IEEE TRANSACTIONS ON
EMERGING TOPICS
IN COMPUTING

Received 14 November 2013; revised 30 January 2014; accepted 25 February 2014. Date of publication 10 March 2014;
date of current version 30 October 2014.

Digital Object Identifier 10.1109/TETC.2014.2310485

ClubCF: A Clustering-Based Collaborative Filtering Approach for Big Data Application

RONG HU[1,2], (Member, IEEE), WANCHUN DOU[1], (Member, IEEE), and JIANXUN LIU[2], (Member, IEEE)

[1]State Key Laboratory for Novel Software Technology, Department of Computer Science and Technology, Nanjing University, Nanjing 210093, China

[2]Key Laboratory of Knowledge Processing and Networked Manufacturing, Hunan University of Science and Technology, Xiangtan 411201, China

CORRESPONDING AUTHOR: R. HU (ronghu@126.com)

This work was supported in part by the National Science Foundation of China under Grants 91318301 and 61321491, in part by the National Key Technology Research and Development Program of the Ministry of Science and Technology under Grant 2011BAK21B06, and in part by the Excellent Youth Found of Hunan Scientific Committee of China under Grant 11JJ1011.

ABSTRACT Spurred by service computing and cloud computing, an increasing number of services are emerging on the Internet. As a result, service-relevant data become too big to be effectively processed by traditional approaches. In view of this challenge, a clustering-based collaborative filtering approach is proposed in this paper, which aims at recruiting similar services in the same clusters to recommend services collaboratively. Technically, this approach is enacted around two stages. In the first stage, the available services are divided into small-scale clusters, in logic, for further processing. At the second stage, a collaborative filtering algorithm is imposed on one of the clusters. Since the number of the services in a cluster is much less than the total number of the services available on the web, it is expected to reduce the online execution time of collaborative filtering. At last, several experiments are conducted to verify the availability of the approach, on a real data set of 6225 mashup services collected from ProgrammableWeb.

INDEX TERMS Big data application, cluster, collaborative filtering, mashup.

I. INTRODUCTION

Big data has emerged as a widely recognized trend, attracting attentions from government, industry and academia [1]. Generally speaking, Big Data concerns large-volume, complex, growing data sets with multiple, autonomous sources. Big Data applications where data collection has grown tremendously and is beyond the ability of commonly used software tools to capture, manage, and process within a ''tolerable elapsed time'' is on the rise [2]. The most fundamental challenge for the Big Data applications is to explore the large volumes of data and extract useful information or knowledge for future actions [3].

With the prevalence of service computing and cloud computing, more and more services are deployed in cloud infrastructures to provide rich functionalities [4]. Service users have nowadays encounter unprecedented difficulties in finding ideal ones from the overwhelming services. Recommender systems (RSs) are techniques and intelligent applications to assist users in a decision making process where they want to choose some items among a potentially overwhelming set of alternative products or services. Collaborative filtering (CF) such as item- and user-based methods are the dominant techniques applied in RSs [5]. The basic assumption of user-based CF is that people who agree in the past tend to agree again in the future. Different with user-based CF, the item-based CF algorithm recommends a user the items that are similar to what he/she has preferred before [6]. Although traditional CF techniques are sound and have been successfully applied in many e-commerce RSs, they encounter two main challenges for big data application: 1) to make decision within acceptable time; and 2) to generate ideal recommendations from so many services. Concretely, as a critical step in traditional CF algorithms, to compute similarity between every pair of users or services may take too much time, even exceed the processing capability of current RSs. Consequently, service recommendation based on the similar users or similar services would either lose its timeliness or couldn't be done at all. In addition, all services

are considered when computing services' rating similarities in traditional CF algorithms while most of them are different to the target service. The ratings of these dissimilar ones may affect the accuracy of predicted rating.

A naïve solution is to decrease the number of services that need to be processed in real time. Clustering are such techniques that can reduce the data size by a large factor by grouping similar services together. Therefore, we propose a Clustering-based Collaborative Filtering approach (ClubCF), which consists of two stages: clustering and collaborative filtering. Clustering is a preprocessing step to separate big data into manageable parts [7]. A cluster contains some similar services just like a club contains some like-minded users. This is another reason besides abbreviation that we call this approach ClubCF. Since the number of services in a cluster is much less than the total number of services, the computation time of CF algorithm can be reduced significantly. Besides, since the ratings of similar services within a cluster are more relevant than that of dissimilar services [8], the recommendation accuracy based on users' ratings may be enhanced.

The rest of this paper is organized as follows. In Section II, A service BigTable is designed for storage requirement of ClubCF. It recruits BigTable and is capable of storing service-relevant big data in distribute and scalable manner. In Section III, ClubCF approach is described in detail step by step. First, characteristic similarities between services are computed by weighted sum of description similarities and functionality similarities. Then, services are merged into clusters according to their characteristic similarities. Next, an item-based CF algorithm is applied within the cluster that the target service belongs to. In Section IV, several experiments are conducted on a real dataset extracted from Programmable Web (http://www.programmableweb.com). Related work is analyzed in Section V. At last, we draw some conclusions and present some future work in Section VI.

II. PRELIMINARY KNOWLEDGE

To measure the similarity between Web services, Liu et al. [9] investigated the metadata from the WSDL (Web Service Description Language) files and defined a Web service as $= \langle N, M, D, O \rangle$, where N is the name that specifies a Web service, M is the set of messages exchanged by the operation invocation, D is the set of data types, and O is the set of operations provided by the Web service. From the definition, three types of metadata from WSDL can be identified for similarity matching: the plain textual descriptions, the operation that captures the purposed functionality and the data type relate to the semantic meanings. For evaluating reputation of service, Li et al. [10] defined a Web service as $WS(id, d, t, sg, rs, dor)$ where id is its identity, d is its text description, t is its classification, sg denotes the level of its transaction volume, rs is its review set, and dor is its reputation degree. In the SOA Solution Stack (S3) [11] proposed by IBM, a service is defined as an abstract specification of one or more business-aligned IT functions. This specification provides consumers with sufficient information to be able to invoke the business functions exposed by a service provider.

Although the definitions of service are distinct and application-specific, they have common elements which mainly include service descriptions and service functionalities. In addition, rating is an important user activity that reflects their opinions on services. Especially in application of service recommendation, service rating is an important element. As more and more services are emerging on the Internet, such huge volume of service-relevant elements are generated and distributed across the network, which cannot be effectively accessed by traditional database management system. To address this problem, Bigtable is used to store services in this paper. Bigtable [12] is a distributed storage system of Google for managing structured data that is designed to scale to a very large size across thousands of commodity servers. A Bigtable is a sparse, distributed, persistent multi-dimensional sorted map. The map is indexed by a row key, column key, and a timestamp; each value in the map is an uninterpreted array of bytes. Column keys are grouped into sets called column families, which form the basic unit of access control. A column key is named using the following syntax: family:qualifier, where 'family' refers to column family and 'qualifier' refers to column key. Each cell in a Bigtable can contain multiple versions of the same data which are indexed by timestamp. Different versions of a cell are stored in decreasing timestamp order, so that the most recent versions can be read first.

In this paper, all services are stored in a Bigtable which is called service Bigtable. The corresponding elements will be drawn from service Bigtable during the process of ClubCF. Formally, service Bigtable is defined as follow.

