are returned to u_{target}; and less computational time is needed as a result.



(a) time costs of three approaches

(b) time costs of $SR_{Amplified\text{-}LSH}$ and $SerRec_{distri\text{-}LSH}$

Figure 5. Comparison of time costs of three approaches

Profile4: Performance of $SR_{Amplified\text{-}LSH}$ with varying values of parameters T, k and t

We measure the variation tendency of the various performance measures (*MAE*, *RMSE*, number of neighbors of u_{target}) of $SR_{Amplified\text{-}LSH}$ with different parameter combinations of T, k and t. In the experiments, T and k both vary from 4 to 10, and t is varied from 2 to 5. Experimental results are presented in table 2. From the table, we will clearly notice the tendency of variation under different parameter settings. Obviously, the performance of $SR_{Amplified\text{-}LSH}$ also varies with changes in parameters T, k and t. We will found in the table that the smallest value of MAE is 4.43 when T, k and t are 6, 4, 2 respectively. Although the differences between different parameter settings may not be large, we could still find the optimal parameter setting according the results.

6. RELATED WORK

The historical quality data of service contain partial user privacy; therefore, it is crucial to make sure that the process of recommendation will not cause the leakages of user privacy. At present, there are several common privacy-preserving methods existing in the field of service recommendation such as partial disclosure, anonymity, encryption, disturbance, decomposition and LSH.

6.1 Partial disclosure

Dou et al. [10] suggested that users only disclose a small amount of optimal quality data they monitored to achieve user privacy protection in service composition recommendation. However, “a small amount of optimal QoS data” cannot objectively reflect true quality level of a service, and a small amount of published quality data will still reveal part of user's privacy. Zheng et al. [11] utilizes “the amount of quality data which users need to expose” as an adjustable parameter, and then models user privacy protection problem as a multi-objective optimization problem with NP-hard complexity, in order to obtain a better compromise between data availability and privacy; However, this method still leads to expose a small amount of quality data; therefore, it may also reveal user privacy.

6.2 Anonymity

Generalization or anonymity is one of the privacy protection strategies commonly used in the field of data security [12,13]. Casino et al. [14] implements K-anonymity of data through

microaggregation technology, protecting users' privacy in the process of service recommendation. Memon et al. [15] uses K anonymity to generalize and blur the user's location information, so as to protect the user's location privacy as much as possible while recommending services to users. However, although abovementioned methods can protect users' sensitive information, the usability of anonymous data usually decreases. Therefore, using anonymous data for service recommendation cannot guarantee high-quality recommendation results.

6.3 Encryption

Shu et al. [22] uses polynomial function to encrypt, match and recommend important data of users and service providers (such as user needs, service functions, etc.) in order to realize privacy-free task outsourcing. Ahila et al. [23] uses homomorphic encryption to protect sensitive service QoS data while reducing the cost of encryption/decryption. However, as a heavy-weight data protection method, the computational cost and time overhead of encryption operations are often high, which is not suitable for light-weight recommendation requirements of some users.

6.4 Disturbance

Zhu et al. [28] used randomized perturbation technology to confuse the original service QoS data and then calculated user similarity and recommended services based on the confused QoS data to achieve a better compromise between recommendation accuracy and data privacy. However, this method is generally aimed at collaborative service recommendation based on Pearson similarity. The scope of application is relatively limited. Differential privacy technology is used by Li et al. [25] and Zhu et al. [26] to inject and confuse sensitive QoS data with noise, and then use noisy QoS data to recommend services, so as to ensure that real QoS data will not leak out in the recommendation process. However, the time complexity of differential privacy algorithm is relatively high; in addition, if the service's QoS data is updated frequently, it will increase the cumulative noise, which will reduce the availability of service's QoS data, and then affect the accuracy of recommendation results.

