2014 IEEE International Conference on Web Services

Web Service Recommendation Based on Watchlist via Temporal and Tag Preference Fusion

Xiuwei Zhang[1,2], Keqing He[1], Jian Wang[1], Chong Wang[1*], Gang Tian[1], Jianxiao Liu[3]
[1]State Key Lab of Software Engineering, School of Computer, Wuhan University, Wuhan, China
[2]No. 94005 Troops of PLA, Jiuquan, China
[3]College of Science, Huazhong Agricultural University,Wuhan, China
{zhangxiuwei, hekeqing, jianwang, cwang, tiangang}@whu.edu.cn, liujianxiao@mail.hzau.edu.cn

***Abstract*—With the increasing number of Web services available on the Internet, how to recommend Web services to interested users effectively and efficiently remains to be a big challenge. At present, collaborative filtering (CF) is the most widely used technique in the design of recommender systems to handle information overload. For Web services, however, it is difficult for user to collect personalized QoS (Quality of Service)data and other explicit feedbacks such as ratings. In most cases, only a part of the implicit feedbacks (e.g., watchlist) is available in service registry. In this paper, we leverage implicit feedback from user's watchlist to build a CF-based recommender system for Web service. Our main contribution is to transform implicit feedbacks into explicit ratings to improve the accuracy of service recommendation. More specifically, we first construct binary user-service rating matrix according to the implicit feedback from the watchlist. Then, temporal and tag preference are combined into the original rating matrix to generate a more accurate pseudo rating matrix, which can reflect users' different preference on services in their own watchlists. Finally, we use traditional user-based CF method to produce a personalized service recommendation list with corresponding pseudo ratings. Moreover, the empirical experiments based on ProgrammableWeb show that compared with traditional log-based CF method, the recommender system with temporal and tag preference is more accurate and precise.**

Keywords- *service recommendation; Web service; implicit feedback; collaborative filtering; watchlist*

I. Introduction

In recent years, large amount of Web services have been developed by different enterprises and organizations with various Web service description languages, and are stored in different repositories. Service consumers are inundated with choices. At present, keyword-based query is one of the most popular approaches for service discovery on the Web. Users pose queries with specific search criteria and find the services with related contents. However, the search engine is designed in a one-fit-all mode and cannot meet personalized demands[1]. The automatic discovery model used in a service recommender system is quite different with search engines. It recommends service items according to user's preference and generated user models, so that it can help service user or service consumer (e.g. system developer, mashup designer, et al.) to select appropriate services in a large space of possible options in a personalized way[2]. In previous researches, CF is the most popular and successful recommendation technique in the design of recommender systems and can be used to deal with information overload.

Temporal information is one of key factors to reflect users' preference drift over time. The key problem of incorporating temporal information into recommendation is how to find the correlation over temporal dynamics with users' preference[3]. As we know, more recent used services will reflect more upcoming interests of users and newer published services will have more abstractive to users. In addition, Web services are often annotated by the service users or domain experts through utilization of tags. Tags assigned to a certain services can be considered as the highly abstracted content features[4]. According to the features of Web service description (e.g. WSDL), these service description documents do not contain sufficient textual description. Although semantic annotation information is added into textual description to extend them such as SAWSDL (Semantic Annotation language for WSDL), OWL-S and WSDL-S, these description languages always require tedious, error-prone manual work and seem to be a hard task for service providers[5]. Therefore, tag information of services are one important clues for service recommendation.

Based on above mentioned two considerations, we intend to merge temporal and tag preference to transform implicit feedbacks into explicit ratings for CF-based Web service recommendation, even in absence of explicit rating and QoS infomration. The implicit data of watchlist is a RSS Feed service which not only looks like a favorite list for user collecting interested services, but also can actively send emails to user for notification of the new updating of services. Based on a real-world data which was crawled from an open service registry and repository ProgrammableWeb[2] (PW), the empirical experimental results show that our method outperforms traditional log-based CF method.

The remainder of this paper is organized as follows. Section II identifies the related work. Section III introduces the basic problem definition. Section IV explains the detailed steps of our recommendation approach. Section V presents the experiments and results which are used to exemplify the potential of our approach. And the final section draws conclusions of our work and outlines the possibilities for future work.

[*] Corresponding author : cwang@whu.edu.cn

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

II. Related Work

The main idea of recommendation is to offer the right products or services at the right time and in the right place to the right customers[6]. With the popularity of recommender system, there is an abundance of real-life applications of recommender systems which helps users to deal with information overload in the Internet. The application domains range from commercial products such as CDs, TV programs and movies, to recommendation of more complex items such as scientific workflow, quality methods and instruments[7]. However, user's explicit rating is not always available for recommendation. In order to avoid the bottleneck of rating data collection, implicit feedback techniques was introduced by inferring something similar to the ratings that a user would assign from observations that are available to the system[8].

In service computing, researchers often take QoS as an important index for service recommendation because of its characteristics in the runtime. These QoS-based approaches for service recommendation use QoS experience of similar users to accurately predict the QoS that an active user may receive from previously unknown services[9]. QoS-based service recommendation methods are often based on the hypothesis that the QoS information is available and accurate to the active user. As a matter of fact, QoS data in real world is always dynamically changed in the client side[10].

A. Implicit feedback-based recommender

1) *Tag-based recommender*

Social tagging is considered an important mechanism for organizing and discovering information on the Web[11]. Many researchers have investigated effective techniques for tag-based recommendation. The tag-based strategies are often categorized into three major groups[12]:

Network-based model: A tag-based network can be viewed as a tripartite graph consisting of three integrated bipartite graphs or a hyper-graph. Zhang, et al.[13] proposed an integrated diffusion on tripartite graph for resource recommendation.

