A collaborative framework of web service recommendation with clustering-extended matrix factorisation

Yueshen Xu, Jianwei Yin and Ying Li*

School of Computer Science and Technology, Zhejiang University, Hangzhou City, Zhejiang Province, China
Email: xyshzjucs@gmail.com
Email: zjuyjw@zju.edu.cn
Email: cnliying@zju.edu.cn
*Corresponding author

Abstract: QoS-based web service recommendation is an important technique to select suitable services to users. In this paper, we aim to achieve superior recommendation accuracy by leveraging the known QoS records. To achieve this goal, we employ the clustering algorithm and Matrix Factorisation model (MF), and propose a collaborative framework of web service recommendation. Using the clustering algorithm, we cluster users and services into different clusters based on their QoS records, and identify similar cluster centres for each user and each service. We propose two clustering-extended MF models, i.e., service clustering-extended MF model (SC-EMF) and user clustering-extended MF model (UC-EMF). In both models, the QoS values are predicted by two parts. One is the invocation experience of the target service or user, and the other is that of the similar centres. The experimental results show the effectiveness of our models.

Keywords: service recommendation; QoS prediction; matrix factorisation; clustering; ensemble models; collaborative framework.


Biographical notes: Yueshen Xu is a full-time PhD candidate in the College of Computer Science and Technology, Zhejiang University, China. He received his bachelor degree in XiDian University, China in 2011. His research interests include service computing, machine learning and business process management. He has published several papers in international conferences and journals, including WISE, WAIM, IEEE SCC, SOCA, ACM DEBS, AMIS, etc.

Jianwei Yin is a Full Professor in the College of Computer Science and Technology, Zhejiang University, China. He received his PhD degree in Zhejiang University in 2001. His research interests include service computing, cloud computing, middleware, business process management, etc. He has published more than 40 journal or conference papers. Some of his works were published in IEEE Internet Computing, IEEE Transaction on Service Computing, IEEE Transaction on Industrial Informatics, IEEE Industrial
Y. Xu, J. Yin and Y. Li

Electronics Magazine, KAIS, IEEE ICWS, IEEE CloudCom, ACM DEBS and ACM CIKM, etc. He has been a PC member over 20 international conferences, and as a reviewer of several international journals.

Ying Li received the BS, MS and PhD degrees in Computer Science from Zhejiang University, China, in 1994, 1997 and 2000 respectively. He is currently an Associate Professor in the College of Computer Science and Technology, Zhejiang University, and a visiting Professor in the University of California, Santa Barbara. His research interests include service computing, cloud computing and business process management. He published more than 30 journal or conference papers. Some of his works were published in Information Sciences, International Journal of Web Services Research, IEEE Transactions on Industrial Informatics, ICSOC, IEEE ICWS, IEEE SCC, etc.

1 Introduction

As a popular way to provide configurable resources on the Internet, Web services have become the underlying components in building enterprise application integration (EAI) and many other information systems (Kreger, 2003; Papazoglou, 2003; Papazoglou and Georgakopoulos, 2009). Following the service-oriented architecture (SOA), Web services are utilised in many cloud computing platforms as the accessible interfaces for various services, such as data storage service and data access service (Amazon Storage Service, 2015; Amazon Database Service, 2015; GoogleAdWords, 2015; Google Apps Script, 2015). As a result, the number of Web services is increasing dramatically, which leads to the difficulty in selecting suitable services for users. Efficient techniques of personalised Web service recommendation need to be developed urgently.

In Web service recommendation, quality of service (QoS for short) is a widely used evaluation criterion for service selection, especially in the case that candidate services have similar function (Yu et al., 2007). However, in many cases, a large part of QoS values are unknown due to the following reasons. First, the number of Web services is so large that it is time-consuming and expensive to invoke all of them. Second, the QoS values of a service are changeable due to the sensitivity to the configuration of computing resources. For instance, the average connection speed in Massachusetts is much faster than that in Arkansas (Akamai, 2015). When users invoke the same service for airline reservation, the users living in Massachusetts likely acquire shorter response time than that acquired by users living in Arkansas. Similarly, services configured with different computing resources usually provide different QoS to the same user. For example, many cloud computing companies offer various configuration schemes to service providers (Google Cloud Platform, 2015; Windows Azure, 2015). The services that are assigned more computing resources can provide better QoS. It is an urgent task to develop effective approaches to predict QoS values from both service side and user side.

As a mature technique to predict ratings in recommender systems, Collaborative Filtering (CF for short) has been also employed to predict QoS values (Shao et al., 2007; Zheng et al., 2009; Chen et al., 2010). There are three steps in the CF-based algorithm.
A collaborative framework of web service recommendation

The first step is to calculate the similarity of each pair of services (or users). The second step is to select similar neighbours based on the similarity. The last step is to get the final predicted value as the weighted average of the neighbours’ known values (Resnick et al., 1994; Sarwar et al., 2001; Su and Khoshgoftaar, 2009). Although CF-based approaches indeed achieve a large improvement in Web service recommendation, such approaches have the following defects. First, the similarity computation is hard to generate accurate results under the case of high data sparsity. In the “cold-start” case, that is, a user did not invoke any service, or a service was not invoked by any user, the similarity even cannot be computed (Lika et al., 2014). Second, the time complexity of similarity computation is quadratic to the data size, which is inefficient, especially in the case of the large dataset. So it is necessary to build more effective and efficient models.

Techniques based on Matrix Factorisation (MF for short) perform well in rating prediction for recommender systems (Salakhutdinov and Mnih, 2007; Ma et al., 2008; Ma et al., 2009; Diao et al., 2014), and have been used for QoS value prediction by service computing community in recent years (Lo et al., 2012b; Zheng et al., 2013). The main idea of MF-based approaches is to learn latent factors that impact QoS. In service recommendation, many researchers have demonstrated that the MF-based models can achieve higher prediction accuracy than CF-based models (Lo et al., 2012a; Zheng et al., 2013). However, the time complexity of existing models is quadratic to the data size due to the similarity computation of each pair of users or services. Thus, they are not efficient enough, especially in the era of big data. It is necessary to develop other ways of extending the MF model to improve the efficiency.

In Web service invocation, a large number of factors that determine QoS are related to IT configuration, such as network bandwidth and CPU performance. The users or services which have similar QoS records tend to own similar computing resources. Some types of computing configurations are typical and taken as standard schemes by service providers. For example, in many cloud computing platforms, standard configuration schemes are recommended to customers to choose (Google Cloud Platform, 2015; Windows Azure, 2015). Also, telecommunication operators, such as AT&T, offer standard Internet service packages (AT&T, 2015). Accordingly, we have two intuitions here. First, by analysing the QoS records, we can select the representative users and services, which have representative physical configurations. Second, their QoS values can reflect the features of QoS values of a service (or user) group. In this paper, such representative users and services are named centre users, centre services, or simply centres. The service invocation scenario is shown in Figure 1(a). We employ the K-Medoid algorithm to discover user and service groups. More details are given in Section 3.

