Knowl Inf Syst
DOI 10.1007/s10115-012-0562-1

REGULAR PAPER

Hybrid Collaborative Filtering algorithm for bidirectional Web service recommendation

Jie Cao · Zhiang Wu · Youquan Wang · Yi Zhuang

Received: 15 June 2011 / Revised: 26 April 2012 / Accepted: 21 July 2012

Abstract Web service recommendation has become a hot yet fundamental research topic in service computing. The most popular technique is the Collaborative Filtering (CF) based on a user-item matrix. However, it cannot well capture the relationship between Web services and providers. To address this issue, we first design a cube model to explicitly describe the relationship among providers, consumers and Web services. And then, we present a Standard Deviation based Hybrid Collaborative Filtering (SD-HCF) for Web Service Recommendation (*WSRec*) and an Inverse consumer Frequency based User Collaborative Filtering (IF-UCF) for Potential Consumers Recommendation (*PCRec*). Finally, the decision-making process of bidirectional recommendation is provided for both providers and consumers. Sets of experiments are conducted on real-world data provided by Planet-Lab. In the experiment phase, we show how the parameters of SD-HCF impact on the prediction quality as well as demonstrate that the SD-HCF is much better than extant methods on recommendation quality, including the CF based on user, the CF based on item and general HCF. Experimental comparison between IF-UCF and UCF indicates the effectiveness of adding inverse consumer frequency to UCF.

Keywords Web service · Bidirectional recommendation · Collaborative Filtering · Hybrid Collaborative Filtering

J. Cao · Z. Wu (✉)
Jiangsu Provincial Key Laboratory of E-Business, Nanjing University of Finance and Economics, Nanjing, China
e-mail: zawu@seu.edu.cn

Y. Wang
College of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China

Y. Zhuang
College of Computer and Information Engineering, Zhejiang Gongshang University, Hangzhou, China

Published online: 10 October 2012

1 Introduction

Web service is a series of platform-independent operations for cooperative design over a network. As services proliferate on the Web, several services may provide similar functionalities while hosted in different sites [22,28]. It is necessary to select the "best" Web service from available candidates. On the other hand, Web services are overloaded, and it is increasingly difficult for the users to locate the right Web service at the right time. Therefore, Web service recommendation system has emerged to help user select suitable Web services. Quality of Service (QoS) encompasses important functional and nonfunctional characteristics of Web service [31]. By combining QoS attributes with preferences and needs of users, recommender systems will more effectively provide the user with intelligent and proactive Web services. This combination can be implemented with the Collaborative Filtering (CF).

Collaborative Filtering (CF) is the most widely used technique for recommender systems. Tapestry and GroupLens are the two earliest CF-based recommender systems for mail and news, respectively [10,15]. Amazon, one of the most famous e-business recommender systems, employs CF algorithm for book recommendation [21]. Facebook, the largest social network service, utilizes CF algorithm for advertisement recommendation [9]. The CF algorithm has also been applied to Web service recommendation [33] and can automatically predict the QoS attributes of a Web service perceived by *consumers*. Most of the present recommender systems describe the relationship between consumers (service users) and Web services with a 2D matrix, called *user-item matrix*. Usually, an element in the *user-item matrix* can represent a QoS vector perceived by a user on a service, without the information concerning service providers. Therefore, *user-item matrix* model cannot reveal the relationship between Web services and providers. However, it is observed that in real applications a service provider often hosts a number of services and many similar services are provided by different providers. In other words, service providers play an important role in Web service and must be taken into account for making high-quality recommendations.

If a recommender system can exploit the information of service providers, the prediction accuracy will be improved substantially. However, exploiting the information of service providers reckons on a well-represented relationship among providers, consumers and services. In the meanwhile, most recommender systems can only recommend Web services for consumers, but cannot recommend consumers for providers. However, a perfect recommender system is desired to recommend consumers for providers, which leads to research into *Bidirectional Web Service Recommendation*.

In this paper, Web service recommendation problem is handled with a *cube* model that explicitly describes relationship among providers, consumers and services. The cube model is represented by three 2D matrixes: consumer-service QoS matrix, provider-service binary matrix and consumers-provider matrix. Consumer-service QoS matrix is similar to traditional *user-item matrix* which represents QoS vectors perceived by users on services. Provider-service binary matrix indicates the provider of each Web service. Consumer-provider matrix describes the level of interest of consumers to providers. Based on the cube model, a Standard Deviation based Hybrid Collaborative Filtering (SD-HCF) is proposed for Web Service Recommendation (*WSRec*), and an Inverse consumer Frequency based User Collaborative Filtering (IF-UCF) is designed for Potential Consumers Recommendation (*PCRec*). The SD-HCF considers both Web service similarity and consumer similarity, where its weights of bias-from-mean of k nearest neighbors are adjusted to the standard deviation. We also consider the impact of service providers on the selection of k nearest neighbors. The IF-UCF follows the process of UCF with employing the *inverse consumer frequency* to adjust the weight of providers. Several experiments are conducted based on real-world dataset

provided by Planet-Lab and demonstrate that the designed SD-HCF and IF-UCF are efficient and promising.

The rest of this paper is organized as follows. Section 2 recalls the related work. Section 3 presents the cube model and problem formalization. We elaborately describe the SD-HCF in Sect. 4 and the IF-UCF in Sect. 5, respectively. Experimental evaluation is presented in Sect. 6. And we conclude our work in Sect. 7.

2 Related work

Recommender systems are changing the way people interact with the Web. Until now, a multitude of algorithms and techniques are developed to improve the recommender systems. Some typical algorithms and techniques in recommender systems are reviewed in Sect. 2.1. And those recommendation algorithms and techniques for Web Service personalization are introduced in Sect. 2.2.

2.1 Recommender systems

There are at least two tasks of recommender systems [5]. One is to predict the utility of a given item to a user. For example, movie recommender systems should predict the rating the user would give to a movie. Another is to help users find other items they may interest by giving a list of relevant items. For instance, Amazon recommends potentially interesting books to users.

The initial method used in recommender systems is the Content-based algorithm which suggests items that are similar to the ones the current users has shown a preference for in the past [24]. Content-based algorithm relies on rich content descriptions of the items. For instance, if a recommender system wants to suggest a coat, the Content-based algorithm will depend on information such as brand, style, color, price and so on. Therefore, the developer should obtain abundant domain knowledge which may not be readily available or straightforward to maintain.

Different from the Content-based algorithm, the CF (Collaborative Filtering) algorithm collects opinions from users in the form of ratings on items. Its advantage over the Content-based algorithm is that it does not need a representation of the items but only the historical information of users. Existing CF algorithms can be divided into two categories, *neighborhood-based approach and latent factor models* [3,17]. Neighborhood-based CF algorithms contain User-based CF (UCF) and Item-based CF (ICF). The UCF is probably the most successful and widely used technique because it has been adopted by a number of well-known recommender systems, such as Tapestry, MovieLens, Amazon Book Store, YouTube, Facebook and so forth. However, the computation capacity required by the UCF grows with the number of users and items linearly. When there are millions of users and items, the UCF suffers from serious scalability problem, the ICF is thus proposed [25]. Since the number of items is more stable than users, the ICF adopts the *pre-computed* model to quickly generate recommendation results [17]. However, it is common that users have few conjunct ranked items, or items have few conjunct ranking users. This leads to that the UCF or ICF can generate poor recommendation.

Latent factor models, such as Singular Value Decomposition (SVD), SVD extended (SVDe) and Asymmetric Factor Model (AFM), create an alternative approach by transforming both items and users to the same latent factor space and make them directly comparable. Latent factor models offer high expressive ability to describe various aspects of the

data. Therefore, they usually provide more accurate results than neighborhood-based models [4,16,13]. However, most literature and commercial systems are based on the neighborhood-based models, such as Amazon, TiVo [1] and Netflix. The prevalence of neighborhood-based models is partly attributed to their relative simplicity and intuitiveness. The proposed SD-HCF belongs to neighborhood-based CF algorithms.

Hybrid Collaborative Filtering (HCF) can be divided into two classes. One type of HCF combines UCF with ICF, which is firstly proposed in [23], and then, Li et al. [20] applied HCF to e-business applications. Another is to combine content-based algorithm and CF. For instance, Gunawardana et al. [11] utilized unified Boltzmann machines to encode collaborative and content information as features and then learn weights reflecting how well each feature predicts user actions. The information about Web Services provider can be considered as a kind of contextual information. The proposed SD-HCF employs provider information to guide the recommendation. Different from the above, our proposed cube model is readily extended to support more contextual information.