Tensor-based model: the tensor factorization (TF)-based method has been paid increasing attention because it is applied in designing recommendation algorithms with social tags. The tensor factorization is based on singular value decomposition (SVD), with which the ternary relation can be reduced to low dimensions[12].

Topic-based model: The topic-based method exploits tags in a probabilistic framework. It views each tag as an indicator of a topic and then estimates the probability of annotating an item by summing the transition probabilities through all tags[11]. It supposes that if a user likes an item in a specific topic, he will also like the other items in this topic. In the past two decades, the most used topic-based methods include Latent Semantic Analysis (LSA), TF-IDF, Latent Dirichlet Allocation (LDA), Ontology, etc.

2) *Temporal-based recommender*

Some works aim to investigate time information in recommendations. For example, user purchase time and item launch time were considered in [14] to improve recommendation accuracy. Furthermore, they analyzed a variety of temporal information including item launch time, users' buying time, and the time difference between those two time. The users's rating time was considered in Ding et al.[15] in order to improve the precision of item-based collaborative filtering. Zheng and Li[16] present implicit feedback based recommender system with time information for social tagging system. Particularly, they used exponential time decay function to compute time weights for different items according to bookmark time and item launched time. The experimental study shows that time information is a significant factor in recommendation. Xiang, et al.[3] proposed a session-based temporal graph (STG) which simultaneously models users' long-term and short-term preferences over time.

B. QoS-based service recommender

QoS mainly consists of performance factors such as availability, response time, reliability, throughput, etc. QoS has been widely used to facilitate service selection[17], service composition[18] and service recommendation [10,19,20]. Zheng, et al.[10] presents a Web service recommender system called WSRec, using the hybrid of user-based CF and item-based CF as its key algorithm. Jiang[20] improved the traditional hybrid recommendation with deviation of the QoS information. Cao, et al.[19] also presents a standard deviation based hybrid CF for Web service recommendation and an inverse consumer frequency based CF for potential consumer recommendation. It not only recommends services to service users but also recommends service users to service provider. In [21], region information is considered in a novel hybrid collaborative filtering algorithm named RegionKNN to support large scale of Web service recommendation.

The recommendation methods mentioned above pay little attention to service recommendation with users' implicit feedback data of watchlist with a public open service registry. Therefore, this paper leverages users' implicit feedbacks of watchlist information to recommend Web services.

III. Problem Definition

To avoid the absence of users' explicit rating and QoS data, this paper intends to transform the accessible and easy-to-get implicit feedbacks (e.g., watchlist) into a personalized pseudo rating matrix rather than explicit rating matrix. Then traditional user-based CF method can be used to recommend services for users. In our approach, the similarity between users is based on the following hypothesis: 1) The user who have common services in the watchlist are supposed to be more similar; 2) The users using common tags are supposed to be similar to each other; 3)Users' preference is slightly drifting with the time passed.

This section introduces, the notations and definitions required in this paper. Formally, there are four different kinds of communities i.e. users, services, tags and timestamp. $U = \{u_1, u_2, \dots u_{|U|}\}$ denotes the set of users, $S =$

$\{s_1, s_2, \dots s_{|S|}\}$ denotes the set of all services, and $T = \{t_1, t_2, \dots, t_{|T|}\}$ denotes the tags set. $TS = \{ts_1, ts_2, \dots, ts_{|TS|}\}$ denotes the timestamp set for service publishing. The relationship among services, tags and timestamps is $\text{CR} = \{< s,\ t_1|t_2 \dots |t_i\ , ts_s, r_s\ \}$, in which s denote the service name, t is the tags of service s, ts_s denotes the time of service s publishing or updating in service registry. r_s denotes the average rate of service s.

Definition 1. (*Watchlist): a RSS Feed service delivers the regular changes of web service to any users who wants it.* It can be expressed as:

$$\boldsymbol{\Gamma_u} = < u, S_u, T_u, CR_u > \qquad (1)$$

where $\text{u} \in \text{U}$ denotes the owner of the watchlist, S_u is the services set collected in the watchlist by user u ($S_u \subseteq S$). T_u denotes the tag set of user u ($\text{T}_\text{u} \subseteq T$). $CR_u (CR_u \subseteq CR)$ denotes the relationship among S_u, T_u and TS_u. An abstraction of watchlist in PW is shown in Figure1. Left part is a snapshot of real watchlist in PW. The right part is a graphic description of watchlist defined by Eq. (1).

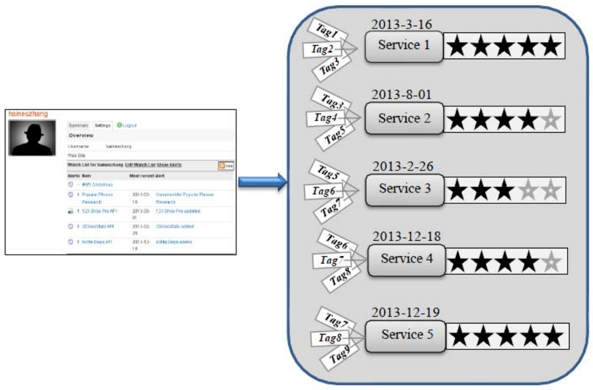

Figure 1. Snapshot of watchlist of user Haines in PW

According to Definition 1, the watchlist in Figure 1 can be formally represented as follows:

$$u =' Haines' ;\ S_u = \{s_1, s_2, \dots, s_5\} ;\ T_u = \{t_1, t_2 \dots t_9\}$$

$$CR_u = \begin{Bmatrix} s_1, t_1|t_2|t_3, 20130316, 5 \\ s_2, t_3|t_4|t_5, 20130801, 4 \\ s_3, t_5|t_6|t_7, 20130226, 3 \\ s_4, t_6|t_7|t_8, 20131218, 4 \\ s_5, t_7|t_8|t_9, 20131219, 5 \end{Bmatrix}$$

