RegionKNN: A Scalable Hybrid Collaborative Filtering Algorithm for Personalized Web Service Recommendation

Xi Chen, Xudong Liu, Zicheng Huang, and Hailong Sun
School of Computer Science and Engineering
Beihang University
Beijing, China
{chenxi, liuxd, huangzc, sunhl}@act.buaa.edu.cn

***Abstract*—Several approaches to web service selection and recommendation via collaborative filtering have been studied, but seldom have these studies considered the difference between web service recommendation and product recommendation used in e-commerce sites. In this paper, we present RegionKNN, a novel hybrid collaborative filtering algorithm that is designed for large scale web service recommendation. Different from other approaches, this method employs the characteristics of QoS by building an efficient region model. Based on this model, web service recommendations will be generated quickly by using modified memory-based collaborative filtering algorithm. Experimental results demonstrate that apart from being highly scalable, RegionKNN provides considerable improvement on the recommendation accuracy by comparing with other well-known collaborative filtering algorithms.**

***Keywords*-web service recommendation; personalization; collaborative filtering; QoS**

I. INTRODUCTION

Web service, a software system designed to support interoperable machine-to-machine interaction over a network [1], is of great importance in Service Oriented Architecture (SOA). As cloud computing, commonly adopting SOA as its architecture, has become a hot issue and widely attracted attentions from both industry and academia, the number of public web services is increasing.

Web service discovery has been extensively studied, which mainly deals with functional properties [14], i.e., locating services that can meet a specified functional description. However, due to the large amount of services with identical or similar functionalities, users will be overwhelmed by the candidates. While web service discovery alone cannot tackle this problem, effective approaches to web service selection and recommendation have become more necessary, which is a key issue in the field of service computing [2].

With respect to the distributed and dynamic nature of web services, user preferences and more service properties should be considered in the phase of service selection, especially the non-functional service properties, also known as Quality of Service (QoS). QoS is a set of properties including response time, price, reputation, correctness, etc. Many researchers propose that QoS should be a key factor in the success of building critical service-oriented applications [3,4]. However, it's not practical for a consumer to acquire the comprehensive QoS information of each candidate service from his friends, nor to evaluate them by himself, since it is time-consuming, and some properties are hard to measure through several web service invocations, such as reputation and reliability.

Several work [5,6,7,14] has been done to apply collaborative filtering (CF) in web service recommendation. In these approaches, QoS values are predicted for an active user based on the QoS records provided by users who have similar historical QoS experiences on some web services. However, they failed to recognize the characteristics of QoS. According to our observation, QoS, regarded as a set of user-perceived properties, highly relates to users' physical locations. For example, in December, 2006, Chinese users had no access to those web services provided by U.S. companies, because Taiwan earthquake disrupted the Internet and telecoms connectivity in parts of Asia, while users from other parts of the world were not affected by this disaster.

To address this problem, we propose RegionKNN, an innovative scalable hybrid CF algorithm, for QoS-based web service recommendation. Our algorithm will first cluster users into several regions based on their physical locations and historical QoS similarities. Then region-sensitive services are identified. After that, modified nearest neighbor-based approach is used to automatically predict the QoS of the candidate web services for an active user by leveraging historical QoS information gathered from users of highly correlated regions. Based on the prediction, the service with the best predicted QoS will be recommended to the active user.

The main contributions of this paper include: 1) We propose a new region model for clustering users and identifying region-sensitive web services. 2) We design a hybrid model-based and memory-based CF algorithm for web service recommendation, which significantly improves the recommendation accuracy comparing to traditional CF algorithms as well as the state-of-the-art one. 3) We demonstrate RegionKNN's scalability advantage over traditional CF algorithms via time-complexity analysis.

The remainder of this paper is organized as follows. Section Ⅱ introduces the concept of collaborative filtering and related work on web service recommendation. Section

978-0-7695-4128-0/10 $26.00 © 2010 IEEE
DOI 10.1109/ICWS.2010.27

Ⅲ states the motivating scenario. Section Ⅳ describes the proposed approach in detail. Section Ⅴ presents experiments and results. Finally we conclude our work in Section Ⅵ.

II. BACKGROUND

In this section, we will briefly introduce the concept of collaborative filtering and the related work on web service recommendation.

