Cold-Start Web Service Recommendation Using Implicit Feedback

Gang Tian[1,2], Jian Wang[1*], Keqing He[1], Weidong Zhao[2], Panpan Gao[1]
[1] State Key Laboratory of Software Engineering, School of Computer, Wuhan University, China
[2]College of Information and Science Engineering, Shandong University of Science and Technology, China
{tiangang, *jianwang, hekeqing}@whu.edu.cn, {zwdslj, g_panpan}@163.com

***Abstract*—Service recommendation becomes an increasingly important issue when more and more Web services are published on the Internet. Existing Web service recommendation approaches based on collaborative filtering (CF) seldom consider recommending services based on users' ratings on services since such kind of explicit feedback is difficult to collect. In addition, the new user cold-start problem is also an important issue due to the lack of accuracy in service recommendations since the new users have not yet cast a significant numbers of votes. In this paper, a dataset consisting of much user-service interaction data is reported. The interaction data created according to users' behaviors can highly represent their preferences. Therefore, we construct pseudo ratings based on this kind of implicit feedback. We develop a novel service recommendation approach which can partially deal with cold-start problem using an online learning model. Experiments show the proposed approach can achieve satisfied results in prediction accuracy and time cost.**

Keywords-cold-start Web service recommendation; implicit feedback; probability matrix factorization;

I. INTRODUCTION

With the continuously increasing of Web services published on the Internet, recommending suitable Web services becomes a challenging issue [1, 2]. Existing works on recommendation systems of Web services are mainly based on collaborative filtering (CF), which computes the similarity of users or services to predict missing ratings based on the interdependencies among users and items.

Many CF-based Web service recommendation systems have been proposed in recent years [3, 4], most of which mainly focus on QoS prediction, while seldom consider recommending services based on users' explicit feedback since it is very hard to collect a large scale dataset that contains explicit feedback (e.g., explicit ratings of Web services assigned by users) or implicit feedback (e.g., service invocation history of real users). Although some artificial datasets are reported, they are very hard to simulate the diversity of real-world users' preferences.

Moreover, most existing Web service recommendation approaches based on CF pay little attention to the new user cold-start problem, which means that the systems may fail to provide any recommendation for new users since no historical information of these users are provided in the systems.

In this paper, we study the above mentioned issues of Web service recommendation, which particularly consider the situation where only implicit feedback is available. Based on the online learning pattern, we propose a novel service recommendation approach. In particular, contributions of this paper include:

- We present a Web service recommendation approach that can partially deal with cold-start recommendation based on the implicit feedback recorded by users.
- We conduct experiments on a real-world Web services dataset, which consists of about 280,000 user-Web service interactions are collected from more than 65,000 users and more than 15,000 Web services and mashups, to verify our method. Experiments show that our recommendation system can achieve good results in prediction accuracy and time cost.

The rest of this paper is organized as follows. Section II discusses related work. Section III gives an overview of the dataset and basic concepts used in Web service recommendation. Section IV introduces our online Web service recommendation approach in detail. Section V reports the experiments and gives comparisons with existing approaches. Section VI concludes the paper.

II. RELATED WORK

CF methods, content-based methods and hybrid methods are three kinds of methods that are widely used in Web service recommendation.

A. CF Methods

The memory-based and model-based methods are two kinds of CF techniques that are widely used in recommendation systems. Well-known memory-based methods include user-based approaches [7] and item-based approaches [8]. Memory-based CF techniques have been recently adopted to provide QoS-aware recommendations [9, 10]. Shao et al. [9] propose a typical user-based CF method to predict QoS values which supposes that similar users tend to receive similar QoS from similar services, and they use Pearson Correlation Coefficient (PCC) to compute similarity between users. Zheng et al. develop a model which enhances the user-based CF by fusing item-based CF [10].

The model-based method allows the system to make intelligent predictions for the collaborative filtering tasks based on some learned models [5, 6]. Matrix factorization (MF) is one of the representative works. In [11], MF is used to construct a global model for predicting Qos data, which can achieve higher prediction accuracy. Yu et al. [13] propose a matrix completing approach using an effective

iterative algorithm. The method takes into account both the low-rank structure and the clustered representation of QoS data.