Definition 2 (*Preference Drift Function, PDF*): *a kind of user preference drift model, which describes the features of change tendency associated with users' preference as the time goes by.* In this paper, the preference drift function is defined as an adaptive exponential forgetting function that is similar to Newton's law of cooling[11]:

$$\text{w}_{\text{PDF}}(u, s) = \exp\{-\alpha * time(u, s)\} \qquad (2)$$

where $\text{w}_{\text{PDF}}(u, s)$ denotes the interest drift weight which is the declined degree of a user's preferences. α denotes the temporal attenuate parameter which is used to adjust the preference drift speed. $time(u, s)$ is a non-negative integer ($time(u, s) \geq 0$), which denotes the number of services in user u's watchlist with temporal order. The newest updating service in watchlist is set to 0 and the penultimate service is set to 1, and so on. The primary reason of importing behind this temporal weight strategy is based on the fact that human beings interest always drifts with time passing. The preference drift function curve with different α is shown in Figure 2, in which α is an adjusted parameter to determine the users' preference drift speed. As Figure 2 prsents, the preference drift speed will drift steeply with increment of α.

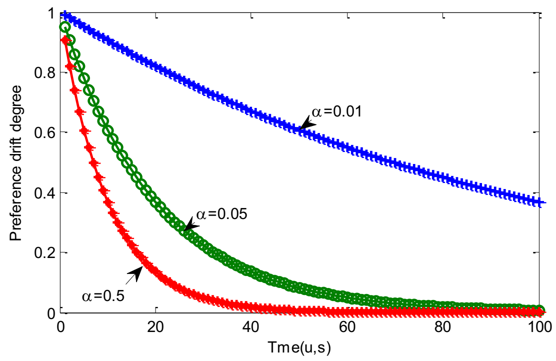

Figure 2. Preference drift function curve with different α

IV. Recommendation Approach

In this section, we present the architecture and algorithm of our approach. Figure 3 illustrates the architecture of our recommendation model. It consists of three parts i.e. implicit feedback database, pseudo rating matrix generator and recommender system. The key part of this recommendation model is the pseudo rating generator which includes baseline rating matrix generator, average rating matrix generator, temporal-weighted rating matrix generator, tag-weighted rating matrix generator and hybrid rating matrix generator. The pseudo rating generator is designed to transform implicit feedback into explicit ratings by merging the temporal and tag preference together to produce a more accurate pseudo rating for service recommendation.

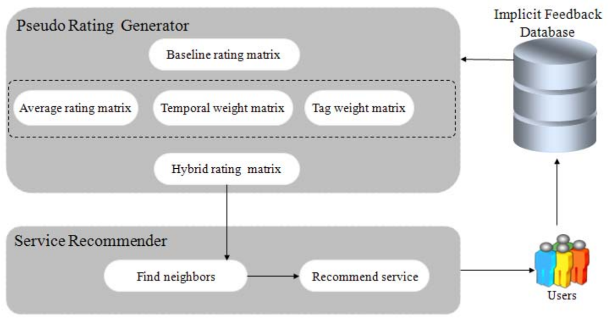

Figure 3. Architecture of our recommendation model

The following algorithm illustrates how to leverage the temporal and tag information as positive factors to improve the accuracy compared to the implicit binary rating matrix. Algorithm 1 describes the procedures of our approach for service recommendation. The input of Algorithm 1 include a user set U (|U|=m), a service set S (|S|=n), a watchlist set $\Gamma(|\Gamma| = m)$, a temporal attenuate parameter α, and a weight balance parameter λ that is used to adjust the significance of temporal weight and tag weight. The output is the recommended Top-K services list sets L.

Algorithm 1 Service recommendation procedure

```
Input:  $U$, $S$, $\Gamma$, $\alpha$, $\lambda$
Output: $L$; //a Top-K services list sets for all users ;
1  Generate baseline rating matrix $\mathbb{A}_{m\times n} \leftarrow \Gamma_u$;
2  Generate average rating matrix $\mathbb{B}_{m\times n} \leftarrow \Gamma_u$;
3  Generate temporal weighted matrix; $\mathbb{C}_{m\times n} \leftarrow (\Gamma_u, \alpha)$;
4  Generate tag weighted matrix $\mathbb{D}_{m\times n}$;
5  Fusion matrix $\mathbb{E}_{m\times n} \leftarrow (\lambda\mathbb{C}_{m\times n} + (1-\lambda)\mathbb{D}_{m\times n}) \cdot \mathbb{B}_{m\times n}$;
6  for each $u \in U$ as active user
7      for each $v \in U$ and $v \neq u$
8          user similarity matrix $\mathbb{S}_{m\times m} \leftarrow sim(u,v) = \frac{\vec{u}\cdot\vec{v}}{||\vec{u}||\cdot||\vec{v}||}$;
9      end for
10     Sort $sim(u)$;
11     Find active user $u$ neighborhood $\mathrm{Neighbor(u)} \subseteq$ U;
12     for each s ∈ S
13         Computing the degree of u prefer service s ∈ $S$
           $\mathrm{Prefer(u,s)} = \frac{\sum_{v\in Neighbor(u)} R\times sim(u,v)}{|\sum_{v\in Neighbor(u)} sim(u,v)|}$;
14         Sort Prefer(u, s);
15         Add user u Top-K recommended services $L_u$ into $L$
16     end for
17 end for
18 return $L$;
```

Next, a simple example will be explained in detail to thow the procedures of generating pesudo rating matrix. The sample data is in Table I. The value in the square bracket is the average rating for service. $time(u,s)$ is given in braces. The date of publishing a service in registry is followed with the pattern 'yyyymmdd'. Tags of one service is splitted with '|'.

