PERSONALIZED SERVICE RECOMMENDATION BASED ON PSEUDO RATINGS BY MERGING TIME AND TAG PREFERENCE

Xiuwei Zhang1,2, Chong Wang1*, Jian Wang1, Jianxiao Liu3, Tian Gang1 and Keqing He1
1 State Key Laboratory of Software Engineering, School of Computer, Wuhan University, China;
2 No. 94005 Troops of PLA, Jiuquan, China
3 College of Informatics, Huazhong Agricultural University, Wuhan, China
{zhangxiuwei, cwang, jianwang, tiangang, liujianxiao, hekeqing}@whu.edu.cn

Abstract
With the rapid development of service-oriented computing technologies, large amounts of Web services have been released on the Internet to facilitate system construction. Consequently, personalized service recommenders are emerging to handle information overload in service computing. Collaborative filtering (CF) is popular for service recommendations based on explicit ratings or Quality of Service (QoS) information provided by users. In real cases, however, it is difficult to collect explicit feedbacks since most available feedbacks are implicit. In this paper, the authors propose an implicit-to-explicit rating approach, which can leverage the implicit feedback from user's collecting behavior to build a CF-based recommender system for Web services. Firstly, a user-service binary matrix is constructed based on the collection records in the watchlist. Secondly, the reputation rating, publishing time and tag information of services are combined into the previous binary matrix to generate a pseudo rating matrix, which can reflect users' preference changing more precisely. Thirdly, both the traditional and enhanced user-based CF methods are used to generate a personalized service list with these pseudo ratings. Finally, a set of experiments are designed to validate the proposed service recommendation approach based on a large scale and real-world dataset from a well-known service registry center Programmableweb (PWeb). The experimental results show that our hybrid recommendation method is more efficient and accurate than the traditional log-based CF methods.

Keywords: Service recommendation; Web service; implicit feedback; collaborative filtering; pseudo ratings

1. INTRODUCTION
Nowadays, more and more recommendation systems are constructed to help people sift through available web sites, books, articles, movies, music, jokes, hotel, restaurants, grocery products, Web services and so forth to satisfy their personalized interests or valuable information requirements (Su & Khoshgoftaar, 2009). In particular, large amounts of Web services have been developed by different enterprises and organizations, represented in different description languages, and stored in different repositories. This leads to the fact that service consumers or users, including system developer, mashup designer, and end user et al., are inundated with the mass of services. Keyword search is one of the most popular approaches for service discovery on the Web, which uses specific search criteria to find the interested services with relevant information. However, keyword-based search engine is designed in a one-fit-all mode and cannot meet users’ personalized demands (Adomavicius & Tuzhilin, 2005). Actually, the automatic discovery model in a service recommender system is quite different with that of search engines. Service recommender systems intend to recommend services according to user’s preference and profile, so it is helpful for service users or service consumers to select appropriate services in a large space of possible options with a personalized way (X. Zhang, He, Wang, Li, & Liu, 2013).

Recently, collaborative filter (CF) seems to be the most popular and successful recommendation technique when designing recommender systems for information overload. The fundamental assumption of CF is that if several users rate some items similarly or have similar operations (e.g., using, buying, listening, watching, clicking, tagging, et al.), then they may have similar behaviors or give similar rate on other items (Su & Khoshgoftaar, 2009). In detail, CF defines a matrix to record user’s preferences for some items, which can used to predict potential topics or products that a new user might like. In typical CF scenario, the preference ratings can be either explicit feedbacks (e.g., 1–5 stars or levels) or implicit indications (e.g., purchase rate or click-through) (Miller, Konstan, & Riedl, 2004).