6.5 Decomposition

Li et al. [27] adopted the "decomposition-merge" mechanism to randomly decompose a QoS data into several segments, and distribute each segment to different users for custody. Then, the recommendation system merges multiple segments held by each user to calculate user similarity and recommend services. In the recommendation process, each user can only hold part of the information of a certain quality of service data, but can not know all of its information, so as to achieve the purpose of privacy protection; however, this method will still disclose some of the privacy information of users in the recommendation process, such as: the intersection of services invoked by two users together.

6.6 LSH

LSH is an effective method for fast neighbor search in massive high-dimensional data and has been gradually applied in the field of collaborative service recommendation in recent years. Qi et al. [16-18] combines LSH with "user-based collaborative filtering", converts user-sensitive quality data into LSH hash value (i.e. user index) with low privacy (or even no privacy) and then searches for similar friends of target users quickly and efficiently according to the user index table

generated offline, and makes service recommendation. Similarly, Zhang et al. [19] introduced LSH into "item-based collaborative filtering" to construct low-privacy/no-privacy service index tables offline, and then quickly and accurately recommend services based on service index tables. Yan et al. [40] introduced LSH technique to protect the private QoS data with big range. And Chen et al. [41] took the technique to alleviate the cold-start problems in recommendations. However, the above research work is only a preliminary attempt by researchers to use LSH for service recommendation-privacy protection, and still faces many unsolved scientific problems.

Based on above analyses, we can make a conclusion that current researches fall short in dealing with problems involving users' privacy in distributed environment. Considering these abovementioned shortcomings, a unique amplified LSH-based approach named $SR_{Amplified\text{-}LSH}$ which will be described in the section 4.

7. CONCLUSION

We propose an effective recommendation approach based on amplified LSH technique, i.e., $SR_{Amplified\text{-}LSH}$, to deal with the challenges in the distributed environment. Through $SR_{Amplified\text{-}LSH}$, the similar users of a particular target user are decreased dramatically. As a result, the recommendation process can be accelerated significantly. In addition, through amplified LSH technique, users' privacy are protected very well. Finally, we prove the feasibility of our proposal via a variety of experiments conducted on *WS-DREAM* dataset. Through our proposed $SR_{Amplified\text{-}LSH}$ approach, the recommendation accuracy is improved significantly. In conclusion, our proposed method can protect user privacy while guarantee the accuracy and efficiency in distributed recommendation.

8. FUTURE WORK

In the future, we are supposed to study further the theory of LSH and try our best to figure out current issues. In our paper, there is still existing shortcomings described as follows. First of all, since LSH technique is a probabilistic method, so we supposed to study the relevant mathematics knowledge of probability in order to deal with some situations performing worse than compared methods and improve the accuracy of recommendation results. Besides, due to the inherent shortcoming of LSH, currently we cannot evaluate or quantify the privacy-preservation effects of LSH directly when performing LSH-based service recommendation. So we need to further refine our work by providing more objective privacy measurement manners. Last but not least, we are supposed to further refine our proposal by introducing more factors, such as efficiency consumption and time delay in work [37-38].

REFERENCES

1. 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.
2. Lianyong Qi, Ruili Wang, Shancang Li, Qiang He, Xiaolong Xu, Chunhua Hu. Time-aware Distributed Service Recommendation with Privacy-preservation. Information Sciences, 480: 354-364, 2019.
3. 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.
4. Lianyong Qi, Yi Chen, Yuan Yuan, Shucun Fu, Xuyun Zhang, Xiaolong Xu, A QoS-Aware Virtual Machine Scheduling Method for Energy Conservation in Cloud-based Cyber-Physical Systems. World Wide Web

Journal, 2019, DOI: 10.1007/s11280-019-00684-y.