TABLE I THE SAMPLE DATA

	s_1 [4]	s_2 [3]	s_3 [5]
u_1	$t_1\|t_2\|t_3$ **20130101** {1}	#	$t_3\|t_4$ **20130108** {0}
u_2	$t_1\|t_2$ **20130206** {0}	$t_2\|t_5$ **20130204** {1}	#
u_3	#	#	$t_3\|t_5$ **20130102** {0}

A. Generate pseudo rating matrix

There are five main steps when generating pseudo rating matrix. Each step will produce a user-service matrix (i.e. from line 1 to line 5 in Algorithm 1) which is filled with pseudo rating values with specific mechanism.

1) Pseudo rating matrix of baseline

We can assign 1 as a rating value when service i exist in user j's watchlist, then $a_{ij} = 1$, otherwise, $a_{ij} = 0$. The baseline rating matrix is a 0-1 matrix that can be annotated as $\mathbb{A}_{m\times n}$, where m denotes the number of users and n denotes the number of services. Based on the sample data, the baseline matrix is produced as follows, which shows that u_1 add s_1 and s_3 in his watchlist, u_2 adds s_1 and s_2 in watchlist, and u_3 only adds s_3 in watchlist.

$$\mathbb{A} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2) Pseudo rating matrix of average rate

To relieve the lack of individulized rating data, this paper uses the average score which is often indicated as stars from one to five in PW. Then the matrix of average rate is constructted to replace '1' in the matrix of baseline with the service average score shown in Table I. The matrix of average score is expressed as $\mathbb{B}_{m\times n}$. And the average rate matrix $\mathbb{B}$ of sample is shown as follow:

$$\mathbb{B} = \begin{bmatrix} 4 & 0 & 5 \\ 4 & 3 & 0 \\ 0 & 0 & 5 \end{bmatrix}$$

3) Pseudo rating matrix of temporal weighted

Generally speaking, preference drifting uses either time window or forgetting functions to learn and track the changes of users' behavior as time passing. Most of the time window methods have completely ignored the old information [16]. In our method, the preference drift function is derived from Newton's law of cooling because the process of preference drift can be simulated as the cooling process of a hot object, whose temperature will gradually low till it is equal to environment temperature. Similarly, the newly published services in the watchlist can reflect more upcoming preference than the old services in the watchlist. Eq. (2) is used to the measure preference drifts in temproal cases. The temporal weighted matrix is expressed as $\mathbb{C}_{m\times n}$. The matrix below shows the calculating results if the temporal attenuate parameter $\alpha = 0.5$.

$$\mathbb{C} = \begin{bmatrix} 0.6065 & 0 & 1 \\ 1 & 0.6065 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

4) Pseudo rating matrix of tag weighted

Tags are usually produced from service publishers, domain experts and users. This paper only focus on the service tag annotated by service publishers and the domain experts. In this way, each user will have the same tags when adhering their preference to the same service. Tag weight is defined as Eq.(3)[16] to measure how user u is interested in service s, i.e. a user's preference for a specified service.:

$$w_{TAG}(u,s) = \sum_{t\in T_{(u,s)}} w_{u,t} \quad (3)$$

where tag weight $w_{TAG}(u,s)$ measures how user u is interested in service s, which implies a user's preference for a service. $T(u,s)$ denotes the tags set which is used to annotate services in user u's watchlist. $w_{u,t}$ denotes tag score of each tag *t* in $T_{(u,s)}$. Tag score $w_{u,t}$ is calculated as follows:

$$w_{u,t} = \frac{\mathrm{freqence(u,t)}}{\sum_{i=1}^{k}\mathrm{frequence(u,t_i)}} \quad (4)$$

where $\mathrm{freqence(u,t)}$ denotes the number of tag t that appears in watchlist of user u. $\sum_{i=1}^{k} frequence(u,t_i)$ is the number of all tags in user u's watchlist. Tag score is a non-negative real number between 0 and 1 ($w_{u,t} \in [0,1]$). Note that $\sum_{i=1}^{k} w_{u,t_i} = 1$. Using Eq.(3) and Eq.(4), the tag-weighted matrix as $\mathbb{D}_{m\times n}$ can be calculated as follows:

$$\mathbb{D} = \begin{bmatrix} 0.8 & 0 & 0.6 \\ 0.75 & 0.75 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

5) Pseudo rating matrix of hybrid

Finally, the temporal and tag weighted matrix are combined with parameter λ, which is given in Eq.(5). Meanwhile, the average score is also considered in the ratings. The hybrid rating matrix $\mathbb{E}$ is defined as:

$$\mathbb{E}_{\mathrm{m}\times\mathrm{n}} = (\lambda\mathbb{C}_{m\times n} + (1-\lambda)\mathbb{D}_{\mathrm{m}\times\mathrm{n}}) \cdot \mathbb{B}_{\mathrm{m}\times\mathrm{n}} \tag{5}$$

where parameter λ is introduced to adjust the weights of temporal-based matrix and tag-based matrix. '.' is the dot multiply between two same dimension matrix, which means that only corresponding element in matrix can be multiplied. For instance, $\lambda = 0.5$ means that time weight and tag weight has the same significance. The final pseudo rating matrix of the sample data is calculated as follows.

$$\mathbb{E} = \begin{bmatrix} 2.513 & 0 & 4.5 \\ 3.75 & 1.847 & 0 \\ 0 & 0 & 5 \end{bmatrix}$$

Note that, the tag, temporal and average rating matrix can be used alone or composed with each other according to the real implicit feedback data on your dataset.

