A Distributed Locality-Sensitive Hashing-based Approach for Cloud Service Recommendation from Multi-Source Data

Lianyong Qi*, Xuyun Zhang, Wanchun Dou* and Qiang Ni

***Abstract*—To maximize the economic benefits, a cloud service provider needs to recommend its services to as many users as possible based on the historical user-service quality data. However, when a cloud platform (e.g., Amazon) intends to make a service recommendation decision, considering only its own user-service quality data is insufficient because a cloud user may invoke services from multiple distributed cloud platforms (e.g., Amazon and IBM). In this situation, it is promising for Amazon to collaborate with other cloud platforms (e.g., IBM) to utilize the integrated data for the service recommendation to improve the recommendation accuracy. However, two challenges are present in the above collaboration process, where we attempt to use multi-source data for the service recommendation. First, protecting users' privacy is challenging when IBM releases its own data to Amazon. Second, the recommendation efficiency and scalability are often low when the user-service quality data of Amazon and IBM update frequently. Considering these challenges, a privacy-preserving and scalable service recommendation approach based on distributed locality-sensitive hashing (LSH), i.e., *SerRec*$_{distri\text{-}LSH}$, is proposed in this paper to handle the service recommendation in a distributed cloud environment. Extensive experiments on the *WS-DREAM* dataset validate the feasibility of our approach in terms of service recommendation accuracy, scalability and privacy preservation.**

***Index Terms*—Distributed service recommendation, cloud platform, collaboration, privacy, scalability.**

I. Introduction

THE increasing number of cloud users and their activities in cloud platforms have produced a vast amount of precious historical data. Cloud service providers often hope to exploit the accumulated user-service quality data (i.e., users' experienced service quality) to assist in the decision of cloud service recommendation to provide better services or to maximize revenue. Specifically, a recommendation method, e.g., the seminal collaborative filtering (CF) [1], can be used directly on the data to find a target user's similar users (i.e., neighbors) based on historical user-service quality data.

The "pay-as-you-go" feature and dynamics of resource provision in cloud computing [2] enable a cloud user to employ multiple cloud services from different cloud providers to seek a more economical combination of cloud services, thereby resulting in multi-source user-service quality data of cloud users. Consequently, each cloud provider only has part of the entire data about cloud users, and if a cloud service provider makes recommendation decisions based on such an incomplete view of the data, the recommendation accuracy can be considerably compromised.

A possible approach is to integrate the data from each cloud platform together and execute the recommendation algorithms on the complete data. However, this approach is often infeasible in practical collaborations between companies even if they understand that the collaboration can lead to higher profits (for example, honest collaboration between the tourism-planning service of TripAdvisor and the flight-reservation service of United Airlines can benefit both companies). One fundamental reason is that companies seldom share the original data with each other due to conflicts of economic interests or user privacy concerns. Another reason is that the volumes of the original datasets often become increasingly massive with updates over time, and sharing of such large volumes of data with others is often impractical due to inefficiency. Considering these issues, a promising approach is to run the recommendation methods in a distributed manner on these multi-source big data. The entire data can be regarded as the union of a set of vertically partitioned sub datasets owned by each party. However, existing service recommendation methods fail to effectively and efficiently handle the problem of service recommendation from multi-source data.

In light of these challenges, we develop a new distributed locality-sensitive hashing (LSH) mechanism for service recommendation from multi-source data, and we propose a novel privacy-preserving and scalable service recommendation approach based on distributed LSH named *SerRec*$_{distri\text{-}LSH}$. Our *SerRec*$_{distri\text{-}LSH}$ can achieve a good tradeoff among service recommendation accuracy, scalability and privacy preservation in a distributed cloud environment.

In general, our contributions in this paper are three-fold.

(1) To the best of our knowledge, existing work has rarely considered the problem of service recommendation from multi-source data across different cloud platforms. In this paper, we formulate this important recommendation problem

L. Qi is with the School of Information Science and Engineering, Chinese Academy of Education Big Data, Qufu Normal University, Rizhao, China. Email: lianyongqi@gmail.com.

X. Zhang is with the Department of Electrical and Computer Engineering, University of Auckland, Auckland, New Zealand. Email: xuyun.zhang@auckland.ac.nz.

W. Dou is with the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China. Email: douwc@nju.edu.cn.

Q. Ni is with the School of Computing & Communications, Lancaster University, Lancaster, UK. Email: q.ni@lancaster.ac.uk.

This paper is partially supported by the National Key Research and Development Program of China (No. 2017YFB1001800), Natural Science Foundation of China (No. 61402258, No. 61672276), EU FP7 CROWN project under Grant PIRSES-GA-2013-610524 and UoA Faculty Research Development Fund (No. 3714668). Dou and Qi are corresponding authors.

Manuscript received xx; revised xxx.

and highlight its efficiency issues and privacy concerns, which are different from those in existing works.

(2) We propose a distributed LSH mechanism for the service recommendation problem from multi-source user-service quality data in the cloud environment to protect user privacy and improve the recommendation scalability in the cross-cloud collaboration process.

(3) We conduct a wide range of experiments on the *WS-DREAM* [3] dataset (a multi-source user-service quality dataset) to validate the feasibility of our approach. The experimental results show that our approach can achieve high improvements in service recommendation accuracy and efficiency while guaranteeing privacy preservation.

The remainder of this paper is organized as follows. Related work is briefly surveyed in Section II. In Section III, we formulate the cloud service recommendation problem from multi-source data and present our research motivation. In Section IV, we propose a distributed LSH-based service recommendation approach to achieve privacy-preserving and scalable service recommendation from multi-source data. In Section V, extensive experiments are conducted on the *WS-DREAM* dataset to validate the feasibility of our approach in terms of service recommendation accuracy, scalability and privacy preservation. Finally, in Section VI, we conclude the paper and indicate some directions for future work.

II. Related Work

Due to its domain-independent and easy-to-explain characteristics, collaborative filtering (CF) has become one of the most effective techniques in various service recommendation systems [4]–[16]. We briefly review the CF-based service recommendation approaches from three perspectives: recommendation accuracy, capability of privacy preservation and scalability.