Missing value estimation is the precondition of recommendation. Zhu et al. classify methods for dealing with missing values into three categories: (1) case deletion, (2) learning without handling of missing values and (3) missing value imputation [35]. Most recommendation algorithms have employed the first method which simply omits those cases with missing values and only utilizes the remaining instances to finish the learning assignments. Zhang et al. [32] propose a novel and efficient algorithm for missing value estimation, called NIIA (Nonparametric Iterative Imputation). Applying these novel algorithms to the prediction phase of the recommendation algorithm will be studied in the future.

2.2 Web service personalization

Key algorithms and techniques in recommender systems have been utilized in Web Service personalization. Zheng et al. [33] presented a Web service recommender system called *WSRec* in which the key algorithm is HCF proposed in [23]. Then, they provided WS-DREAM dataset for conducting experiment on Web service recommendation. Chen et al. [6] extended *WSRec* [33] and proposed a novel HCF called *RegionKNN* by considering region during Web service recommendation. Li et al. [20] and Mclaughlin and Herlocker [23] made QoS prediction on the basis of local average QoS vector of user and item. Different from the above, our personalization algorithm, called SD-HCF, utilizes standard deviation to adjust the weight of bias-from-mean of k nearest neighbors. In Sect. 5.3, WS-DREAM dataset is used to compare the prediction quality of SD-HCF and general HCF.

The difference between recommendation of Web service and e-commerce (or movie) is that the measuring criterion in Web Service is a multi-dimensional QoS vector rather than a numeral rating in e-commerce or movie. QoS has been widely used for Web service composition, selection and resource scheduling. QoS attributes of Web services are multi-dimensional and include response time, trust, failure rate, price and so on [31]. The global performance of multi-dimensional QoS attributes is measured by User Satisfaction Degree (USD) [19,27,29]. Including these functions, our work also supports multi-dimensional QoS attributes and adopts USD to measure Web service recommendation.

The rapid development of Web 2.0 facilitates the development of new techniques for recommender system. Kim et al. [14] utilized collaborative tagging to enhance the quality of CF. Song et al. [30] proposed automatic tag recommendation algorithms based on machine learning for social recommender systems. Leung et al. [18] utilized sentiment analysis to augment ratings for performing the CF, and Fan et al. [8] presented a sentiment-oriented contextual advertising. Integrating sentiment analysis and the CF can utilize user-generated

reviews in the context of the article instead of numerical ratings. Bechetti et al. [2] have investigated the effectiveness of fully decentralized CF in pervasive scenarios. According to the aforementioned researches, we concluded that applying recommender system to various Web applications and enhancing the quality of recommender system utilizing various information scatted in Web are two essential topics.

3 Problem formalization

3.1 Cube model

There are three kinds of roles in Web service, namely provider, consumer and Web service. A service provider may host different Web services. A set of Web services with similar functions might be offered by several service providers, and QoS vectors of these similar Web services are distinguishing. In what follows, we propose to use three matrixes to describe the relation among provider, consumer and Web service. It is interesting to note that three matrixes are three sides of a provider-consumer-service cube, R as depicted in Fig. 1. Each cell inside the cube $\mathbf{R}(c_i, p_j, s_m)$ contains three attributes as shown in Q, A and V, respectively.

Definition 3.1 (*Consumer-service QoS matrix,* $\boldsymbol{Q}$). Assume the set of consumers is $C = \{c_1, c_2, \ldots, c_{|C|}\}$ and the set of Web services is $S = \{s_1, s_2, \ldots, s_{|S|}\}$. The rows in $\boldsymbol{Q}$ represent consumers, and the columns represent Web services. If consumer c_i invoked Web service m, QoS vector perceived by the consumer is recorded in matrix $\boldsymbol{Q}$, denoted as $\boldsymbol{Q}(c_i, s_m)$.

Definition 3.2 (*Provider-service binary matrix,* A). Assume the set of providers is $P = \{p_1, p_2, \ldots, p_{|P|}\}$. The rows in $\boldsymbol{A}$ represent providers, and the columns represent Web services where $\boldsymbol{A}$ represents the host relationship between provider and Web service, in which $\mathbf{A}(p_j, s_m)$ is 1 if the service s_m is hosted by the provider p_j or 0 if not. P=$\{p_1, p_2, \ldots, p_{|P|}\}$.

Definition 3.3 (*Consumer-provider matrix,* V). The columns represent providers. The element $\mathbf{V}(c_i, p_j)$ represents the number of times that the consumer c_i has *successfully* invoked Web services hosted by the provider p_j.

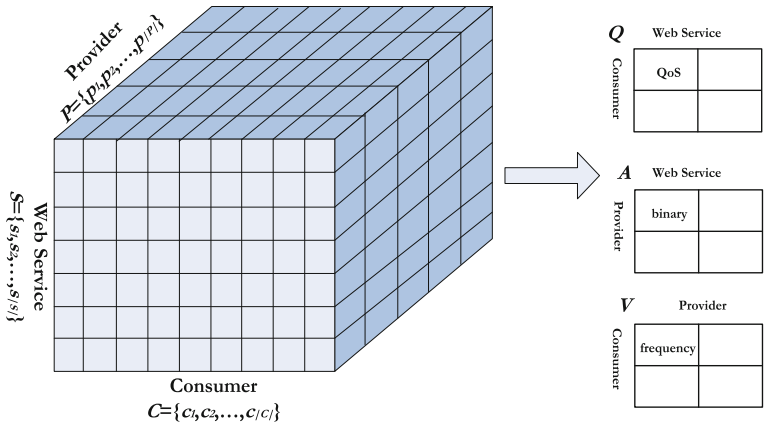


Fig. 1 An example of cube $\boldsymbol{R}$

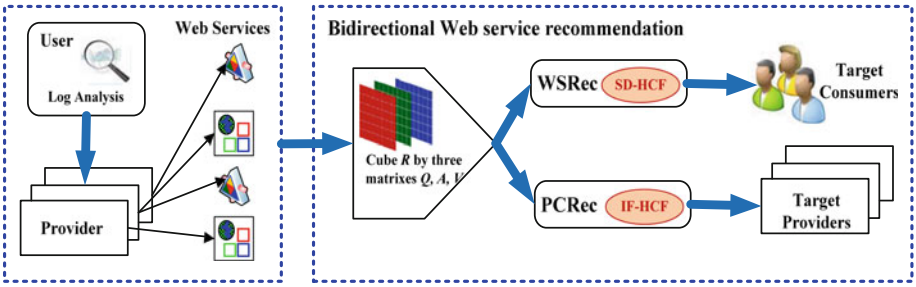


Fig. 2 Illustration of bidirectional Web service recommendation

The formal statement for *bidirectional Web service recommendation* problem is stated as follows. The original input is the data cube $\boldsymbol{R}$ described by the three matrixes $\boldsymbol{Q}$, $\boldsymbol{A}$, $\boldsymbol{V}$. Firstly, Web services are recommended to active users (*WSRec* for short). Let $\boldsymbol{T}$ be the set of known perceived QoS vector in $\boldsymbol{Q}$, and $\boldsymbol{T}$ is far less than the set of unknown perceived QoS vector, namely $|\boldsymbol{T}| \ll |\boldsymbol{Q}|$. The recommender needs to predict unknown perceived QoS vectors (also called missing data) on the basis of $\boldsymbol{T}$, and to employ the predicted QoS vectors to recommend Web services to active users. If we denote online consumers (also called *active users*) as $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_g\} (\alpha \in C, g < |C|)$, the set of Web services recommended is written Ω_1:

$$\Omega_1 : \Omega_1 = \{s_n | \forall \alpha_i \in \alpha, s_n = \underset{n \in [1,|S|]}{\text{DecisionFun}}[fus_i(\mathbf{Q}(\alpha_i, s_n))]\} \tag{1}$$

where fus_i is the user satisfaction function of ith QoS attribute.