B. Content Based Methods

The content based methods mainly focused on providing a mechanism to formalize users' preference, resource, and the description of Web services, and recommendations are generated based on the predefined semantic models.

Zhao et al. [15] provide a way to model services and their linkages by semantic algorithm. Based on the input keywords, users can get a set of recommendations with linkages to the query. Blake and Nowlan [24] compute a 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. Mehta et al. [5] add quality and usage pattern to the service description to provide more information to discover a service that meets user requirements. Maamar et al. [6] propose a model for the context of Web service interactions and highlighted the resource on which the Web service performed.

C. Hybrid Method

Since hybrid methods which often combine CF with other techniques can provide more accurate predictions, they are widely used. Numerous hybrid models have been presented that involve other related factors to improve service recommendation quality, such as users' locations [16, 17], social network information [18] and temporal effects [19]. Chen et al. [16] propose a CF algorithm which takes into account of users' physical locations and design a region model for large-scale Web service recommendation. Tang et al. [17] demonstrate a location aware CF model by incorporating locations of both users and services. Tang et al. [18] propose a trust-aware recommendation method with social network which integrates some social relation. Amin et al. [19] denote an approach that integrates ARIMA and GARCH models to capture the QoS attributes' volatility.

All the above mentioned approaches do not take into considerations the cold-start problem in service recommendation. There are some approaches focusing on overcoming the cold start problem in recommendation systems. For instance, Yu [12] integrates MF with decision tree learning to bootstrap service recommendation. MF is used to predict missing Qos data and then decision tree is used to handle new user cold-start issue. Bobabdilla et al. [25] design new similarity metrics using optimization based on neural learning which provides greater precision to mitigate new user cold start situations. However, works [12, 25] are both focus on providing more precise classification of new users, while we mainly deal with dynamic scenario. Ling et al [14] develop an online learning framework for collaborative filtering to handle dynamic scenario. We leverage their theory of online learning to handle cold-start problem in service recommendation.

III. BACKGROUND

A. Cold-start Problems in Web Service Recommendation

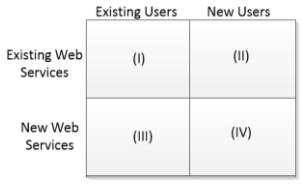

Figure 1. Partitions of Web service recommendation

Fig.1 is used to illustrate the cold-start problem in the service-oriented recommendation. Different partitions require different recommendation approaches. Available approaches for each partition are as follows:

Partition (I) (recommendation on existing Web services for existing users): this is the standard CF techniques. There is no cold-start problem in this case.

Partition (II) (recommendation on existing Web services for new users): for new users without historical ratings, the "most popular" strategy that recommends the highly-rated Web services to new users serves as a baseline. In this case, new users may rate one or more services recommended by the recommendation system. After Web services are rated by new users, the recommendation system needs to learn the new recommendation model. If the model is retrained by a batched pattern where all data is used, the time or memory cost is too high to accept. Therefore, it is reasonable that an online model training method is used to address this problem. In this paper, we mainly focus on this type of cold-start problems.

Partition (III) (recommendation on new Web services for existing users): content-based recommendation can recommend new Web services to existing users based on the users' historical ratings and features of Web services.

Partition (VI) (recommendation on new Web services for new users): in this scenario, the "random" strategy can be applied to deal with the recommendation problem.

B. DataSet in Programmableweb

Programmableweb [1] (PW) is a well-known service registration center where Web APIs, member profiles and other data can be retrieved through its APIs. By using these APIs, we crawled a data set including 66 Web service categories, 17 service protocols, 7,155 mashups, 10,050 Web services, 69,384 users, and 280,611 user-service interactions in users' watch lists (by 2013-11-25). Please note that in PW, when a user is browsing services, if he is interested in a certain service, he can click on the button "track this API" and a piece of user-service interaction information will be recorded in the watch list in the form of (user, service, time). Therefore, services in users' watch lists can highly represent their preferences.