To address the issues above, in this paper, we propose a collaborative framework of service recommendation based on clustering-extended MF models. We identify centre services and centre users using the clustering algorithm, and compute the similarity between each member (user or service) and the centres. For each user and each service, we select the centres that have similar QoS records as similar centres. Based on the MF model, we first propose the service clustering-extended MF model (SC-EMF for short). The QoS value is learned from two parts. One is the target service’s latent factors, and the other is the latent factors of the similar centres. Similarly, we propose the user
clustering-extended MF model (UC-EMF for short). The QoS value is also learned from two parts, which are the latent factors of the target user and his (or her) similar centres. In order to leverage as many invocation experiences as possible, we unify the two proposed models into an ensemble model by aggregating the results in two different ways. Note that, the time complexity of all our models is linear to the data size. We conduct extensive experiments on real-world datasets in comparison with existing well-known models. The results demonstrate that our proposed models can achieve superior prediction accuracy.

**Figure 1** Web service invocation scenario in real world (see online version for colours)

1.1 Our contributions

This paper makes the following contributions:

1. It proposes two clustering-extended MF models, which are verified to be efficient and effective. It also gives a detailed derivation procedure to gain optimal results from the proposed models. We believe that the derivation procedure will facilitate the implementation of our models in practice.

2. We propose two ways of combining the results of the two models together, which can achieve further higher prediction accuracy.

3. The time complexity of our models is linear to the data size, which is lower than that of the existing models. We believe that a low time complexity improves the applicability of our models, and gives high competitiveness with other models.

4. We conduct sufficient experiments on real-world datasets under various experimental cases, and the results demonstrate the effectiveness of our models.

The rest of this paper is organised as follows: Section 2 summarises related works. Section 3 states the clustering algorithm. Section 4 presents our two proposed models. Section 5 presents the two ensemble models. Section 6 demonstrates the experimental results. Section 7 concludes the paper and discusses the future work.
2 Related work

There are mainly two types of algorithms to predict QoS values for Web service. One is Collaborative Filtering (CF), which utilises the QoS records of similar neighbours. The other is Matrix Factorisation (MF), which explores the latent factors that determine QoS.

There are extensive CF-based approaches, which extend from the basic CF algorithm and become more competent in QoS prediction (Zheng et al., 2009; Chen et al., 2010; Chan et al., 2010; Cao et al., 2013; Xiong et al., 2014). Shao et al. (2007) distinguished the positive and negative correlations among users. The users that have a positive relation with the target user were regarded as similar neighbours, and the users in a negative relation were dissimilar neighbours. Based on that, they proposed a hybrid user-based CF model. Chen et al. (2010) partitioned users into different regions through a hierarchical clustering algorithm, and aggregated small regions into a bigger region. By identifying the user that was cluster centre, they computed the predicted value as the weighted average of QoS values of the region centres that were similar to the target user. Although this algorithm could achieve a good result, the hierarchical clustering algorithm was time-consuming and hard to be extended. In this paper, we use the K-Medoid algorithm, and leave enough space to enhance the clustering validation indices (Halkidi et al., 2001).

Zheng et al. (2009) built a Web service recommender system based on a hybrid CF-based model. They first computed two kinds of intermediate predicted results using user-based and service-based CF algorithms separately. Then, the intermediate results were combined together. They collected several real-world datasets, which significantly propelled the related research. Although the CF-based algorithms are not hard to implement, they suffer from bad prediction accuracy when the data density is sparse. Moreover, because of the similarity computation, the time complexity of CF-based algorithms is quadratic to the data size. Due to the high complexity, the CF-based algorithms cannot be applied on a large dataset. In contrast, as shown in the experiment results (see Section 6), our models can achieve superior prediction accuracy in the case of high sparsity. Also, as explained in Section 5.5, the time complexity of our models is linear to the data size.

The MF model and its extensions achieve good performance in recommender systems, and have been employed to predict QoS values in recent years (Zheng et al., 2013; Lo et al., 2012a; Xu et al., 2013). Zheng et al. (2013) proposed a user neighbourhood extended MF model named NIMF, which identified user neighbours through similarity computation of each pair of users. The predicted value was learned from the latent factors of the target user and his (or her) similar neighbours. Lo et al. (2012a) selected similar neighbours for each user and each service respectively through similarity calculation. They constructed two regularisation terms, which tried to minimise the difference between latent factors of the target service (or user) and the neighbours. Finally, they built a model named EMF_F. Apparently, due to the similarity computation, the time complexity of the two models (NIMF and EMF_F) is also quadratic to the data size. Xu et al. (2013) proposed a location-based model. For each user, their model selected the similar neighbours according to the geographic distance. The similarity between each user and each his neighbour was computed based on the distance with a predefined function. The predicted value was learned by both the latent factors of the target user and his neighbours. Although this model achieves good
prediction accuracy, it suffers from bad efficiency when the data size is large. Also, it is hard to define a suitable similarity function to measure the similarity according to the geographic information.

Clustering is a mature technique, which works on unlabeled data and can categorise multidimensional variables into different clusters (Han et al., 2011). As an unsupervised learning technique, there are diverse clustering algorithms, such as K-Means, K-Medoid, Spectral Clustering and Gaussian Mixture Model. The performance of the clustering algorithm is determined by various factors, including data distribution, variable dimension and some others. In this paper, we use the K-Medoid algorithm to cluster users and services. More details are given in Section 3.

3 Identifying centre users and centre services

In this paper, we choose to use the K-Medoid algorithm due to the following three reasons. First, the time complexity of K-Medoid is linear to the data size. Second, as demonstrated in (Han et al., 2011), the K-Medoid has high applicability on different datasets. That is, K-Medoid is relatively insensitive to the data distribution and variable dimension. Third, in this paper, our goal is to select a centre user or service that really exists in the dataset. Although as a popular clustering algorithm, the K-Means algorithm can also cluster data effectively, a cluster centre found by K-Means may be a virtual data point, which does not exist in the dataset. In our case, if we use the K-Means algorithm, the final centre user or service may not exist in the real data. In the rest of this section, we present the K-Medoid algorithm (see Section 3.1), and state the weight computation of centres (see Section 3.2).

3.1 The K-Medoid algorithm

Since the clustering algorithm is the same for users and services, we take the clustering procedure of services for example to explain. Let \( R = \{ r_{ij} \} \in \mathbb{R}^{M \times N} \) represent the user-service invocation matrix (see Figure 1(b)), where \( M \) and \( N \) denote the number of users and services respectively. For service clustering, \( R \) is regarded as a set of multidimensional data points, in which each column of \( R \) contains the QoS values of service \( j \). \( C = \{ c_{k} | 1 \leq k \leq K \} \) is a set consisting of cluster centres, where \( K \) is the number of clusters. \( L = \{ L_{k} | 1 \leq k \leq K \} \) is a set that includes each cluster \( L_{k} \), and the centre of \( L_{k} \) is \( c_{k} \). The K-Medoid algorithm is given in Algorithm 1.