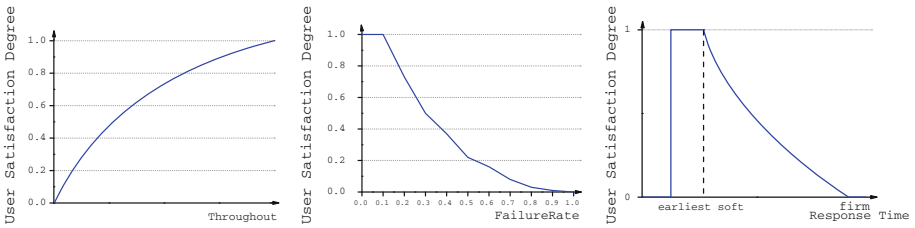
Secondly, active service users who may be interested to a provider are recommended to the provider as its potential customers (*PCRec* for short). Let Ω_2 be the set of active users recommended to p_i:

$$\Omega_2 : \Omega_2 = \{c_i | \forall p_j \in p, c_i = \underset{i \in [1,|C|]}{\text{DecisionFun}}[\mathbf{V}(c_i, p_j)]\} \tag{2}$$

There are a great deal of missing data in $\boldsymbol{Q}$ and $\boldsymbol{V}$. Therefore, we should first predict the missing data and fill them into $\boldsymbol{Q}$ and $\boldsymbol{V}$. In this article, we propose SD-HCF and IF-UCF for predicting the missing data of $\boldsymbol{Q}$ and $\boldsymbol{V}$, respectively.

In Eqs. (1) and (2), $DecisionFun$ is the decision-making function determining which services should be recommended to users or which users to providers. We propose two methods to define $DecisionFun$ in Sect. 4.4.

Figure 2 schematically illustrates the proposed approach for *bidirectional Web service recommendation*. Cube $\mathbf{R}$ is generated by processing logs of consumers and relationship between a provider and its Web services. Generally, the log file records Web service invocations by a consumer including Web service ID, values of QoS attributes and status of invocation (e.g., OK or error). A tabular representation of the log contains Web service ID, WSDL address, Web service name, provider name and its location. Based on cube $\boldsymbol{R}$ represented by $\boldsymbol{Q}$, $\boldsymbol{A}$, $\boldsymbol{V}$, we propose SD-HCF algorithm for *WSRec* in Sect. 4, and IF-UCF algorithm for *PCRec* in Sect. 5.



(a) Throughout **(b)** Failure-rate **(c)** Response Time

Fig. 3 An example of user satisfaction function

3.2 QoS model

The global performance of multi-dimensional QoS attributes is measured by User Satisfaction Degree (USD). User Satisfaction Function (USF) defines the mapping relation between QoS attribute and USD.

Definition 3.4 (*User Satisfaction Degree, USD*): a number in [0,1]. As the USD increases, users are satisfied by the service more. When the USD reaches 1, users are satisfied perfectly.

Definition 3.5 (*User Satisfaction Function, USF*): fus_i^u is the USF of ith QoS attribute for user u and describes the mapping relation between the ith QoS attribute and USD. The definition domain field of fus_i^u varies with specific QoS attribute, and the range of fus_i^u is [0,1].

USF is *user-specific* and *attribute-specific*, as shown in Fig. 3. To illustrate the user-specific using the example shown in Fig. 3c, if users define different *soft* response time for applications, USD perceived by users for a certain response time is different. Also, three curves in Fig. 3 represent three kinds of QoS attribute, which indicates that USF is attribute-specific.

Assuming the vector $Q(c_i, s_n)$ has q-dimension, the set *fus* contains q USFs. Weighted average of USD values of q-dimension QoS attributes is used to obtain the rating of c_i to s_n, as shown in Eq. (3).

$$\text{rating}_{u,i} = \frac{\sum_{j=1}^{q} w_j^u fus_j^u(Q_j^u)}{q}, \qquad \text{where} \qquad \sum_{j=1}^{q} w_j^u = 1 \tag{3}$$

In Eq. (3), Q_j^u is the jth QoS attribute, w_j^u is its weight, and fus_j^u is its USF for user u. $\text{rating}_{u,i}$ is actually the weighted average of q-dimension QoS vector. So, this user-specific rating can reveal the user preference on perceived QoS.

4 SD-HCF for Web service recommendation

The novel SD-HCF algorithm is able to be divided into three phases including similarity computation, k nearest neighbors selection and unknown QoS vector prediction. Although the three-phase framework is generalized, SD-HCF has improved each phase in some ways. Specifically, (1) a weight is added to the similarity computation to solve the overestimation of PCC measure; (2) dissimilar users are eliminated from top-k neighbors, and Web services with the same provider are added to top -k neighbors; and (3) HCF and standard deviation

Table 1 The main notations used in the paper

Notations	Description
S_{c_i}	A set of Web services invoked by consumer c_i
$\overline{S_{c_i}}$	The vector of average QoS values of consumer c
C_{s_n}	A set of consumers who have invoked the Web service s_n
$\overline{C_{s_n}}$	The vector of average QoS values perceived by all consumers on the Web service s_n
$Qsim(c_i, c_j)$	The similarity between consumer c_i and consumer c_j based on consumer-service QoS matrix, $\boldsymbol{Q}$
$Qsim(s_m, s_n)$	The similarity between Web service s_m and Web service s_n based on consumer-service QoS matrix, $\boldsymbol{Q}$
$Vsim(c_i, c_j)$	The similarity between consumer c_i and consumer c_j based on consumer-provider matrix, $\boldsymbol{V}$
$Vsim(s_m, s_n)$	The similarity between Web service s_m and Web service s_n based on consumer-provider matrix, $\boldsymbol{V}$
$N(c_i)$	k nearest neighbors of consumer c_i
$N(s_n)$	k nearest neighbors of Web service s_n

weight are incorporated to the prediction phase. All important notations used in the paper are given in Table 1.

4.1 Similarity computation

Previous work has proposed a host of methods to compute similarity, such as cosine, correctional cosine, Pearson correlation coefficient (PCC) and so forth. PCC was introduced in an army of recommender systems for similarity computation [12,14,20,33], because it can be easily implemented and can achieve high accuracy. Therefore, we also employ PCC to compute similarity between consumers and between Web services. The similarity between consumer c_i and consumer c_j is computed by Eq. (4).

$$Qsim(c_i, c_j) = \frac{\sum_{s_n \in S}(Q(c_i, s_n) - \overline{S_{c_i}}) \cdot (Q(c_j, s_n) - \overline{S_{c_j}})}{\sqrt{\sum_{s_n \in S}(Q(c_i, s_n) - \overline{S_{c_i}})^2}\sqrt{\sum_{s_n \in S}(Q(c_j, s_n) - \overline{S_{c_j}})^2}} \tag{4}$$

where $S = S_{C_i} \cap S_{C_j}$ is the subset of Web services which user c_i and c_j commonly invoked. In Eq. (4), $Qsim(c_i, c_j)$ is in $[-1, 1]$, and consumer c_i and c_j are more similar when $Qsim(c_i, c_j) > 0$ and with a larger value. Conversely, $Qsim(c_i, c_j) \leq 0$ indicates that consumer c_i and c_j are not similar. Note that PCC measure is 0 when none of the Web service is invoked both by c_i and c_j. In the same way, the similarity between Web service s_m and s_n is computed by Eq. (5).

$$Qsim(s_m, s_n) = \frac{\sum_{c_i \in C}(Q(c_i, s_m) - \overline{C_{s_m}}) \cdot (Q(c_i, s_n) - \overline{C_{s_n}})}{\sqrt{\sum_{c_i \in C}(Q(c_i, s_m) - \overline{C_{s_m}})^2}\sqrt{\sum_{c_i \in C}(Q(c_i, s_n) - \overline{C_{s_n}})^2}} \tag{5}$$

where $C = C_{s_m} \cap C_{s_n}$ is the subset of consumers who invoke both Web service s_m and s_n.

PCC often overestimates the similarities of consumers who are actually not similar but happen to have similar QoS experience on a few co-invoked Web services [23]. To remedy

that, a weight is added to reduce the influence of a small number of similar co-invoked Web services. Eqs. (6) and (7) present the weighted PCC.

$$Qsim'(c_i, c_j) = \frac{|S_{c_i} \cap S_{c_j}|}{|S_{c_i} \cup S_{c_j}|} Qsim(c_i, c_j) \tag{6}$$

$$Qsim'(s_m, s_n) = \frac{|C_{s_m} \cap C_{s_m}|}{|C_{s_m} \cup C_{s_n}|} Qsim(s_m, s_n) \tag{7}$$

From Eqs. (6) and (7), when $|S_{c_i} \cap S_{c_j}|$ (or $|C_{s_m} \cap C_{s_n}|$) is smaller than $|S_{c_i} \cup S_{c_j}|$ (or $|C_{s_m} \cup C_{s_n}|$), the weight will decrease the similarity between consumers (or Web services). The idea behind the weight definition is that for two users, the more invoked services, the more likely they co-invoke many Web services by chance. So their similarity should be reduced. Also, it makes sense that the similarity between two consumers with a small set of commonly invoked Web services is small.