[1] http://www.programmableweb.com/

IV. Online Web Service Recommedation

In this section, we firstly discuss how to quantify the implicit feedback of users, and then introduce our proposed approach.

A. Pseudo Rating From Implicit Feedback

According to our observation, each time when a service is updated, a record will be added to all the watch lists that contain this service automatically. Our statistics shows there are 133 services that are updated more than 5 times, which belong to 105 providers. Most providers have only one such updated frequently service, while the following providers are exceptions: Google has 18 frequently updated services, Yahoo has 5, Amazon has 2, and so on. Clearly a frequently updated service has a large probability of belonging to well-known companies, and thus may have a better quality.

We further investigate whether these frequently updated services are more preferred by users than these infrequently updated services. The data analysis shows that the answer is yes. For example, Google has 18 frequently updated services and 103 infrequently updated services in PW. For each frequently updated service provided by Google, it has an average user number of 72.8, that is, about 72.8 users selected this service into their watch lists, while the number is 22.75 for each infrequently updated service. The data collected from other companies also shows similar results.

Therefore, we can make an assumption that if a service is updated more frequently, it is maintained more frequently, and thus it will be more attractive to users. Therefore, it will have a larger probability to be rated higher by users.

The user-service interaction is represented as *(u, s, rating)*, where *u* denotes the user, *s* denotes the service, and *rating* is the implicit feedback value calculated by Equation (1).

$$rating(u,s) = \frac{freq(s, Watchlist_u)}{|Watchlist_u|} \quad (1)$$

where $freq(s, Watchlist_u)$ is the number that a given service s occurs in a user's watch list $Watchlist_u$. $|Watchlist_u|$ is the number of services in user u's watchlist.

B. MF and Probabilistic Matrix Factorization

Matrix Factorization is one of the most common approaches for recommender systems [21]. Suppose that there are m existing users and n services. $M \in \mathbb{R}^{m \times n}$ denotes the user-service interaction, where M_{ij} represents the rating that user i assigns to service j. MF model finds a joint latent factor space of dimensionality k. MF computes two low-rank matrices $U \in \mathbb{R}^{m \times k}$ and $S \in \mathbb{R}^{n \times k}$ which can minimize the Frobenius norm $||M - US^T||_F$ to approximate the original matrix M, as follows:

$$M_{u,s} \approx \hat{r}(u,s) = \sum_{k=1}^{K} U_{u,k} S_{k,s} \quad (2)$$

Probabilistic Matrix Factorization (PMF) adopts a probabilistic linear model with Gaussian observation noise [22]. It assumes Gaussian distribution on the residual noise of observed data and also places Gaussian priors on the latent matrices U and S. Since PMF is the basis of our approach, we briefly introduce it first. The objective function of PMF for the frequency data is defined as follows:

$$\mathrm{p}(\mathrm{R}|\mathrm{U},\mathrm{S},\sigma^2) = \prod_{u=1}^{m}\prod_{s=1}^{n}[\mathcal{N}(R_{u,s}|U_u S_s^T, \sigma^2)]^{W_{u,s}} \quad (3)$$

where $\mathcal{N}(x|\mu,\sigma^2)$ is the probability density function of the Gaussian distribution with mean μ and variance σ^2, $R_{u,s}$ represents the rating that user u rates services s, and $W_{u,s}$ is the indicator function that equals to 1 if user u rated service s and equals to 0 otherwise.

To calculate model parameters (e.g., U_u and S_s^T), optimization techniques should be used. Model parameters optimality is usually defined with a loss function l and the objective task is to minimize the sum of losses on the observed data. The loss function for PMF with quadratic regularization terms is defined as:

$$l = -\frac{1}{2}\sum_{u=1}^{m}\sum_{s=1}^{n} W_{u,s}\left(g(R_{u,s}) - g(U_u S_s^T)\right)^2 + \frac{\lambda_U}{2}||U||_F^2 + \frac{\lambda_S}{2}||S||_F^2 \quad (4)$$

where $g(x) = 1/(1+e^{-x})$ is the logistic function used to map the value into the range of [0, 1], $W_{u,s}$ is the indicator function, and $||\cdot||_F^2$ denotes the Frobenius norm.