In the initial phrase, the cluster centres are selected randomly, and each service is allocated to the initialised cluster based on the similarity with each centre (line 1, Algorithm 1). In the traditional K-Medoid algorithm, similarity is measured by the distance between two data points (Han et al., 2011). In our algorithm, the similarity between a service \( j \) and a centre service \( c_{k} \) is measured by Pearson Correlation Coefficient (PCC for short), which is computed with

\[
Sim(j, c_{k}) = \frac{\sum_{out}(r_{ij} - \bar{r}_{j})(r_{ik} - \bar{r}_{k})}{\sqrt{\sum_{out}(r_{ij} - \bar{r}_{j})^2} \sqrt{\sum_{out}(r_{ik} - \bar{r}_{k})^2}}
\] (1)
A collaborative framework of web service recommendation

where \( u \in U \) represents the users who invoked both service \( j \) and \( c_k \) before. \( r_{ujr} \) and \( r_{kucr} \) are the QoS values that user \( u \) acquired after the invocation. \( \overline{r}_j \) and \( \overline{r}_c \) are the means of the known QoS values of service \( j \) and \( c_k \). To avoid the overestimation of the similarity, equation (1) is extended to

\[
\text{Sim}'(j, c_k) = \frac{2 \times |U_j \cap U_{c_k}|}{|U_j| + |U_{c_k}|} \text{Sim}(j, c_k)
\]

where \(|U_j|, |U_{c_k}|, |U_j \cap U_{c_k}|\) separately represent the number of users who invoked service \( j, c_k \) and both of them before.

Algorithm 1 The K-Medoid Algorithm

**Input:** \( R \) : the set of QoS records, \( K \) : cluster number, \( A \) : iterative times

**Output:** \( L \) : a set of \( K \) clusters, \( C \) : a set of \( K \) cluster centers

1. Initialize: randomly select \( K \) cluster centers \( c_k \) \((1 \leq k \leq K)\)
2. For each service \( j \), assign \( j \) to cluster \( L_j \) with the largest \( \text{Sim}'(j, c_j) \) (\( c_j \) is the center of \( L_j \))
3. Compute \( \text{SimSum} \) as the sum of all \( \text{Sim}'(j, c_j) \)
4. while \( a < A \) do
5. Randomly select \( c_{\text{rand}} \) as the temporary center of \( L_{\text{rand}} \)
6. for each service \( j \) do
7. if \( c_j \neq c_{\text{rand}} \) then
8. if \( \text{Sim}'(j, c_{\text{rand}}) > \text{Sim}'(j, c_j) \) then
9. Assign \( j \) to \( L_{\text{rand}} \) temporarily
10. end if
11. else
12. For all \( c_k \) including \( c_{\text{rand}} \), select the largest \( \text{Sim}'(j, c_j) \) among all \( \text{Sim}'(j, c_k) \), and assign \( j \) to \( L_j \) temporarily
13. end if
14. end for
15. Compute \( \text{SimSum}' \) as the temporary sum of total similarity
16. if \( \text{SimSum}' > \text{SimSum} \) then
17. Assign \( c_{\text{rand}} \) as the center of \( L_{\text{rand}} \) formally
18. Keep the center assignment for each service permanently
19. \( \text{SimSum} \leftarrow \text{SimSum}' \)
20. else
21. Restore original assignment for each service
22. Restore \( c_{\text{rand}} \) as a common service
23. end if
24. \( a \leftarrow a + 1 \)
25. end while
26. Compute \( \text{Sim}'(j, c_k) \) between each service \( j \) and each center \( c_k \) \((1 \leq k \leq K)\)
27. return \( L \) and \( C \)
The main part of this algorithm is a procedure of iterations (lines 4–25). In each iteration, a service \( c_{\text{rand}} \) is randomly selected to temporarily replace the centre of cluster \( L_{\text{rand}} \) in which \( c_{\text{rand}} \) is (line 5). For each service \( j \), we compute the similarity \( \text{Sim}'(j, c_{\text{rand}}) \) between service \( j \) and \( c_{\text{rand}} \). For the services in different clusters with \( c_{\text{rand}} \), if \( \text{Sim}'(j, c_{\text{rand}}) \) is larger than the similarity between service \( j \) and its current cluster centre, service \( j \) will be reassigned to \( L_{\text{rand}} \) temporarily (lines 7–10). Otherwise, service \( j \) remains in the original cluster. For the service \( j \) in the same cluster with \( c_{\text{rand}} \), it is necessary to compute the similarity \( \text{Sim}(j, c_j) \) between service \( j \) and all other cluster centres \( c_k \) (lines 11–13). If \( \text{Sim}'(j, c_{\text{rand}}) \) is the largest, service \( j \) will be assigned to \( L_{\text{rand}} \). If \( \text{Sim}'(j, c_k) \) is the largest, service \( j \) will be assigned to \( L_k \). In the end of each iteration, the total sum of similarity \( \text{SimSum}' \) is computed (line 15). If \( \text{SimSum}' \) is larger than the similarity sum \( \text{SimSum} \) of the last iteration, \( c_{\text{rand}} \) will replace the original centre formally (lines 16–19). In contrast, if \( \text{SimSum}' \) is smaller, \( c_{\text{rand}} \) will enlarge the distance between services on the whole. In such a case, \( c_{\text{rand}} \) is ignored, and the original cluster centre will be restored, which means that this iteration changes nothing (lines 20–23). Similarly, we can use the same procedure to conduct clustering for users, and find centre users. In the end, we acquire the cluster set \( L \) and cluster centre set \( C \) (lines 26–27).

### 3.2 Weight calculation

After clustering and similarity computation, we define a set \( C(j) = \{c|c \in C \text{ and } \text{Sim}'(j, c) > 0\} \) that contains the similar centre services of service \( j \). The weight of each centre service \( c \) is

\[
w_j = \frac{\text{Sim}'(j, c)}{\sum_{i \in C(j)} \text{Sim}'(j, c_j)}
\]

Similarly, for each user \( i \), the similar centre users are selected and form the set \( C(i) = \{c|c \in C \text{ and } \text{Sim}'(i, c) > 0\} \). The weight of each centre \( c \) in \( C(i) \) is

\[
w_i = \frac{\text{Sim}'(i, c)}{\sum_{j \in C(i)} \text{Sim}'(i, c_j)}
\]

### 4 QoS prediction with matrix factorisation

In this section, we briefly review the MF model (see Section 4.1) and elaborate our proposed models (see Sections 4.2 and 4.3).