A. Collaborative Filtering

Collaborative Filtering, firstly proposed by Rich [15], is widely used in commercial recommender systems like Amazon.com [10]. The basic idea of CF is to predict and recommend the potential favorite items for a particular user by leveraging rating data collected from similar users. Formally, a CF domain consists of a set of n users $\{u_1, u_2, ..., u_n\}$, a set of m items $\{i_1, i_2, ..., i_m\}$, and users' ratings on items, which is often denoted by a *user-item* matrix as depicted in Table Ⅰ. Entry $r_{x,y}$ ($1 \le x \le n$, $1 \le y \le m$) in this matrix represents user x' s rating on item y. The rating score always has a fixed range, like 1-5, and 0 means the user does not rate the corresponding item. Since users only express their preference on a small number of items, the matrix is very sparse.

TABLE I. USER-ITEM MATRIX

	i_1	i_2	…	i_m
u_1	0	0	…	3
u_2	2	3	…	0
…	…	…	…	…
u_n	1	0	…	4

Essentially, CF is based on processing the *user-item* matrix. In [16], Breese et al. introduce a classification of CF algorithm that divides them into two broad classes: *memory-based* algorithms and *model-based* algorithms.

Memory-based algorithms such as item-based KNN [10] and user-based KNN [11,17] use the entire user-item matrix when computing recommendations. These algorithms are easy to implement, require little or no training cost, and can easily take new user's ratings into account. But they cannot cope well with large number of users and items, since their online performance is often slow.

Alternatively, model-based algorithms, including K-means clustering [13], Bayesian model [12], etc., always learn the model from the dataset using statistical and machine learning techniques. These algorithms can quickly generate recommendations and achieve good online performance. However, the drawback is that the model must be performed anew when new users or items are added to the matrix.

B. Web Service Recommendation

Recommendation techniques have been used in recent research projects to enhance web service discovery. In [20], Mehta, Niederee, Stewart, Muscogiuri, and Neuhold find that semantics and syntax are insufficient to discover a service that meet users' needs, and they add two more dimensions of service description: quality and usage pattern. Based on this service description, they propose an architecture for recommendation-based service mediation. In [8], Blake and Nowlan propose to compute a web service recommendation score by matching strings collected from the user's operational sessions and the description of the web services. Based on this score, they judge whether a user is interested in the service. In [9], Maamar, Mostefaoui and Mahmoud propose a model for the context of web service interactions, and highlight the resource on which the web service is performed. These approaches focus on providing a mechanism to formalize users' preference, resource and the description of web services. With this predefined semantic and ontology models, recommendations are generated. Comparing with these methods, our approach is quite different, since the recommendations are generated by mining the QoS records that are automatically collected from interactions between users and services.

Limited work has been done to apply CF to web service recommendation. Shao et al. [6] propose a user-based CF algorithm to predict QoS values. Zheng, Ma, Lyu and King [5] propose a hybrid user-based and item-based CF algorithm to recommend web services. However, since neither of them recognizes the different characteristics between web service QoS and user ratings, the prediction accuracy of these methods is unsatisfied. Work [14,7] applies the idea of CF to their systems, and uses MovieLens [11] data for experimental analysis, thus their results are less compelling.

Different from the existing methods, which always suffer from low prediction accuracy and poor scalability, we propose an effective hybrid CF algorithm for web service recommendation with consideration of the region factor. Comprehensive experiments conducted with real QoS records show that our method outperforms others consistently.

III. A MOTIVATING SCENARIO

Suppose that Alice and Bob are two software engineers. As Fig. 1 depicts, Alice is working in India with a low bandwidth, and Bob is in America with a higher one. Both of them need an email filtering web service with low response time. Then they check the same public service registry located in America. From the recorded QoS values, they find that service *A* provided by an American provider outperforms others. After trying it, however, Alice finds that the response time of service *A* is much higher than her expectation, while Bob thinks *A* is what he wants.

Alice then recognizes that some QoS properties, like response time and availability, highly relate to the network environment of the region where she locates. She then turns to her colleagues, and they suggest service *B* provided by a local company based on their past experiences.

Our motivating problem is to make more accurate service recommendations for users located quite differently based on the collected QoS records.

To achieve this goal, several challenges need to be addressed. 1) How to build a region model that can reflect

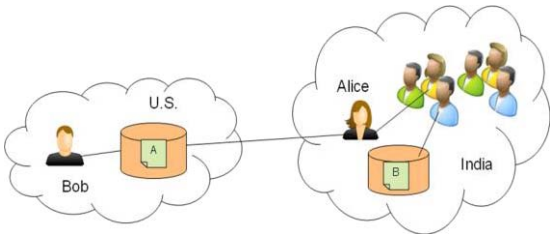


Figure 1. The motivating scenario

QoS characteristics? 2) How to refine CF algorithm to suit for web service recommendation with consideration of the region information? 3) How to verify the prediction results and use them in web service recommendation?