C. Algorithm

When we are using PMF, there is an assumption that all rating values are available before model learning, which makes it inappropriate for dynamic scenarios. However, the ratings in our system are extracted from users' watch lists. To capture a newly entered user's ratings, the model has to be retrained using all data. As is known to all, the time and memory consumption are very expensive to retrain the model. Therefore, online learning methods which incrementally modify the model according to the newly observed datum can highly deal with this situation.

In the following, we present our algorithms generalized from PMF named online web service recommendation (OWSR). As discussed above, we mainly deal with new user cold start problems. As shown in Fig.2, all existing ratings are organized into a sequential dataset Q to train the model. Then the "most popular" strategy is applied to recommend the highly-rated Web services to new users. Finally we could adjust the recommendation model based on the new users' ratings using an online pattern. After the model is retrained, new recommendations can be offered to existing users.

In Fig.2, the "most popular" strategy provides the same recommendation to all new users based on the global popularity of services. Let C_s be the users who have rated service s, the global popularity of a Web service is defined as follows:

$$MP_s = \frac{\overline{rating}_s * n_s + \overline{rating} * \alpha}{n_s + \alpha} \quad (5)$$

where the average rating $\overline{rating}_s$ is defined as $\frac{1}{n_s}\sum_{u \in \mathsf{C}_s} rating_{us}$, n_s is the number of users who have rated service s, $\overline{rating}$ denotes the average of all ratings

and α denotes the shrinkage parameter which can regularize ill posed ratings. In this way, the highest rated services in the current time can be recommended to new users.

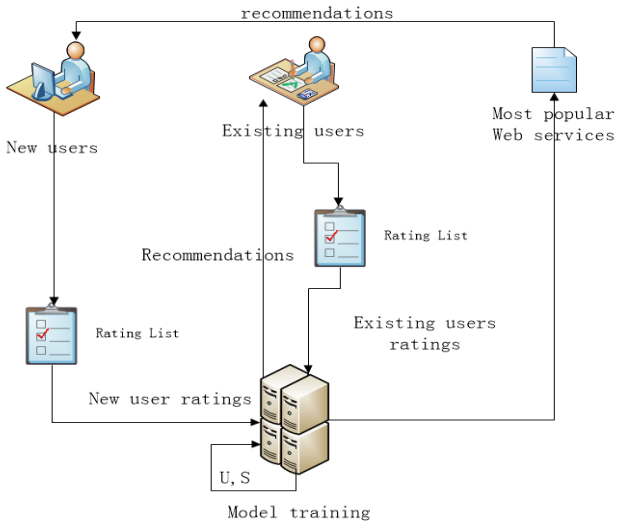

Figure 2. Cold-start Web service recommendation

The procedure of OWSR is described as follows.

ALGORITHM1: *OWSR*

Input: $Interaction\ \ data\ of\ existing\ users$: Q

Output: $Recommendations$, *Most popular Web services*

1. *Train the model using existing ratings.*

Initialize $U_u \leftarrow U_0$, $S_s \leftarrow S_0$ *randomly*

(U_u, S_s) = ***ModelTraining***(U_u, S_s, Q)

2. *Based on* U_u, S_s, Q, *calculate* MP_s, *recommend items to new users, get interaction data of new users:* $Q_{newuser}$, *and get most popular Web service list MPList.*

3. *Retrain the model using new users' ratings* $Q_{newuser}$ *and* U_u, S_s *obtained by step* ***1****.*

(U_u, S_s) = ***ModelTraining***$(U_u, S_s, Q_{newuser})$

return $U_u \times S_s$ and $MPList$

Note that Q and $Q_{newuser}$ are sequential datasets which record the user-service interactions. Each record in Q and $Q_{newuser}$ has the form of *(u, s, rating).*