A. Recommendation Accuracy

User-based CF and item-based CF are recruited for high-quality service recommendation in [4] and [5], respectively. To combine their advantages, a hybrid CF recommendation approach is proposed in [6]. The experimental results show that the hybrid approach significantly improves the recommendation performance. Because the quality of a service often varies with the service invocation context (e.g., time and location), time-aware CF and location-aware CF are proposed in [7] and [8], respectively, to improve the recommendation accuracy. However, the above approaches only recruit objective service quality for recommendation and therefore fail to consider users' subjective preference, which also plays a crucial role in users' final service selection decisions. In light of this shortcoming, a preference-aware recommendation approach is proposed to satisfy the personalized service recommendation requirements from various users [9].

However, all the above CF approaches assume that the recommendation bases, i.e., user-service quality data, are centralized without considering the situations where quality data are from multiple sources in a distributed environment. Therefore, these approaches fail in handling the important problems of service recommendation from multi-source data produced by different cloud platforms.

B. Privacy Preservation

Protecting users' privacy is often regarded as a precondition of a successful service recommendation. An encryption technique is often used to protect user privacy in the domain of information retrieval, e.g., encryption-based multi-keyword search scheme for personalized user preferences in [17], keyword vector encryption scheme for privacy-preserving information search over cloud data in [18], and encrypted semantic search based on conceptual graphs in clouding computing in [19]. However, encryption often introduces considerable computation cost and is hence not suitable for light-weight service recommendation. To achieve light-weight privacy-preservation solutions, in [10], the authors suggest that a user should release only a small portion of user-service quality data to the public so that the remaining majority of the data are secure. However, the released small portion of data can still reveal part of a user's private information. To completely protect user privacy, the data obfuscation technique is employed in [11] to hide the real user-service quality data. However, as the data used to make service recommendations have been obfuscated, the recommendation accuracy often decreases accordingly. Similarly, a segment-based data hiding approach is developed in [12], where each piece of user-service quality data is divided into several data segments, and then the data segments are employed to calculate user similarity approximately and make further service recommendations. However, the work in [12] fails to protect some important privacy information appropriately, e.g., the information of the service intersection commonly invoked by two users. A privacy-preserving recommendation approach is proposed in our previous work [16]; however, this approach cannot handle the service recommendation scenario where the service quality data observed by a user are distributed in multiple platforms.

C. Scalability

Traditional CF approaches (e.g., user-based, item-based and hybrid CF) are designed based on the in-memory computing scheme, resulting in poor efficiency and scalability. Therefore, to improve the recommendation efficiency and scalability, a variety of model-based CF approaches have been proposed, such as matrix factorization-based CF [13], LDA-based approaches [14] and clustering-based approaches [15]. In these approaches, the recommendation process is divided into two phases: model training and recommendation. Because the first phase (i.e., model training) can be executed offline, the overall recommendation efficiency can be considerably improved. However, when the user-service quality data in each cloud platform update over time, the models need to be re-trained and updated frequently. Consequently, these approaches also fail to satisfy the users' quick response requirements due to their poor efficiency and scalability.

From the above analyses, we conclude that the existing research work falls short in handling the problems of privacy-preserving and scalable cloud service recommendation from

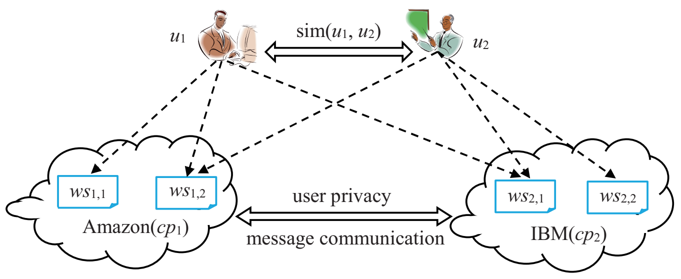


Fig. 1. Cloud service recommendation from multi-source data: an example

multi-sourced data. Considering these drawbacks, we propose a novel service recommendation approach based on distributed LSH, as elaborated in the next section.

III. Problem Formulation and Motivation

In this section, we first formulate the problem of cloud service recommendation from multi-source user-service quality data. Then, an intuitive example is presented to motivate our research. Table I recapitulates the symbols used in this paper.

A. Problem Formulation

Concretely, the problem of cloud service recommendation from multi-source data can be formulated as a five-tuple $\textit{Distri\_Ser\_Rec}(CP, WS, U, u_{target}, q)$, where

(1) $CP = \{cp_1, \ldots, cp_z\}$: $cp_k (1 \le k \le z)$ denotes the k-th cloud platform, which provides the k-th part of the user-service quality data of cloud users.

(2) $WS = \{WS_1, \ldots, WS_z\}$: $WS_k (1 \le k \le z)$ denotes the web service set in cloud platform cp_k. To ease the discussion, we assume that each platform owns m services, i.e., $WS_k = \{ws_{k,1}, \ldots, ws_{k,m}\}$, where $ws_{k,i} (1 \le i \le m)$ denotes the i-th web service in cloud platform cp_k.

(3) $U = \{u_1, \ldots, u_n\}$: $u_j (1 \le j \le n)$ denotes the j-th cloud user. Here, for user u_j, his/her user-service quality data are recorded by multiple cloud platforms in set CP and are hence multi-sourced.

(4) u_{target}: a target user to whom a cloud platform intends to recommend services. Here, $u_{target} \in U$ holds.

(5) q is a quality dimension of web services, e.g., *response time* and *throughput*. For simplicity, we subsequently only consider one quality dimension.

B. Research Motivation

We employ the example in Fig. 1 to demonstrate the motivation for our research herein. Assume that there are two cloud platforms, Amazon (denoted as cp_1) and IBM (denoted as cp_2), in Fig. 1. Two web services $ws_{1,1}$ and $ws_{1,2}$ are in Amazon, and another two services $ws_{2,1}$ and $ws_{2,2}$ are in IBM. Two cloud users u_1 and u_2 invoked services $\{ws_{1,1}, ws_{1,2}, ws_{2,1}\}$ and $\{ws_{1,2}, ws_{2,1}, ws_{2,2}\}$, respectively. Now, inspired by economic interests, Amazon and IBM agree to share their respective data with each other to attract more service users.

According to the traditional user-based CF recommendation approach, if Amazon intends to recommend its own services to u_2, the first step is to calculate the similarity between u_1 and u_2, i.e., $sim(u_1, u_2)$, based on the quality data of their invoked services (including $ws_{1,1}$ and $ws_{1,2}$ in Amazon and $ws_{2,1}$ and $ws_{2,2}$ in IBM). However, two challenges arise in the above collaboration process between Amazon and IBM. (1) Due to user privacy concerns, IBM cannot reveal its own user-service quality data to Amazon, which makes the collaboration process infeasible and renders the recommended results inaccurate. (2) For both Amazon and IBM, the volumes of their user-service quality data have become increasingly massive with updates over time. In this situation, the collaboration efficiency and scalability between Amazon and IBM are significantly reduced and cannot satisfy cloud users' quick response requirements.