IV. THE REGIONKNN APPROACH

A. Problem Definition

The aim of our approach is to recommend web services with optimal QoS to the active user with consideration of the region factor. QoS is a set of user-perceived properties, including response time, also called round-trip time (RTT), price, availability, etc. Different properties have different characteristics. For instance, service price is comparatively stable, while RTT and availability highly relate to the user's physical location. We mainly focus on the latter kind of properties, and take RTT as an example to describe our approach.

We define the notations that are used throughout this paper as follows:

$S = \{s_1, s_2, \ldots, s_m\}$ is a set of services with similar functionality, where $s_i (1 \le i \le m)$ denotes a service, and m is the number of all services.

$U = \{u_1, u_2, \ldots, u_n\}$ is a set of users in the database, where $u_i (1 \le i \le n)$ denotes a user, and n is the number of all users. u_a is an active user who has provided some RTTs and now needs web service recommendation.

$T_u = \{R_u(s_1), R_u(s_2), \ldots, R_u(s_m)\}$ is an RTT vector or a user's RTT profile associated with each user, where $R_u(s_i)$ $(1 \le i \le m)$ denotes the RTT of service s_i provided by user u. We set $R_u(s_i) = 0$ if it is unknown. $\overline{R}_u$ denotes the average RTT provided by user u. $region(u)$ denotes the region of user u.

RegionKNN is a hybrid of the model-based and memory-based CF algorithms and keeps the advantages of both types. This algorithm has two phases: region model building (offline) and the generation of QoS prediction (online).

B. Region Model Building

Region model is an intuitive model for compressing QoS data by clustering users into different regions. We define a *region* as a group of users who are closely located with each other and have similar RTT profiles.

Region-Sensitive Service Detection. From large number of QoS records, we observe that for a certain number of web services, their RTTs vary greatly from region to region, and some services are unavailable to some regions. Our objective is to distinguish those services from others.

The set of non-zero RTTs $\{R_1(s), R_2(s), \ldots, R_k(s)\}$ $(1 \le k \le n)$ collected from all users of service s is a sample from population $\boldsymbol{R}$. To estimate the mean μ and the standard deviation σ of $\boldsymbol{R}$, we use two robust measures: median and median absolute deviation (MAD). Median is the numeric value separating the higher half of a sample from the lower half. MAD is defined as the median of the absolute deviations from the sample's median.

$$MAD = median_i(|R_i(s) - median_j(R_j(s))|) \quad i = 1,\ldots,k,\ j = 1,\ldots,k. \tag{1}$$

The two estimators are:

$$\hat{\mu} = median_i(R_i(s)) \quad i = 1,\ldots,k \tag{2}$$

$$\hat{\sigma} = 1.4862 MAD_i(R_i(s)) \quad i = 1,\ldots,k. \tag{3}$$

Definition1. **[Region-Sensitive Service]** Let $R = \{R_1(s), R_2(s), \ldots, R_k(s)\}$ be the set of RTTs of service s provided by users from all regions. Service s is a sensitive service to region M *iff* $\exists R_j(s) \in R((R_j(s) > \hat{\mu} + 3\hat{\sigma}) \wedge region(j) = M)$, where $\hat{\mu} = median(R)$, and $\hat{\sigma} = 1.4862 MAD(R)$.

Definition2. **[Region Sensitivity]** The sensitivity of region M is the fraction between the number of sensitive services in region M over the total number of services.

Definition3. **[Sensitive Region]** Region M is a sensitive Region *iff* its region sensitivity exceeds the sensitivity threshold λ.

Definition4. **[Region center]** The center of region M is defined as the median vector of all the RTT vectors provided by users in region M.