5. Wenwen Gong, Lianyong Qi, Yanwei Xu. Privacy-aware Multidimensional Mobile Service Quality Prediction and Recommendation in Distributed Fog Environment. Wireless Communications and Mobile Computing, vol. 2018, Article ID 3075849, 8 pages, 2018.
6. Lianyong Qi, Wanchun Dou, Wenping Wang, Guangshun Li, Hairong Yu, Shaohua Wan. Dynamic Mobile Crowdsourcing Selection for Electricity Load Forecasting. IEEE ACCESS, 6: 46926-46937, 2018.
7. 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.
8. Yanwei Xu, Lianyong Qi, Wanchun Dou, Jiguo Yu. Privacy-preserving and Scalable Service Recommendation based on SimHash in A Distributed Cloud Environment. Complexity, Volume 2017, Article ID 3437854, 9 pages, 2017.
9. Lianyong Qi, Xuyun Zhang, Wanchun Dou, Qiang Ni. A Distributed Locality-Sensitive Hashing based Approach for Cloud Service Recommendation from Multi-Source Data. IEEE Journal on Selected Areas in Communications, 35(11): 2616-2624, 2017.
10. Wanchun Dou, Xuyun Zhang, Jianxun Liu and Jinjun Chen. HireSome-II: Towards Privacy-aware Cross-cloud Service Composition for Big Data Applications. IEEE Transactions on Parallel and Distributed Systems, 26(2): 455-466, 2015.
11. Xu Zheng, Zhipeng Cai, Jianzhong Li, Hong Gao. Location-Privacy-Aware Review Publication Mechanism for Local Business Service Systems. IEEE International Conference on Computer Communications (INFOCOM'17), 2017.
12. A. Machanavajjhala, J. Gehrke, D. Kifer, M. Venkitasubramaniam. l-diversity: Privacy beyond K-anonymity. International Conference on Data Engineering (ICDE' 06), 2006.
13. Ninghui Li, Tiancheng Li and Suresh Venkatasubramanian. t-Closeness: Privacy Beyond k-Anonymity and l-Diversity. International Conference on Data Engineering (ICDE' 07), 2007.
14. Fran Casino, Josep Domingo-Ferrer, Constantinos Patsakis, Domènec Puig and Agusti Solanas. A K-anonymous Approach to Privacy Preserving Collaborative Filtering. Journal of Computer and System Sciences, 81(6): 1000-1011, 2015.
15. Imran Memon. Authentication User's Privacy: An Integrating Location Privacy Protection Algorithm for Secure Moving Objects in Location Based Services. Wireless Personal Communications, 82(3): 1585-1600, 2015.
16. Lianyong Qi, Wanchun Dou, Xuyun Zhang, and Shui Yu: Amplified Locality-Sensitive Hashing for Privacy-Preserving Distributed Service Recommendaiton. SpaCCS 2017, LNCS 10656, pp. 280-297, 2017.
17. Lianyong Qi, Shunmei Meng, Xuyun Zhang, Ruili Wang, Xiaolong Xu, Zhili Zhou, Wanchun Dou. An Exception Handling Approach for Privacy-preserving Service Recommendation Failure in A Cloud Environment. Sensors, 18(7): 1-11, 2018.
18. Lianyong Qi, Xuyun Zhang, Wanchun Dou, Chunhua Hu, Chi Yang, Jinjun Chen. A Two-stage Locality-Sensitive Hashing Based Approach for Privacy-Preserving Mobile Service Recommendation in Cross-Platform Edge Environment. Future Generation Computer Systems, 88: 636-643, 2018.
19. Kunpeng Zhang, Shaokun Fan and Harry Jiannan Wang. An Efficient Recommender System Using Locality Sensitive Hashing. The 51th Annual Hawaii International Conference on System Sciences (HICSS'18), 2018.
20. Xiaolong Xu, Yuan Xue, Yuan Yuan, Lianyong Qi, Xuyun Zhang, Tariq Umer, Shaohua Wan. An Edge Computing-Enabled Computation Offloading Method with Privacy Preservation for Internet of Connected Vehicles. Future Generation Computer Systems, vol. 96, pp. 89-100, 2019.
21. Lee Rodgers, Joseph, and W. Alan Nicewander. Thirteen Ways to Look at the Correlation Coefficient. The