Considering these challenges, we propose a privacy-preserving and scalable recommendation approach based on distributed LSH, named $\textit{SerRec}_{\textit{distri-LSH}}$, to handle the problem of cloud service recommendation from multi-source data, as elaborated in the following section.

IV. A Distributed Service Recommendation Approach from Multi-source Data: $\textit{SerRec}_{\textit{distri-LSH}}$

Our proposed cloud service recommendation approach $\textit{SerRec}_{\textit{distri-LSH}}$ is a new distributed LSH approach. In subsection IV.A, we first briefly introduce the LSH technique. Then, in subsection IV.B, we introduce the details of our proposal.

A. Locality-Sensitive Hashing

Locality-sensitive hashing (LSH) was introduced by Aristides Gionis in 1999 [20] and has been proven to be an effective approach for approximate nearest neighbor (ANN) search, such as the LSH-based privacy-preserving image content and feature protection approach in [21] and the LSH-based multi-keyword fuzzy search approach over encrypted outsourced data in [22]. The main idea of LSH is as follows: select a hashing function (or a hashing function family) such that (1) two neighboring points in the original data space are still neighbors after hashing with high probability and (2) two non-neighboring points in the original data space are still not neighbors after hashing with high probability.

If a hashing function satisfies the above two conditions, then it is called a LSH function. Formally, function $h(\cdot)$ is a LSH function iff both conditions in (1) and (2) hold. Here, x and y are two points in the original data space, $d(x, y)$ denotes the distance between x and y, $h(x)$ represents the hashing value (or index) of x, $P(X)$ denotes the probability that event X holds, and $\{d_1, d_2, p_1, p_2\}$ are a set of thresholds. If the conditions in (1) and (2) hold simultaneously, then LSH function $h(\cdot)$ is called (d_1, d_2, p_1, p_2)-sensitive.

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

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

We utilize the example in Fig. 2 to illustrate the basic process of LSH-based ANN search. Assume that there are n points $\{o_1, \ldots, o_n\}$ in the original data space. Then, through a LSH function $h(\cdot)$ (or a LSH function family), $\{o_1, \ldots, o_n\}$ are projected into corresponding buckets $b_1, \ldots, b_t$, where each bucket $b_i (1 \le i \le t)$ contains $n_i (n_i \ll n \text{ and } \sum n_i = n)$

neighboring data points. Thus, if a target user (denoted by data point X) intends to find his/her similar neighbors from $\{o_1, \ldots, o_n\}$, we can first calculate the hashing value (or index) $h(X)$ based on pre-selected hashing function $h(\cdot)$ and then determine the bucket (assume b_i) corresponding to index $h(X)$. Finally, according to the nature of LSH, all the n_i data points in bucket b_i could be regarded as similar neighbors of X (with high probability). Thus, the LSH-based ANN search process ends successfully.

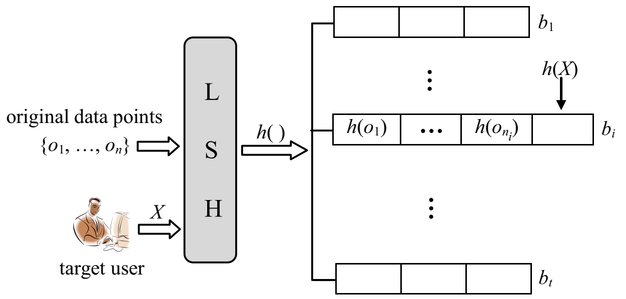


Fig. 2. LSH-based ANN search process

From the above example, we have observed three advantages of LSH. First, through LSH, distributed data points $\{o_1, \ldots, o_n\}$ could be integrated into a single hashing table, which enables the subsequent uniform computation. Second, original data o_i are transparent to the target user as he/she can only know the hashing value $h(o_i)$ with little (or null) privacy; therefore, user privacy is protected by LSH. Third, the hashing table can be constructed offline, through which the ANN search efficiency and scalability are significantly improved. Therefore, the LSH technique is extended in this paper to realize the privacy-preserving and scalable service recommendation from multi-source data in a distributed cloud environment.

B. *SerRec$_{distri\text{-}LSH}$: Service Recommendation based on Distributed LSH*

In this subsection, we propose a distributed LSH-based service recommendation approach called *SerRec$_{distri\text{-}LSH}$*. Generally, our proposal mainly consists of the four steps shown in Fig. 3. Here, u_{target} is the target user to whom a cloud platform intends to recommend services. Other symbols and their meanings can be found in Table 1.

Step 1: Building user sub-indices offline. For a cloud platform $cp_k (1 \leq k \leq z)$, a LSH function family $H_k()$ is chosen to hash each user $u (\in U)$ to be $H_k(u)$ offline based on u's user-service quality data in cp_k. Then, $H_k(u)$ is regarded as the k-th sub-index of u.

Step 2: Building user index by merging sub-indices offline. User u's sub-indices $H_k(u) (1 \leq k \leq z)$ derived in Step 1 are merged into a complete index $H(u) = (H_1(u), \ldots, H_z(u))$ offline. A hash table is created for users based on index $H(u)$.

Step 3: Online neighbor search for u_{target}. Calculate u_{target}'s index $H(u_{target})$ online based on Step 1 and Step 2. Then, find the bucket whose number is equal to $H(u_{target})$, and take the users in the bucket as u_{target}'s similar neighbors.

Step 4: Top-K service recommendation. According to u_{target}'s neighbors derived in Step 3, predict the quality of services never invoked by u_{target} and return the top-K services to u_{target}.

Fig. 3. Four steps of service recommendation approach *SerRec$_{distri\text{-}LSH}$*

TABLE I
SPECIFICATIONS OF SYMBOLS USED IN THIS PAPER

symbol	specification
z	number of cloud platforms
m	number of web services in each cloud platform
n	number of cloud users (or original data points)
q	a quality dimension of web services
$d(,)$	distance between two original data points (or users)
$P(.)$	probability value
d_1, d_2, p_1, p_2	thresholds recruited in LSH definition
$h(.)$	a LSH function
$H(.)$	a LSH function family
T	number of LSH tables
r_k	number of LSH functions chosen by cloud platform cp_k
$b_1, \ldots, b_t$	buckets in a hashing table
n_i	number of data points (or cloud users) in bucket b_i
X	profile that depicts a target user
β	density of user-service quality matrix

Step 1: Building user sub-indices offline.