Region Similarity. The similarity of two regions M and N is measured by the similarity of their region centers. In memory-based CF algorithms, Pearson Correlation Coefficient (PCC) is often employed to define the similarity of two users. PCC has a range of [-1, 1]. PCC>0 indicates the two users have similar preferences, while PCC<0 means that their preferences are opposite. Based on PCC, the similarity of the two region centers m and n is formally defined as:

$$Sim(m,n) = \frac{\sum_{s \in S(n) \cap S(m)} (R_m(s) - \overline{R}_m) \cdot (R_n(s) - \overline{R}_n)}{\sqrt{\sum_{s \in S(n) \cap S(m)} (R_m(s) - \overline{R}_m)^2} \sqrt{\sum_{s \in S(n) \cap S(m)} (R_n(s) - \overline{R}_n)^2}} \tag{4}$$

where $S(m)$ is the set of services invoked by users in region M, and $S(n)$ is the set of services invoked by users in region N. Since PCC only accounts for the RTT difference between the services invoked by both regions, it often overestimates the similarities when the two regions have few co-invoked services [17]. To adjust it, we use:

$$Sim'(m,n)=\frac{|S(m)\cap S(n)|}{|S(m)\cup S(n)|}Sim(m,n) \qquad (5)$$

where $|S(m)\cap S(n)|$ is the number of web services invoked by both regions, and $|S(m)\cup S(n)|$ is the number of web services invoked either by region M or by region N.

Region Aggregation Algorithm. Since each user only provides a limited number of QoS values, the QoS dataset of each region is very sparse, which always leads to poor recommendation. To address this problem, we propose region aggregation, which is a bottom-up hierarchical clustering algorithm [18]. It treats users with similar IP addresses as a region at the outset and then successively aggregates pairs of regions until the stopping criterion has been met.

The algorithm is shown in Fig. 2. C is an $N\times N$ matrix storing similarities between each two regions. To be more efficient, we use priority queues P to sort the rows of C in decreasing order of similarity. $P[k].MAX()$ returns the region in $P[k]$ that is most similar to region k. I is a vector indicates which regions are non-sensitive and still available to be aggregated. The result is stored as a list of aggregates in A. Function $SIM(r_i, r_j)$computes the similarity between two regions using adjusted PCC. $ISSENSITIVE(r)$ judges whether region r is sensitive, and $SIM(m, i, j)$ computes the similarity of region m with the aggregates of region i and j. After creating the aggregated regions r_i and $r_j (i\leqslant j)$, r_i is used as its representative.

We first compute similarity matrix C and vector I. In each iteration, the two most similar and non-sensitive regions are selected and aggregated, if their similarity exceeds threshold μ. Then similarity matrix C and vector I are updated. The algorithm executes at most N-1 steps, in case that all regions are non-sensitive, extremely correlates to each other and finally aggregates into one region. The time complexity of region aggregation algorithm is $O(N^2 logN)$, and N is the number of regions at the outset.

Algorithm 1 Region Aggregation

```
in: regions r_1,...,r_N
for n←1 to N - 1
    for i←n + 1 to N
        C[n][i].sim←SIM(r_n,r_i)
        C[n][i].index←i
    end for
   I[n].sensitivity←ISSENSITIVE(r_n)
    if I[n].sensitivity=0
        then I[n].aggregate←1
    else I[n].aggregate←0
    P[n]←priority queue for C[n] sorted on sim
end for
calculate and set the sensitivity and aggregate of I[N]
A ←[]
while true
        k_1←argmax_{k:I[k].aggregate=1} P[k].MAX().sim
        if k_1=null or sim<μ
            then return A
        k_2←P[k_1].MAX().index
        A.APPEND(<k_1, k_2>)
        I[k_2].aggregate←0
        P[k_1]←[]
        I[k_1].sensitivity ←ISSENSITIVE(k_1)
        if I[k_1].sensitivity=1
            then I[k_1].aggregate←0
                for each i with I[i].aggregate=1
                    P[i].DELETE(C[i][k_1])
                    P[i].DELETE(C[i][k_2])
                end for
        else
            for each i with I[i].aggregate=1 ∧ i≠k_1
                P[i].DELETE(C[i][k_1])
                P[i].DELETE(C[i][k_2])
                C[i][k_1].sim←SIM(i, k_1, k_2)
                P[i].INSERT(C[i][k_1])
                C[k_1][i].sim←SIM(i, k_1, k_2)
                P[k_1].INSERT(C[k_1][i])
            end for
end while
```

Figure 2. Region aggregation algorithm

C. Neighbor Selection and QoS Prediction

Searching neighbors of an active user is an important step of CF algorithm before making predictions. Traditional method [5,6,11] is to search the entire dataset, which suffers from poor scalability. Different from that, after the phase of region aggregation, thousands of users are clustered into a certain number of regions based on their physical locations and historical QoS similarities.

The feature of the group of users in a region is represented by the region center. So the similarity between the active user and users of a region is calculated by the similarity between the active user and the region center.