Definition 1: A service Bigtable is defined as a table expressed in the format of $< Service\_ID > < Timestamp >$

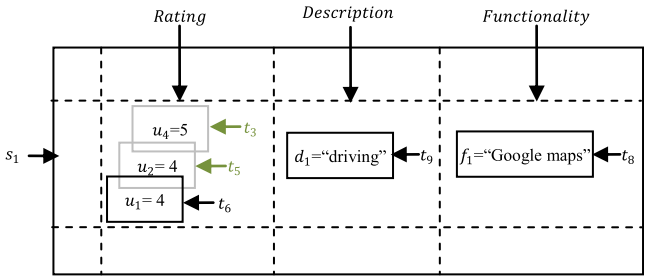
$$\begin{aligned}&\{< Description >: [< d_1 >, < d_2 >, \ldots];\\ &< Functionality > [:< f_1 >, < f_2 >, \ldots];\\ &< Rating >: [< u_1 >, < u_2 >, \ldots]\}\end{aligned}$$

The elements in the expression are specified as follows:

1. *Service_ID* is the row key for uniquely identifying a service.
2. *Timestamp* is used to identify time when the record is written in service Bigtable.
3. *Description*, *Functionality* and *Rating* are three column families.
4. The identifier of a description word, e.g. d_1 and d_2, is used as a qualifier of *Description*.
5. The identifier of a functionality e.g. f_1 and f_2 is used as a qualifier of *Functionality*.
6. The identifier of a user, e.g. u_1 and u_2 is used as a qualifier of *Rating*.

A slice of service Bigtable is illustrated in Table I. The row key is s_1. The *Description* column family contains the words for describing s_1, e.g., ''driving''. The *Functionality* column family contains the service functionalities,

TABLE 1. A slice of a service bigtable.



e.g., "Google maps". And the *Rating* column family contains the ratings given by some users at different time, e.g., "4" is a rating that "u_1" gave to "s_1" at timestamp "t_6".

III. A CLUSTERING-BASED COLLABORATIVE FILTERING APPROACH (ClubCF) FOR BIG DATA APPLICATION

According to **Definition 1**, a service could be expressed as a triple, $s = (D, F, R)$, where D is a set of words for describing s, F is a set of functionalities of s, R is a set of ratings some users gave to s. Five kinds of service similarities are computed based on D, F and R during the process of ClubCF, which are defined as follow.

Definition 2: Suppose $s_t = \langle D_t, F_t, R_t \rangle$ and $s_j = \langle D_j, F_j, R_j \rangle$ are two services. The similarity between s_t and s_j is considered in five dimensions which are description similarity $D\_sim(s_t, s_j)$, functionality similarity $F\_sim(s_t, s_j)$, characteristic similarity $C\_sim(s_t, s_j)$, rating similarity $R\_sim(s_t, s_j)$ and enhanced rating similarity $R\_sim'(s_t, s_j)$, respectively.

With this assumption, a ClubCF approach for Big Data application is presented, which aims at recommending services from overwhelming candidates within an acceptable time. Technically, ClubCF focuses on two interdependable stages, i.e., clustering stage and collaborative filtering stage. In the first stage, services are clustered according to their characteristic similarities. In the second stage, a collaborative filtering algorithm is applied within a cluster that a target service belongs to.

Concretely, Fig. 1 depicts the specification of the ClubCF approach step by step.

A. DEPLOYMENT OF CLUSTERING STAGE

Step 1.1: Stem Words

Different developers may use different-form words to describe similar services. Using these words directly may influence the measurement of description similarity. Therefore, description words should be uniformed before further usage. In fact, morphological similar words are clubbed together under the assumption that they are also semantically similar. For example, "map", "maps", and "mapping" are forms of the equivalent lexeme, with "map" as the morphological root form. To transform variant word forms to their common root called stem, various kinds of stemming algorithms, such as Lovins stemmer, Dawson Stemmer, Paice/Husk Stemmer, and Porter Stemmer, have been proposed [13]. Among them, Porter Stemmer (http://tartarus.org/martin/PorterStemmer/) is one of the most widely used stemming algorithms. It applies cascaded rewrite rules that can be run very quickly and do not require the use of a lexicon [14].

Stage 1: **Clustering stage**

Step 1.1: Stem the words in D_t and D_j using Porter Stemmer. The stemmed words in D_t are put into D_t' and the stemmed words in D_j are put into D_j'.

Step 1.2: Compute $D\_sim(s_t, s_j)$ and $F\_sim(s_t, s_j)$ using Jaccard similarity coefficient respectively.

Step 1.3: Compute $C\_sim(s_t, s_j)$ by weighted sum of $D\_sim(s_t, s_j)$ and $F\_sim(s_t, s_j)$. Create a matrix D each entry of which is a characteristic similarity.

Step 1.4: Cluster services according to their characteristic similarities in D using an agglomerative hierarchical clustering algorithm.

Stage 2: **Collaborative filtering stage**

Step 2.1: Compute $R\_sim(s_t, s_j)$ using Pearson correlation coefficient if s_t and s_j belong to the same cluster, and compute $R\_sim'(s_t, s_j)$ by weighting $R\_sim(s_t, s_j)$.

Step 2.2: Select services whose enhanced rating similarity with s_t exceed a rating similarity threshold γ, and put them into a neighbors set.

Step 2.3: Compute the predicted rating of s_t for an active user. If the predicted rating exceeds a recommending threshold, it will be recommended to the active user.

FIGURE 1. Specification of the ClubCF Approach.

In ClubCF approach, the words in D_t are gotten from service Bigtable where row key = "s_t" and column family = "*Description*". The words in D_j are gotten from service Bigtable where row key = "s_j" and column family = "*Description*". Then these words are stemmed by Porter Stemmer and put into D_t' and D_j', respectively.

Step 1.2: Compute Description Similarity And Functionality Similarity

Description similarity and functionality similarity are both computed by Jaccard similarity coefficient (JSC) which is a statistical measure of similarity between samples sets [15]. For two sets, JSC is defined as the cardinality of their intersection divided by the cardinality of their union. Concretely, description similarity between s_t and s_j is computed by formula (1):

$$D\_sim(s_t, s_j) = \frac{|D_t' \bigcap D_j'|}{|D_t' \bigcup D_j'|} \quad (1)$$

It can be inferred from this formula that the larger $\left|D_t^{'} \bigcap D_j^{'}\right|$ is, the more similar the two services are. $\left|D_t^{'} \bigcup D_j^{'}\right|$ is the scaling factor which ensures that description similarity is between 0 and 1.

The functionalities in F_t are gotten from service Bigtable where row key = "s_t" and column family = "*Functionality*". The functionalities in F_j are gotten from service Bigtable where row key = "s_j" and column family = "*Functionality*". Then, functionality similarity between s_t and s_j is computed using JSC as follow:

$$F\_sim(s_t, s_j) = \frac{\left|F_t \bigcap F_j\right|}{\left|F_t \bigcup F_j\right|} \tag{2}$$

Step 1.3: Compute Characteristic Similarity

Characteristic similarity between s_t and s_j is computed by weighted sum of description similarity and functionality similarity, which is computed as follow:

$$C\_sim(s_t, s_j) = \alpha \times D\_sim(s_t, s_j) + \beta \times F\_sim(s_t, s_j) \tag{3}$$

In this formula, $\alpha \in [0, 1]$ is the weight of description similarity, $\beta \in [0, 1]$ is the weight of functionality similarity and $\alpha + \beta = 1$. The weights express relative importance between these two.

Provided the number of services in the recommender system is n, characteristic similarities of every pair of services are calculated and form a $n \times n$ characteristic similarity matrix D. An entry $d_{t,j}$ in D represents the characteristic similarity between s_t and s_j.

Step 1.4: Cluster Services

Clustering is a critical step in our approach. Clustering methods partition a set of objects into clusters such that objects in the same cluster are more similar to each other than objects in different clusters according to some defined criteria.

Generally, cluster analysis algorithms have been utilized where the huge data are stored [16]. Clustering algorithms can be either hierarchical or partitional. Some standard partitional approaches (e.g., $K-$means) suffer from several limitations: 1) results depend strongly on the choice of number of clusters K, and the correct value of K is initially unknown; 2) cluster size is not monitored during execution of the K-means algorithm, some clusters may become empty ("collapse"), and this will cause premature termination of the algorithm; 3) algorithms converge to a local minimum [17]. Hierarchical clustering methods can be further classified into agglomerative or divisive, depending on whether the clustering hierarchy is formed in a bottom-up or top-down fashion. Many current state-of-the-art clustering systems exploit agglomerative hierarchical clustering (AHC) as their clustering strategy, due to its simple processing structure and acceptable level of performance. Furthermore, it does not require the number of clusters as input. Therefore, we use an AHC algorithm [18], [19] for service clustering as follow.