Using an index is an effective approach for boosting the search speed in the information retrieval domain, e.g., tree-based keyword index for efficient information search over cloud data in [23]. Likewise, we build a user index for efficient service recommendation in this paper. Concretely, in this step, for each cloud platform $cp_k (1 \leq k \leq z)$, a LSH function family $H_k(\cdot)$ is chosen to build a sub-index for each user $u \in U$ offline based on u's experienced quality of services in cp_k (we assume that each cloud platform releases its data to other platforms honestly). Here, the selection of LSH function family $H_k(\cdot)$ depends on the "distance" type in the LSH definition (see subsection IV.A), while Pearson correlation coefficient (PCC) [24] is often used as the "distance" measurement in CF-based recommendation. Therefore, we need to determine the LSH functions $H_k(\cdot)$ corresponding to PCC, which mainly consists of the following three substeps (more intuitive specifications of these three substeps are presented in Fig. 4).

First, for a user u, his/her experienced quality over m services $\{ws_{k,1}, \ldots, ws_{k,m}\}$ in platform cp_k is transformed into an m-dimensional vector $\overrightarrow{u_{(k)}} = (ws_{k,1}.q, \ldots, ws_{k,m}.q)$, where q is a quality dimension of web services and $ws.q = 0$ if user u has never invoked service ws before.

Second, according to [25], the hash value of vector $\overrightarrow{u_{(k)}}$, i.e., $h_k(u)$ is calculated based on the LSH function in (3). Here, $\vec{v}$ is an m-dimensional vector $(v_1, \ldots, v_m)$, where $v_i (1 \leq i \leq m)$ is a random value in range $[-1, 1]$; symbol "$\circ$" represents the dot product between two vectors. To ease the understanding for readers, we explain the physical meaning of (3) as follows: vector $\vec{v}$ is considered as a hyperplane for space partition; if two vectors $\vec{x}$ and $\vec{y}$ are located on the same side of $\vec{v}$ (i.e., both $\vec{x} \circ \vec{v} > 0$ and $\vec{y} \circ \vec{v} > 0$ hold, or both $\vec{x} \circ \vec{v} \leq 0$ and

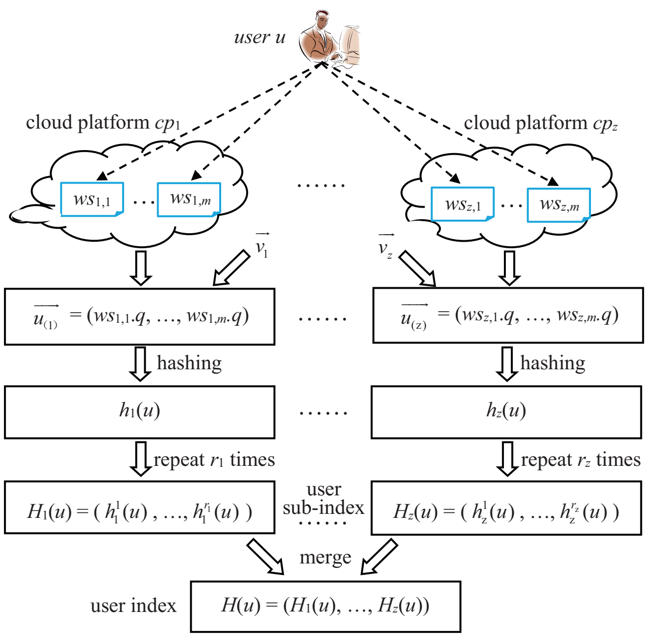


Fig. 4. Index building process for user u (i.e., Step 1 and Step 2)

$\vec{y} \circ \vec{v} \leq 0$ hold), then $\vec{x}$ and $\vec{y}$ could be regarded as similar (with high probability).

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

Third, repeat the above hash process r_k times based on different vectors $\vec{v}$ (generated randomly), and then the sub-index for user u, i.e., $H_k(u) = (h_k^1(u), \ldots, h_k^{r_k}(u))$, could be obtained, where $h_k^g(u)(1 \leq g \leq r_k)$ is calculated by (3). Here, sub-index $H_k(u)$ is actually an r_k-dimensional 0-1 vector, and the value of r_k should be "the larger the better" (LSH is essentially a probability-based search approach; therefore, more hash functions often mean higher search accuracy).

Step 2: Building user index by merging sub-indices offline.

In Step 1, we have derived z sub-indices $H_1(u), \ldots, H_z(u)$ of user u based on u's user-service quality data in cloud platforms $cp_1, \ldots, cp_z$, respectively. Next, we merge the z sub-indices into a complete index for u offline, i.e., $H(u) = (H_1(u), \ldots, H_z(u))$, which has $(r_1+r_2+\ldots+r_z)$ dimensions. Next, for each $u \in U$, we repeat the above process to build his/her index $H(u)$. Subsequently, the mapping relationships of "$u \rightarrow H(u)$" are recorded in a hash table *H_Table*.

Step 3: Online neighbor search for u_{target}.

According to the hash function family $H_k(.)(1 \leq k \leq z)$ chosen in Step 1 and the merging operation in Step 2, we calculate target user u_{target}'s index $H(u_{target})$ online. Then, we search for the bucket whose number is equal to $H(u_{target})$ from the hash table *H_Table* derived in Step 2. If a qualified bucket is found, then all the users in the bucket are considered as u_{target}'s similar neighbors and placed in the neighbor set *NB_Set*.

Otherwise, qualified buckets are not present; however, in this situation, we cannot simply conclude that u_{target} has no similar neighbors because LSH is actually a probability-based search approach and may overlook partial similar neighbors of u_{target}. Thus, we adopt the "OR" operation to relax the conditions for neighbor search to overlook as few neighbors as possible. Concretely, rather than creating only a hash table *H_Table* in Step 2, we repeat Step 1 and Step 2 to create T hash tables: $H\_Table_1, \ldots, H\_Table_T$. Then, if the condition in (4) holds, we can conclude that u_{target} has similar neighbors, and the users in the bucket whose number is equal to $H(u_{target})_x$ are neighbors of u_{target} and placed in set *NB_Set*.

$$\begin{aligned} &\exists u (\in U) \text{ and } x (\in \{1, \ldots, T\}), \\ &\text{satisfy } H(u)_x = H(u_{target})_x \text{ in } H\_Table_x \end{aligned} \tag{4}$$

Step 4: Top-K service recommendation.