B. Find Neighbors

Based on the generated pseudo rating matrix $\mathbb{E}$, the traditional user-based CF method is performed to obtain the recommendation list. There are many ways to calculate the similarity between users, such as Pearson correlation coefficient, Spearman correlation coefficient, cosine similarity, Jaccard similarity, etc. This paper adopts cosine similarity to find neighbors as Eq.(6) defines[16]:

$$sim(u,v) = \frac{\vec{u}\cdot\vec{v}}{\|\vec{u}\|\cdot\|\vec{v}\|} = \frac{\sum_{x\in X(u,v)} \mathcal{M}_{u,x}\times\mathcal{M}_{v,x}}{\sqrt{\sum_{x\in X(u,v)} \mathcal{M}_{u,x}^2}\times\sqrt{\sum_{x\in X(u,v)} \mathcal{M}_{v,x}^2}} \tag{6}$$

where $X(u,v)$ is the services set in the watchlists of both user u and user v, and $\mathcal{M}_{v,x}$ is the pseudo rating value of service x specified by user v. After computing the similarity between users, the scores of similarity will be sorted in descending order. Then the nearest neighborhood can be selected for service recommendation.

C. Recommend services

In order to measure the preference degree between user u and service s, we use Eq. (7) to calculate the value of preference degree. The higher the value, the higher rank position in recommendation list.

$$\mathrm{prefer}(\mathrm{u},\mathrm{s}) = \frac{\sum_{v\in Neighbor(u)} \mathcal{M}_{v,s}\times sim(u,v)}{|\sum_{v\in Neighbor(u)} sim(u,v)|} \tag{7}$$

where $Neighbor(u)$ denotes the neighbor set of active user u. After computing the predicted preference scores, the Top-K high score services will be recommended to an active user.

V. EXPERIMENTS AND EVALUATION

In this section, some experiments are designed to find out whether temporal and tag information will really impact users' preference or not. More specifically, several experiments are performed on a real-world dataset. Our experiments were implemented using MatLab 2010a with SQL Server 2005. All the experiments are performed in a PC with Windows 7 Professional, Intel Core (TM) i5 cpu @2.80G (4 core), and memory of 4G RAM.

A. Dataset

Firstly we give a brief introduction of our dataset. The experimental dataset comes from the public Web service registry center Programmableweb. PW is a well-known service registration center where Web APIs, mashups, member profiles and other data can be searched and retrieved through its own APIs[3]. By using these APIs, we leverage a crawler to fetch registered user's profile, user's watchlist and the corresponding service profiles including average ratings, timestamps and service tags (crawled on Nov-25-2013). All the data are stored in a database of SQL Server 2005 and released as public dataset file on the Internet[4]. From the all open data, we select 4568 users who have more than 20 services in their watchlist, and 6362 services (include APIs and mashups) are involved by these users. In our dataset, only the service publishing or updating time take into account other than the service collection time.

In order to simplify the complexity of the experiments and ensure the experimental result are independent with the particular dataset, we randomly selected 1000 users as the user set U and repeat our experiments on 5 randomly selected sampled subsets by using resampling technique. These users tracked services in their watchlist as service set S. Each dataset is consisted of many entries, each entry follows the form $\{\mathrm{u},\mathrm{s},\mathrm{t}_1 \,|\mathrm{t}_2\, | \ldots |\mathrm{t}_\mathrm{h},\mathrm{ts},\mathrm{r}\}$, in which h denotes the number of tags. ts denotes the timestamp of publishing or updating a service and r is the average rate of service s. After partitioning the subset, each subset is divided into two parts: the testing set covers 20% of most recent published services in the watchlist of each user, and the remaining 80% is the training set. Table II lists the statistical description of the whole dataset and each subset.

TABLE II DATASET STATISTIC DESCRIPTION

dataset	user	service	tags	density	watchlist
Overall	4568	6362	1869	0.004	116,589
Subset1	1000	3138	1338	0.0082	25,795
Subset2	1000	3252	1373	0.0079	25,724
Subset3	1000	3113	1354	0.0081	25,329
Subset4	1000	3228	1352	0.0080	25,670
Subset5	1000	3372	1389	0.0077	25,838

B. Evaluation metrics

We select **hit-ratio**, **hit-rank** and **diversification** as the metrics to evaluate the effectiveness of our method. The envaluation is based on the average values of these five subsets. All of the experiments are performed on each subset

[3] http://www.programmableweb.com/profile/username/alerts#watchlist

[4] http://pan.baidu.com/s/1c0vgjPA

to generate a fixed number of Top-10 services in the recommendation list.

1) Hit-ratio[22] quantifies the accuracy by calculating the intersection of recommended services and the services in the testing set for each user. It can be defined as Eq.(8):

$$\text{hit-ratio}=\frac{\sum_{u\epsilon U}\ number\_of\_hit}{m*|L|} \quad (8)$$

where m is the total number of the users and L is the length of the recommendation list ($|L|=10$). Hit-ratio=1 indicates that the recommender always provides right services, whereas hit-ratio=0 means that the recommender cannot give any right services to the user. However, one limitation of the hit-ratio measure method is that it treats all the hits equally regardless of their appearance in the top-K list.