American Statistician, 42(1): 59-66, 1988

22. Jiangang Shu, Xiaohua Jia, Kan Yang and Hua Wang. Privacy-Preserving Task Recommendation Services for Crowdsourcing. IEEE Transactions on Services Computing, 2018. DOI: 10.1109/TSC.2018.2791601.
23. S. Sobitha Ahila and K. L. Shunmuganathan. Role of Agent Technology in Web Usage Mining: Homomorphic Encryption Based Recommendation for E-commerce Applications. Wireless Personal Communications, 87(2): 499–512, 2016.
24. Xiaolong Xu, Shucun Fu, Lianyong Qi, Xuyun Zhang, Qingxiang Liu, Qiang He, Shancang Li. An IoT-Oriented data placement method with privacy preservation in cloud environment. Journal of Network and Computer Applications, vol. 124, pp. 148-157, 2018.
25. Chao Li, Balaji Palanisamy and James Joshi. Differentially Private Trajectory Analysis for Points-of-Interest Recommendation. IEEE International Congress on Big Data (BigData Congress' 17), 2017.
26. Tianqing Zhu, Gang Li, Wanlei Zhou, Ping Xiong and Cao Yuan. Privacy-preserving Topic Model for Tagging Recommender Systems. Knowledge and Information Systems, 46(1): 33-58, 2017.
27. Dongsheng Li, Chao Chen, Qin Lv, Li Shang, Yingying Zhao, Tun Lu and Ning Gu. An Algorithm for Efficient Privacy-preserving Item-based Collaborative Filtering. Future Generation Computer Systems, 55: 311-320, 2016.
28. 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 (ICWS' 15), 2015.
29. Xiaolong Xu, Yuancheng Li, Tao Huang, Yuan Xue, Kai Peng, Lianyong Qi, Wanchun Dou. An Energy-Aware Computation Offloading Method for Smart Edge Computing in Wireless Metropolitan Area Networks, Journal of Network and Computer Applications, vol. 133, pp. 75-85, 2019.
30. https://wsdream.github.io/ (accessed on 1st December 2018)
31. Xiaolong Xu, Qingxiang Liu, Yun Luo, Kai Peng, Xuyun Zhang, Shunmei Meng, Lianyong Qi. A Computation Offloading Method over Big Data for IoT-Enabled Cloud-Edge Computing, Future Generation Computer Systems, vol. 95, pp. 522-533, 2019.
32. Xiaolong Xu, Yi Chen, Yuan Yuan, Tao Huang, Xuyun Zhang, Lianyong Qi，Blockchain-based cloudlet management for multimedia workflow in mobile edge computing, Multimedia Tools and Applications, 2019, DOI: 10.1007/s11042-019-07900-x.
33. Xiaokang Wang, Laurence T. Yang, Hongguo Li, Man Lin, Jianjun Han, Bernady O. Apduhan, "NQA: A Nested Anti-Collision Algorithm for RFID Systems", ACM Transactions on Embedded Computing Systems, Vol.18, no. 4, doi：10.1145/3330139, 2019.
34. Gionis, A., Indyk, P., Motwani, R.: Similarity search in high dimensions via hashing. VLDB 99(6), 518–529 , 1999
35. Xiaokang Wang, Laurence T. Yang, Liwei Kuang, Xingang Liu, Qingxia Zhang and M. Jamal Deen, "A Tensor-based Big Data-Driven Routing Recommendation Approach for Heterogeneous Networks", IEEE Network Magazine, Vol. 33, no.1, pp64-69, 2019.
36. Anand Rajaraman and Jeffrey David Ullman. Mining of massive datasets. Vol. 77. Cambridge University Press Cambridge, 2012.
37. Qiang He, Guangming Cui, Xuyun Zhang, Feifei Chen, Shuiguang Deng, Hai Jin, Yun Yang, A Game-Theoretical Approach for User Allocation in Edge Computing Environment, IEEE Transactions on Parallel and Distributed Systems, DOI: 10.1109/TPDS.2019.2938944, 2019.
38. Phu Lai, Qiang He, Mohamed Abdelrazek, Feifei Chen, John Hosking, John Grundy, and Yun Yang, Optimal Edge User Allocation in Edge Computing with Variable Sized Vector Bin Packing, 16th International Conference on Service-Oriented Computing, pp. 230-245, Hangzhou, China, 2018.