Moreover, it is more reasonable to predict the QoS value for an active user based on his region, for users in the same region are more likely to have similar QoS experiences on web services, especially on those region-sensitive ones. In order to compute the RTT prediction $\hat{R}_u(s_i)$ for the active user u and service s_i, we take the following steps:

- Get the active user's IP address and find the region the user belongs to. If no appropriate region is found, the active user will be treated as a member of a new region.
- Identify whether service s_i is sensitive to the specific region. If it is region-sensitive, then the prediction is generated from the region center:

$$\hat{R}_u(s_i)=R_{center}(s_i) \qquad (6)$$

- Otherwise, use adjusted PCC to compute the similarity between the active user and each region center that has evaluated service s_i, and find up to k most similar centers $\{c_1, c_2,\ldots, c_k\}$.

- If the active user's region center has the RTT value of s_i, i.e., $R_{center}(s_i) \neq 0$, the prediction is computed using the following equation:

$$\hat{R}_u(s_i) = R_{center}(s_i) + \frac{\sum_{j=1}^{k}(R_{c_j}(s_i) - \overline{R}_{c_j})Sim'(u,c_j)}{\sum_{j=1}^{k} Sim'(u,c_j)} \quad (7)$$

- Otherwise,

$$\hat{R}_u(s_i) = \frac{\sum_{j=1}^{k} R_{c_j}(s_i)Sim'(u,c_j)}{\sum_{j=1}^{k} Sim'(u,c_j)} \quad (8)$$

Note that previous CF-based web service recommendation algorithms [5,6] use the following equation, a rating aggregate method commonly adopted in recommendation systems [11,17], to predict the missing QoS value.

$$\hat{R}_u(s_i) = \overline{R}_u + \frac{\sum_{j=1}^{k}(R_{c_j}(s_i) - \overline{R}_{c_j})Sim'(u,c_j)}{\sum_{j=1}^{k} Sim'(u,c_j)} \quad (9)$$

However, we find it not applicable in our context. Because this equation is based on the assumption that each user's rating range is subjective and comparatively fixed (e.g., critical users always rate items with lower ratings), whereas the range of RTT varies largely from service to service. As a result, the average RTT of all services provided by user u cannot reveal the performance of a specific web service s_i. Instead, as (7) states, we turn to the RTT profile of the region center, and use its RTT of service s_i to predict the missing value.

The time complexity of RegionKNN can be divided into two parts: one for the offline model building, the other for the online prediction, which highly relates to the user experience. Let l be the number of regions, in the online part, O(l) similarity weight calculations are needed, each of which takes O(m) time. Therefore, the online time-complexity is $O(lm) \approx O(m)$. Compared to the user-based CF algorithm used in [5,6] with O(mn) online time-complexity, RegionKNN is more efficient and well suited for large dataset.

We employ the predicted QoS values in web service recommendation. When the active user requires a web service with specific functional description, the one with the optimal predicted QoS value will be recommended to him based on his region and the QoS values he has provided. We also provide the recommended service's performance in other regions for the active user as a reference.

V. EXPERIMENTS

QoS prediction is the fundamental of web service recommendation, and we conduct a set of experiments to examine the prediction accuracy of our new approach with other methods. Particularly, we address the following questions:

1) How do the thresholds λ and μ affect the performance of prediction? The two thresholds are used in the phase of region model building. Threshold λ determines whether a region is too sensitive to be aggregated, and μ judges whether the two regions are similar enough that can be aggregated. The two thresholds have direct influence on the number of aggregated regions, which highly relates to the prediction accuracy. Experiments are carried out to find the impact of the two thresholds.

2) How does the parameter k influence the prediction accuracy? Parameter k controls the number of similar regions that are employed in QoS prediction. Picking a larger k will result in too much noise for those active users who have high correlates, while a smaller one may cause poor predictions for those who only have low correlates. Experiments are conducted to examine the impact of this parameter on the performance of RegionKNN.

3) How does the sparsity of RTT matrix affect the prediction accuracy? In real situation, the amount of RTTs provided by users is limited. We vary the number of RTTs given by both active users and training users to study its influence.

4) How does our approach compare with other CF-based web service recommendation methods? We compare our approach to work [5,6] and item-based CF algorithm which is widely used in commercial recommendation systems [10].