2) Hit-rank[22] considers the position of each service in the hits of the recommendation list. The result can be used to effectively investigate the temporal impact in the watchlist. It is defined in Eq.(9):

$$\text{hit-rank}=\frac{1}{m*|L|}\sum_{u\in U}\sum_{i=1}^{h}\frac{1}{p_i} \quad (9)$$

where h is the number of hits occurring at the positions p_1, p_2, …,p_h within the recommendation list. So this paper uses hit-rank as the supplement metric of hit-ratio.

3) Diversification[13] identifies the uniqueness of the recommendation lists provided by different users. So it can be treated as the inter-user diversity.

$$\text{diversification}=\frac{2}{m(m-1)}\sum_{i\neq j}(1-\frac{|L_i\cap L_j|}{|L|}) \quad (10)$$

where L_i denotes the recommendation list of user u_i, $|L|$ is the length of the recommendation list, and m is the user number. In our experiment, |L|=10 and m=1000. Different values of diversification represent the variety of personalization in users' recommendation lists. We use diversification to find out whether the diversification degree varies for each method or not.

C. *Result*

In this section, four groups of experiments are discussed to evaluate our approach. The first group is used to measure the dataset sensitivity. The second group intends to find out the impact of temporal attenuate parameter α which can help us find out the best range of adjusted parameter value. The third group concentrates on the impact of weight balance parameter λ. The fourth group compares our method to the log-based method with different neighborhood sizes. The results of hit-ratio, hit-rank and diversification is the average results over 5 subsets.

1) *Dataset sensitivity*

In the first group, the temporal attenuate parameter is set as $\alpha=0.001$ and weight balance parameter $\lambda=0.1$. Suppose neighborhood size is 60, and the corresponding experiment results are shown in Table III. For each subset, the first row describes the data of hit-ratio, the second row is for hit-rank and the third row is diversification. The results show that the hybrid-based method brings the highest improvement of hit-ratio reaching 3.5% on subset 2. The average improvement of hit-ratio reaches 1.26%, and the average hit-rank increases nearly 0.43%. The diversification of tag-based method is better than other methods. Therefore, our method can get a relatively stable performance with different subsets.

2) *Impact of temporal attenuate parameter* α

The temporal attenuate parameter is used to adjust the decline speed of users' preference with time passing. The parameter λ changes from 0 to 1 with an interval of 0.05. Figure 4a shows that the best range of attenuate parameter is less than 0.01 (α<0.01). If the parameter is out of the specified range, the recommendation accuracy will decrease, even lower to the log-based method. In following tests, the attenuate parameter is set as $\alpha=0.001$.

TABLE III RESULT COMPARISON WITH DIFFERENT SUBSETS

Dataset	**User=1000**	**Neighborhood=60**	**topK = 10**	$\boldsymbol{\alpha=0.001}$	$\boldsymbol{\lambda=0.1}$
	Log-based	**Rate-based**	**Temporal-based**	**Tag-based**	**Hybrid-based**
Subset1	*0.1218*	*0.1115*	*0.1364*	0.1155	*0.1266*
	0.0632	*0.0557*	*0.0685*	*0.0537*	*0.0708*
	0.1246	*0.1244*	*0.1257*	*0.1498*	*0.1144*
Subset2	*0.1081*	*0.1348*	*0.1177*	*0.1311*	*0.1431*
	0.0544	*0.0656*	*0.0544*	*0.0550*	*0.0710*
	0.1044	*0.1045*	*0.1031*	*0.1419*	*0.1116*
Subset3	*0.1226*	*0.1311*	*0.1319*	*0.1312*	*0.1323*
	0.0696	*0.0648*	*0.0695*	*0.0532*	*0.0736*
	0.0978	*0.1193*	*0.0967*	*0.1491*	*0.1342*
Subset4	*0.1201*	*0.1282*	*0.1351*	*0.1149*	*0.1375*
	0.0654	*0.0657*	*0.0712*	*0.0503*	*0.0700*
	0.0975	*0.0946*	*0.0983*	*0.1460*	*0.1273*
Subset5	*0.1350*	*0.1353*	*0.1442*	*0.1250*	*0.1455*
	0.0699	*0.0687*	*0.0700*	*0.0561*	*0.0733*
	0.1293	*0.1300*	*0.1294*	*0.1637*	*0.1468*
Average	*0.1215*	*0.1282*	*0.1331*	*0.1235*	*0.1370*
	0.0645	*0.0641*	*0.0667*	*0.0537*	*0.0718*
	0.1107	*0.1146*	*0.1106*	*0.1501*	*0.1269*

Figure 4(b) illustrates the hit-ratio, hit-rank and diversification of recommendations versus the weight balance parameter λ respectively. The parameter λ is set from 0 to 1 with an interval of 0.1. The result shows that the parameter λ will influence the actual performance of recommendation. And the performance is the best when λ is 0.1, which will lead to the highest performance of hit-ratio and hit-rank. However, the diversification of hybrid-based method is even lower to that of log-based method when λ locate in [0.2, 0.5]. So it is necessary to mix these two weights with right balance parameter. Otherwise it will lead to negative effects.