Assume there are n services. Each service is initialized to be a cluster of its own. At each reduction step, the two most similar clusters are merged until only $K(K < n)$ clusters remains.

Algorithm 1 AHC algorithm for service clustering

Input: A set of services $S = \{s_1, \ldots, s_n\}$,
a characteristic similarity matrix $D = \left[d_{i,j}\right]_{n \times n}$,
the number of required clusters K.
Output: $Dendrogram_k$ for $k = 1$ to $|S|$.

```
1. C_i = {s_i} , ∀i;
2. d_{C_i,C_j} = d_{i,j}, ∀ij;
3. for k = |S| down to K
4.     Dendrogram_k = {C_1, ..., C_k};
5.     lm = d_{C_i,C_j};
6.     C_l = Join (C_l, C_m);
7.     for each C_h ∈ S
8.         if C_h ≠ C_l and C_h ≠ C_m
9.             d_{C_l,C_h} = Average (d_{C_l,C_h}, d_{C_m,C_h}) ;
10:        end if
11:    end for
12.    S = S − {C_m} ;
13. end for
```

B. DEPLOYMENT OF COLLABORATIVE FILTERING STAGE

Up to now, item-based collaborative filtering algorithms have been widely used in many real world applications such as at Amazon.com. It can be divided into three main steps, i.e., compute rating similarities, select neighbors and recommend services.

Step 2.1: Compute Rating Similarity

Rating similarity computation between items is a time-consuming but critical step in item-based CF algorithms. Common rating similarity measures include the Pearson correlation coefficient (PCC) [20] and the cosine similarity between ratings vectors. The basic intuition behind PCC measure is to give a high similarity score for two items that tend to be rated the same by many users. PCC which is the preferred choice in most major systems was found to perform better than cosine vector similarity [21]. Therefore, PCC is applied to compute rating similarity between each pair of services in ClubCF. Provided that service s_t and s_j are both belong to the same cluster, PCC-based rating similarity [22] between s_t and s_j is computed by formula (4):

$$R\_sim(s_t, s_j) = \frac{\sum_{u_i \in U_t \bigcap U_j} \left(r_{u_i,s_t} - \overline{r}_{s_t}\right)\left(r_{u_i,s_j} - \overline{r}_{s_j}\right)}{\sqrt{\sum_{u_i \in U_t \bigcap U_j} \left(r_{u_i,s_t} - \overline{r}_{s_t}\right)^2} \sqrt{\sum_{u_i \in U_t \bigcap U_j} \left(r_{u_i,s_j} - \overline{r}_{s_j}\right)^2}} \tag{4}$$

Here, U_t is a set of users who rated s_t while U_j is a set of users who rated s_j, u_i is a user who both rated s_t and s_j, r_{u_i,s_t} is the rating of s_t given by u_i which is gotten from service Bigtable where row key = "s_t" and column key = "*Rating* : u_i," r_{u_i,s_j} is the rating of s_j given by u_i which is gotten from

service Bigtable where row key = "s_j" and column key = "*Rating* : u_i," $\overline{r}_{s_t}$ is the average rating of s_t, and $\overline{r}_{s_j}$ is the average rating of s_j. It should be noted that if the denominator of formula (4) is zero, we make 0, in order to avoid division by 0.

Although PCC can provide accurate similarity computation, it may overestimate the rating similarities when there are a small amount of co-rated services. To address this problem, the enhanced rating similarity [23] between s_t and s_j is computed by formula (5):

$$R\_sim'(s_t,\ s_j) = \frac{2 \times |U_t \cap U_j|}{|U_t| + |U_j|} \times R\_sim(s_t, s_j) \qquad (5)$$

In this formula, $|U_t \cap U_j|$ is the number of users who rated both service s_t and s_j, $|U_t|$ and $|U_j|$ are the number of users who rated service s_t and s_j, respectively. When the number of co-rated services is small, for example, the weight $\frac{2\times|U_t\cap U_j|}{|U_t|+|U_j|}$ will decrease the rating similarity estimation between these two users. Since the value of $\frac{2\times|U_t\cap U_j|}{|U_t|+|U_j|}$ is between the interval of [0,1] and the value of $R\_sim(s_t, s_j)$ is in the interval of $[-1, 1]$, the value of $R\_sim'(s_t, s_j)$ is also in the interval of $[-1, 1]$.

Step 2.2: Select Neighbors

Based on the enhanced rating similarities between services, the neighbors of a target service s_t are determined according to constraint formula (6):

$$N(s_t) = \left\{s_j | R\_sim'(s_t, s_j) > \gamma, s_t \neq s_j\right\} \qquad (6)$$

Here, $R\_sim'(s_t, s_j)$ is the enhanced rating similarity between service s_t and s_j computed by formula (5), γ is a rating similarity threshold. The bigger value of γ is, the chosen number of neighbors will relatively less but they may be more similar to the target service, thus the coverage of collaborative filtering will decrease but the accuracy may increase. On the contrary, the smaller value of γ is, the more neighbors are chosen but some of them may be only slightly similar to the target service, thus the coverage of collaborative filtering will increase but the accuracy would decrease. Therefore, a suitable γ should be set for the tradeoff between accuracy and coverage. While γ is assigned, s_j will be selected as a neighbor of s_t and put into the neighbor set $N(s_t)$ if $R\_sim'(s_t, s_j) > \gamma$.

Step 2.3: Compute Predicted Rating

For an active user u_a for whom predictions are being made, whether a target service s_t is worth recommending depends on its predicted rating. If $N(s_t) \neq \Phi$, similar to the computation formula proposed by Wu et al. [24], the predicted rating $P(u_a s_t)$ in an item-based CF is computed as follow:

$$P_{u_a,s_t} = \overline{r}_{s_t} + \frac{\sum_{s_j \in N(s_t)} \left(r_{u_a,s_j} - \overline{r}_{s_j}\right) \times R\_sim'\left(s_t, s_j\right)}{\sum_{s_j \in N(s_t)} R\_sim'\left(s_t, s_j\right)} \qquad (7)$$

Here, $\overline{r}_{s_t}$ is the average rating of s_t, $N(s_t)$ is the neighbor set of s_t, $s_j \in N(s_t)$ denotes s_j is a neighbor of the target service s_t, r_{u_a,s_j} is the rating that an active user u_a gave to s_j, $\overline{r}_{s_j}$ is the average rating of s_j, and $R\_sim'(s_t, s_j)$ is the enhanced rating similarity between service s_t and s_j computed using formula (5).

If the predicted rating of a service exceeds a recommending threshold, it will be a recommendable service for the active user. A service is generally rated on a five-point scale from 1 (very dissatisfied) to 5 (very satisfied). Therefore, we set the recommending threshold to 2.5 which is the median value of the max rating. All recommendable services are ranked in non-ascending order according to their predicted ratings so that users may discover valuable services quickly.

C. TIME COMPLEXITY ANALYSIS

The time complexity of ClubCF can be divided into two parts: 1) the offline cluster building; and 2) the online collaborative filtering.

There are two main computationally expensive steps in the AHC algorithm. The first step is the computation of the pairwise similarity between all the services. Provided the number of services in the recommender system is n, the complexity of this step is generally $O(n^2)$. The second step is the repeated selection of the pair of most similar clusters or the pair of clusters that best optimizes the criterion functionality. A naive way of performing this step is to re-compute the gains achieved by merging each pair of clusters after each level of the agglomeration, and select the most promising pair. During the lth agglomeration step, this will require $O((n-l)^2)$ time, leading to an overall complexity of $O(n^3)$. Fortunately, if the priority queue is implemented using a binary heap, the total complexity of delete and insert operations is $O((n-l)\log(n-l))$. The overall complexity over the $n-1$ agglomeration steps is $O(n^2 logn)$ [25].