Since ratings enter the system sequentially, as shown in Algorithm 1, we adjust the model using an online pattern. That is, we train the model using existing ratings first, and then adjust the model according to new users' ratings. The dual-average method for PMF which absorbs previous rating information in an approximate average gradient of the loss is one of the most widely used methods to solve optimization problems [14]. Given the *t*-th coming interaction data (*u, s, rating*), we can update the average gradient $\overline{Gu_t}$ and $\overline{Gs_t}$ with respect to u_t and s_t respectively by the following rules:

$$\overline{Gu_t} = \frac{t_u - 1}{t_u}\overline{Gu_{t-1}} + \frac{1}{t_u}\left(g(R_{u,s}) - g(U_u S_s^T)\right) g'\left(U_u S_s^T\right) S_s \quad (6)$$

$$\overline{Gs_t} = \frac{t_s - 1}{t_s}\overline{Gs_{t-1}} + \frac{1}{t_s}\left(g(R_{u,s}) - g(U_u S_s^T)\right) g'(U_u S_s^T) U_u \quad (7)$$

where t_u denotes the number of services u has rated and t_s denotes the number of users who rate services s.

The model training algorithm is detailed as follows:

ALGORITHM 2: *Model Training*

Input: U, S, Q

Output: $new\ \ U, S$

1. *Initialization:*

$\overline{Gs_t} \leftarrow 0,\ \overline{Gu_t} \leftarrow 0$

for *t = 1, 2, 3,…,*$|Q|$ ***do***

2. *Given the function* l_t, *compute the subgradient on* $U_t, S_t, \overline{Gu_t}, \overline{Gs_t}$, *For the coming instances*$(u_t, s_t, rating_t)$

$l_t \leftarrow \left(g(R_{u,s}) - g(U_u S_s^T)\right)^2$

$Gu_t \leftarrow \left(g(R_{u,s}) - g(U_u S_s^T)\right) g'(U_u S_s^T) S_s$

$Gs_t \leftarrow \left(g(R_{u,s}) - g(U_u S_s^T)\right) g'(U_u S_s^T) U_u$

3. *Update the average subgradient* $\overline{Gu_t}, \overline{Gs_t}$

$$\overline{Gu_t} = \frac{t_u - 1}{t_u}\overline{Gu_{t-1}} + \frac{1}{t_u}Gu_t$$

$$\overline{Gs_t} = \frac{t_s - 1}{t_s}\overline{Gs_{t-1}} + \frac{1}{t_s}Gs_t$$

4. *Calculate the next iteration*

$U_{t+1} \leftarrow \arg min_W \{\overline{Gu_t} W + \lambda_U ||W||_F^2\}$

$S_{t+1} \leftarrow \arg min_W \{\overline{Gs_t} W + \lambda_S ||W||_F^2\}$

endfor

return *U, S*

Note that λ_U and λ_S are the regularization parameters, which are used to reduce over-fitting.

As for the time complexity, for each interaction record (*u, s, rating*)$\in Q$, only O(*k*) steps should be used to adjust the model, where k is the latent feature size and is much smaller than *m* and *n*. Since *k* is very small, it can be treated as constant time and thus the time cost scales linearly with the length of the sequential dataset Q.

V. EXPERIMENTS

A. Evaluation Metrics

The experiments are conducted on a dataset which is described in section III. The dataset contains 280,611 user-web service interactions. Since most users have very few rating information, we select 510 users who have more than 50 services in their watch list and totally 4,424 services are included. Thus we have a 510×4424 matrix. The experiments are conducted in a PC with Core 1.8GHz processor and 4GB memory running in an operating system of Windows 8.

To measure the prediction quality of our method in comparison with other approaches, Root Mean Square Error (RMSE) metrics is embodied which is defined as:

$$RMSE = \sqrt{\frac{\sum_{u,s}(r_{u,s} - \hat{r}_{u,s})^2}{N}} \quad (8)$$

where $r_{u,s}$ is the actual rating of Web service *s* assigned by user *u*, $\hat{r}_{u,s}$ is estimated value, and *N* indicates the number of predicted values.