In Step 3, we have obtained the similar neighbor set of target user u_{target}, i.e., *NB_Set*. Next, we utilize *NB_Set* to make service recommendations to u_{target}. Concretely, we predict service *ws*'s quality over dimension q by u_{target}, i.e., $ws.q_{target}$ based on (5). Here, *ws* is a service in *cp* (here, *cp* denotes the cloud platform that intends to recommend its own services to u_{target}) but never invoked by u_{target} before, and $ws.q_j$ denotes *ws*' quality over dimension q observed by user u_j. Finally, we rank all the candidate services in cloud platform *cp* by their predicted quality in (5) and return the optimal top-K services as the final recommendation results.

$$ws.q_{target} = \frac{1}{|NB\_Set|} * \sum_{u_j \in NB\_Set} ws.q_j \tag{5}$$

Through the above four steps of $SerRec_{distri\text{-}LSH}$, a cloud platform can recommend its optimal K (at most) services to a target user in a privacy-preserving and scalable manner. Formally, our proposal is specified by the following pseudocode.

Algorithm: $SerRec_{distri\text{-}LSH}$

Inputs: $CP = \{cp_1, \ldots, cp_z\}$: cloud platform set
$WS = \{WS_1, \ldots, WS_z\}$: service set in cloud platforms
$U = \{u_1, \ldots, u_n\}$: user set
q: a quality dimension of web services
u_{target}: a target user
cp: a cloud platform that intends to recommend services to u_{target}

Outputs: *Result_Set*: recommended results to u_{target}

```
/* Step 1: Building user sub-indices offline*/
1  for $k = 1$ to $z$ do
3    for $g = 1$ to $r_k$ do
4      for $i = 1$ to $m$ do
5        $h_{kgi}$ = random $[-1, 1]$
6      end for
7      $\vec{v_g} = (h_{kg1}, \ldots, h_{kgm})$
8      for $j = 1$ to $n$ do
9        $\overrightarrow{u_{j(k)}} = (ws_{k,1}.q_j, \ldots, ws_{k,m}.q_j)$
10       if $\overrightarrow{u_{j(k)}} \circ \vec{v_g} > 0$
11         then $h_k^g(u_j) = 1$
12         else $h_k^g(u_j) = 0$
13       end if
14     end for
15   end for
```

```
16     for $j = 1$ to $n$ do
17        $H_k(u_j) = (h_k^1(u), \ldots, h_k^{r_k}(u))$
18     end for
19 end for
/*Step 2: Building user index by merging sub-indices offline*/
20 for $j = 1$ to $n$ do
21     $H(u_j) = (H_1(u_j), \ldots, H_z(u_j))$
22 end for
23 create hash table $H\_Table$ based on $H(u_1) \ldots H(u_n)$
/*Step 3: Online neighbor search for $u_{target}$ */
24 $NB\_Set = \Phi$
25 for $x = 1$ to $T$ do
26     repeat Step 1 and Step 2 to create $H\_Table_x$
27     find bucket $b$ corresponding to $H(u_{target})_x$
28     if $b \neq Null$
29        then place users in $b$ into $NB\_Set$
30     end if
31 end for
/*Step 4: Top-K service recommendation */
32 $Result\_Set = \Phi$
33 for each service $ws$ in $cp$ do
34     if $ws.q_{target} = 0$
35        then count = 0
36           for $u_j \in NB\_Set$ do
37              if $ws.q_j \neq 0$
38              then count++
39                 $ws.q_{target} = ws.q_{target} + ws.q_j$
40              end if
41           end for
42           $ws.q_{target} = ws.q_{target}$ /count
43     end if
44 end for
45 place Top-K services with highest $ws.q_{target}$ into $Result\_Set$
46 return $Result\_Set$ to $u_{target}$
```

V. Experiments

In this section, a set of experiments are conducted based on the distributed dataset *WS-DREAM* [3] to validate the feasibility of our proposed service recommendation approach *SerRec*$_{distri\text{-}LSH}$. *WS-DREAM* is a real-world service quality set (e.g., *response time* values) obtained from 339 users on 5825 web services from different countries. In our experiments, the two countries that own the most services, i.e., USA and Canada, are used to simulate two geographically distributed cloud platforms, while the services hosted in other countries are eliminated (actually, the number of cloud platforms herein does not substantially affect the final service recommendation result because the user-service quality data in each cloud platform are hashed independently offline). Moreover, only one quality dimension of services, i.e., *response time*, is considered, and the top-3 (at most) services are recommended to each randomly selected target user.

To demonstrate the advantages of our proposal, we compare *SerRec*$_{distri\text{-}LSH}$ with four state-of-the-art approaches: *UPCC* [26], *IPCC* [27], *P-UIPCC* [11] and *PPICF* [12]. Furthermore, the following two evaluation measures are examined and compared (because user privacy can be protected well by the intrinsic nature of LSH, we will not evaluate the privacy-preservation capability of our proposal here).

(1) *Time cost*: time consumed for generating service recommendation results, through which we can test the recommendation efficiency and scalability.

(2) *MAE* (mean absolute error) [28]: average difference between predicted quality and real quality of recommended services, through which we can test the recommendation accuracy.

The experiments are conducted on a Lenovo laptop with 2.40 GHz processors and 12.0 GB of RAM. The machine runs Windows 10, JAVA 8 and MySQL 5.7. Each experiment was performed 10 times, and the average experimental results are reported.

A. *Experimental Results and Analyses*

Concretely, five profiles are tested and compared in our experiments. Here, as specified in Table 1, r_1 and r_2 denote the number of LSH functions chosen by cloud platforms cp_1 (USA) and cp_2 (Canada), respectively; T represents the number of hash tables created in Step 3; and β denotes the density of the user-service quality matrix.

***Profile* 1: accuracy comparison of the five approaches**

In this profile, the recommendation accuracy values of the five approaches are tested and compared. The experimental parameters are set as follows: $T = 10$, $r_1 = 8$, $r_2 = 6$, and β is varied from 5% to 25%. The experimental results are presented in Fig. 5.