3) Impact of neighborhood size

Five methods (i.e. log-based, rate-based, tag-based, temporal-based and hybrid-based) are compared with each other using different neighbor size $Neighbor \in [10, 100]$ by an interval of 10. From Figure 4(c), we can find that the neighborhood size will brings a significant impact on the accuracy of recommendation. The hybrid-based method outperforms the log-based method using metrics of hit-ratio and hit-rank. We can also find that the tag-based method can get even better result of hit-ratio than the hybrid method when neighborhood size less than 50. This makes it possible to dynamically adjust the weight balance parameter to get the best value by changing the size of neighborhood. In some cases, the performance of rate-based methods is lower than that of the log-based method because we only take parts of users as experiment dataset. Howerver, the average rate value is calculated by all users in a repository. Although, hybrid-based method does not always work better than separate performance of either temporal-based or tag-based approaches. But the performance of the hybrid method can always better than the log-based methods if the mixed parameter in the right ranges, which is the main reason why we choose this hybrid method. It is also implies that our hypothesis on both tag and time information is crucial for personalized recommendations of services.

D. Experiment summary

The complexity of both user similarity and the preference degree between users and services is $O(mn)$. m is used to describe the number of users and n is the number of services. In our recommender system, the similarities between users are calculated offline and updated periodically. To shorten the computing time of user similarity, some data are processed and stored on temporary tables. The user-service matrix and similarity matrix is addressed as the sparse matrix in Matlab to reduce the storage space.

Compared to the traditional log-based CF method for service recommendations, our proposed method can make a better performance in most cases. However, some problems need to be addressed such as : (1) suffering from the cold start problem; (2) handling with data sparse problem; and (3) lacksing of scalability with increasing numberof users.

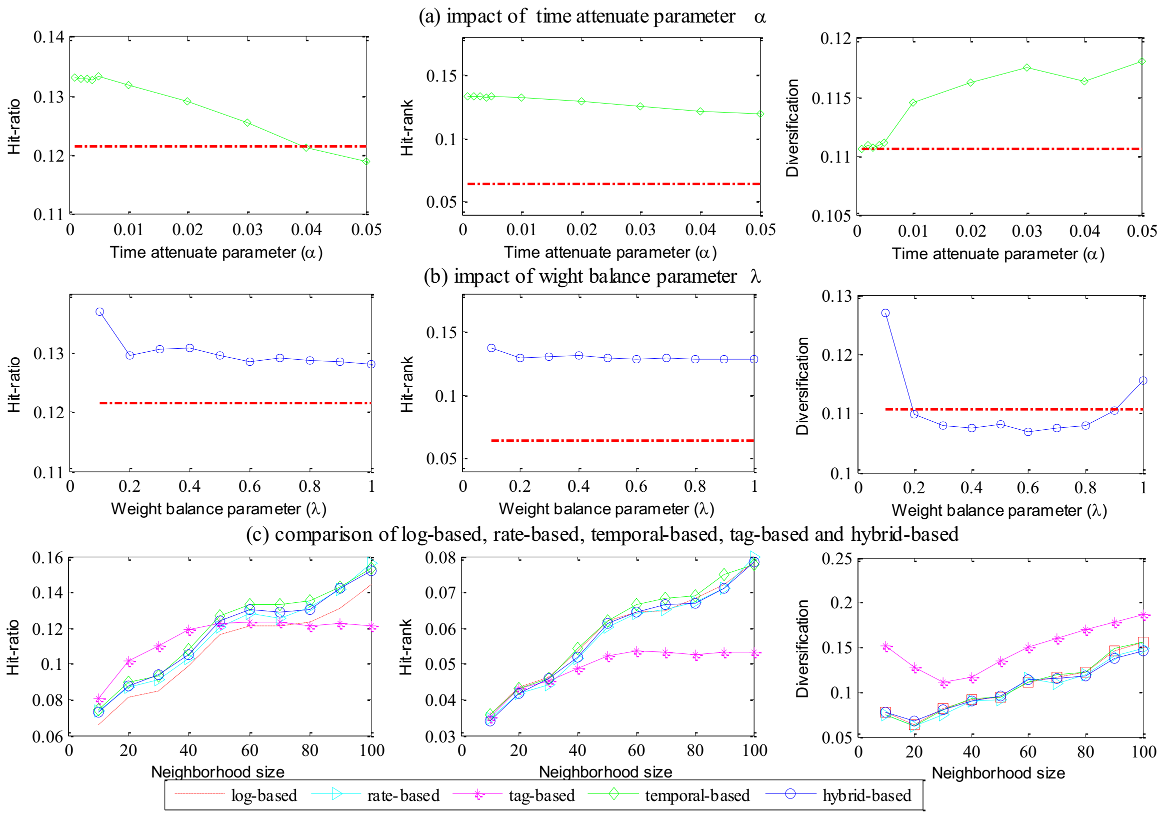

Figure 4. Plot of experimental results

VI. Conclusion

This paper examine how to exploit temporal and tag information to build a more effective recommendation model for Web service. Then a service recommender system is proposed to provide high-quality service recommendation with users' implicit feedbacks even in the absence of explicit ratings and QoS information. One of the key issues of our research is to address whether tag-based and temporal-based methods can capture users' preferences or not. The experimental results show that our approach can provide better performance than traditional log-based method.

However, this paper just presents the start point of designing a hybrid method with temporal and tag information. Some issues need to be studied in the near future. For example, our experimental dataset only treat the service publishing or updating time as temporal information other than service collection time in watchlist. Therefore, it is necessary to apply our method to other dataset to find the correlation between users' preference and service collection time. More specifically, the further work covers three aspects: (1) how to deploy our recommender system in a parallel distributed platform or cloud platform to get a better running environment to improve the speed; (2) how to verify the effectiveness of with other dataset; and (3) how to combine other recommendation strategies such as matrix factorization[23], to overcome the problem of cold start and data sparse.