#### 4.1 Matrix factorisation

The MF model is a latent factor analysis model, which can factorise the high dimensional invocation matrix (see Figure 1(a)) into two low dimensional latent factor matrices in the same space. In the two latent factor matrices, each column represents the user or service latent factor vector, which needs to be learned. Each empty entry in the invocation matrix (i.e., the grey entry in Figure 1(a)) is predicted as the inner product of the corresponding user latent factor vector and service latent factor vector.
A collaborative framework of web service recommendation

Let $U \in \mathbb{R}^{D \times M}$ and $V \in \mathbb{R}^{D \times N}$ represent user and service latent factor matrices respectively, where $M, N$ are the number of users and services. $D$ is the number of latent factors and $D$ is much smaller than $M$ and $N$. The setting of $D$ is given in the experiment section (see Section 6.1). The invocation matrix $R$ is factorised as the product of $U$ and $V$ as

$$R \approx \tilde{R} = U^T V$$

(5)

where $\tilde{R} = \{\tilde{r}_{ij}\} (1 \leq i \leq M, 1 \leq j \leq N)$ is the estimated matrix of $R$, in which $\tilde{r}_{ij}$ is the predicted value of $r_{ij}$. The total prediction errors are minimised as

$$\min_{U, V} \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{N} I_{ij} (r_{ij} - U^T V)$$

(6)

where $U_i$ and $V_j$ are the $i$th and $j$th column of $U$ and $V$. $I_{ij}$ is an indicator function. $I_{ij}$ is equal to 1 if $r_{ij}$ is known and 0 otherwise. It is suggested to add two regularisation terms to forbid overfitting during the learning process, especially in the case of high data sparsity (Koren et al., 2009). We can get the MF model as follows:

$$\varepsilon(U, V) = \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{N} I_{ij} (r_{ij} - U^T V)^2 + \frac{\lambda_U}{2} \|U\|^2 + \frac{\lambda_V}{2} \|V\|^2$$

(7)

where $\|\|$ is Frobenius norm, and $\lambda_U, \lambda_V$ are small constants. The setting of $\lambda_U$ and $\lambda_V$ is given in the experiment section (see Section 6.1). We can gain a local minimum of the MF model using gradient decent algorithm.

4.2 Service clustering-extended matrix factorisation

In Section 3.2, we select the similar centre services for service $j$, and gain the weight of each similar centre service. Based on the discussion in the introduction section, as a cluster centre, the distribution of its QoS values can represent the distribution of QoS values of cluster members. In our problem, that is, each service $j$’s latent factors should be similar to those of its similar centres. To reflect such similarity relation, we first compute the predicted value learned from the similar centres:

$$\tilde{r}_j = \sum_{c \in X(j)} w_c U^T V_c$$

(8)

where $w_c$ represents the weight of centre service $c$ to service $j$ (see equation 3). To leverage the similar centres’ prediction value collaboratively, the missing value is learned by the following two parts:

$$r_{ij} \approx \tilde{r}_j = \alpha U^T V_j + (1 - \alpha) \sum_{c \in X(j)} w_c U^T V_c$$

(9)

Where $\tilde{r}_j$ is the estimated value, and $\alpha$ is a parameter to control the proportion of the two parts. The impact of the setting of $\alpha$ is studied in Section 6.4. We build the first proposed model service clustering-extended MF model (SC-EMF for short) as follows:

$$\varepsilon(U, V) = \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{N} I_{ij} \left( r_{ij} - \left( \alpha U^T V_j + (1 - \alpha) \sum_{c \in X(j)} w_c U^T V_c \right) \right)^2$$

(10)
To use the gradient descent algorithm, we need to get the partial derivatives. The derivative of equation (10) over $U_i$ is

$$
\frac{\partial E}{\partial U_i} = \sum_{j=1}^{N} I_{ij} \left( \alpha U_i^j V_j + (1-\alpha) \sum_{c \in c(j)} w_c U_i^j V_c - r_{ij} \right) 
\times \left( \alpha V_j + (1-\alpha) \sum_{c \in c(j)} w_c V_c \right) + \lambda_i U_i
$$

(11)

There are two forms of derivatives over $V_j$ oriented to common services and centre services respectively. For centre services, the derivative over $V_j$ is

$$
\frac{\partial E}{\partial V_j} = \alpha \sum_{i=1}^{M} I_{ij} U_i \left( \alpha U_i^j V_j + (1-\alpha) \sum_{c \in c(j)} w_c U_i^j V_c - r_{ij} \right) 
+ (1-\alpha) \sum_{i=1}^{M} \sum_{g \in G(j)} I_{ig} U_i^g \left( \alpha U_i^g V_g + (1-\alpha) \sum_{c \in c(g)} w_c U_i^g V_c - r_{ig} \right) 
\times \sum_{c \in c(g)} w_c U_i^g V_c - r_{ig} \right) + \lambda_V V_j
$$

(12)

where for the services contained in $G(j)$, their similar centre sets include the centre service $j$. Since it is impossible for common services to be in any similar centre set, the derivative over $V_j$ is more concise than equation (12) as

$$
\frac{\partial E}{\partial V_j} = \alpha \sum_{i=1}^{M} I_{ij} U_i \left( \alpha U_i^j V_j + (1-\alpha) \sum_{c \in c(j)} w_c U_i^j V_c - r_{ij} \right) 
+ \lambda_V V_j
$$

(13)

In our experiments, for all centre services, equation (12) is simplified into equation (13) due to the following two reasons. First, since centre services are in minority of the whole services, such simplification will not affect the final predicted results (see the experimental results in Section 6.3). Second, for facilitating the implementation, it is necessary to conduct the learning process in a unified manner. As a result, the time complexity of SC-EMF is further lower than that of existing models, such as NIMF (Zheng et al., 2013) and EMF_F (Lo et al., 2012a). The gradient descent algorithm is run iteratively as

$$
\begin{align*}
U_i' &= U_i - \gamma \times \frac{\partial E}{\partial U_i} \\
V_j' &= V_j - \gamma \times \frac{\partial E}{\partial V_j}
\end{align*}
$$

(14)

where $\gamma$ is the learning rate.