B. Performance Evaluation

To illustrate the performance of our method, we compare the proposed OWSR with another five existing approaches. These approaches are illustrated as follows:

a) UPCC: UPCC is a typical CF algorithm using PCC as a measurement of similar users [7, 9]. We use Mahout0.8[2] and set the number of similar users as 50.

b) IPCC: IPCC is an item based CF method using PCC as a measurement of similar items [8].

c) WSRec: WSRec is a hybrid approach integrating both user and item based CF models [10]. During the evaluation, parameters (k,λ) are set to (32, 0.2).

d) SVT: SVT is a singular value thresholding method that completes the matrix via minimum nuclear norm [23]. We use SVT Package[3], and parameters are set as follows: N=510*4424, R=20, df=98280, m=491400, p=0.217796, τ =7510.39, σ=5.50974, iters=500, ε= 0.00001, and EPS = 0.

e) PMF: PMF is a matrix factorization model which scales linearly with the number of observations [22]. We use PMF Package[4] and parameters are set as follows: $\lambda=\lambda_U=\lambda_V$=0.01, η=1000, maxepoch=500, and num_feat=20.

f) OWSR: parameters of our method are set as follows: $\lambda=\lambda_U=\lambda_V$ = 0.02, maxepoch=500, and num_feat=20.

User-service interactions are randomly divided into two categories in a certain proportion: the training set and the test set. The proportion that the training set accounts for the whole set varies from 0.1 to 0.9, which makes the sparsity of the matrix ranging from 0.9930 to 0.9992, as shown in Table I. The sparsity of the matrix is defined as $sparsity = 1 - \frac{|N|}{|User| \times |Service|}$, where $|\cdot|$ is numbers of $\cdot$.

Table I presents the results of prediction accuracy performance of these six approaches. According to these experiments, we have the following observations.

First of all, compared with the existing four approaches (*IPCC*, *UPCC*, *WSRec*, and *SVT*), we observe that the OWSR algorithm achieves smaller RMSE values for all the cases, which shows the effectiveness of the proposed approach.

Second, the RMSE of PMF outperforms other approaches for all the cases. The average performance of PMF exceeds ours about 12%. The performance gap is probably due to the approximation when computing the gradient.

C. OWSR (online pattern) versus PMF (batch pattern):

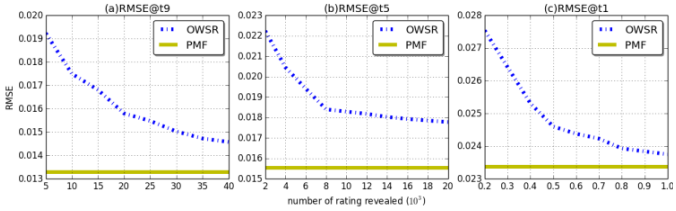

Figure 3. Comparison between OWSR and PMF

To further compare OWSR with PMF, we conduct other experiments. As shown in Fig.3, under RMSE@t9 (90% of all data are randomly chosen for training, and the remaining 10% are for evaluation), PMF exceeds our approach about 9%. Under these three settings, with the increase of the number of ratings revealed, the performance of OWSR converges to a certain value.

In Fig.3, we can see that under RMSE@t1 (10% for training and 90% for evaluation), our approach is closer to PMF compared with under RMSE@t9 and RMSE@t5. The most likely cause is that under the scenario of few training data, our model is less likely to fall into local optimum. As a whole, the performance of OWSR is approaching that of the PMF.

D. Time Overhead

Since two algorithms (PMF and OWSR) we test are matrix factorization based, employing the same latent feature size is fair for comparison. The PMF spends more than 26 seconds in finishing 500 iterations when there are 19642 rating. OWSR spends only about 14 seconds in reaching a similar performance. As shown in Fig.4, when the number of ratings grows bigger, PMF takes much more time than OWSR. OWSR is much better than PMF to this point.