4.2 k nearest neighbors selection

The nearest-neighbor technique utilizes nearest neighbors to describe the characteristics of the test example. Selecting neighbors of active consumers is an important step for accurate recommendation. According to Sect. 4.1, two consumers (or Web services) are not similar if their similarity is equal to or smaller than 0. Therefore, some consumers (or Web services) have limited similar counterparts or even none. Traditional top-k algorithm ignores this problem and still chooses the top k most similar neighbors even if they are not similar at all, which will decrease the prediction quality dramatically. In this paper, a neighbor will be removed from the set of the top k similar neighbors if its similarity is equal or smaller than 0. We denote the set of k nearest neighbors of consumer c_i as $N(c_i)$ given by the following equation:

$$N(c_i) = \{c_j | Qsim'(c_i, c_j) > 0,\ c_j \in T_k(c_i)\} \tag{8}$$

where $T_k(c_i)$ is a set of the top k similar neighbors to the consumer c_i. In other words, $T_k(c_i)$ is k nearest neighbors found by traditional top-k algorithm. Our proposed method selects the subset of $T_k(c_i)$ where the similarity is larger than 0. So, we have $|T_k(c_i)| = k$, but $|N(c_i)| \leq k$.

For selecting nearest neighbors for a Web service, we consider the impact of providers as well as the elimination of dissimilar services. Indeed, Web services hosted by the same provider exhibit similar characteristic and are more likely to be similar neighbors. For our experimental dataset WS-DREAM, Table 2 gives the percentage of neighbors with the same provider in Top-5 and Top-10 nearest neighbors sets, which are computed based on the total dataset (i.e., training data plus test data). According to this chart, other services hosted by the same provider are apt to be the nearest-neighbor of a certain Web service. However, when known user-perceived QoS vectors set $\boldsymbol{T}$ (i.e., training data) is very sparse, top-k nearest neighbors set computed based on $\boldsymbol{T}$ often does not include Web services with the same provider. Therefore, we present a novel method for selecting k nearest neighbors of Web service s_n, as shown in Eq. (9).

$$N(s_n) = \{s_m | sim'(s_m, s_n) > 0,\ s_m \in T_k(s_n) \vee \exists p_i,\ \mathbf{A}(p_i, s_m) = \mathbf{A}(p_i, s_n) = 1\} \tag{9}$$

where $T_k(s_n)$ is a set of the top k similar Web services to the Web service s_n. Apart from the top k similar neighbors whose similarity is greater than 0, several Web services with the same provider are added to $N(s_n)$. It is interesting to note that $|T_k(s_n)| = k$, but $|N(c_i)| \geq k$ may be tenable. In Sect. 5.2, we conduct several experiments to verify the improvement.

Table 2 The percentage of neighbors with the same provider of top k in WS-DREAM data

Provider	Web service name	Total	Top-5	Top-10
ualberta.ca	BacMapGetGeneCardSer	6	2	2
	getColiCardIDs_by_blattner_number		2	2
dyndns.org	Echo	5	2	3
	CaptchaAudio		2	2
bsc.es	runTcoffeeFromProfiles	2	1	1
uva.nl	SynsetServerService	2	1	1
mpg.de	get_other_database_accessions	2	1	1

4.3 Unknown QoS vector prediction

SD-HCF predicts the missing QoS vectors in matrix $\boldsymbol{Q}$ for active consumers. Let $P(c, s_n)$ be QoS vector predicted by consumer c of Web service s_n. SD-HCF utilizes a weighted sum of $U(c, s_n)$ and $I(c, s_n)$ which are QoS vectors predicted by UCF and ICF, respectively. Standard deviation σ is adopted to adjust the weight of k nearest neighbors of c and s_n. Given a list of elements which are not equal to 0, standard deviation σ can be defined as:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{n}} \tag{10}$$

where the set $\{x_1, \ldots, x_n\}$ is the nonzero list of elements and $x_i \in \{x_1, \ldots, x_n\}$. $U(c, s_n)$ is defined as the sum of the vector of average QoS values of consumer c (vector $\overline{S_c}$) and deviations from the neighbor's mean:

$$\begin{aligned} U(c, s_n) &= \overline{S_c} + \sigma_c \frac{\sum_{c_j \in N(c)} Qsim'(c, c_j) z_{c_j}}{\sum_{c_j \in N(c)} Qsim'(c, c_j)} \\ \text{where} \quad z_{c_j} &= \frac{Q(c_j, s_n) - \overline{S_{c_j}}}{\sigma_{c_j}} \end{aligned} \tag{11}$$

The weight of z_{c_j}, namely $1/\sigma_{c_j}$, indicates that as σ_{c_j} increases, perceived QoS vectors of c_j fluctuate more considerably, and the bias-from-mean of c_j holds a smaller proportion. The reason of adding σ_c to $U(c, s_n)$ is that as σ_c increases, predicted QoS vector should more deviate from $\overline{S_c}$. In the same way, $I(c, s_n)$ can be defined as:

$$\begin{aligned} I(c, s_n) &= \overline{C_{s_n}} + \sigma_n \frac{\sum_{s_m \in N(s_m)} Qsim'(s_m, s_n) z_m}{\sum_{s_m \in N(s_m)} Qsim'(s_m, s_n)} \\ \text{where} \quad z_m &= \frac{Q(c, s_m) - \overline{C_{s_m}}}{\sigma_m} \end{aligned} \tag{12}$$

SD-HCF blends UCF with ICF, where UCF and ICF are represented by $U(c, s_n)$ and $I(c, s_n)$, respectively. Since different recommenders deal with different applications whose dataset has different distribution rule, the degree of dependence on UCF and ICF of SD-HCF will also be changed. Therefore, we adopt parameter λ ($0 \leq \lambda \leq 1$) to adjust the weight of UCF and ICF. $P(c, s_n)$ is defined as:

$$P(c, s_n) = \lambda U(c, s_n) + (1 - \lambda) I(c, s_n) \tag{13}$$

When the value of λ increases, SD-HCF squints toward UCF. Conversely, SD-HCF is altered to squints toward ICF. If $\lambda = 1$ SD-HCF is altered to UCF and if $\lambda = 0$, SD-HCF is altered to ICF. Based on $P(c_i, s_n)$, we can utilize QoS model introduced in Sect. 3.2 to compute rating of c_i to s_n.

4.4 Pseudo-code of SD-HCF

When unknown QoS vectors in cube $\boldsymbol{Q}$ for active consumers are predicted, we can make decision on recommendation for both active consumers and providers. In fact, the process of making decision on recommendation is to define *DecisionFun* in Eqs. (1) and (2). We have two methods to define *DecisionFun*:

METHOD I. Select top N target objects with higher USD value for active consumers and providers. The method is sure to generate recommendation results. However, some selected target objects may have low USD value, and they are useless active consumers or providers.

METHOD II. Make binary decision. The consumer either will or will not utilize the Web service. If the rating scale is not already binary, we need to transform it into a binary scale. We can define a threshold value, and if the USD value of a target object is greater than the threshold value, the target object is relevant.

SD-HCF utilizes the second method for decision making. The pseudo-code of SD-HCF is given in Algorithm 4.4.

Algorithm 1 Pseudo-code of the SD-HCF algorithm

Input:
$\boldsymbol{Q}$: consumer-service QoS matrix
α: a set of online consumers
Output:
Ω_1: a list of Web services recommended to all consumers in α
1: Compute similarity between consumers in α and all other consumers
2: Construct k nearest neighbors set of consumers in α
3: Compute similarity among all Web services
4: Construct k nearest neighbors set of all Web services
5: **for** all missing values $< c_i, s_n >$ in $\boldsymbol{Q}$
6: Compute $P(c_i, s_n)$ using Eq. (13)
7: **end for**
8: Construct Ω_1 based upon METHOD II

Lines 1–4 are for similarity computation of users and Web services. UCF suffers serious scalability problems due to the constant change of consumers. Since Web services are relatively steadier than consumers, we can utilize pre-computed model to speed up the SD-HCF. Lines 3–4 of SD-HCF can be pre-computed in an idle duration of servers, and the middle results should be kept in the servers. Lines 5–7 show the prediction process. The computational details have been given in Sect. 4.3. Finally, Line 8 chooses and returns the best Web services according to user satisfaction degrees.