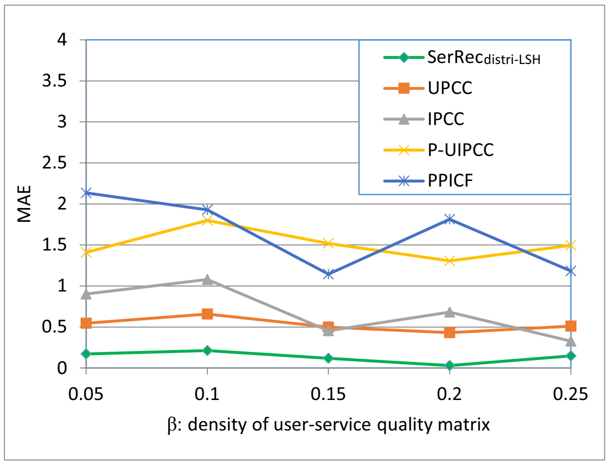


Fig. 5. Recommendation accuracy comparison

As shown in Fig. 5, the recommendation accuracy values of the five approaches all increase (i.e., *MAE* values all decrease) with increasing β (i.e., density of the user-service quality matrix). This result occurs because more useful recommendation information is available when the matrix becomes denser. Moreover, the recommended results in both the *P-UIPCC* and *PPICF* approaches are not sufficiently accurate because many approximate operations are recruited in these two approaches to protect user privacy. Our proposed *SerRec*$_{distri\text{-}LSH}$ approach outperforms the other four approaches in terms of recommendation accuracy because only the most similar neighbors of a target user could be found and utilized for service recommendation based on the nature of LSH.

***Profile* 2: efficiency comparison of the five approaches**

In this profile, we compare the efficiencies of the five recommendation approaches. The experimental parameters are set as follows: $T = 10$, $r_1 = 8$, $r_2 = 6$, and β is varied from 5% to 25%. The concrete experiment results are presented in Fig. 6.

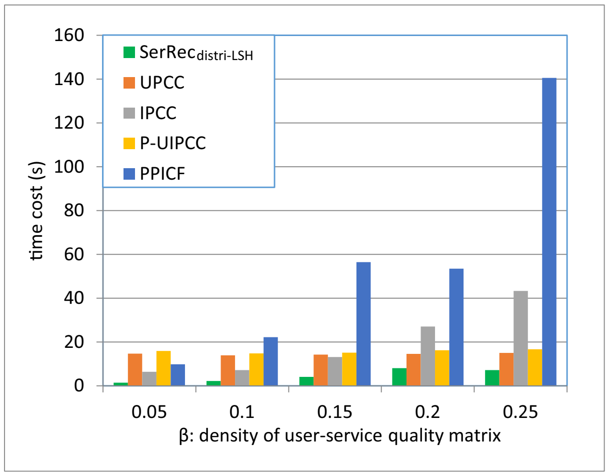


Fig. 6. Recommendation efficiency comparison

As shown in Fig. 6, the time cost of the *PPICF* approach is the highest in most cases (except when $\beta = 5\%$) because the data segment division and merging operations in *PPICF* require substantial computational time. Because there are a total of 339 users and 5825 web services in *WS-DREAM*, the number of services increases faster than the number of users when the user-service quality matrix becomes denser (i.e., when β increases). Therefore, the time cost of *IPCC* (actually a type of item-based CF approach) increases faster than those of the other two user-based CF approaches, i.e., *UPCC* and *P-UIPCC*. Our proposed *SerRec*$_{distri\text{-}LSH}$ approach outperforms the other four approaches in terms of computational time because most of the jobs (e.g., hash table creation) in *SerRec*$_{distri\text{-}LSH}$ could be finished offline, while the time complexity of the remaining job (i.e., online neighbor search) is almost $O(1)$ [29].

Profile* 3: recommendation accuracy of *SerRec*$_{distri\text{-}LSH}$ w.r.t. *r

In our *SerRec*$_{distri\text{-}LSH}$ approach, the number of hash functions in each hash table, i.e., r, plays an important role in service recommendation accuracy because more hash functions often mean stricter conditions for finding the similar neighbors of a target user. Thus, in this profile, we test the recommendation accuracy of our proposal with respect to r. The experimental parameters are set as follows: T is varied from 6 to 14, $r_1 = r_2 = r$ while r is varied from 4 to 7 (as larger r may induce recommendation failures), and $\beta = 25\%$. The experimental results are presented in Fig. 7.

As shown in Fig. 7, the *MAE* values of our proposal remain increasingly stable with increasing r (e.g., when $r = 7$). This result occurs because a larger r value often means a

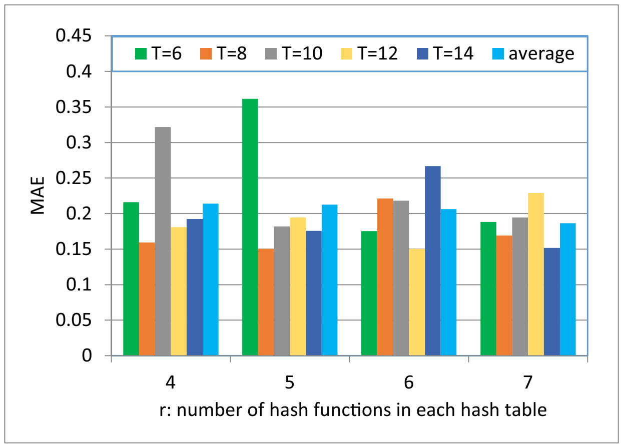


Fig. 7. Recommendation accuracy of *SerRec*$_{distri\text{-}LSH}$ w.r.t. r

stricter condition for neighbor search; consequently, only a few fixed and "really similar" neighbors of a target user could be found and utilized for subsequent service recommendation, and therefore, the recommendation accuracy (i.e., *MAE*) does not fluctuate much. For the same reason, the recommendation accuracy increases (i.e., *MAE* decreases) when r increases, which can also be observed from the "average" columns in Fig. 7.

Profile* 4: recommendation efficiency of *SerRec*$_{distri\text{-}LSH}$ w.r.t. *r

In this profile, we test the recommendation efficiency of our *SerRec*$_{distri\text{-}LSH}$ approach with respect to r. The experimental parameters are set as follows: T is varied from 6 to 14, $r_1 = r_2 = r$ while r is varied from 4 to 7, and $\beta = 25\%$. The experimental results are presented in Fig. 8.