4.3 User clustering-extended matrix factorisation

For the proposed user clustering-extended model, the intuition is similar to that of SC-EMF. Since user $i$’s QoS values have similar distribution with that of similar centres, it
can be inferred that their latent factor vectors should be similar to some extent. To leverage such similarity relation, the QoS value is learned with the assistance of the latent factors of similar centre users. The average predicted value that is learned from the similar centre users is
\[
\tilde{r}_i = \sum_{c \in C(i)} w_c U_i^c V_j^c
\]  
(15)
where \( C(i) = \{ c | c \in C \text{ and } \text{Sim}(i, c) > 0 \} \) consists of the similar centre users of user \( i \). The predicted value \( r_{ij} \) is learned by two parts as
\[
r_{ij} \approx \tilde{r}_i = \beta U_i^j V_j + (1 - \alpha) \sum_{c \in C(i)} w_c U_i^c V_j^c
\]  
(16)
where \( \alpha \) is also a parameter to control the proportion of the two parts. The impact of the setting of \( \alpha \) is given in Section 6.4. We build the second proposed model user clustering extended MF model (UC-EMF for short) as
\[
\varepsilon(U, V) = \frac{1}{2} \sum_{i \in I} \sum_{j \in J} \left( r_{ij} - \left( \alpha U_i^j V_j + (1 - \alpha) \sum_{c \in C(i)} w_c U_i^c V_j^c \right) \right)^2
\]  
\[
+ \frac{\lambda_u}{2} \| U \|_F^2 + \frac{\lambda_v}{2} \| V \|_F^2
\]  
(17)
To achieve a local minimum of equation (17) by the gradient descent algorithm, we also need to acquire the partial derivatives. The derivative of equation (17) over \( V_j \) is
\[
\frac{\partial \varepsilon}{\partial V_j} = \sum_{i \in I} \sum_{j \in J} \left( \alpha U_i^j V_j + (1 - \alpha) \sum_{c \in C(i)} w_c U_i^c V_j^c - r_{ij} \right) \times \left( \alpha U_i^j + (1 - \alpha) \sum_{c \in C(i)} w_c U_i^c \right) + \lambda_v V_j
\]  
(18)
The derivative over \( U_i \) has two forms oriented to common users and centre users. For common users, the derivative is
\[
\frac{\partial \varepsilon}{\partial U_i} = \alpha \sum_{j \in J} I_i^j V_j \left( \alpha U_i^j V_j + (1 - \alpha) \sum_{c \in C(i)} w_c U_i^c V_j^c - r_{ij} \right) + \lambda_u U_i
\]  
(19)
For centre users, the derivative is
\[
\frac{\partial \varepsilon}{\partial U_i} = \alpha \sum_{j \in J} I_i^j V_j \left( \alpha U_i^j V_j + (1 - \alpha) \sum_{c \in C(i)} w_c U_i^c V_j^c - r_{ij} \right)
\]  
\[
+ (1 - \alpha) \sum_{b \in B(i)} \sum_{j \in J} I_{ib} w_b V_j \left( \alpha U_b^j V_j + (1 - \alpha) \right)
\]  
\[
\times \sum_{c \in C(b)} w_c U_c^j V_j^c - r_{ij} \right) + \lambda_u U_i
\]  
(20)
where for the common user \( h \) in \( H(i) \), his (or her) similar centre set contains the centre user \( i \). For the same reason for the simplification of the derivatives of \( SC-EMF \) (see Section 4.2), equation (19) is used for the centre users instead of equation (20). Using the gradient descent algorithm (see equation 14), we can find a local minimum of the \( UC-EMF \) model.

5 A collaborative framework of web service recommendation

In this section, we present our proposed collaborative recommendation framework, which provides a whole procedure for Web service recommendation (shown in Figure 2). As Figure 2 shows, the framework contains five parts, including QoS value collection, clustering, model building, model ensemble and service recommendation. This paper focuses on the three parts surrounded by the dash line in the box, including clustering, model building and model ensemble. All the three steps can be conducted with linear time complexity (see details in Section 5.5). In contrast, most of existing models are inefficient due to the quadratic complexity (Lo et al., 2012a; Xu et al., 2013). We give more details about the framework below.

In the QoS value collection, the QoS values are recorded after the invocation process, and organised into the user-service invocation matrix (see Figure 1(b)). Usually, the matrix is quite sparse, and the missing values (i.e., the grey entries in Figure 1(b)) are needed to be predicted. Based on the QoS values, the second step is to cluster services (or users) into different groups using clustering algorithm. We adopt the K-Medoid algorithm (see Algorithm 1 in Section 3.1), and further select the similar cluster centres for each service (or user). In model building, we propose two MF-extended models (\( SC-EMF \) and \( UC-EMF \)), and also explain the parameter estimation solution, i.e., the gradient descent algorithm. For the last two steps, i.e., model ensemble and service recommendation, we present the two steps in the rest of this section. Also, we discuss the time complexity.

Figure 2 The collaborative framework of web service recommendation (see online version for colours)

5.1 Comparison of different fusion ways

We explore several ways to combine \( SC-EMF \) and \( UC-EMF \) together due to the following reasons. First, we want to fully leverage the invocation experiences of all centres, including centre users and centre services. Second, previous works demonstrated
A collaborative framework of web service recommendation

that the ensemble of basic models likely generates a more effective model (Opitz and Maclin, 1999; Strehl and Ghosh, 2002). Our principle for model ensemble is that the time complexity of the ensemble model should be the same as that of each standalone model. We investigate two ways of model ensemble. The first way is model fusion, which integrates the latent factor vectors of similar centre users and services into a unified model. In detail, \( r_{ij} \) is learned as

\[
r_{ij} \approx \tilde{r}_i = \alpha_1 U_i^T V_j + \alpha_2 \sum_{c \in C(i)} w_c U_c^T V_j + \alpha_3 \sum_{c \in C(j)} w_c U_i^T V_c
\]

(21)

where \( \alpha_1, \alpha_2, \alpha_3 \) are parameters to control the proportions of the three parts, and \( \sum_{i, j}^3 \alpha_i = 1 (\alpha_i > 0) \).

The other way is result fusion, which combines the results of SC-EMF and UC-EMF together. In this paper, we adopt the second way due to the following reasons. First, the derivatives of equation (21) are complicated, which increases the difficulty of achieving prediction results. Second, SC-EMF and UC-EMF are independent prediction models, and both can achieve superior accuracy (see Tables 2 and 3 in Section 6.3). It is natural to get an aggregated result after the independent modelling of SC-EMF and UC-EMF. We will discuss two ways of result fusion

5.2 Static result fusion

In our first proposed way of result fusion, the weights of intermediate results are static. The predicted value \( r_{ij} \) is computed as

\[
r_{ij} \approx \tilde{r}_i = \beta \times \tilde{r}_{SC\text{-}MF} (i, j) + (1 - \beta) \times \tilde{r}_{UC\text{-}EMF} (i, j)
\]

(22)

where \( \beta (0 < \beta < 1) \) is a parameter, which is the same for all pairs of user \( i \) and service \( j \). \( \tilde{r}_{SC\text{-}MF} (i, j) \) and \( \tilde{r}_{UC\text{-}EMF} (i, j) \) are predicted results of SC-EMF and UC-EMF respectively. This model is named static ensemble-extended MF model (SE-EMF for short).