5 IF-UCF for potential consumers recommendation

In this section, IF-UCF algorithm is proposed for *PCRec* based on consumer-provider matrix $\boldsymbol{V}$. Inverse consumer Frequency based UCF (IF-UCF) employs *the inverse consumer*

frequency to adjust the weight of providers. The proposed IF-UCF algorithm also consists of three phases. Since the element in $\boldsymbol{V}$ represents frequency which is different from the element in $\mathbf{Q}$ used by SD-HCF, phases of similarity computation and unknown frequency prediction are different from those of SD-HCF. Therefore, in what follows, we are able to focus on the discrimination between IF-UCF and SD-HCF, rather than the generalized framework.

5.1 Similarity computation

IF-UCF employs PCC to compute similarity between consumers based on consume-provider matrix $\boldsymbol{V}$. The number of consumers invoking services from a provider is taken into consideration. For instance, assume there are two providers, p_1 and p_2. If only 10 consumers have successfully invoked services hosted by p_1, while 100 consumers have successfully invoked services hosted by p_2, provider p_1, with fewer consumers, is relatively more reliable for similarity computation than provider p_2. The information gain of p_1 is larger than that of p_2 in terms of information theory. Similar to the inverse document frequency in Web, *the inverse consumer frequency* is presented as follows.

Definition 5.1 (*Inverse consumer frequency*). Let C be the set of consumers with size $|C|$, n_{p_i} be the number of consumers who have successfully invoked Web services hosted by p_i, and icf_{p_i} be *the inverse consumer frequency* of p_i. We have the inverse consumer frequency icf_{p_i} for the ith provider as follows.

$$icf_{p_i} = \begin{cases} \log \frac{1}{n_{p_i}} & \text{if } n_{p_i} < |C| \\ 0 & \text{if } n_{p_i} = |C| \end{cases} \tag{14}$$

Then, we utilize the inverse consumer frequency to modify PCC for similarity computation between consumer c_i and consumer c_k as shown in Eq. (15).

$$Vsim(c_i, c_k) = \frac{\sum_{p_j \in S} icf_{p_j}^2 \cdot s(c_i, p_j) \cdot s(c_k, p_j)}{\sqrt{\sum_{p_j \in S} icf_{p_j}^2 \cdot s(c_i, p_j)^2} \sqrt{\sum_{p_j \in S} icf_{p_j}^2 \cdot s(c_k, p_j)^2}} \tag{15}$$

$$\text{where } s(c_i, p_j) = V(c_i, p_j) - \overline{V_{c_i}}, s(c_k, p_j) = V(c_k, p_j) - \overline{V_{c_k}}$$

$Vsim(c_i, c_k)$ obtained by Eq. (15) is still in $[-1, 1]$, and its implication is similar to Eq. (4).

5.2 k nearest neighbors selection

kNN selection of IF-UCF is same as kNN selection for consumers of SD-HCF. We also get rid of similar neighbors whose similarity is equal to or smaller than 0 from top k similar neighbors. We denote the set of k nearest neighbors of consumer c_i as $N'(c_i)$ in Eq. (16).

$$N'(c_i) = \{c_j | sim'(c_i, c_j) > 0,\ c_j \in T_k'(c_i)\} \tag{16}$$

where $T_k'(c_i)$ is a set of the top k similar neighbors to the consumer c_i. In other words, $T_k'(c_i)$ is k nearest neighbors found by traditional top-k algorithm. Our proposed method selects the subset of $T_k'(c_i)$ where the similarity is greater than 0. Note that $T_k'(c_i)$ is obtained by similarity computed on $\boldsymbol{V}$, but $T_k(c_i)$ is based on the similarity from $\boldsymbol{Q}$.

5.3 Unknown frequency prediction

This step tries to predict missing frequency data in matrix $\boldsymbol{V}$ for target providers. If we want to compute the prediction value of the consumer c_i for the provider p_j, the weighted sum of

frequency for the provider p_j given by kNN of c_i is utilized. The similarity between c_i and its neighbor is employed as the weight. Formally, the prediction value of $V(c_i, p_j)$ can be obtained by Eq. (17).

$$V(c_i, p_j) = \sum_{c_k \in T'_k(c_i)} Vsim(c_i, c_k) \cdot V(c_k, p_j) \tag{17}$$

In Eq. (17), $T'_k(c_i)$ is the selected nearest neighbors of c_i, $V(c_k, p_j)$ is the frequency value of neighbor user c_k in $\boldsymbol{V}$, and $Vsim(c_i, c_k)$ is the similarity computed by Eq. (15).

Algorithm 2 Pseudo-code of the IF-UCF algorithm

Input:
$\boldsymbol{V}$: consumer-provider matrix
β: a set of target providers
Output:
Ω_2: top N potential consumers for all providers in β
1: **for** all consumers in $C = \{c_1, c_2, \cdots, c_{|C|}\}$, let c_i be the current consumer, $i \leq |C|$
2: Compute similarity between c_i and all other consumers using Eq. (15)
3: Construct k nearest neighbors set of c_i, $T'_k(c_i)$
4: Predict all missing values in i-th row in matrix $\boldsymbol{V}$ using Eq. (17)
5: **end for**
6: Construct Ω_2 based upon METHOD I

IF-UCF traverses each row of matrix $\boldsymbol{V}$ in order to compute all prediction values of missing elements in $\boldsymbol{V}$, because prediction values are generated by kNN of consumers. Based on known values and prediction values in $\boldsymbol{V}$, IF-UCF selects top N potential consumers with higher frequency value as mentioned in METHOD I in Sect. 4.4.

In summary, SD-UCF has used the data from $\boldsymbol{Q}$ and $\boldsymbol{A}$, but IF-UCF only used the data from $\boldsymbol{V}$. The data from $\boldsymbol{A}$ help SD-UCF to construct the set of kNN as shown in Eq. (9).

6 Experimental results

In this section, a series of experiments are conducted to evaluate algorithms proposed in this paper. Firstly, we conduct comparison among SD-HCF, HCF, UCF and ICF for $WSRec$ to demonstrate the advantage of proposed SD-HCF. Secondly, we investigate the affect of service providers for $WSRec$ to show the necessity for selecting services with the same provider into the nearest-neighbor set. Finally, the performance of IF-UCF for $PCRec$ is demonstrated.

6.1 Experimental setup

Our experiments employ the real-world Web service QoS data set.[1] The dataset includes 100 publicly available Web services hosted by more than 20 providers and 150 distributed computers from Planet-Lab [7,34]. The service consumers (distributed computer from Planet-Lab) observe and collect QoS attributes of Web services. By processing the open source files, we construct a $150 \times 100 \times 20$ consumer-service-provider cube. Each entry of consumer-service

[1] https://www.wsdream.net.

QoS matrix $\boldsymbol{Q}$ is made up of two QoS attributes which are RTT (round-trip time) and failure rate. RTT indicates the time duration between the client sending a request and receiving a response. Failure rate represents the probability that a request is correctly responded within the maximum expected time which is set to 20 s in default. RTT and failure rate are the average value of many invocations on 100 Web services by a service consumer. Provider-service binary matrix $\boldsymbol{A}$ is obtained from a relational table of Web Service. Consumer-provider matrix $\boldsymbol{V}$ is obtained by processing Web service invocations recorded in log files of consumers. Matrixes $\boldsymbol{A}$ and $\boldsymbol{V}$ are further divided into training set and test set. The percentage of training set to the whole dataset is called *density*.

Herlocker et al. [12] classify performance metrics into *statistical accuracy* and *decision-support accuracy metrics*. The experiments adopt these two kinds of metrics to evaluate the performance of algorithms. *Mean absolute error* (MAE) metric is a representative example of a statistical accuracy measure and is widely adopted to measure the prediction accuracy of Collaborative Filtering. MAE is defined as:

$$MAE = \frac{\sum_{i,n} |Q(c_i, s_n) - \widehat{Q}(c_i, s_n)|}{M} \tag{18}$$

where $Q(c, s_n)$ denotes the actual QoS vector perceived by consumer c of Web service n and $\widehat{Q}(c_i, s_n)$ denotes the predicted QoS vector. M is the number of predicted QoS vectors.