Suppose there are m users and n services. The relationship between users and services is denoted by a $m \times n$ matrix. Each entry $r_{i,j}$ represents the rating of the user u_i on the service s_j. Then the time complexity of PCC-based item similarity measures is $O(nm^2)$ [26]. Fortunately, the number of service in a cluster is much less than the whole number of services. Suppose that the number of services in a cluster C_k is n_k and the number of users who rated at least one service in C_k is m_k, then the time complexity of similarity computation is $O(n_k m_k^2)$. If the number of the target service's neighbors reaches to the max value, the worst-case time complexity of item-based prediction is $O(n_k)$. Since $n_k \ll n$ and $m_k \ll m$, the cost of computation of ClubCF may decrease significantly. Through the analysis above, it can be inferred that ClubCF may meet the demand of real-time recommendation to some extent.

IV. EXPERIMENTS AND EVALUATION

A. EXPERIMENTAL BACKGROUND

To verify ClubCF, a mashup dataset is used in the experiments. Mashup is an ad hoc composition technology of Web applications that allows users to draw upon content retrieved from external data sources to create value-added

services [27]. Compared to traditional "developer-centric" composition technologies, e.g., BPEL (Business Process Execution Language) and WSCI (Web Service Choreography Interface), mashup provides a flexible and easy-of-use way for service composition on web [28]. Recently, "mashup" has become one of the hottest buzzwords in the area of web applications, and many companies and institutions provide various mashup solutions or re-label existing integration solutions as mashup tools. For example, HousingMaps (http://www.housingmaps.com) combines property listings from Craigslist (http://www.craigslist.org/) with map data from Google Maps (http://maps.google.com/) in order to assist people moving from one city to another and searching for housing offers. More interesting mashup services include Zillow (http://www.zillow.com/) and SkiBonk (http://www.skibonk.com/).

Manual mashup development requires programming skills and remains an intricate and time consuming task, which prevents the average user from programming own mashup services. To enable even inexperienced end users to mashup their web services, many mashup-specific development tools and frameworks have emerged [29]. The representative approaches of end user mashup tools include Google Mashup Editor, Yahoo Pipes, Microsoft Popfly, Intel Mash Maker, and IBM's QEDWiki [30]. These tools speed up the overall mashup development process, resulting in an explosion in the amount of mashup services available on the Internet. Meanwhile, a large number of mashup services are similar to each other, in their components and in the logic [31]. Over mashup-oriented big data, ClubCF is a suitable approach for recommending ideal mashup services for users.

The data for our experiments was collected from ProgrammableWeb, a popular online community built around user-generated mashup services. It provides the most characteristic collection [32]. The extracted data was used to produce datasets for the population of mashup services. The dataset included mashup service name, tags, and APIs used. As of Dec 2012, 6,225 mashup services and related information are crawled from this site, which are labeled with 20,936 tags among which 1,822 tags are different. And, 15,450 APIs are used by these mashup services among which 1,499 APIs are different in name. The tags are stemmed using Porter Stemmer algorithm and 1,608 different stems of tags are obtained.

Since there are very few ratings available by now, we generate pseudorandom integers in the range 0 to 5 as the ratings of mashup services. Assume there are 500 users that have rated some mashup services published on the website. Then the user-item matrix consists of 500 rows and 6,225 columns. In total, 50,000 non-zero ratings are generated. The sparsity level of the matrix is 98.39% (sparsity level = 1-50000/500 $*$ 6226 = 0.9839).

We add an empirical evaluation based on a well known statistical test, namely the l-fold cross validation [33]. The ratings records are split into l mutually exclusive subsets (the folds) of equal size. During each step, it is tested on fold and trained on the rest. The cross-validation process is then repeated l times, with each of the l subsets used exactly once as the validation data. In this paper, 5-fold cross validation is applied (i.e., $l = 5$). In order to distribute test data and training data over all clusters, 20% services of each cluster was included in test data and 80% of it was included in training data for each data split.

TABLE 2. The experimental environments.

	Host	Cluster	Private Cloud
Hard-ware	DELL OPTIPLEX380 machine with 2 Intel E5400, 2.69GHz processors and 2GB RAM.	**Masters** (18 nodes): Intel(R) Quad Core E5620 Xeon(R), 2.4Ghz/12M Cache, 24GB (6x4GB) 1333MHz Dual Ranked RDIM for 2 Processors, 2TB 3.5-inch 7.2K RPM SATA II Hard Drive **Slaves** (17 nodes): Intel(R) Quad Core E5620 Xeon(R), 2.4Ghz/12M Cache, 24GB (6x4GB) 1333MHz Dual Ranked RDIM for 2 Processors, 2TB 3.5-inch 7.2K RPM SATA II Hard Drive	1) 10 computing pool, 2) 20TB stor-age capacity 3) 8-core processor, 20G RAM, and 100GB disk for a user
Soft-ware	Windows XP and VC 6.0.	Redhat Enterpreise Linux Server 6.0, Java 1.6.0, and Hadoop-0.20.205.0	Four OS for selecting: 1) Windows server 2008, 2) Windows7, 3) Ubuntu Linux, and 4) Centos Linux; KVM virtualization software
Net-work	Broadcom NetXtreme 57xx Gigabit Controller, and the network bandwidth is 100Mbps.		

B. EXPERIMENTAL ENVIRONMENTS

The experiments are conducted in a hybrid network environment that consists of three local hosts. The mashup services recruited in the experiment are distributed among the three hosts. As listed in Table II, the cluster (HDFS: http://114.212.190.91:50070, and JobTracker: http://114.212.190.91:50030, Campus Network) consists of an 18 node cluster, with one master node and 17 slave nodes. Each node is equipped with two Intel(R) Quad Core E5620 Xeon(R) processors at 2.4GHz and 24GB RAM. For the master node, a 2TB disk is mounted while, for each slave node, two 2TB disks are equipped. The cluster runs under Redhat Enterprise Linux Server 6.0, Java 1.6.0 and Hadoop-0.20.205.0. For the private cloud (http://cs-cloud.nju.edu.cn), there is totally 20TB storage capacity in the system. A user can apply for an 8-core processor, 20G memory and 100GB disk [34].

V. THE EXPERIMENT ENVIRONMENTS

A. Experimental Case Study

According to ClubCF, experimental process is promoted by two stages as we specified in Section III: Clustering stage and collaborative filtering stage.

TABLE 3. Case of mahsup services.

No.	Name	APIs (F_i)	Tags (D_i)	Stemmed Tags ($D_i^{'}$)
s_1	4Wheelz RouteMate	Google Maps	driving, google, maps, streetview	drive, google, map, streetview
s_2	GuruLib	Amazon Product Advertising	books, library, videos	book, library, video
s_3	100 Destinations	Google Maps + Twitter	fun, mapping, photo, social, travel	fun, map, photo, social, travel
s_4	Anuncios Total	Google Maps + Twitter	ads, deadpool, shopping	ads, deadpool, shop
s_5	22books	Amazon Product Advertising	books, lists, shopping, social	book, list, shop, social
s_6	Favmvs	Google Search + MTV	deadpool, MTV, music, video	deadpool, MTV, music, video
s_7	FlickrCash	Flickr	photos, shopping	photo, shop

In the first stage, characteristic similarities between mashup services are first computed. Then, all mashup services are merged into K clusters using Algorithm 1. In the second stage, rating similarities between mashup services that belong to the same cluster are computed. As PCC may overestimate the rating similarities, enhanced rating similarities are calculated. Then some mashup services whose enhanced rating similarities with the target mashup service exceed a threshold are selected as neighbors of the target mashup service. At last, the predicted rating of the target mashup service is computed.