5.3 Dynamic result fusion

To control the weights of the two standalone models dynamically, we adopt the combination method proposed by Zheng et al. (2009). For user \( i \) and service \( j \), two confidence weights are computed to evaluate the prediction confidence for SC-EMF and UC-EMF:

\[
con_i = \frac{\sum_{c \in C(i)} Sim'(i, c)^2}{\sum_{c \in C(i)} Sim'(i, c)} \quad con_j = \frac{\sum_{c \in C(j)} Sim'(j, c)^2}{\sum_{c \in C(j)} Sim'(j, c)}
\]

(23)

where \( C(i), C(j) \) are similar centre sets, and \( Sim'(i, c), Sim'(j, c) \) are the similarity values (see equation (2) in Section 3.1). Two weights \( w_{ic} \) and \( w_{jc} \) are computed to control the proportion of the two models in the predicted value, which are presented below.

\[
w_{ic} = \frac{con_i \times \beta}{con_i \times \beta + con_j \times (1 - \beta)} \quad w_{jc} = \frac{con_j \times (1 - \beta)}{con_i \times \beta + con_j \times (1 - \beta)}
\]

(24)
where $w_{ic} + w_{jc} = 1$, and $\beta (0 < \beta < 1)$ is a parameter. Finally, $r_y$ is computed with

$$ r_y \approx \tilde{r}_y = w_r \times \tilde{r}_{SC\_MF} (i, j) + w_{jc} \times \tilde{r}_{UC\_EMF} (i, j) $$

(25)

where $\tilde{r}_{SC\_MF} (i, j)$ and $\tilde{r}_{UC\_EMF} (i, j)$ have the same meaning as in equation (22). The setting of $\beta$ is given in the experiment section, and the impact of $\beta$ is studied in Section 6.7. We name this model dynamic ensemble-extended MF model (DE-EMF for short).

5.4 Web service recommendation

The prediction results of SC-EMF and UC-EMF can be the input of ensemble models, or used for recommendation independently. If a service’s predicted QoS values are better than the given threshold, this service will be recommended to users. Let us see an example. A user is seeking services for weather forecast, and requires that the response time of the invocation must be shorter than 0.5s. After running our models, those services whose predicted response time (QoS value) is less than 0.5s will be recommended to the user. In this paper, we draw the recommendation decision based on a single type of QoS. For the problem of multivariate decision, in which a service is evaluated by several QoS properties jointly, please refer to the work on service composition (Paik et al., 2014).

5.5 Complexity analysis

The time complexity of our models comes from two sources. One is the clustering algorithm, and the other is the gradient descent algorithm. The time complexity of service clustering and user clustering is $O(K_sN)$ and $O(K_uM)$ respectively. $K_s, K_u$ are the number of service and user clusters, and $N, M$ are the number of services and users respectively.

Since the time complexity of the gradient descent algorithm for SC-EMF and UC-EMF is the same, we take SC-EMF as the example to explain. Because the iterative times in the gradient descent algorithm are a constant, the time complexity results from the derivatives (see equation (11) and equation (13) in Section 4.2). The time complexity of equation (11) is $O(\rho R \epsilon_D + \rho R \epsilon_s)$, and that of equation (13) is $O(\rho R \epsilon_s D)$. $\rho_R$ is the number of the known entries in the invocation matrix (see Figure 1(b)), $D$ is the number of latent factors and $\epsilon_s$ is less than or equal to $K_s$. $\epsilon_s$ represents the average number of similar centres of each service. The total time complexity is aggregated into $O(\rho R \epsilon_s D + \rho R \epsilon_s + K_s N)$, which can be simplified into $O(\rho R \epsilon_s D + K_s N)$. Similarly, the time complexity of UC-EMF is $O(\rho R \epsilon_s D + K_u M)$, where $\epsilon_u$ is the average number of similar centres of each user. Since $N$ and $M$ are much less than $\rho_R$ (see Table 1), the total complexity is linear to the data size. Table 1 reports the statistics of $N, M$ and $\rho_R$ under different data densities (The “data density” refers to the training set density, and see details in Section 6.1). The statistics in Table 1 are computed on the dataset used in our experiments.

For SE-EMF, since there is no other additional computation involved in, the time complexity is $O(\rho R (\epsilon_s + \epsilon_u) D + K_s N + K_u M)$. As for DE-EMF, the time complexity of the computation of $coni$ and $conj$ (see equation (24)) is $O(M + N)$. Thus, the final time complexity of DE-EMF is the same as that of SE-EMF. In summary, both SE-EMF and DE-EMF have linear scalability to the data size.
A collaborative framework of web service recommendation

6 Experiment and evaluation

In this section, we conducted sufficient experiments on two real-world datasets to answer the following questions: (1) How do our models compare with other well-known models? (2) How do the parameters impact the performance of our models?

6.1 Dataset and parameter setting

In the experiments, we use the two datasets released by Zheng et al. (2009). The two datasets contain 339 users, 5825 Web services, and their QoS values that were collected in a real world invocation scenario. One dataset records the response time values, and this dataset is named RT dataset. The other dataset records the throughput values, and this dataset is named TP dataset. Many works evaluated their models’ performance on the two datasets (Zheng et al., 2010; Lo et al., 2012a).

Training/Testing Set. In our experiment, the whole dataset is divided into two subsets for training and testing. We randomly select a part of data from the whole dataset to form the training set, and the remaining data is to be the testing set. In the experiment, we create six training sets with different data densities, i.e., 5%, 7.5%, 10%, 12.5%, 15% and 17.5%.

Parameter Setting. For all models, the default parameter setting is the same. That is, $\alpha$ is set to 0.4, the dimension (D) of $U_i$ and $V_j$ is set to 10, and $\beta$ is set to 0.4 (see $\beta$ in Section 5.3). The number of clusters is set to 20. $\lambda_U$ and $\lambda_V$ are both set to 0.001 in all experiments.

6.2 Evaluation metrics

We employ RMSE (Root Mean Squared Error) and MAE (Mean Absolute Error) as the evaluation metrics of prediction accuracy. RMSE and MAE are defined as follows:

$$RMSE = \sqrt{\frac{1}{\rho_{testing}} \sum_{i,j} (r_{ij} - \bar{r}_{ij})^2}, \quad MAE = \frac{1}{\rho_{testing}} \sum_{i,j} |r_{ij} - \bar{r}_{ij}|$$

where $r_{ij}$ is the known QoS value in the testing dataset, $\bar{r}_{ij}$ is the predicted value, and $\rho_{testing}$ denotes the total number of $r_{ij}$. A smaller RMSE value and MAE value mean better performance.

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Statistics of $M$, $N$ and $\rho_R$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M$</td>
<td>$N$</td>
</tr>
<tr>
<td>$N$/$\rho_R$</td>
<td>$M$/$\rho_R$</td>
</tr>
<tr>
<td>339</td>
<td>5825</td>
</tr>
<tr>
<td>Data Density = 15%</td>
<td>Data Density = 20%</td>
</tr>
<tr>
<td>$N$/$\rho_R$</td>
<td>$M$/$\rho_R$</td>
</tr>
<tr>
<td>1.97%</td>
<td>0.11%</td>
</tr>
</tbody>
</table>
6.3 Performance comparison

Several well-known models are compared to our models as baselines, including:

1. **UserMean**: $r_{ij}$ is predicted as the mean of all known QoS values of user $i$.

2. **ItemMean**: $r_{ij}$ is predicted as the mean of all known QoS values of service $j$.

3. **UPCC**: this model first computes the similarity of each pair of users, and selects a certain number of similar users for each user as his (or her) neighbours. $r_{ij}$ is predicted as the weighted average of all the neighbours’ QoS values (Resnick et al., 1994).