A. DataSet

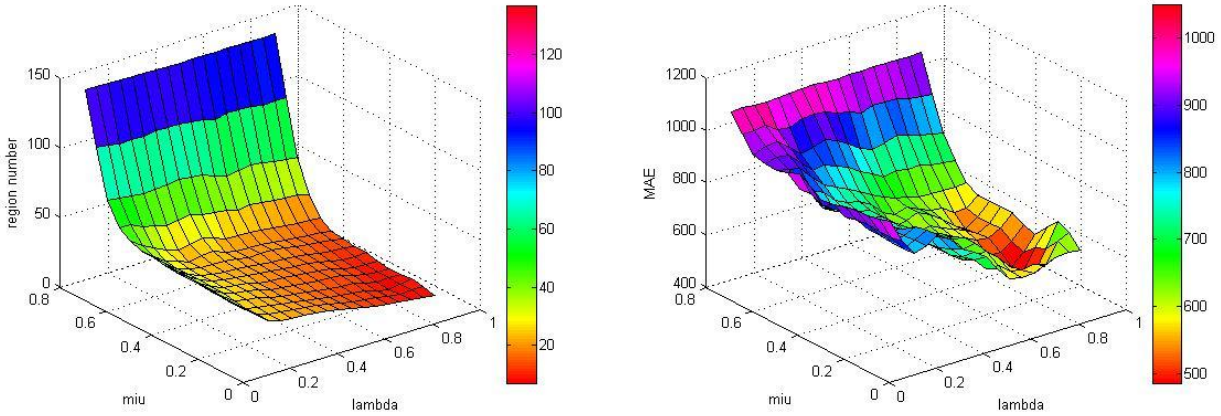
We adopt the WSRec [5] dataset, which contains about 1.5 million web service invocation records of 100 web services from more than 20 countries. RTT records are collected by 150 computer nodes from Planet-Lab, which are distributed in more than 20 countries. We extract a subset of 300,000 RTT records, which ranges from 2 to 31407 milliseconds. 3000 users are generated, each of whom is associated with a RTT profile of all the 100 services.

B. Evaluation Metric

Mean Absolute Error (MAE) is a statistical accuracy metric and commonly used in collaborative filtering to measure the prediction quality. MAE is the average absolute deviation of predictions to the ground truth data. For all test service RTTs and test users:

$$MAE = \frac{\sum_{u,s} | R_u(s) - \hat{R}_u(s) |}{L} \quad (10)$$

where $R_u(s)$ denotes the actual RTT of web service s given by user u, $\hat{R}_u(s)$ denotes the predicted one, and L denotes the number of tested RTTs. Smaller MAE means better prediction accuracy.



(a) (b)

Figure 3. (a) Impact of λ and μ on region number (b) Impact of λ and μ on MAE

The 3000 users are divided into two parts, one as training users and the rest as active (test) users. To simulate the real situation, we randomly remove some RTT entries of the training matrix to make a set of sparse matrices with density ranging from 10% to 50%. In terms of the active users, we vary the number of RTT values given by them from 10, 20 to 30, and name them *Given* 10, *Given* 20, and *Given* 30, respectively. The removed values of the active users are used to study the prediction accuracy. Each experiment is evaluated by 10 times 10-fold cross validation.

In 10-fold cross validation, the 3000 users are randomly partitioned into 10 subsets. In each fold, 9 subsets (90% of the users) are used as the set of training users, a single subset (10% of the users) is retained as the set of active users for validation. The process is repeated 10 times to predict the missing RTTs of test users based on RTTs of the corresponding training users. The advantage of this method is that all users are used for both training and validation, and each user is used for validation exactly once. To get a reliable error estimate, the 10-fold cross validation is performed 10 times in each experiment and the average MAE is reported.

C. Performance Evaluation

In order to show the performance of our approach to collaborative filtering, we compare RegionKNN to some traditional ones: user-based algorithm using PCC (UPCC) [6,19], item-based algorithm using PCC (IPCC) [10], and the state-of-the-art one: WSRec [5].

In this experiment, we set parameters in RegionKNN with λ = 0.8, μ = 0.3, k = 10. As to the parameter λ in WSRec, it achieves best performance when λ = 0.9 according to our experiment.

Table Ⅱ shows the prediction performance of different methods employing the 0.1 density training matrix. We observe that our new method significantly improves the prediction accuracy, and outperforms other methods consistently. Further, as the given number increases from 10 to 30, MAE becomes smaller which means that better accuracy is achieved by providing more QoS data.