1) Deployment of Clustering Stage

Step 1.1: Stem Words

Generally, a mashup service s_i is described with some tags and functionalize with some APIs [35]. As an experimental case, seven concrete mashup services (i.e., s_1, s_2, s_3, s_4, s_5, s_6 and s_7) the corresponding tags and APIs are listed in Table III. APIs of s_i are put into F_i, tags of s_i are put into D_i. Tags in D_i are stemmed using Porter stemmer and put into $D_i^{'}$.

Step 1.2: Compute Description Similarity And Functionality Similarity

Description similarities between mashup services are computed using formula (1). For instance, there are one same stemmed tag (i.e., "book") among the six different stemmed tags in D_2 and D_5, therefore, $D\_sim(s_2, s_5) = \frac{|D_2^{'} \bigcap D_5^{'}|}{|D_2^{'} \bigcup D_5^{'}|} = \frac{1}{6}$.

Functionality similarities between mashup services are computed using formula (2). Since there is only one API (i.e., "Amazon Product Advertising") in F_2 and F_5, $F\_sim(s_2, s_5) = \frac{|F_2 \bigcap F_5|}{|F_2 \bigcup F_5|} = 1$.

Step 1.3: Compute Characteristic Similarity

Characteristic similarity is the weight sum of the description similarity and functionality similarity, which is computed using formula (3). Without loss of generality, the weight of description similarity α is set to 0.5. Then the characteristic similarity between s_2 and s_5 is computed as $C\_sim(s_2, s_5) = \alpha \times D\_sim(s_2, s_5) + (1 - \alpha) \times F\_sim(s_2, s_5) = 0.5 \times \frac{1}{6} + 0.5 \times 1 \cong 0.583$. It should be noted that all the computation results retain 3 digits after the decimal point, thereafter.

TABLE 4. Characteristic similarity matrix (keeping three decimal places).

	s_1	s_2	s_3	s_4	s_5	s_6	s_7
s_1	/	0	0.313	0.25	0	0	0
s_2	0	/	0	0	0.583	0.083	0
s_3	0.313	0	/	0.5	0.063	0	0.083
s_4	0.25	0	0.5	/	0.083	0.083	0.125
s_5	0	0.583	0.063	0.083	/	0	0.1
s_6	0	0.083	0	0.083	0	/	0
s_7	0	0	0.083	0.125	0.1	0	/

TABLE 5. Initial similarity matrix ($k = 7$).

	C_1	C_2	C_3	C_4	C_5	C_6	C_7
C_1	/	0	0.313	0.250	0	0	0
C_2	0	/	0	0	**0.583**	0.083	0
C_3	0.313	0	/	0.500	0.063	0	0.083
C_4	0.250	0	0.500	/	0.083	0.083	0.125
C_5	0	**0.583**	0.063	0.083	/	0	0.100
C_6	0	0.083	0	0.083	0	/	0
C_7	0	0	0.083	0.125	0.1	0	/

Characteristic similarities between the seven mashup services are all computed by the same way, and the results are shown in Table IV.

Step 1.4: Cluster Services

In this step, Algorithm 1 is processed in the specified order. Initially, the seven services $s_1 \sim s_7$ are put into seven clusters $C_1 \sim C_7$ one by one and the characteristic similarities between each pair of services in Table IV are assigned to similarity of the corresponding clusters. The highlighted data in Table V is the maximum similarity in the similarity matrix.

The reduction step of Algorithm 1 is described as follows.

Step 1. Search for the pair in the similarity matrix with the maximum similarity and merge them.

Step 2. Create a new similarity matrix where similarities between clusters are calculated by their average value.

Step 3. Save the similarities and cluster partitions for later visualization.

Step 4. Proceed with 1 until the matrix is of size K, which means that only K clusters remains.

TABLE 6. Algorithm 1: reduction step 1 ($k = 6$).

	C_1	(C_2, C_5)	C_3	C_4	C_6	C_7
C_1	/	0	0.313	0.250	0	0
(C_2, C_5)	0	/	0.032	0.042	0.042	0.050
C_3	0.313	0.032	/	**0.500**	0	0.083
C_4	0.250	0.042	**0.500**	/	0.083	0.125
C_6	0	0.042	0	0.083	/	0
C_7	0	0.050	0.083	0.125	0	/

TABLE 7. Algorithm 1: reduction step 2($k = 5$).

	C_1	(C_2, C_5)	(C_3, C_4)	C_6	C_7
C_1	/	0	**0.282**	0	0
(C_2, C_5)	0	/	0.037	0.042	0.050
(C_3, C_4)	**0.282**	0.037	/	0.042	0.104
C_6	0	0.042	0.042	/	0
C_7	0	0.050	0.104	0	/

Let $K = 3$ as the termination condition of Algorithm 1, the reduction steps are illustrated in Table VI∼Table IX.

As for reduction Step 1 as shown in Table VI, since the maximum similarity in the similarity matrix is d_{C_2,C_5}, C_2 and C_5 are merged into (C_2, C_5). And the similarity between (C_2, C_5) and other clusters is calculated by their average value. For example, $d_{(C_2,C_5),C_3} = (d_{C_2,C_3} + d_{C_5,C_3})/2 = (0 + 0.063)/2 \cong 0.032$.

As for reduction Step 2 as shown in Table VII, since the maximum similarity in the similarity matrix is d_{C_3,C_4}, C_3 and C_4 are merged into (C_3, C_4). And the similarity between (C_3, C_4) and other clusters is calculated by their average value. For example, $d_{(C_2,C_5),(C_3,C_4)} = (d_{(C_2,C_5),C_3} + d_{(C_2,C_5),C_4})/2 = (0.032 + 0.042)/2 = 0.037$.

As for reduction Step 3 as shown in Table VIII, since the maximum similarity in the similarity matrix is $d_{C_1,(C_3,C_4)}$, C_1 and (C_3, C_4) are merged into (C_1, C_3, C_4). And the similarity between (C_1, C_3, C_4) and other clusters is calculated by their average value. For example, $d_{(C_1,C_3,C_4),C_6} = (d_{C_1,C_6} + d_{(C_3,C_4),C_6})/2 = (0 + 0.042)/2 = 0.021$.

As for reduction Step 4 as shown in Table IX, since the maximum similarity in the similarity matrix is $d_{(C_1,C_3,C_4),C_7}$, (C_1, C_3, C_4) and C_7 are merged into (C_1, C_3, C_4, C_7). And the similarity between (C_1, C_3, C_4, C_7) and other clusters is calculated by their average value. For example, $d_{(C_1,C_3,C_4,C_7),(C_2,C_5)} = (d_{(C_1,C_3,C_4),(C_2,C_5)} + d_{C_7,(C_2,C_5)})/2 = (0.019 + 0.050)/2 \cong 0.035$.

Now, there are only 3 clusters remaining and the algorithm is terminated.

By using Algorithm 1, the seven mashup services are merged into three clusters, where s_2 and s_5 are merged into a cluster named C_1, $s_1 s_3$, s_4 and s_7 are merged into a cluster named C_2, and s_6 is separately merged into a cluster named C_3.

TABLE 8. Algorithm 1: reduction step 3 ($k = 4$).

	(C_1, C_3, C_4)	(C_2, C_5)	C_6	C_7
(C_1, C_3, C_4)	/	0.019	0.021	**0.052**
(C_2, C_5)	0.019	/	0.042	0.050
C_6	0.021	0.042	/	0
C_7	**0.052**	0.050	0	/

TABLE 9. Algorithm 1: reduction step 4 ($k = 3$).

	(C_1, C_3, C_4, C_7)	(C_2, C_5)	C_6
(C_1, C_3, C_4, C_7)	/	0.035	0.011
(C_2, C_5)	0.035	/	0.042
C_6	0.011	0.042	/

TABLE 10. Rating matrix.

	C_1		C_2				C_3
	s_2	s_5	s_1	s_3	s_4	s_7	s_7
u_1	5	4	4	3	3	1	0
u_2	1	1	4	5	4	4	2
u_3	4	4	1	2	0	2	3
u_4	5	4	5	5	5	1	5

TABLE 11. Rating similarities and enhanced rating similarities with s_4.