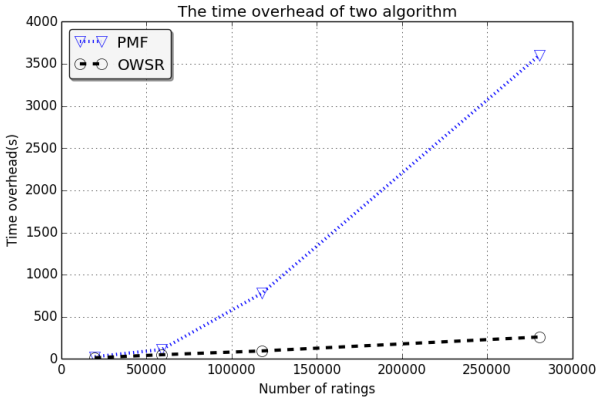

Figure 4. Time overhead of two algorithms

According to these experiments, we can conclude that the prediction accuracy of OWSR is close to PMF, while OWSR costs much shorter time.

In addition, as for the parameters used in OWSR, *num_feat* is the number of latent feature indicates the degree of compression of the proposed model. We vary the latent feature number from 10 to 100 and an optimal RMSE is achieved when *num_feat* = 20.

Parameter λ manages the trade-off between the regularization and the loss. We set $\lambda_U=\lambda_V=\lambda$ for simplicity. An optimal result is achieved when λ = 0.02.

[2] http://mahout.apache.org/

[3] http://statweb.stanford.edu/~candes/svt/code.html

[4] http://www.cs.toronto.edu/~rsalakhu/BPMF.html

TABLE I. RMSE PERFORMANCE

Algorithm	*Data Sparsity*								
	0.9930	*0.9938*	*0.9945*	*0.9953*	*0.9961*	*0.9969*	*0.9977*	*0.9984*	*0.9992*
	Train Set Ratio								
	0.9	*0.8*	*0.7*	*0.6*	*0.5*	*0.4*	*0.3*	*0.2*	*0.1*
IPCC	0.02350	0.03623	0.03749	0.04878	0.03841	0.03563	0.02257	0.03721	0.03773
UPCC	0.01929	0.03057	0.03491	0.04142	0.02230	0.03127	0.02691	0.02893	nan
WSRec	0.02956	0.02575	0.02192	0.02275	0.02160	0.02201	0.02275	0.02434	0.02652
SVT	0.02629	0.03157	0.02581	0.02639	0.03722	0.03028	0.03807	0.03130	0.03220
PMF	0.01330	0.01366	0.01520	0.01990	0.01555	0.02178	0.02172	0.02170	0.02338
OWSR	0.01458	0.01545	0.01610	0.02014	0.01797	0.02199	0.02235	0.02186	0.02375

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed an online CF algorithm based on implicit feedback data for Web service recommendation. We first introduce how to calculate ratings from the implicit feedback data. To deal with cold-start problems in Web service recommendation, we adopt the "most popular" strategy and propose a matrix factorization CF model using the online pattern. Experiments show the proposed model can achieve satisfied result in both prediction accuracy and time cost.

Our future work includes improving the performance of OWSR and exploring how to solve the cold-start problem discussed in partition III of Section III.

ACKNOWLEDGEMENT

The work is supported by the National Basic Research Program of China under grant No. 2014CB340404, the National Science & Technology Pillar Program of China under grant No. 2012BAH07B01, the National Natural Science Foundation of China under grant No. 61202031 and 61373037.

REFERENCES

[1] L.-J. Zhang, J. Zhang, and H. Cai. Services computing. In Springer and Tsinghua University Press, 2007.

[2] Q. Yu, X. Liu, A. Bouguettaya, and B. Medjahed, "Deploying and Managing Web Services: Issues, Solutions, and Directions". The VLDB Journal, 17(3), pp. 537–572, 2008.

[3] Y. Jiang, J. Liu, M. Tang, et al, "An effective Web service recommendation based on personalized collaborative filtering", In ICWS, pp.213-218, 2011.