4. **IPCC**: this model is similar to UPCC, but the similarity is computed for each pair of services, and $r_{ij}$ is predicted as the weighted average of the neighbours’ QoS values (Sarwar et al., 2001).

5. **UIPCC**: this model integrates UPCC and IPCC into a unified model by aggregating their predicted results together (Zheng et al., 2009).

6. **MF**: this approach is the basic Matrix Factorisation model (Koren et al., 2009), and has been explicated in Section 4.1.

7. **EMF_F**: this model first identifies the similar neighbours of each user (or service), and then minimises the difference of latent factors of each user (or service) and the neighbours (Lo et al., 2012a).

8. **WLEMF**: this model computes the similarity between two users based on their geographic coordinates, and selects each user’s geographic neighbours. $r_{ij}$ is predicted by the invocation experiences of two targets, i.e., the target user and his (or her) neighbours (Xu et al., 2013).

<table>
<thead>
<tr>
<th>Table 2</th>
<th>Accuracy comparison on response time (a smaller value means better performance)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approach</td>
<td>Training Set Density (TD)</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>UserMean</td>
<td>RMSE</td>
</tr>
<tr>
<td>ItemMean</td>
<td>1.8588</td>
</tr>
<tr>
<td>UPCC</td>
<td>1.5726</td>
</tr>
<tr>
<td>IPCC</td>
<td>1.7331</td>
</tr>
<tr>
<td>UIPCC</td>
<td>1.6589</td>
</tr>
<tr>
<td>MF</td>
<td>1.5317</td>
</tr>
<tr>
<td>EMF_F</td>
<td>1.5371</td>
</tr>
<tr>
<td>WLEMF</td>
<td>1.5265</td>
</tr>
<tr>
<td>SC-EMF</td>
<td>1.4751</td>
</tr>
<tr>
<td>UC-EMF</td>
<td>1.4621</td>
</tr>
<tr>
<td>SE-EMF</td>
<td>1.3752</td>
</tr>
<tr>
<td>DE-EMF</td>
<td>1.3725</td>
</tr>
<tr>
<td></td>
<td>1.3802</td>
</tr>
</tbody>
</table>
Table 3  Accuracy comparison on throughput (a smaller value means better performance)

<table>
<thead>
<tr>
<th>Approach</th>
<th>Training Set Density (TD)</th>
<th>RMSE</th>
<th>RMSE</th>
<th>RMSE</th>
<th>RMSE</th>
<th>RMSE</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TD=5%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UserMean</td>
<td>110.4593</td>
<td>110.4991</td>
<td>110.5147</td>
<td>110.3946</td>
<td>110.4179</td>
<td>110.3572</td>
<td></td>
</tr>
<tr>
<td>ItemMean</td>
<td>66.575</td>
<td>65.6443</td>
<td>65.2560</td>
<td>64.6815</td>
<td>64.4674</td>
<td>64.3612</td>
<td></td>
</tr>
<tr>
<td>UPCC</td>
<td>69.3207</td>
<td>61.6455</td>
<td>57.4295</td>
<td>54.4661</td>
<td>52.3451</td>
<td>51.2218</td>
<td></td>
</tr>
<tr>
<td>IPCC</td>
<td>69.4329</td>
<td>63.8765</td>
<td>60.8373</td>
<td>58.7788</td>
<td>57.4926</td>
<td>56.1547</td>
<td></td>
</tr>
<tr>
<td>UIPCC</td>
<td>64.3460</td>
<td>59.3630</td>
<td>56.2403</td>
<td>53.6599</td>
<td>51.7780</td>
<td>50.6382</td>
<td></td>
</tr>
<tr>
<td>MF</td>
<td>56.1293</td>
<td>50.8476</td>
<td>47.8242</td>
<td>46.6344</td>
<td>45.0469</td>
<td>44.9052</td>
<td></td>
</tr>
<tr>
<td>EMF_F</td>
<td>54.8751</td>
<td>50.5114</td>
<td>47.7375</td>
<td>46.2906</td>
<td>44.8351</td>
<td>44.1905</td>
<td></td>
</tr>
<tr>
<td>WLEMF</td>
<td>53.8142</td>
<td>49.8561</td>
<td>47.2471</td>
<td>46.1552</td>
<td>44.6712</td>
<td>44.1011</td>
<td></td>
</tr>
<tr>
<td>SC-EMF</td>
<td>52.8493</td>
<td>49.3567</td>
<td>46.8543</td>
<td>45.7296</td>
<td>43.7796</td>
<td>43.4201</td>
<td></td>
</tr>
<tr>
<td>UC-EMF</td>
<td>53.4048</td>
<td>49.4486</td>
<td>46.9370</td>
<td>46.0282</td>
<td>43.9882</td>
<td>43.5986</td>
<td></td>
</tr>
<tr>
<td>SE-EMF</td>
<td>52.9443</td>
<td>48.9791</td>
<td>46.2452</td>
<td>44.9895</td>
<td>43.2559</td>
<td>42.6114</td>
<td></td>
</tr>
<tr>
<td>DE-EMF</td>
<td>52.6737</td>
<td>48.9222</td>
<td>46.2307</td>
<td>44.9945</td>
<td>43.2938</td>
<td>42.7355</td>
<td></td>
</tr>
</tbody>
</table>

Tables 2 and 3 present the experimental results of our models and the baselines on the RT dataset and TP dataset. The following observations can be made from Tables 2 and 3.

1. Our proposed models SC-EMF and UC-EMF consistently outperform all baseline models on the two datasets. For a detailed example, on average over all training set densities (i.e., 5% s 17.5%), UC-EMF achieves 8.53% improvement than UIPCC on the RT dataset and 15.57% improvement than UIPCC on the TP dataset. Such improvements indicate the following points: 1) For a target service (or a user) and the similar centres, their latent factor vectors are indeed similar to some extent. 2) Such similarity is leveraged appropriately in our proposed models. 3) The gradient descent algorithm for solving SC-EMF and UC-EMF is effective.

2. Both SE-EMF and DE-EMF perform further better, which verifies the effectiveness of model fusion. Let us see a detailed example. On average, DE-EMF achieves 9.08% improvement than UIPCC on the RT dataset and 16.94% improvement than UIPCC on the TP dataset. Also, DE-EMF achieves 5.56% improvement and 4.25% improvement than MF model on the two datasets.

3. Even though DE-EMF behaves better SE-EMF in most cases, their performance is close to each other on both datasets. Intuitively, fusing models in a dynamical way is usually more effective than in a static way. However, for easy implementation, it is enough to employ SE-EMF to acquire superior results.

4. Along with the increase of training set density, the prediction error is continuously to decrease. It is a natural trend, since more training data can provide more historical invocation information to learn latent factors more accurately.