TABLE II. MAE PERFORMANCE EVALUATION

Training User = 2700 Density = 0.1			
Method	Given 10	Given 20	Given 30
IPCC	2224.9841	2075.7625	2003.3556
UPCC	1280.9547	1145.8	1085.8475
WSRec	976.008	805.5951	772.3278
RegionKNN	**629.1583**	**621.9307**	**618.4066**

D. Impact of λ and μ

λ and μ play a very important role in the model building phase of RegionKNN. As discussed in Section Ⅳ, the two thresholds directly determine how many regions can be aggregated, and only those with the similarity higher than μ and the sensitivity less than λ are able to be aggregated. Moreover, λ and μ also have great impact on the final performance of RegionKNN.

In this experiment, we set the neighborhood size $k = \lceil 0.8 \times regionNumber \rceil$, density = 0.1. The threshold λ is varied from 0.1 to 0.9, and μ is varied from 0.1 to 0.8 both with a step of 0.1. Fig. 3 shows how λ and μ affect the number of regions and the final performance. It's obvious that lower μ and higher λ result in fewer regions and lower MAE, which means better prediction accuracy. Since a very sparse matrix is employed in this experiment, it is natural that regions only have low correlates, and the more regions are aggregated, the better prediction accuracy will be achieved.

Note that the optimal values of λ and μ are closely related to the dataset, especially its density. Moreover, smaller μ does not necessarily mean better performance, especially with dense training matrix. Because at that time, regions always have high correlates and smaller μ will cause too much noise.

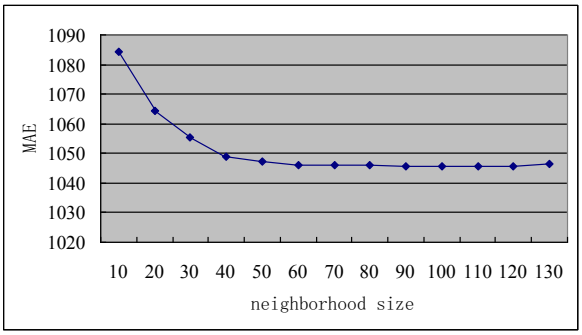


Figure 4. Impact of k (0.1 density)

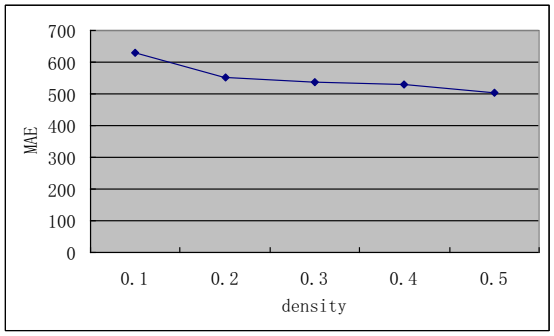


Figure 5. Impact of data sparsity (given 10)

E. *Impact of neighborhood size k*

Parameter k determines how many region neighbors are employed in the phase of QoS prediction, and highly relates to the prediction accuracy. In this experiment, we use 2700 training users to generate a training matrix with density 0.1. To get more regions, we set $\lambda = 0.1$, $\mu = 0.8$. After the model building phase, we obtain 137 regions. To study the impact of neighborhood size, we vary k from 10 to 130 with a step value of 10. As shown in Fig. 4, MAE decreases sharply with an increasing neighborhood size at the beginning, and then stays around 1045, since low correlates contribute little to the final prediction.

F. *Data Sparsity*

This experiment is carried out to investigate the effect of data sparsity on the performance of RegionKNN. Fig. 5 depicts its performance when the density of the training matrix is varied from 0.1 to 0.5, and better accuracy is achieved with denser matrix. Parameters of RegionKNN are tuned according to Table Ⅲ. We observe that the optimal value of λ is comparatively stable, while μ varies according to density, which indicates that the aggregation of highly correlated regions will result in good performance when the training matrix is not that sparse.

TABLE III. PARAMETER SETTING OF REGINKNN

Density	λ	μ	k
0.1	0.8	0.3	10
0.2	0.9	0.7	10
0.3	0.8	0.5	10
0.4	0.9	0.85	10
0.5	0.9	0.9	10

VI. CONCLUSTION AND FUTURE WORK

We have presented an innovative approach to web service recommendation. Different from other approaches, this algorithm employs the characteristics of QoS by building a region model. Based on this model, refined nearest-neighbor algorithm is used to generate QoS prediction and recommendation. Furthermore, the operator of the recommender system can tune the parameters in the region model to trade off speed and recommendation accuracy.