$FMeasure$ is employed as a representative example of the decision-support accuracy metric. $FMeasure$ combines $precision$ and $recall$ into a single number and evaluates how effectively predictions help a consumer select high-relevant services [12]. Precision is defined by the $NMAE$ (Normalized MAE) as Eq. (19).

$$precision = 1 - NMAE \quad \text{where} \quad NMAE = \frac{MAE}{(\sum_{i,n} Q(c_i, s_n))/M} \tag{19}$$

$Precision$ is in [0,1] and decreases with the increase in $NMAE$. The definition of $recall$ is based upon binary decision discussed in Sect. 4.4. $Recall$, shown in Eq. (20), is defined as the ratio of relevant services selected to total number of relevant items available. $Recall$ represents the probability that a relevant service will be selected.

$$recall = \frac{N_{rs}}{N_s} \tag{20}$$

Based on $Precision$ and $Recall$, $FMeasure$ can be defined, shown in Eq. (21).

$$FMeasure = \frac{2 \times recall \times precision}{recall + precision} \tag{21}$$

6.2 Comparison among SD-HCF, HCF, UCF and ICF for $WSRec$

SD-HCF utilizes parameter λ to adjust the weight of UCF and ICF. If $\lambda = 1$, SD-HCF is altered to UCF, and if $\lambda = 0$, SD-HCF is altered to ICF. To compare SD-HCF with UCF and ICF, we vary the value of λ from 0 to 1 and the interval is set to 0.1. Other parameters influencing results include cube density, neighborhood size k and the number of active consumers.

Previous HCF algorithms do not employ standard deviation σ to adjust the weight of bias-from-mean of k nearest neighbors, such as [6,20,23,33]. In order to validate the effectiveness of σ, our experiments also compare SD-HCF with HCF. Tables 3 and 5 show $NMAE$ and $FMeasure$ performance comparison of different methods employing 10 % $\sim$ 40 % density training cube. We have also conducted the statistical test to compare the performance of these four variants of CF. Tables 4 and 6 show the pairwise U-test [26] values of the

Table 3 $NMAE$ performance comparison

QoS		RTT				Failure rate			
Density (%)	Given	UCF	ICF	HCF	SD-HCF	UCF	ICF	HCF	SD-HCF
10	5	0.194	0.198	0.189	0.189	0.023	0.035	0.021	0.019
	10	0.195	0.198	0.189	0.188	0.016	0.028	0.016	0.015
	20	0.194	0.197	0.188	0.188	0.018	0.032	0.017	0.015
20	5	0.190	0.198	0.188	0.187	0.018	0.052	0.018	0.018
	10	0.189	0.198	0.188	0.188	0.019	0.060	0.018	0.018
	20	0.189	0.198	0.187	0.187	0.021	0.052	0.019	0.019
30	5	0.185	0.195	0.180	0.180	0.015	0.017	0.014	0.014
	10	0.189	0.197	0.181	0.180	0.136	0.017	0.013	0.013
	20	0.185	0.195	0.180	0.179	0.165	0.220	0.017	0.016
40	5	0.200	0.178	0.177	0.176	0.019	0.034	0.016	0.014
	10	0.178	0.198	0.177	0.177	0.014	0.019	0.012	0.011
	20	0.178	0.198	0.177	0.177	0.015	0.022	0.013	0.012

Table 4 U-test results: the *NMAE* case

	UCF	ICF	HCF	UCF	ICF	HCF
ICF	2.835***			0.431		
HCF	−2.339***	−5.785***		−1.648**	−2.128**	
SD-HCF	−2.507***	−5.950***	−0.206	−1.705**	−2.182**	−0.785*

The comparative direction is from the row item to the column item * $-\alpha = 0.22$; ** $-\alpha = 0.05$; *** $-\alpha = 0.01$; others are not significant

Table 5 $FMeasure$ performance comparison

QoS		RTT				Failure rate			
Density (%)	Given	UCF	ICF	HCF	SD-HCF	UCF	ICF	HCF	SD-HCF
10	5	0.890	0.834	0.901	0.901	0.988	0.961	0.990	0.990
	10	0.889	0.835	0.899	0.901	0.992	0.963	0.992	0.993
	20	0.890	0.835	0.901	0.901	0.890	0.835	0.901	0.901
20	5	0.891	0.766	0.894	0.895	0.992	0.979	0.994	0.994
	10	0.892	0.782	0.901	0.901	0.993	0.985	0.994	0.995
	20	0.898	0.782	0.901	0.902	0.990	0.973	0.992	0.993
30	5	0.890	0.834	0.901	0.901	0.988	0.961	0.990	0.990
	10	0.889	0.835	0.899	0.901	0.992	0.963	0.992	0.993
	20	0.890	0.835	0.901	0.901	0.990	0.960	0.992	0.992
40	5	0.891	0.766	0.894	0.895	0.992	0.979	0.994	0.994
	10	0.898	0.782	0.901	0.901	0.993	0.985	0.994	0.995
	20	0.898	0.782	0.901	0.902	0.990	0.973	0.992	0.993

Table 6 U-test results: the *FMeasure* case

	UCF	ICF	HCF	UCF	ICF	HCF
ICF	−10.092***			−1.652**		
HCF	5.897***	10.982***		0.207	1.874**	
SD-HCF	6.635***	11.067***	0.665*	0.252	1.908**	0.048

The comparative direction is from the row item to the column item * $-\alpha = 0.58$; ** $-\alpha = 0.05$; *** $-\alpha = 0.01$; others are not significant