Existing researches on Web service recommendation usually take QoS (Quality of Service) as the runtime indicator for service recommendation (Z. Zheng, Ma, Lyu, & King, 2011). Most of them focus on how to optimize QoS data because it not only identifies the significant non-functional characteristics of Web services but also can be captured and utilized easily. Generally, QoS-based CF methods use similar QoS experience of users to accurately predict that of an active user receiving from previously unknown services (Yu, 2012). In most cases, however, it is difficult for users to precisely define and describe QoS. Moreover, it is also hard to collect QoS since it is always changing dynamically. To relief the lack of both user’s explicit rating and QoS information, this paper intends to convert the available implicit feedbacks (e.g., bookmarking behavior) into personalized pseudo ratings, rather than
explicit ratings. The implicit feedbacks related to service bookmarking (or watchlist records) of users was crawled and collected from an open APIs and mashups repository, called ProgrammableWeb\(^1\) (PWeb for short). Since both APIs and mashups in PWeb are reusable service components, we will use the terms API and mashup as service interchangeably throughout this paper.

1.1 Motivation

Recently, Web services are increasing on the Internet due to its interoperability and reusability. Considering PWeb, more than 10,000 APIs and 7,000 mashups have been released until June 2014, and the main service protocols or types include REST, SOAP, JSON-RPC, etc. PWeb provides not only the keyword-based search engine and category filter, but also the most popular and the latest recommendation method for service selection. However, the recommendation mechanism in PWeb is based on one-fit-all modes and lack of personalization. Particularly, not all the users can express their personalized service requests clearly and definitely, which is beyond the capability of the keyword-based search engine in PWeb.

\[\text{Figure 1. The Motivating Example}\]

Suppose there are three PWeb users, i.e. Alice, Bob, and Cathy. Each of them wants to find some services in PWeb to construct a service-oriented system respectively. In PWeb, a user can choose his/her interested services by collecting the service. In this case, to add a service into his/her watchlist means that the user is interested in this service. As Figure 1 shows, Alice tracked two services named “3D Geology Maps” and “Amazon Discount”, Bob tracked “3D Geology Maps” and “AmazonHive”, and Cathy only tracked the service called “Amazon Discount”. Mostly, services might not be recommended to users accurately with service bookmarking records only. The main reason is that the user’s preference for service is constantly changing. Considering the motivating example in Figure 1, the system needs to recommend some services to the potential service user named Alice. Usually, it will try to find out her neighbors who have the similar preferences, and then recommends the most similar services in her neighbors’ favorites to Alice. In this case, “Amazon Discount” tracked by Bob will be recommended to Alice because both Alice and Bob collects the same service “3D Geology Maps”. Then, Bob will be one of the neighbors of Alice. Similarly, Cathy is also a neighbor of Alice because both of them are interested in “Amazon Discount” service. It is obvious that both Bob and Cathy bookmarked one service with Alice. If the system will recommend the service collected by the nearest neighbor of Alice, then the problem is to decide who her nearest neighbor is.

1.2 Contribution

Although great deal of valuable information can be deduced from the service bookmarking operation of users, the challenge is how to use such kind of information for personalized service recommendation. Service tags, service publishing time and service reputation ratings are three main factors in personalized service recommendation. In order to provide a reasonable and efficient solution for the motivating case in Figure 1, this paper intends to use the implicit feedbacks relevant to service publishing time and service tags to enrich and quantify users’ preference. More specifically, the main contribution of this paper can be addressed from the following three aspects:

- Time related information and tag-based ratings are merged with weight balance parameter to transform users’ implicit feedbacks into the explicit ratings of services. Meanwhile, a preference drift function is defined as an adaptive exponential forgetting function to resolve the cold start problem in service recommendation.
- The traditional user-based CF method is extended with case amplification to recommend services by analyzing user’s historic implicit feedback records, rather than collecting and processing explicit ratings and QoS data of services.
- A set of experiments are designed based on a large scale and real-world dataset from PWeb to verify the proposed recommendation method. The results show that our hybrid method works better than log-based methods. And the experimental dataset is released online\(^2\) to extend and improve our method in the near future.

The remainder of this paper is organized as follows. Section 2 discusses the related work. Section 3 addresses the problem and foundation of our approach. Section 4 explores the overview and details of our approach to personalized service recommendation. Section 5 evaluates the proposed approach with the experiments and the corresponding results. Section 6 summaries our paper, followed by the future work.

2. RELATED WORK

\(^1\)http://www.programmableweb.com

\(^2\)http://pan.baidu.com/s/1nt81acD
In recent years, service recommendation becomes one of the hot issues in service computing. This section mainly concentrates on the existing researches related to QoS-aware service recommender and the implicit feedback based recommender.