Experimental results show that compared with other well-known methods, RegionKNN significantly improves the prediction accuracy regardless of the sparsity of the training matrix. We also demonstrate its scalability advantage over traditional CF algorithms.

For future work, we plan to investigate more QoS properties and their variation with time. Moreover, since some QoS properties are correlated (e.g., availability and response time), we will conduct more research on their internal relations.

ACKNOWLEDGMENT

We would like to thank Zibin Zheng for the dataset he provided. This research is partly supported by China 863 program (No.2007AA010301, 2009AA01Z419), 973 program (No.2005CB321803), and the National Natural Science Foundation of China (No. 60731160632, 60903149).

REFERENCES

[1] H.Haas, and A.Brown, "Web services glossary," W3C Working Group Note 11, http://www.w3.org/TR/ws-gloss/.

[2] L.-J.Zhang, J.Zhang, and H.Cai, Service computing, Springer and Tsinghua University Press, 2007.

[3] O. Moser, F. Rosenberg, and S. Dustdar, "Non-intrusive monitoring and service adaptation for WS-BPEL [C]," Proc. 17th International Conference on World Wide Web (WWW 2008), pp. 815–824, 2008.

[4] M. P. Papazoglou and D. Georgakopoulos, "Service-Oriented computing," Communications of the ACM, pp.46(10):24–28, 2003.

[5] Z. Zheng, H. Ma, M.R. Lyu, and I.King, "WSRec: a collaborative filtering based web service recommendation system," Proc. 7th International Conference on Web Services (ICWS 2009), pp. 437-444, 2009.

[6] L.Shao, J.Zhang, Y.Wei, J.Zhao, B.Xie, and H.Mei, "Personalised QoS prediction for web services via collaborative filtering," Proc. 5th International Conference on Web Services (ICWS 2007), pp. 439-446, 2007.

[7] W.Rong, K.Liu, and L.Liang, "Personalised web service ranking via user group combining association rule," Proc. 7th International Conference on Web Services (ICWS 2009), pp. 445-452, 2009.

[8] M. B. Blake and M. F. Nowlan, “A web service recommender system using enhanced syntactical matching,” Proc. 5th International Conference on Web Services (ICWS 2007), 2007.

[9] Z.Maamar, S.K. Mostefaoui, and Q.H. Mahmoud. “Context for personalized web services,” In System Sciences. Proc. the 38th Annual Hawaii International Conference, 2005.

[10] G. Linden, B. Smith, and J. York, “Amazon.com recommendations: item-to-item collaborative filtering,” IEEE Internet Computing, Jan./Feb. 2003.

[11] B.N. Miller, I. Albert, S.K. Lam, J.A. Konstan, and J. Riedl, “MovieLens unplugged: experiences with an occasionally connected recommender system,” Proc. Int’l Conf. Intelligent User Interfaces, pp.263-266, 2003.

[12] Y.-H. Chien and E.I. George, “A bayesian model for collaborative filtering,” Proc. Seventh Int’l Workshop Artificial Intelligence and Statistics, 1999.

[13] L.H. Ungar and D.P. Foster, “Clustering methods for collaborative filtering,” AAAI Workshop on Recommendation Systems, pp.114-129, 1998.

[14] R. M. Sreenath and M. P. Singh. “Agent-based service selection,” Journal of Web Semantics, pp. 261–279, 2003.

[15] E.Rich, “User modeling via stereotypes,” Cognitive Science, Vol.3, No.4, 1979.

[16] J.S.Breese, D.Heckerman, and C.Kadie, “Empirical analysis of predictive algorithms for collaborative filtering,” Proc. 14th Conference on Uncertainty in Artificial Intelligence(UAI-98), pp. 43-52, July 1998.

[17] M.R. McLaughlin and J. L. Herlocker, “A collaborative filtering algorithm and evaluation metric that accurately model the user experience,” SIGIR, pp.329-336, 2004.

[18] Christopher D. Mining, Prabhakar Raghavan, and Hinrich Schütze, An Introduction to Information Retrieval, Cambridge University Press, Cambridge, England, 2009.

[19] Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordan, and John Riedl, “GroupLens: applying collaborative filtering to usenet news,” Communications of the ACM, pp.40(3):63-65, March 1997.

[20] B. Mehta, C. Niederee, A. Stewart, C. Muscogiuri, and E. J.Neuhold, “An architecture for recommendation based service mediation,” ICSNW, pp.250-262, 2004.