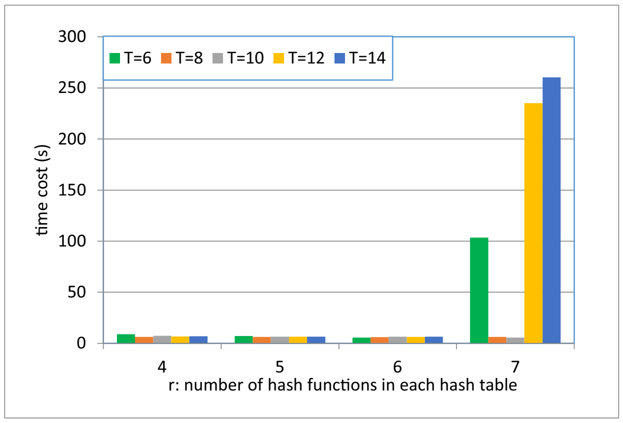


Fig. 8. Recommendation efficiency of *SerRec*$_{distri\text{-}LSH}$ w.r.t. r

As shown in Fig. 8, the time costs of our proposal are small and remain approximately stable when r is small (e.g., when $r = 4$, 5, and 6). This result occurs because most jobs (e.g., hash table creation) can be finished offline, while the time complexity of remaining jobs (i.e., online neighbor search) is almost $O(1)$. However, as shown in Fig. 8, the time cost significantly increases when r is varied from 6 to 7. This is because when r is larger (e.g., when $r = 7$), the condition for neighbor search becomes stricter, and hence, it is probable that no similar neighbors could be found through one iteration process in our approach. In this situation, multiple iterations are needed to find at least one qualified neighbor, and hence, more computational time is consumed.

***Profile* 5: number of outputted similar neighbors in *SerRec*$_{\textit{distri-LSH}}$ w.r.t. T and r**

In Step 3 of our proposed *SerRec*$_{\textit{distri-LSH}}$ approach, we have derived the similar neighbors of the target user and placed them in set *NB_Set*. In this profile, we test the relationship between the size of set *NB_Set* and parameters T and r (here, T and r represent the number of hash tables and the number of hash functions in each hash table, respectively). The experimental parameters are set as follows: T is varied from 6 to 14, r is varied from 4 to 7, and $\beta = 25\%$. The concrete experimental results are shown in Fig. 9.

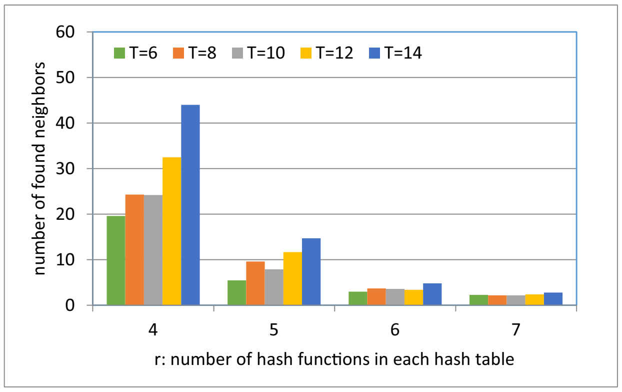


Fig. 9. Number of outputted neighbors w.r.t. T and r

As shown in Fig. 9, the number of outputted neighbors quickly decreases with increasing r; this is because a larger r value often means a stricter filtering condition for neighbor search, and consequently, only fewer qualified neighbors are finally obtained. Moreover, the number of outputted neighbors increases approximately when parameter T is varied from 6 to 14; this is because a larger T value means a looser condition for neighbor search (see the "OR" operation in condition (4)) under which more candidate users become "qualified" neighbors of the target user.

B. Time Complexity Analyses

In this subsection, we analyze the time complexity of our proposed cloud service recommendation approach *SerRec*$_{\textit{distri-LSH}}$. Suppose that there are n users and z cloud platforms, each platform contains m services, and T and r represent the number of hash tables and number of hash functions in each hash table, respectively. Next, we discuss the time costs of the four steps in *SerRec*$_{\textit{distri-LSH}}$.

In Step 1 and Step 2, the user index (including user sub-indices) building process can be completed offline; therefore, the time costs of these two steps are not considered in the complexity analyses.

In Step 3, we first build the z sub-indices for the target user, whose time complexity is $O(m*r)$ as the z sub-indices can be built by z cloud platforms in parallel; subsequently, the obtained z sub-indices are merged to be a complete index, whose time complexity is $O(1)$. Therefore, the time complexity of building an index for the target user is $O(m*r)$. Furthermore, because a total of T indices are necessary for a target user, the time complexity of Step 3 is $O(m*r*T)$.

In Step 4, we evaluate the quality of each candidate service (at most m candidates) based on the service quality data observed by similar neighbors (at most $n-1$ neighbors) in Step 3. Therefore, the time complexity of Step 4 is $O(m*n)$.

With the above analyses, we can conclude that the time complexity of our proposed *SerRec*$_{\textit{distri-LSH}}$ approach is $O(m*(r*T+n))$. The polynomial time complexity means that the recommendation efficiency and scalability of our proposal are often high, which has already been validated by the experimental results in subsection V.A.

VI. Conclusions

The pay-as-you-go feature of cloud platforms has enabled cloud users to employ multiple cloud services from different service providers to reduce costs for their business. This leads to the problem of cloud service recommendation from the users' behavioral data distributed across multiple cloud platforms. This problem is of both practical and theoretical importance as it is promising to use the multi-source data to improve the service recommendation performance in a distributed environment, while we need to protect user privacy to encourage data sharing between cloud providers. In this paper, we have proposed a novel service recommendation approach named *SerRec*$_{\textit{distri-LSH}}$ by developing a new distributed LSH scheme to handle the cloud service recommendation problems from multi-source data. With the use of LSH, user privacy has been protected in cross-cloud collaboration due to the inherent nature of hashing techniques. Furthermore, recommendation efficiency and scalability have been significantly improved as most computation in the recommendation process (e.g., hash table creation) can be finished offline. Finally, extensive experiments conducted on the multi-source service quality dataset *WS-DREAM* have validated the feasibility of our proposal in terms of recommendation accuracy, scalability and the capability of privacy preservation.

References

[1] Bo Yang, Yu Lei, Jiming Liu, and Wenjie Li. Social Collaborative Filtering by Trust. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016.