Mashup service pair	Rating similarity	Enhanced rating similarity
(s_4, s_1)	0.544	0.467
(s_4, s_3)	0.736	0.631
(s_4, s_7)	0	0

TABLE 12. Rating similarities and enhanced rating similarities with s_1.

Mashup service pair	Rating similarity	Enhanced rating similarity
(s_1, s_3)	0.839	0.839
(s_1, s_4)	0.544	0.467
(s_1, s_7)	-0.187	-0.187

2) Deployment of Collaborative Filtering Stage

Step 2.1: Compute Rating Similarity

Suppose there are four users (i.e., $u_1 u_2 u_3 u_4$) who rated

the seven mashup services. A rating matrix is established as Table X. The ratings are on 5-point scales and 0 means the user did not rate the mashup. As u_3 does not rate s_4 (a not-yet-experienced item), u_3 is regarded as an active user and s_4 is looked as a target mashup. By computing the predicted rating of s_4, it can be determined whether s_4 is a recommendable service for u_3. Furthermore, s_1 is also chosen as another target mashup. Through comparing the predicted rating and real rating of s_1, the accuracy of ClubCF will be verified in such case.

Since s_4 and s_1 are both belong to the cluster C_2, rating similarity and enhanced rating similarity are computed between mashup services within C_2 by using formula (4) and (5). The rating similarities and enhanced rating similarities between s_4 and every other mashup service in C_2 are listed in Table XI while such two kinds of similarities between s_1 and every other mashup service in C_2 are listed in Table XII.

Step 2.2: Select Neighbors

Rating similarity is computed using Pearson correlation coefficient which ranges in value from -1 to $+1$. The value of -1 indicates perfect negative correlation while the value of $+1$ indicates perfect positive correlation. Without loss of generality, the rating similarity threshold γ in formula (6) is set to 0.4.

Since the enhanced rating similarity between s_4 and s_1 is 0.467 (i.e., $R\_sim'(s_4, s_1) = 0.467$) and the enhanced rating similarity between s_4 and s_3 is 0.631 (i.e., $R\_sim'(s_4, s_3) = 0.631$), which are both greater than γ, s_1 and s_3 are chosen as the neighbors of s_4, i.e., $N(s_4) = \{s_1, s_3\}$.

Since the enhanced rating similarity between s_1 and s_3 is 0.839 (i.e., $R\_sim'(s_1, s_3) = 0.839$) and the enhanced rating similarity between s_1 and s_4 is 0.467 (i.e., $R\_sim'(s_1, s_4) = 0.467$), which are both greater than γ, s_3 and s_4 are chosen as the neighbors of s_1, i.e., $N(s_1) = \{s_3, s_4\}$.

Step 2.3: Compute Predicted Rating

According to formula (7), the predicted rating of s_4 for u_3, i.e., $P_{u_3,s_4} = 1.97$ and the predicted rating of s_1 for u_3, i.e., $P_{u_3,s_1} = 1.06$.

Thus, s_4 is not a good mashup service for u_3 and will not be recommended to u_3. In addition, as the real rating of s_1 given by user u_3 is 1 (see Table X) while its predicted rating is 1.06, it can be inferred that ClubCF may gain an accurate prediction.

B. Experimental Evaluation

To evaluate the accuracy of ClubCF, Mean Absolute Error (MAE), which is a measure of the deviation of recommendations from their true user-specified ratings, is used in this paper. As Herlocker et al. [36] proposed, MAE is computed as follow:

$$\text{MAE} = \frac{\sum_{i=1}^{n} \left| r_{a,t} - P(u_a, s_t) \right|}{n} \tag{8}$$

In this formula, n is the number of rating-prediction pairs, $r_{a,t}$ is the rating that an active user u_a gives to a mashup service s_t, $P(u_a, s_t)$ denotes the predicted rating of s_t for u_a.

In fact, ClubCF is a revised version of traditional item-based CF approach for adapting to big data environment. Therefore, to verify its accuracy, we compare the MAE of ClubCF with a traditional item-based CF approach (IbCF) described in [26]. For each test mashup service in each fold, its predicted rating is calculated based on IbCF and ClubCF approach separately.

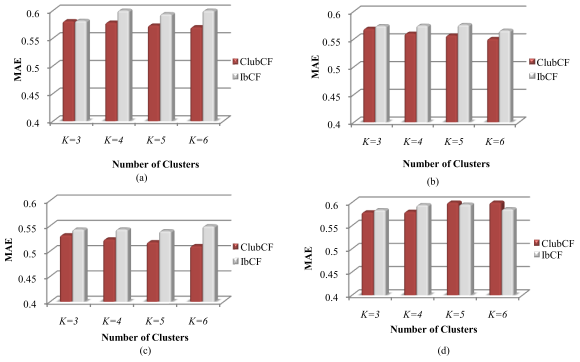


FIGURE 2. Comparison of MAE with IbCF and ClubCF. (a) $\gamma = 0.1$. (b) $\gamma = 0.2$. (c) $\gamma = 0.3$. (d) $\gamma = 0.4$.

The mashup services published on ProgrammableWeb focus on six categories which labeled with keywords: "photo," "google," "flash," "mapping," "enterprise," and "sms". Therefore, without loss of generality in our experiment, the value of K, which is the third input parameter of Algorithm 1, is set to 3, 4, 5, and 6, respectively. Furthermore, rating similarity threshold γ is set to 0.1, 0.2, 0.3 and 0.4. Under these parameter conditions, the predicted ratings of test services are calculated by ClubCF and IbCF. Then the average MAEs of ClubCF and IbCF can be computed using formula (8). The comparison results are shown in Fig. 2(a)–(d), respectively. There are several discoveries as follows.

- While the rating similarity threshold $\gamma < 0.4$, MAE values of ClubCF decrease as the value of K increases. Since services are divided into more clusters, the services in a cluster will be more similar with each other. Furthermore, neighbors of a target service are chosen from the cluster that the target service belongs to. Therefore, these neighbors may be more similar to the target service. It results in more accurate prediction. In contrast, γ plays no role in IbCF because it is not considered in IbCF.
- When $\gamma < 0.4$, MAE values of ClubCF and IbCF both decrease as the value of γ increases. It is due to that the neighbors will be more similar to the target service when the value of γ increases. It also results in the predicted ratings of target services computed according to the history ratings of the neighbors are approximate to their actual value.
- When $\gamma < 0.4$, MAE values of ClubCF are lower than IbCF. In ClubCF, services are first clustered according to their characteristic similarities. And then rating similar-

ities are measured between services in the same cluster. Since ratings of characteristic-similar services are more relevant with each other, it is more accurate to compute rating similarities between services in the same cluster. And the neighbors chosen based on the rating similarities are more similar to the target services. Consequently, the predicted ratings of the target services will be more precise than that of IbCF.

- While $\gamma = 0.4$, MAE values of ClubCF and IbCF both increase. When $k = 5$ and $k = 6$, MAE values of ClubCF are even more than that of IbCF. By checking the intermediate results of these two approaches, we found that a lot of test services have few or even no neighbors when the rating similarity threshold is set to 0.4, especially when neighbors have to be selected from a smaller cluster. It results in comparatively large deviations between the predict ratings and the real ratings. With more sparse user-rating data, it will be more difficult to find high-similar services. Therefore, if sparsity is inevitable, rating similarity threshold should be adjusted dynamically according to the accuracy requirement of application or the sparsity of the dataset.

In addition, to evaluate the efficiency of ClubCF, the online computation time of ClubCF is compared with that of IbCF, as shown in Fig. 3(a)–(d). There are several discoveries as follows.