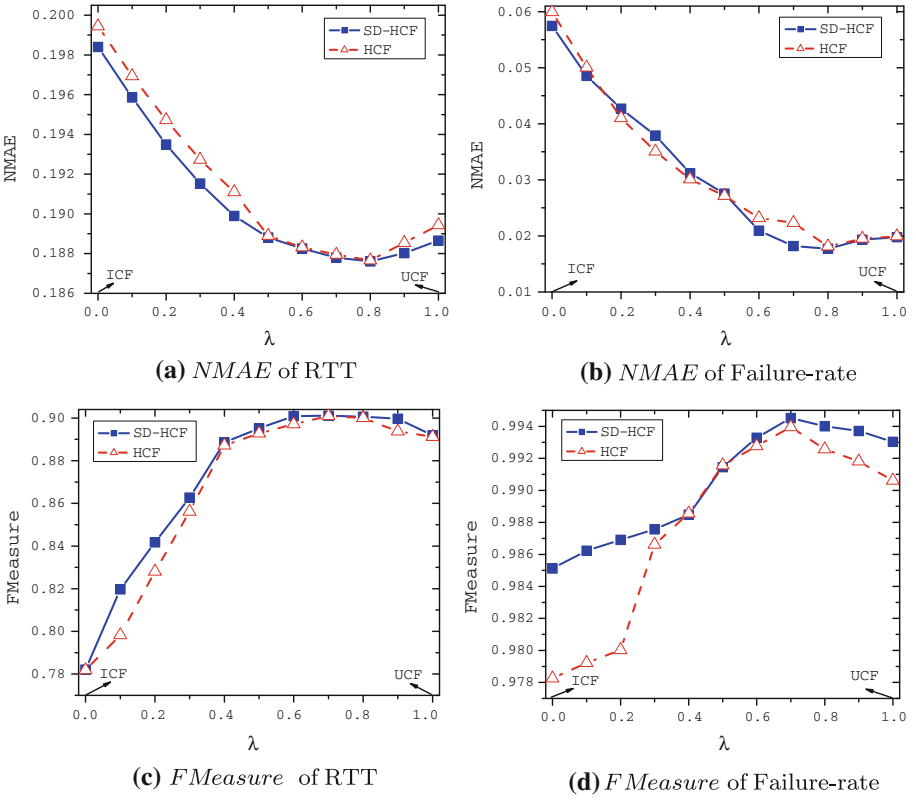


Fig. 4 Impact of λ on $NMAE$ and $FMeasure$ for SD-HCF and HCF

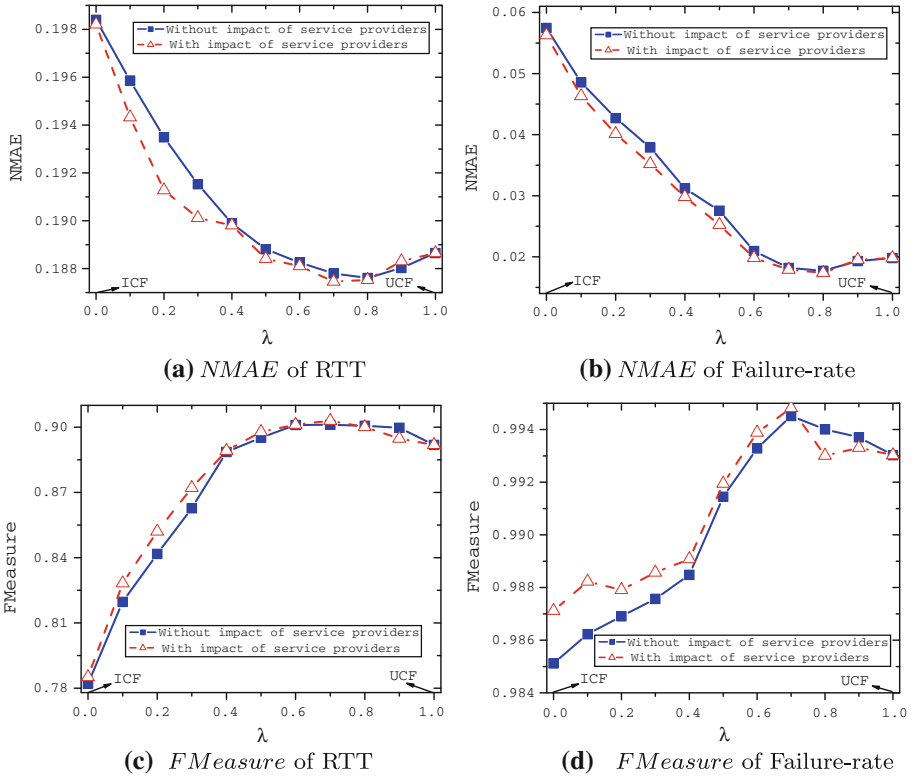
results by $NMAE$ and $FMeasure$, respectively. As can be seen from the $NMAE$ case in Table 4, while SD-HCF performs slightly better than HCF at the significance level $\alpha = 0.22$, they outperform UCF and ICF at the significance levels $\alpha = 0.05$ and $\alpha = 0.01$. The $FMeasure$ case shows similar results, except that SD-HCF and HCF show more comparable performance.

In order to illustrate how λ affects the performance of SD-HCF and HCF, we vary λ from 0 to 1 and set the interval to 0.1. Figure 4, above which some parameters are given, shows $NMAE$ and $FMeasure$ values of RTT and failure rate varying with the increase in λ. Curves of $NMAE$ (or $FMeasure$) exhibit similar variation tendency. The curves hit a trough (or reach a peak) when the λ value is intermediate. λ value of the minimal $NMAE$ (or the

maximal $FMeasure$) changes with the specific cube density and QoS values. Therefore, we can utilize *cross-validation approach* to determine the optimal λ value for the specific data. As is exhibited in Fig. 4, SD-HCF is always better than HCF except some points in Fig. 4b. However, the minimal $NMAE$ of failure rate of SD-HCF is smaller than the minimal value of HCF. So, Tables 3, 5 and Fig. 4 are sufficient to lead us the conclusion that SD-HCF has a great advantage over HCF, UCF and ICF.

6.3 Impact of service providers for $WSRec$

SD-HCF proposed in this paper considers the impact among services with the same provider. Since a provider hosts a small quantity of services in WS-DREAM data, all Web services with the same provider are added to the nearest-neighbor set in the experiments. The impact of service providers on prediction performance is investigated. Figure 5 gives the comparison results in scenes with and without the impact of service providers. We set k to 10, density to 20% and active consumer to 10. It is observed that SD-HCF with the impact of service providers obtains smaller $NMAE$ values and larger $FMeasure$ values than that without the impact of service providers. The $NAME$ values and $FMeasure$ values are equal in many cases due to two reasons: (1) a lot of services do not have the service with the same provider and (2) services with the same providers have been included in the k nearest neighbors set. Anyway, selecting k nearest neighbors by considering the impact of service providers is able to improve prediction performance.



(a) $NMAE$ of RTT **(b)** $NMAE$ of Failure-rate

(c) $FMeasure$ of RTT **(d)** $FMeasure$ of Failure-rate

Fig. 5 Comparison between scenes with and without the impact of service providers

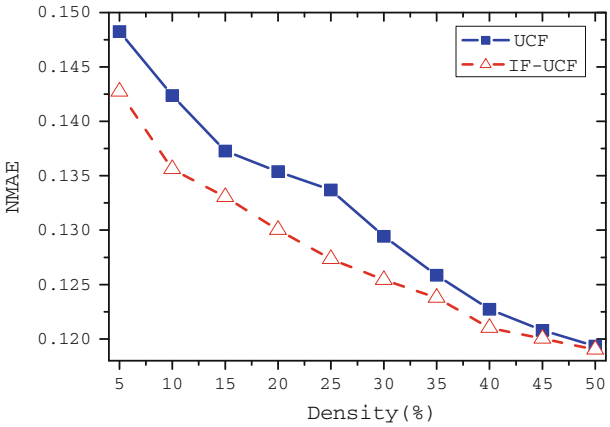


Fig. 6 Comparison of $NMAE$ between IF-UCF and UCF with the alteration of density

6.4 Performance of IF-UCF for $PCRec$

The experiments in this section compare IF-UCF with UCF to illustrate the effectiveness of inverse consumer frequency. $NMAE$ is employed as the performance metric. Consumer-provider matrix $\boldsymbol{V}$ is divided into training set and test set. The percentage of training set to the whole dataset is called density. We range density from 5 to 50 %, and the interval is set to 5 %. Figure 6 shows comparison between IF-UCF and UCF on $NMAE$ varying with the density. The comparison of the plots in Fig. 6 indicates that the $NMAE$ of IF-UCF is always smaller than UCF. Figure 6 also shows that with the increase in density, the difference of $NMAE$ between IF-UCF and UCF decreases, because both IF-UCF and UCF are ready to make accurate prediction utilizing abundant data in training set.

7 Conclusion

We have presented a novel data model and two algorithms for bidirectional Web service recommendation. Different from existing approaches, the data model is a three-dimensional cube which can explicitly describe relationship among providers, consumers and Web services. Based on the cube data model, the whole process of bidirectional Web service recommendation was summarized. We then presented Standard Deviation based Hybrid Collaborative Filtering (SD-HCF) for Web Service Recommendation ($WSRec$) and Inverse consumer Frequency based User Collaborative Filtering (IF-UCF) for Potential Consumers Recommendation ($PCRec$), respectively.

The SD-HCF algorithm utilizes PCC to measure the similarity and avoid overestimating the similarity. In k nearest neighbors selection, the SD-HCF algorithm eliminates the dissimilar neighbors and consider the impact of service providers. Our SD-HCF adopts standard deviation to adjust the weight of bias-from-mean of k nearest neighbors. The IF-UCF takes the number of consumers invoking services from a provider into consideration and employs $theinverseconsumerfrequency$ to adjust the weight of providers. In k nearest neighbors selection, the IF-UCF algorithm also eliminates the dissimilar neighbors.

At last, several experiments have been conducted based on real-world dataset provided by Planet-Lab. From these experimental results, the novel SD-HCF provides better recommendation quality than Collaborative Filtering (CF) based on user, CF based on item

and traditional HCF dramatically. Moreover, selecting k nearest neighbors by considering the impact of service providers can improve prediction performance. The prediction accuracy of IF-UCF is better than the UCF especially when the density of training set is small.

In the future work, the trustworthiness of the perceived QoS vector of consumer will be investigated. How to identify malicious users and produce trusted recommendation results will be further explored.

Acknowledgments This research is supported by National Natural Science Foundation of China (Nos. 71072172, 61103229, 61003074), Industry Projects in the Jiangsu S&T Pillar Program (No. BE2011198), Jiangsu Provicial Colleges and Universities Outstanding S&T Innovation Team Fund (No. 2001013), Key Project of Natural Science Research in Jiangsu Provincial Colleges and Universities (No. 12KJA520001), National Key Technologies R&D sub Program in 12th five-year-plan (No. SQ2011GX07E03990), International S&T Cooperation Program of China (No. 2011DFA12910), Program of Natural Science Foundation of Zhejiang Province (Nos. Z1100822, Y1110644, Y1110969, Y1090165) and Key Laboratory of Network and Information Security of Jiangsu Province of China (Southeast University) (No. BM2003201).