[2] Yan Kong, Minjie Zhang and Dayong Ye. A belief propagation-based method for Task Allocation in Open and Dynamic Cloud Environments. *Knowledge-based Systems*, 115: 123-132, 2016.

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

[4] Huigui Rong, Shengxu Huo, Chunhua Hu, and Jinxia Mo. User Similarity-based Collaborative Filtering Recommendation Algorithm. *Journal on Communications*, 35(2): 16-24, 2014.

[5] Kyung-Yong Chung, Daesung Lee, and Kuinam J. Kim. Categorization for Grouping Associative Items Using Data Mining in Item-based Collaborative Filtering. *Multimedia Tools and Applications*, 71(2): 889-904, 2014.

[6] Cuiqing Jiang, Rui Duan, Hemant K. Jain, Shixi Liu, and Kun Liang. Hybrid Collaborative Filtering for High-involvement Products: A Solution to Opinion Sparsity and Dynamics. *Decision Support Systems*, 79: 195-208, 2015.

[7] 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.

[8] 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.

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

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

[12] 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.

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

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

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

[16] Lianyong Qi, Haolong Xiang, Wanchun Dou*, Chi Yang, Yongrui Qin and Xuyun Zhang. Privacy-preserving Distributed Service Recommendation based on Locality-Sensitive Hashing. *IEEE International Conference on Web Services*, 2017.

[17] Zhangjie Fu, Kui Ren, Jiangang Shu, Xingming Sun, and Fengxiao Huang. Enabling Personalized Search over Encrypted Outsourced Data with Efficiency Improvement. *IEEE Transactions on Parallel and Distributed Systems*, 27(9): 2546C2559, 2016.

[18] Zhihua Xia, Xinhui Wang, Xingming Sun, and Qian Wang. A Secure and Dynamic Multi-keyword Ranked Search Scheme over Encrypted Cloud Data. *IEEE Transactions on Parallel and Distributed Systems*, 27(2): 340-352, 2015.

[19] Zhangjie Fu, Fengxiao Huang, Xingming Sun, Athanasios V. Vasilakos, and Ching-Nung Yang. Enabling Semantic Search based on Conceptual Graphs over Encrypted Outsourced Data. *IEEE Transactions on Services Computing*, DOI: 10.1109/TSC.2016.2622697.

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

[21] Zhihua Xia, Xinhui Wang, Liangao Zhang, Zhan Qin, Xingming Sun, and Kui Ren. A Privacy-preserving and Copy-deterrence Content-based Image Retrieval Scheme in Cloud Computing. *IEEE Transactions on Information Forensics and Security*, 11(11): 2594-2608, 2016.

[22] Zhangjie Fu, Xinle Wu, Chaowen Guan, Xingming Sun and Kui Ren. Toward Efficient Multi-keyword Fuzzy Search over Encrypted Outsourced Data with Accuracy Improvement. *IEEE Transactions on Information Forensics and Security*, 11(12): 2706-2716, 2016.

[23] Zhangjie Fu, Xingming Sun, Qi Liu, Lu Zhou, and Jiangang Shu. Achieving Efficient Cloud Search Services: Multi-keyword Ranked Search over Encrypted Cloud Data Supporting Parallel Computing. *IEICE Transactions on Communications*, E98-B(1): 190-200, 2015.

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

[25] Yannis Ioannidis, Yannis Kotidis, Theofilos Mailis, Ralf Möller, Christian Neuenstadt, Özgür L. Özçep, and Christoforos Svingos. Data Mining and Query Log Analysis for Scalable Temporal and Continuous Query Answering, *http://www. optique-project. eu/*, 2015.

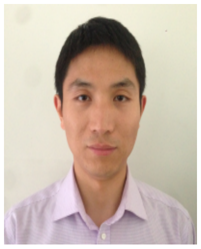
[26] Breese JS, Heckerman D, Kadie C. Empirical Analysis of Predictive Algorithms for Collaborative Filtering, 1998. John S. Breese, David Heckerman, and Carl Kadie. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. *International Conference on Uncertainty in Artificial Intelligence*,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

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

[28] D.G. Mayer and D.G. Butler. Statistical Validation. *Ecological Modelling*, 68(1): 21-32, 1993.

[29] Malcolm Slaney and Michael Casey. Locality-sensitive Hashing for Finding Nearest Neighbors. *IEEE Signal Processing Magazine*, 25(2): 128-131, 2008.

Lianyong Qi received his PhD degree in Department of Computer Science and Technology from Nanjing University, China, in 2011. Now, he is an associate professor of the School of Information Science and Engineering, Qufu Normal University, China. His research interests include big data, cloud computing and service computing.



Xuyun Zhang is a lecturer in the Department of Electrical & Computer Engineering at the University of Auckland, New Zealand. Prior to his current appointment, he worked as a postdoctoral fellow in the Machine Learning Research Group of NICTA (currently Data61, CSIRO) in Australia. He received his PhD degree from University of Technology, Sydney (UTS, Australia) in 2014, as well as his ME and BS degrees in Computer Science from Nanjing University (China) in 2011 and 2008, respectively. His primary research interests include IoT and smart cities, big data, cloud computing, scalable machine learning & data mining, data privacy & security, and Web service technology.

Wanchun Dou received his PhD degree in Mechanical and Electronic Engineering from Nanjing University of Science and Technology, China, in 2001. From 2001 to 2002, he did his postdoctoral research in the Department of Computer Science and Technology, Nanjing University, China. Now, he is a full professor of the State Key Laboratory for Novel Software Technology, Nanjing University, China. From Apr. 2005 to Jun. 2005 and from Nov. 2008 to Feb. 2009, he respectively visited Hong Kong University of Science and Technology, as a visiting scholar. His research interests include workflow, cloud computing, big data and service computing.

Qiang Ni (M'04CSM'08) received the B.Sc., M.Sc., and Ph.D. degrees from the Huazhong University of Science and Technology, China, all in engineering. He is a Professor and the Head of Communication Systems Research Group, School of Computing and Communications, Lancaster University, InfoLab21, Lancaster, U.K. His research interests include future generation communications and networking systems, including green communications and networking, cloud systems, cognitive radio network systems, heterogeneous networks, 5G, SDN, IoTs, big data analytics and vehicular networks in which areas he had already published more than 180 papers. He is a Voting Member of IEEE 1932.1 standard. He was an IEEE 802.11 Wireless Standard Working Group Voting member and a Contributor to the IEEE Wireless Standards.