Acknowledgment

This research project was supported by the National Basic Research and Development Program (973 program) of China (No.2014CB340404), the National Natural Science Foundation of China under Grant No.61373037, No. 61202031 and No. 61100017, National Science & Technology Pillar Program of China under grant No.2012BAH07B01, the Central Grant Funded Cloud Computing Demonstration Project of China undertaken by Kingdee Software (China) co., ltd., the Science and Technology Innovation Program of Hubei Province No.2013AAA020.

References

[1] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," IEEE Transactions on Knowledge and Data Engineering, vol. 17, pp. 734-749, 2005.

[2] X. Zhang, K. He, C. Wang, Z. Li, and J. Liu, "Interest-Driven Web Service Recommendation Based on MFI-7,", 2013 IEEE International Conference on Services Computing (SCC), pp. 438-445, 2013.

[3] L. Xiang, Q. Yuan, S. Zhao, L. Chen, X. Zhang, Q. Yang, et al., "Temporal recommendation on graphs via long-and short-term preference fusion," in Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 723-732 ,2010.

[4] L. Chen, L. Hu, Z. Zheng, J. Wu, J. Yin, Y. Li, et al., "Wtcluster: Utilizing tags for web services clustering," in Service-Oriented Computing, ed: Springer, pp. 204-218, 2011.

[5] U. Chukmol, A. N. Benharkat, and Y. Amghar, "Enhancing web service discovery by using collaborative tagging system," 2008. NWESP'08. 4th International Conference on Next Generation Web Services Practices, pp. 54-59,2008.

[6] A. Sunikka and J. Bragge, "Applying text-mining to personalization and customization research literature–Who, what and where?," Expert Systems with Applications, vol. 39, pp. 10049-10058, 2012.

[7] N. Manouselis and C. Costopoulou, "Experimental Analysis of Multiattribute Utility Collaborative Filtering on a Synthetic Data Set," Personalization Techniques and Recommender Systems, Series in Machine Perception and Artificial Intelligence, vol. 70, pp. 111-134, 2008.

[8] D. W. Oard and J. Kim, "Implicit feedback for recommender systems," in Proceedings of the AAAI workshop on recommender systems, 1998, pp. 81-83.

[9] Q. Yu, "Decision Tree Learning from Incomplete QoS to Bootstrap Service Recommendation," 2012 IEEE 19th International Conference on Web Services (ICWS), pp. 194-201, 2012.

[10] Z. B. Zheng, H. Ma, M. R. Lyu, and I. King, "QoS-Aware Web Service Recommendation by Collaborative Filtering," IEEE Transactions on Services Computing, vol. 4, pp. 140-152, Apr-Jun 2011.

[11] J. Peng, D. D. Zeng, H. Zhao, and F.-y. Wang, "Collaborative filtering in social tagging systems based on joint item-tag recommendations," in Proceedings of the 19th ACM international conference on Information and knowledge management, pp. 809-818, 2010.

[12] Z. K. Zhang, T. Zhou, and Y. C. Zhang, "Tag-aware recommender systems: a state-of-the-art survey," Journal of Computer Science and Technology, vol. 26, pp. 767-777, 2011.

[13] Z. K. Zhang, T. Zhou, and Y. C. Zhang, "Personalized recommendation via integrated diffusion on user–item–tag tripartite graphs," Physica A: Statistical Mechanics and its Applications, vol. 389, pp. 179-186, 2010.

[14] T. Q. Lee, Y. Park, and Y. T. Park, "A time-based approach to effective recommender systems using implicit feedback," Expert systems with applications, vol. 34, pp. 3055-3062, 2008.

[15] Y. Ding, X. Li, and M. E. Orlowska, "Recency-based collaborative filtering," in Proceedings of the 17th Australasian Database Conference-Volume 49, pp. 99-107, 2006.

[16] N. Zheng and Q. Li, "A recommender system based on tag and time information for social tagging systems," Expert Systems with Applications, vol. 38, pp. 4575-4587, 2011.

[17] X. Wang, Z. Wang, and X. Xu, "An Improved Artificial Bee Colony Approach to QoS-Aware Service Selection," 2013 IEEE 20th International Conference on Web Services, pp. 395-402, 2013.

[18] Y. Feng, L. D. Ngan, and R. Kanagasabai, "Dynamic Service Composition with Service-Dependent QoS Attributes," 2013 IEEE 20th International Conference on Web Services, pp. 10-17, 2013.

[19] J. Cao, Z. Wu, Y. Wang, and Y. Zhuang, "Hybrid Collaborative Filtering algorithm for bidirectional Web service recommendation," Knowledge and Information Systems, pp. 1-21, 2012.

[20] Y. Jiang, J. Liu, M. Tang, and X. Liu, "An effective web service recommendation method based on personalized collaborative filtering," 2011 IEEE International Conference on Web Services (ICWS), pp. 211-218, 2011.

[21] X. Chen, X. Liu, Z. Huang, and H. Sun, "RegionKNN: A scalable hybrid collaborative filtering algorithm for personalized web service recommendation," in 2010 IEEE International Conference on Web Services (ICWS), pp. 9-16, 2010.

[22] M. Deshpande and G. Karypis, "Item-based top-n recommendation algorithms," ACM Transactions on Information Systems (TOIS), vol. 22, pp. 143-177, 2004.

[23] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," Computer, vol. 42, pp. 30-37, 2009.