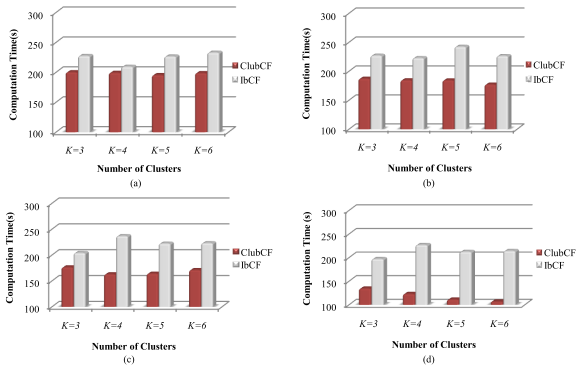


FIGURE 3. Comparison of Computation Time with ClubCF and IbCF. (a) $\gamma = 0.1$. (b) $\gamma = 0.2$. (c) $\gamma = 0.3$. (d) $\gamma = 0.4$.

- In all, ClubCF spends less computation time than Item-based CF. Since the number of services in a cluster is fewer than the total number of services, the time of rating similarity computation between every pair of services will be greatly reduced.
- As the rating similarity threshold γ increase, the computation time of ClubCF decrease. It is due to the number of neighbors of the target service decreases when γ increase. However, only when $\gamma = 0.4$, the decrease of computation time of IbCF is visible. It is due to the number of neighbors found from a cluster may less than that of found from all, and then it may spend less time on computing predicted ratings in ClubCF.
- When $\gamma = 0.4$, as K increase, the computation time of ClubCF decrease obviously. Since a bigger K means fewer services in each cluster and a bigger γ makes less neighbors, the computation time of predicted ratings based on less neighbors may decrease.

According to the computation complex analysis in Section III-C and these illustrations of experimental results, it can draw a conclusion that ClubCF may gain good scalability via increase the parameter K appropriately. Along with adjustment of γ, recommendation precision is also improved.

VI. RELATED WORK

Clustering methods for CF have been extensively studied by some researchers. Mai et al. [37] designed a neural networks-based clustering collaborative filtering algorithm in e-commerce recommendation system. The cluster analysis gathers users with similar characteristics according to the web visiting message data. However, it is hard to say that a user's preference on web visiting is relevant to preference on purchasing. Mittal et al. [38] proposed to achieve the predictions for a user by first minimizing the size of item set the user needed to explore. K-means clustering algorithm was applied to partition movies based on the genre requested by the user. However, it requires users to provide some extra information. Li et al. [39] proposed to incorporate multidimensional clustering into a collaborative filtering recommendation model. Background data in the form of user and item profiles was collected and clustered using the proposed algorithm in the first stage. Then the poor clusters with similar features were deleted while the appropriate clusters were further selected based on cluster pruning. At the third stage, an item prediction was made by performing a weighted average of deviations from the neighbor's mean. Such an approach was likely to trade-off on increasing the diversity of recommendations while maintaining the accuracy of recommendations. Zhou et al. [40] represented Data-Providing (DP) service in terms of vectors by considering the composite relation between input, output, and semantic relations between them. The vectors were clustered using a refined fuzzy C-means algorithm. Through merging similar services into a same cluster, the capability of service search engine was improved significantly, especially in large Internet-based service repositories. However, in this approach, it is assumed that domain ontology exists for facilitating semantic interoperability. Besides, this approach is not suitable for some services which are lack of parameters. Pham et al. [41] proposed to use network clustering technique on social network of users to identify their neighborhood, and then use the traditional CF algorithms to generate the recommendations. This work depends on social relationships between users. Simon et al. [42] used a high-dimensional parameter-free, divisive hierarchical clustering algorithm that requires only implicit feedback on past user purchases to discover the relationships within the users. Based on the clustering results, products of high interest were recommended to the

users. However, implicit feedback does not always provide sure information about the user's preference.

In ClubCF approach, the description and functionality information is considered as metadata to measure the characteristic similarities between services. According to such similarities, all services are merged into smaller-size clusters. Then CF algorithm is applied on the services within the same cluster. Compared with the above approaches, this approach does not require extra inputs of users and suits different types of services. Moreover, the clustering algorithm used in ClubCF need not consider the dependence of nodes.

VII. CONCLUSION AND FUTURE WORK

In this paper, we present a ClubCF approach for big data applications relevant to service recommendation. Before applying CF technique, services are merged into some clusters via an AHC algorithm. Then the rating similarities between services within the same cluster are computed. As the number of services in a cluster is much less than that of in the whole system, ClubCF costs less online computation time. Moreover, as the ratings of services in the same cluster are more relevant with each other than with the ones in other clusters, prediction based on the ratings of the services in the same cluster will be more accurate than based on the ratings of all similar or dissimilar services in all clusters. These two advantageous of ClubCF have been verified by experiments on real-world data set.

Future research can be done in two areas. First, in the respect of service similarity, semantic analysis may be performed on the description text of service. In this way, more semantic-similar services may be clustered together, which will increase the coverage of recommendations. Second, with respect to users, mining their implicit interests from usage records or reviews may be a complement to the explicit interests (ratings). By this means, recommendations can be generated even if there are only few ratings. This will solve the sparsity problem to some extent.

REFERENCES

[1] M. A. Beyer and D. Laney, "The importance of 'big data': A definition," Gartner Inc., Stamford, CT, USA, Tech. Rep., 2012.

[2] X. Wu, X. Zhu, G. Q. Wu, and W. Ding, "Data mining with big data," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 1, pp. 97–107, Jan. 2014.

[3] A. Rajaraman and J. D. Ullman, *Mining of Massive Datasets*. Cambridge, U.K.: Cambridge Univ. Press, 2012.

[4] Z. Zheng, J. Zhu, and M. R. Lyu, "Service-generated big data and big data-as-a-service: An overview," in *Proc. IEEE Int. Congr. Big Data*, Oct. 2013, pp. 403–410.

[5] A. Bellogín, I. Cantador, F. Díez, P. Castells, and E. Chavarriaga, "An empirical comparison of social, collaborative filtering, and hybrid recommenders," *ACM Trans. Intell. Syst. Technol.*, vol. 4, no. 1, pp. 1–37, Jan. 2013.

[6] W. Zeng, M. S. Shang, Q. M. Zhang, L. Lü, and T. Zhou, "Can dissimilar users contribute to accuracy and diversity of personalized recommendation?" *Int. J. Modern Phys. C*, vol. 21, no. 10, pp. 1217–1227, Jun. 2010.

[7] T. C. Havens, J. C. Bezdek, C. Leckie, L. O. Hall, and M. Palaniswami, "Fuzzy c-means algorithms for very large data," *IEEE Trans. Fuzzy Syst.*, vol. 20, no. 6, pp. 1130–1146, Dec. 2012.

[8] Z. Liu, P. Li, Y. Zheng, and M. Sun, "Clustering to find exemplar terms for keyphrase extraction," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, May 2009, pp. 257–266.

[9] X. Liu, G. Huang, and H. Mei, "Discovering homogeneous web service community in the user-centric web environment," *IEEE Trans. Services Comput.*, vol. 2, no. 2, pp. 167–181, Apr./Jun. 2009.

[10] H. H. Li, X. Y. Du, and X. Tian, "A review-based reputation evaluation approach for Web services," *J. Comput. Sci. Technol.*, vol. 24, no. 5, pp. 893–900, Sep. 2009.

[11] K. Zielinnski, T. Szydlo, R. Szymacha, J. Kosinski, J. Kosinska, and M. Jarzab, "Adaptive SOA solution stack," *IEEE Trans. Services Comput.*, vol. 5, no. 2, pp. 149–163, Jun. 2012.

[12] F. Chang *et al.*, "Bigtable: A distributed storage system for structured data," *ACM Trans. Comput. Syst.*, vol. 26, no. 2, pp. 1–39, Jun. 2008.

[13] R. S. Sandeep, C. Vinay, and S. M. Hemant, "Strength and accuracy analysis of affix removal stemming algorithms," *Int. J. Comput. Sci. Inf. Technol.*, vol. 4, no. 2, pp. 265–269, Apr. 2013.