[4] Z. Zheng, H. Ma, M.R.Lyu, et al, "QoS-aware Web service recommendation by collaborative filtering". TSC, 4(2), pp. 140-152, 2011.

[5] B. Mehta, C. Niederee, A. Stewart, C. Muscogiuri, and E.J.Neuhold, "An Architecture for Recommendation Based Service Mediation," Semantics of a Networked World, vol. 3226, pp. 250-262,2004.

[6] Z. Maamar, S.K. Mostefaoui, and Q.H. Mahmoud, "Context for Personalized Web Services," roc. 38th Ann. Hawaii Int'l Conf.,pp. 166b-166b, 2005.

[7] J.S.Breese, D. Heckerman, C. Kadie Empirical Analysis of Predictive Algorithms for Collaborative Filtering, In UAI,pp. 43-52, 1998

[8] M. Deshpande and G. Karypis, " Item-Based Top-N Recommendation", ACM Trans. Information System,vol. 22, no. 1, pp. 143-177, 2004.

[9] L. Shao, J. Zhang et al, "Personalized qos prediction for web services via collaborative filtering", In ICWS, pp. 439-446, 2007.

[10] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Wsrec: A collaborative filtering based web service recommender system", In ICWS, pages 437–444, 2009.

[11] Z. Zheng, H. Ma, M. R. Lyu, and I. King. Collaborative Web Service QoS Prediction via Neighborhood Integrated Matrix Factorization. IEEE Transactions on Service Computing (TSC), 2012.

[12] Q. Yu, "Decision Tree Learning from Incomplete QoS to Bootstrap Service Recommendation", In ICWS, pp. 194-201, 2012.

[13] Q. Yu, Z. Zheng, H. Wang, "Trace norm regularized matrix factorization for service recommendation", In ICWS, pp.34-41, 2013.

[14] Ling G, Yang H, King I, et al. Online learning for collaborative filtering[C],Neural Networks (IJCNN), The 2012 International Joint Conference on. IEEE, 2012: 1-8.

[15] C. Zhao, C. Ma, J. Zhang, J. Zhang, L. Yi, and X. Mao,"HyperService: Linking and Exploring Services on the Web," Proc. Int'l Conf. Web Services,pp. 17-24, 2010.

[16] X. Chen, X. Liu, Z. Huang, et al. "RegionKNN: a scalable hybrid collaborative filtering algorithm for personalized Web servicer ecommendation". In ICWS, pp. 9-16, 2010.

[17] Tang M, Jiang Y, Liu J, et al. "Location-aware collaborative filtering for qos-based service recommendation", In ICWS, pp. 202-209 2012.

[18] Tang M, Xu Y, Liu J, et al. Trust-Aware Service Recommendation via Exploiting Social Networks[C] In SCC, 2013: 376-383.

[19] Amin, Ayman, Alan Colman, and Lars Grunske. "An approach to forecasting QoS attributes of web services based on ARIMA and GARCH models", In ICWS, pp.74-81,2012.

[20] D.W. Oard and J. Kim, "Implicit Feedback for Recommender Systems",Proc. 5th DELOS Workshop on Filtering and Collaborative Filtering, pp. 31–36, 1998.

[21] Y. Koren, "Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model," Proc. 14th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining, ACM Press, 2008, pp. 426-434

[22] R. Salakhutdinov and A. Mnih, "Probabilistic Matrix Factorizat ion," Proc. Advances in Neural Information Processing Systems 20(NIPS 07), ACM Press, 2008, pp. 1257-1264.

[23] J F. Cai, E J. Candes, and Z. Shen, "A singular value thresholding algorithm for matrix completion," SIAM J. on Optimization, 20(4):1956–1982, 2010.

[24] M.B.Blake and M.F. Nowlan, "A Web Service RecommenderSystem Using Enhanced Syntactical Matching," Proc. Int'l Conf.Web Services, pp. 575-582, 2007.

[25] Bobadilla J, Ortega F, Hernando A, et al. A collaborative filtering approach to mitigate the new user cold start problem[J]. Knowledge-Based Systems, 2012, 26: 225-238.