In the following experiments, we investigate the impact of parameters, including α, cluster number, D and β.
6.4 Impact of $\alpha$

The parameter $\alpha$ controls the proportion of the two parts in equation (9) and equation (16). If $\alpha$ is larger than 0.5, the invocation experience of the target service (or user) tends to determine more in the predicted results. In contrast, if $\alpha$ is smaller than 0.5, the invocation experience of similar centres will make more contributions to the learning of missing values. We investigate the impact of $\alpha$ on models’ performance on the RT dataset. The other parameters are set to the values that are claimed in the parameter setting (see Section 6.1). The experimental results are shown in Figure 3.

As shown in Figures 3(a) and 3(b), SC-EMF and UC-EMF achieve the lowest prediction error in the range of 0.2 to 0.4. It illustrates that the latent factors of similar centres take a more important role in learning missing values. However, the latent factors of the target service (or user) cannot be ignored, since when $\alpha$ is smaller than 0.2 or 0.3, the performance of the two models becomes much worse. In the whole range (0.1 s 0.9), the prediction error of UC-EMF is lower than that of SC-EMF, especially when $\alpha$ is smaller than 0.4, but close to each other when $\alpha$ is larger than 0.5.

In Figures 3(c) and 3(d), it can be observed that both SE-EMF and DE-EMF achieve the lowest error at $\alpha = 0.3$. The performance of the two ensemble models is close to each other, and has the similar changing trend with the performance of SC-EMF and UC-EMF.

Figure 3 Impact of $\alpha$ (see online version for colours)
6.5 Impact of cluster number

The cluster number determines the number of similar centres. In this paper, we only leverage the latent factors of those centres that are similar to the target service (or user). We investigate the impact of cluster number on the RT dataset. The other parameters are set to the values that are claimed in the parameter setting (see Section 6.1). The experimental results are shown in Figure 4.

As Figures 4(a) and 4(b) demonstrate, for SC-EMF, the lowest prediction error is achieved at 20. However, both RMSE and MAE values change smoothly from 10 to 40. UC-EMF performs consistently better than SC-EMF. The performance of UC-EMF changes smoothly in the range of 5 to 40. It illustrates that our proposed models are not sensitive to the number of clusters, and can gain high performance in various settings of $D$. It is enough to achieve high prediction accuracy with a small number of similar centres (e.g., 20). The similar changing trend can be also observed in Figures 4(c) and 4(d). The performance of SE-EMF and DE-EMF is close to each other, and varies smoothly.

Figure 4 Impact of cluster number (see online version for colours)
6.6 Impact of $D$

The parameter $D$ denotes the number of latent factors. Ideally, only the factors that can really impact QoS should be involved in the learning process. However, it is the only way to acquire the proper value of $D$ through experiments. We investigate the impact of $D$ on the RT dataset. The other parameters are set to the values that are claimed in the parameter setting (see Section 6.1). The experimental results are shown in Figure 5.

As shown in Figures 5(a) and 5(b), for both SC-EMF and UC-EMF, RMSE and MAE values first decrease, and reach the minima within 6 to 14, and increase again. It can be inferred that the factors that influence QoS are not so many. But due to the variety of factors in the invocation process, it cannot be concluded that for all QoS properties, the factors impacting QoS are the same. The prediction accuracy of UC-EMF is higher than that of SC-EMF in most cases.

Figures 5(c) and 5(d) demonstrate that SE-EMF and DE-EMF also achieve the best performance within 6 and 14. Also, the two ensemble models have the similar changing trend in RMSE and MAE with SC-EMF and UC-EMF. The difference is that the performance of the two ensemble models changes more smoothly than that of the two standalone models. That is, SE-EMF and DE-EMF perform not only better but also more steadily.

Figure 5 Impact of $D$ (see online version for colours)
6.7 Impact of $\beta$

The parameter $\beta$ controls the proportion of the intermediate results of SC-EMF and UC-EMF in the final predicted values. In the static ensemble model (i.e., SE-EMF), the proportion is fixed, which is equal to $\beta$. In contrast, in the dynamic ensemble model (DE-EMF), the proportion is different for each pair of intermediate results. We investigate the impact of $\beta$ on the RT dataset and TP dataset. The other parameters are set to the values that are claimed in the parameter setting (see Section 6.1). The experimental results of SE-EMF are shown in Figure 6, and the experimental results of DE-EMF are shown in Figure 7.

Figure 6 Impact of $\beta$ in SE-EMF (see online version for colours)

As shown in Figure 6, for SE-EMF, the lowest prediction error is achieved at $0.4$ for both response time and throughput in the two training set densities (10% and 15%). It is implied that UC-EMF takes a more important role than SC-EMF, which can be explained by the experimental results in Tables 2 and 3 (see the two tables in Section 6.3). The results in the two tables demonstrate that UC-EMF performs better than SC-EMF in most cases. However, since the two ensemble models consistently achieve better performance than the two standalone models, every standalone model cannot be disregarded.
Figure 7 shows that the changing trend of RMSE and MAE in DE-EMF is similar to that in SE-EMF. The minima are achieved at 0.4 for response time, and 0.5 for throughput. Note that, for the two ensemble methods, the variation of the performance is smooth on the two datasets, especially for RMSE.

**Figure 7** Impact of β in DE-EMF

7 Conclusion and future work

In this paper, we proposed two novel prediction models (SC-EMF and UC-EMF), which are based on MF and clustering algorithm. Before building models, we employed clustering algorithm to cluster service (or user) into different groups according to QoS records, and selected similar cluster centres for each service (or user). In the two models, besides the QoS records of the target service (or user), we also incorporated the QoS records of their similar centres into the prediction of missing values. Based on the two standalone models, we proposed two ensemble models. This paper first verifies that the QoS records of the similar centres are indeed helpful in the task of QoS prediction. Also, such QoS records are used effectively in our proposed models. Second, this paper shows
that although the K-Medoid algorithm is simple, it can acquire effective cluster centres. Third, although the dynamic ensemble model (DE-EMF) performs better than the static dynamic model (SE-EMF), it is enough to leverage SE-EMF to acquire superior results. Fourth, the discussion of time complexity verifies that all our proposed models have linear scalability to the data size. Finally, the experimental results demonstrate the effectiveness of our models, as well as the gradient descent algorithm.

These achievements are also valuable for other related research. For example, we plan to model the invocation process involving the time factor (Zhang, Zheng, and Lyu, 2011; Zhang et al., 2014). In the real world, the IT infrastructure of users and services may change along with time, which leads to the variation of cluster centres. So it is necessary to construct a time-aware QoS prediction framework. Also, we are planning to build a real-world service recommender system based on the framework proposed in this paper.

Acknowledgement

This paper is supported by National Natural Science Foundation (No.61272129), National High-Tech Research Program of China (No.2013AA01A213), New-Century Excellent Talents Program by Ministry of Education of China (No.NCET-12-0491), Zhejiang Provincial Natural Science Foundation of China (LR13F020002), Science and Technology Program of Zhejiang Province (No.2012C01037-1) and China Scholarship Council.

References


Google AdWords (2015) AdWords API. Available online at: https://developers.google.com/adwords/api/docs/reference