[14] V. Gupta and G. S. Lehal, "A survey of common stemming techniques and existing stemmers for Indian languages," *J. Emerging Technol. Web Intell.*, vol. 5, no. 2, pp. 157–161, May 2013.

[15] A. Rodriguez, W. A. Chaovalitwongse, L. Zhe, H. Singhal, and H. Pham, "Master defect record retrieval using network-based feature association," *IEEE Trans. Syst., Man, Cybern., Part C, Appl. Rev.*, vol. 40, no. 3, pp. 319–329, Oct. 2010.

[16] T. Niknam, E. Taherian Fard, N. Pourjafarian, and A. Rousta, "An efficient algorithm based on modified imperialist competitive algorithm and K-means for data clustering," *Eng. Appl. Artif. Intell.*, vol. 24, no. 2, pp. 306–317, Mar. 2011.

[17] M. J. Li, M. K. Ng, Y. M. Cheung, and J. Z. Huang, "Agglomerative fuzzy k-means clustering algorithm with selection of number of clusters," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 11, pp. 1519–1534, Nov. 2008.

[18] G. Thilagavathi, D. Srivaishnavi, and N. Aparna, "A survey on efficient hierarchical algorithm used in clustering," *Int. J. Eng.*, vol. 2, no. 9, Sep. 2013.

[19] C. Platzer, F. Rosenberg, and S. Dustdar, "Web service clustering using multidimensional angles as proximity measures," *ACM Trans. Internet Technol.*, vol. 9, no. 3, pp. 11:1–11:26, Jul. 2009.

[20] G. Adomavicius and J. Zhang, "Stability of recommendation algorithms," *ACM Trans. Inf. Syst.*, vol. 30, no. 4, pp. 23:1–23:31, Aug. 2012.

[21] J. Herlocker, J. A. Konstan, and J. Riedl, "An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms," *Inf. Retr.*, vol. 5, no. 4, pp. 287–310, Oct. 2002.

[22] A. Yamashita, H. Kawamura, and K. Suzuki, "Adaptive fusion method for user-based and item-based collaborative filtering," *Adv. Complex Syst.*, vol. 14, no. 2, pp. 133–149, May 2011.

[23] D. Julie and K. A. Kumar, "Optimal web service selection scheme with dynamic QoS property assignment," *Int. J. Adv. Res. Technol.*, vol. 2, no. 2, pp. 69–75, May 2012.

[24] J. Wu, L. Chen, Y. Feng, Z. Zheng, M. C. Zhou, and Z. Wu, "Predicting quality of service for selection by neighborhood-based collaborative filtering," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 43, no. 2, pp. 428–439, Mar. 2013.

[25] Y. Zhao, G. Karypis, and U. Fayyad, "Hierarchical clustering algorithms for document datasets," *Data Mining Knowl. Discovery*, vol. 10, no. 2, pp. 141–168, Nov. 2005.

[26] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "QoS-aware web service recommendation by collaborative filtering," *IEEE Trans. Services Comput.*, vol. 4, no. 2, pp. 140–152, Feb. 2011.

[27] M. R. Catherine and E. B. Edwin, "A survey on recent trends in cloud computing and its application for multimedia," *Int. J. Adv. Res. Comput. Eng. Technol.*, vol. 2, no. 1, pp. 304–309, Feb. 2013.

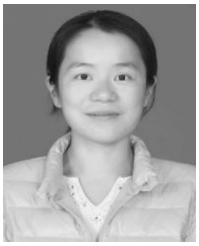
[28] X. Liu, Y. Hui, W. Sun, and H. Liang, "Towards service composition based on mashup," in *Proc. IEEE Congr. Services*, Jul. 2007, pp. 332–339.

[29] X. Z. Liu, G. Huang, Q. Zhao, H. Mei, and M. B. Blake, "iMashup: A mashup-based framework for service composition," *Sci. China Inf. Sci.*, vol. 57, no. 1, pp. 1–20, Jan. 2013.

[30] H. Elmeleegy, A. Ivan, R. Akkiraju, and R. Goodwin, "Mashup advisor: A recommendation tool for mashup development," in *Proc. IEEE Int. Conf. Web Services*, Oct. 2008, pp. 337–344.

[31] O. Greenshpan, T. Milo, and N. Polyzotis, "Autocompletion for mashups," *Proc. VLDB Endowment*, vol. 2, no. 1, pp. 538–549, Aug. 2009.

[32] M. Weiss, S. Sari, and N. Noori, "Niche formation in the mashup ecosystem," *Technol. Innov. Manag. Rev.*, May 2013.
[33] S. An, W. Liu, S. Venkatesh, and H. Yan, "Unified formulation of linear discriminant analysis methods and optimal parameter selection," *Pattern Recognit.*, vol. 44, no. 2, pp. 307–319, Feb. 2011.
[34] W. Dou, X. Zhang, J. Liu, and J. Chen, "HireSome-II: Towards privacy-aware cross-cloud service composition for big data applications," *IEEE Trans. Parellel Distrib. Syst.*, 2013, to be published.
[35] S. Agarwal, and A. Nath, "A study on implementing Green IT in Enterprise 2.0," *Int. J. Adv. Comput. Res.*, vol. 3, no. 1, pp. 43–49, Mar. 2013.
[36] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 5–53, Jan. 2004.
[37] J. Mai, Y. Fan, and Y. Shen, "A neural networks-based clustering collaborative filtering algorithm in e-commerce recommendation system," in *Proc. Int. Conf. Web Inf. Syst. Mining*, pp. 616–619, Jun. 2009.
[38] N. Mittal, R. Nayak, M. C. Govil, and K. C. Jain, "Recommender system framework using clustering and collaborative filtering," in *Proc. 3rd Int. Conf. Emerging Trends Eng. Technol.*, Nov. 2010, pp. 555–558.
[39] X. Li and T. Murata, "Using multidimensional clustering based collaborative filtering approach improving recommendation diversity," in *Proc. IEEE/WIC/ACM Int. Joint Conf. Web Intell. Intell. Agent Technol.*, Dec. 2012, pp. 169–174.
[40] Z. Zhou, M. Sellami, W. Gaaloul, M. Barhamgi, and B. Defude, "Data providing services clustering and management for facilitating service discovery and replacement," *IEEE Trans. Autom. Sci. Eng.*, vol. 10, no. 4, pp. 1–16, Oct. 2013.
[41] M. C. Pham, Y. Cao, R. Klamma, and M. Jarke, "A clustering approach for collaborative filtering recommendation using social network analysis," *J. Univ. Comput. Sci.*, vol. 17, no. 4, pp. 583–604, Apr. 2011.
[42] R. D. Simon, X. Tengke, and W. Shengrui, "Combining collaborative filtering and clustering for implicit recommender system," in *Proc. IEEE 27th Int. Conf. Adv. Inf. Netw. Appl.*, Mar. 2013, pp. 748–755.



RONG HU was born in Xiangtan, Hunan, China, in 1977. She is currently pursuing the Ph.D. degree in computer science and technology from Nanjing University, China. She received the M.A. degree from the College of Information Engineering, Xiangtan University, in 2005. Her current research interests include service computing and big data.

WANCHUN DOU was born in 1971. He is a Professor with the Department of Computer Science and Technology, Nanjing University, China. He received the Ph.D. degree from the Nanjing University of Science and Technology, China, in 2001. Then, he continued his research work as a Post-Doctoral Researcher with the Department of Computer Science and Technology, Nanjing University, from 2001 to 2002. In 2005, he visited the Hong Kong University of Science and Technology as a Visiting Scholar. His main research interests include knowledge management, cooperative computing, and workflow technologies.

JIANXUN LIU was born in 1970. He received the M.S. and Ph.D. degrees in computer science from the Central South University of Technology and the Shanghai Jiao Tong University in 1997 and 2003, respectively. He is currently a Professor with the Department of Computer Science and Engineering, Hunan University of Science and Technology. His current interests include workflow management systems, services computing, and cloud computing.