References

1. Ali K, Stam W (2004) TiVo: making show recommendations using a distributed collaborative filtering architecture. In: Proceedings of the 10th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'04), pp 394–401
2. Becchetti L, Colesanti UM, Marchetti-Spaccamela A, Vitaletti A (2011) Recommending items in pervasive scenarios: models and experimental analysis. Knowl Inf Syst (KAIS) 28(3):555–578
3. Bell RM, Koren Y (2007) Improved neighborhood-based collaborative filtering. In: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'07). California, pp 7–14
4. Bell RM, Koren Y (2007) Lessons from the Netflix prize challenge. SIGKDD Explor 9(2):75–79
5. Bezerra B, Carvalho F (2011) Symbolic data analysis tools for recommendation systems. Knowl Inf Syst (KAIS) 26(3):385–418
6. Chen X, Liu X, Huang Z, Sun H (2010) RegionKNN: a scalable hybrid collaborative filtering algorithm for personalized web service recommendation. In: 2010 IEEE international conference on web services, pp 9–16
7. Chun B, Culler D, Roscoe T, Bavier A, Peterson L, Wawrzoniak M, Bowman M (2003) Planetlab: an overlay testbed for broad-coverage services. ACM SIGCOMM comput Commun Rev 33(3):3–12
8. Fan T, Chang C (2010) Sentiment-oriented contextual advertising. Knowl Inf Syst (KAIS) 23(2):321–344
9. Fang L, Kim H, LeFevre K, Tami A (2010) A privacy recommendation wizard for users of social networking sites. In: Proceedings of the 17th ACM conference on computer and communications, Security, pp 630–632
10. Goldberg D, Nichols D, Oki BM, Terry D (1992) Using collaborative filtering to weave an information tapestry. Commun ACM 35(12):61–70
11. Gunawardana A, Meek C (2009) A unified approach to building hybrid recommender systems. In: Proceedings of ACM conference on recommender systems (RecSys'09). New York, USA, pp 117–124
12. Herlocker J, Konstan J, Terveen L, Riedl J (2004) Evaluating collaborative filtering recommender systems. ACM Trans Inf Syst 22(1):5–53
13. Jahrer M, Töscher A, Legenstein R (2010) Combining predictions for accurate recommender systems. In: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '10), pp 693–701
14. Kim H, Ji A, Ha I, Jo G (2010) Collaborative filtering based on collaborative tagging for enhancing the quality of recommendation. Electron Commer Res Appl 9(1):73–83
15. Konstan J, Miller B, Maltz D, Herlocker J, Gordon L, Riedl J (1997) GroupLens: applying collaborative filtering to Usenet news. Commun ACM 40(3):77–87
16. Koren Y (2008) Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'08), pp 426–434
17. Koren Y (2009) Factor in the neighbors: scalable and accurate collaborative filtering. ACM Trans Knowl Discov Data 4(1):1–24

18. Leung C, Chan S, Chung F, Ngai G (2011) A probabilistic rating inference framework for mining user preferences from reviews. World Wide Web Internet Web Inf Syst 14(2):187–215
19. Li C, Li L (2007) Optimization decomposition approach for layered QoS scheduling in grid computing. J Syst Archit 53(11):816–823
20. Li Y, Liu L, Li X (2005) A hybrid collaborative filtering method for multiple-interests and multiple-content recommendation in E-Commerce. Expert Syst Appl 28(1):67–77
21. Linden G, Smith B, York J (2003) Amazon.com recommendations: item-to-item collaborative filtering. IEEE Intern Comput 7(1):76–80
22. Malik Z, Rater BA (2009) Credibility assessment in web services interactions. World Wide Web Internet Web Inf Syst 12(1):3–25
23. Mclaughlin MR, Herlocker JL (2004) A collaborative filtering algorithm and evaluation metric that accurately model the user experience. In: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in, information retrieval(SIGIR'04), pp 329–336
24. Melville P, Mooney R, Nagarajan R (2002) Content-boosted collaborative filtering for improved recommendations. In: Proceedings of the 18th national conference on, artificial intelligence (AAAI'02), pp 187–192
25. Sarwar B, Karypis G, Konstan J, Riedl J (2001) Item-Based collaborative filtering recommendation algorithms. In: Proceedings of the 10th international world wide web conference, pp 285–295
26. Sprent P, Smeeton NC (2000) Applied nonparametric statistical methods, 3rd edn. Chapman & Hall, London
27. Tao Y, Zhang Y, Lin K (2007) Effective algorithms for web services selection with end-to-end QoS constraints. ACM Trans Web (TWEB) 1(1):1–26
28. Wu Z, Deng S, Li Y, Wu J (2009) Computing compatibility in dynamic service composition. Knowl Inf Syst (KAIS) 19(1):107–129
29. Wu Z, Luo J, Song A, Cao J (2009) QoS guaranteed service resource co-allocation and management. J Softw 12(20):3150–3162
30. Yang S, Zhang L, Giles C (2011) Automatic tag recommendation algorithms for social. ACM Trans Web (TWEB) 5(1): 31 Article 4
31. Zeng L, Benatallah B, Ngu AH, Dumas M, Kalagnanam J, Chang H (2004) QoS-aware middleware for web services composition. IEEE Trans Softw Eng 30(1):311–327
32. Zhang S, Jin Z, Zhu X (2011) Missing data imputation by utilizing information within incomplete instances. J Syst Softw 84(3):452–459
33. Zheng Z, Ma H, Michael R, King I (2009) WSRec: a collaborative filtering based web service recommender system. In: IEEE International Conference on Web Services, pp 437–449
34. Zheng Z, Michael R (2010) Collaborative reliability prediction for service-oriented systems. In: Proceedings of the ACM/IEEE 32nd international conference on software engineering (ICSE'10). Cape Town, South Africa, pp 35–44
35. Zhu X, Zhang S, Jin Z, Zhang Z, Xu Z (2011) Missing value estimation for mixed-attribute data sets. IEEE Trans Knowl Data Eng 23(1):110–121

Author Biographies

Jie Cao received the Ph.D. degree from Southeast University, China, in 2002. He is currently a professor and the director of Jiangsu Provincial Key Laboratory of E-Business at Nanjing University of Finance and Economics. He has been selected in the Program for New Century Excellent Talents in University (NCET) and awarded with Young and Mid-aged Expert with Outstanding Contribution in Jiangsu province. His main research interests include cloud computing, business intelligence and data mining. He has published one book and more than 40 referred papers in various conferences and journals. He has taken charge of over 20 national projects and obtained Jiangsu Provincial Scientific and Technological Progress Second Prizes Awarded (2008).

Zhiang Wu received the Ph.D. degree from Southeast University, China, in 2009. He is currently an associate professor of Jiangsu Provincial Key Laboratory of E-Business at Nanjing University of Finance and Economics. He is the member of the CCF and ACM. Dr. Wu is a recipient of the Excellent Student Paper Award of IFIP Intl. Conf. on Network Parallel Computing (2007). His recent research interests include network computing, recommender systems and data mining. Dr. Wu has authored more than 20 referred papers in international journals and conferences including *SIGKDD, TSMC, RecSys, WWWJ,* etc.

Youquan Wang is currently Ph.D. candidate in the School of Computing Science and Engineering at Nanjing University of Science and Technology, Nanjing, China. He received a B.E. degree from Nanjing University of Finance and Economics, Nanjing, China, in 2007 and an M.E. degree from Nanjing University of Finance and Economics, Nanjing, China, in 2010. His research interests include data mining and machine learning.

Yi Zhuang is a recipient of the CCF Doctoral Dissertation Award conferred by Chinese Computer Federation in 2008 and IBM Ph.D. Fellowship 2007–2008. He is currently an associate professor at the College of Computer & Information Engineering in Zhejiang Gongshang University where he joined as faculty member since May 2008. He obtained his Ph.D. degree in computer science from Zhejiang University in March 2008. From January 2008 to March 2008, supported by IBM Ph.D. Fellowship, Dr. Zhuang has spent three months to participate in the study of an optimal hybrid storage model based on DB2 as a research intern in IBM China Research Lab. His research interests mainly focus on multimedia database and cloud computing.