39. Xiaokang Wang, Laurence T. Yang, Xia Xie, Jirong Jin, and M. Jamal Deen, “A Cloud-Edge Computing Framework for Cyber-Physical-Social Services,” IEEE Communications Magazine, vol.55, no.11, pp. 80-85, 2017.
40. Chao Yan, Xuening Chen, Qinglei Kong: LSH-based private data protection for service quality with big range in distributed educational service recommendations. EURASIP J. Wireless Comm. and Networking 2019: 92. 2019
41. Xuening Chen, Hanwen Liu, Dan Yang: Improved LSH for privacy-aware and robust recommender system with sparse data in edge environment. EURASIP J. Wireless Comm. and Networking 2019: 171. 2019

Table 1. Symbol Definitions

Symbols	*Definition*
$pf_1, \ldots, pf_z$	Platforms that store QoS data
$u_1, \ldots, u_m$	Historical users who invoke web services
$ws_1, \ldots, ws_n$	Candidate web services hosted in all platforms
$A, \quad B$	Two points distributed in space
s	The number of hash functions contained in original LSH family
t	The number of hash functions contained in a member of new LSH family
u_{target}	A target user
$d_{(A, B)}$	Distance between A and B
$f(.)$	Original hash function
d_1, d_2, p_1, p_2	LSH parameters
k	The number of members in a new LSH family
$h(.)$	New hash functions
T	The number of LSH tables

Table 2. Performance variation of $SR_{Amplified}$ with respect to *T*, *k* and *t*

T*k	MAE				RMSE				Num_of_Similar_Users			
	t=2	t=3	t=4	t=5	t=2	t=3	t=4	t=5	t=2	t=3	t=4	t=5
10*4	5.35	5.14	4.95	4.83	14.08	12.66	10.96	10.34	20	5	3	2
10*6	5.06	5.27	4.77	4.91	13.32	12.21	9.95	10.21	20	3	2	2
10*8	4.64	5.25	5.53	6.05	11.50	12.06	19.18	23.54	34	3	118	338
10*10	4.76	5.66	6.06	6.05	11.66	13.61	23.56	23.54	55	4	338	339
8*4	4.81	5.45	4.61	4.66	10.82	13.27	10.31	10.39	23	5	3	2
8*6	4.45	5.30	4.87	4.70	10.83	12.41	10.65	10.17	29	3	2	2
8*8	4.47	5.35	5.21	6.06	11.94	12.49	16.06	23.59	51	4	119	337
8*10	4.75	5.39	5.56	6.05	14.68	12.31	21.89	23.54	78	5	337	339
6*4	4.43	5.55	5.07	5.30	10.71	13.76	12.97	13.35	31	5	3	2
6*6	4.82	5.62	5.03	5.36	14.15	13.92	12.41	13.32	50	5	3	2
6*8	5.43	5.42	4.38	6.05	17.73	13.39	12.50	23.52	80	8	119	338
6*10	5.17	5.87	6.05	6.05	17.93	17.25	23.54	23.54	111	12	338	339
4*4	4.63	5.49	4.84	4.83	12.62	13.60	12.04	11.80	56	9	3	2
4*6	5.12	5.30	5.28	5.43	15.29	14.10	13.59	11.74	90	15	3	2
4*8	5.05	5.23	4.48	6.05	16.88	15.15	13.31	23.55	128	24	120	337
4*10	5.47	4.78	6.12	6.05	20.03	12.85	23.90	23.54	160	34	338	339