2.1 QoS-aware Service Recommender

The increasing number of services with similar functionalities brings the challenge on how to select an appropriate service with higher quality. QoS-aware service recommendations intends to optimize service selection by considering both users preferences and the QoS attributes of candidate Web services with similar functionalities (Deng, Huang, & Xu, 2014). QoS is mainly described by several performance factors, such as availability, response time, reliability, throughput, etc. They have been considered and utilized in previous studies to facilitate service selection (Wang, Wang, & Xu, 2013), service composition (Feng, Ngan, & Kanagasabai, 2013) and service recommendation (Cao, Wu, Wang, & Zhuang, 2013; Jiang, Liu, Tang, & Liu, 2011; Z. Zheng et al., 2011). Particularly, Karta (Karta, 2005) pointed out that the main difference between service selection and service discovery is whether QoS criteria is used or not. And he believed that service recommendation will benefit from more personalized preference than service functional query. Currently, most service recommendation approaches aim to predict the exact QoS values of Web services and extract users’ interests or preferences from the history or logs of service invoking.

CF is the mainstream technique for recommendation in existing researches. Users with similar QoS experiences on Web services are treated as similar neighbors. For any active user, the QoS values of a Web service can be predicted by analyzing the services invoked by the users who are similar to him/her. Then the services with higher QoS values will be recommended to the active user. More specifically, Shao et al. (Shao et al., 2007) proposed a user-based CF algorithm using PCC (Pearson Correlation Coefficient) to compute the similarity between users. Zheng et al. (Z. Zheng et al., 2011) presented a Web service recommender system called WSRec, using the hybrid of user-based and item-based CF in its key recommendation algorithm. Jiang, et al. (Jiang et al., 2011) improved the traditional hybrid recommendation with deviation of the QoS information. Similarly, Cao et al. (Cao et al., 2013) presented a standard deviation based hybrid CF for Web service recommendation and an inverse consumer frequency based CF for potential consumer recommendation. It enables both service recommendation for service users and user recommendation for service provider. Chen et al. (X. Chen, Liu, Huang, & Sun, 2010) proposed a novel hybrid collaborative filtering algorithm named RegionKNN, which introduces user location into traditional CF to support large scale recommendation of Web service. Lin et al. (Lin, Shi, & Ishida, 2012) proposed a service selection approach with context-aware QoS. The context-aware factors with effect on QoS attributes are extracted by analyzing their correlation with QoS attributes. Then, QoS data can be generated from the extracted factors for prediction and evaluation.

In general, QoS-based service recommendation methods are based on the assumption that the QoS information is available and accurately expresses user’s preferences. But in practice, the feedback are simulated. Therefore, it is difficult for them to collect QoS information because of it is always dynamically changing in the client side (Z. Zheng et al., 2011). Thus, implicit feedback is considered by service recommender systems. Compared to QoS-based service recommendation, researches on implicit feedback based service recommendation are seldom reported since it is not a easy work to collect implicit feedbacks in service registries. If simulated implicit feedback data is taken into account, it will significantly impact the authenticity of the recommendation due to the phenomena of “Garbage in, and Garbage out”.

2.2 Implicit Feedback Based Recommender

2.2.1 Tag-based Service Recommender

Tags can help users organize, index and facilitate retrieval of his/her interests. Web services can be annotated by the service users or domain experts with tags when publishing. Tags can be considered as the highly abstracted content features of the attached services (L. Chen et al., 2011). Generally, service description specifications (e.g. WSDL) cannot provide sufficient textual content to extract service tags. How to acquire service tags and use them to represent user’s preference becomes a great challenge in service recommendation. Although semantic annotation can be added into textual description to extend the capability of existing service description languages, such as SAWSDL (Semantic Annotation language for WSDL), OWL-S and WSDL-S, tedious and error-prone manual work for service tagging is required inevitably for service providers (Chukmol, Benharkat, & Amghar, 2008).

It is well known that social tagging plays an important role in organizing and discovering information on the Web (Peng, Zeng, Zhao, & Wang, 2010). Many researchers investigated different strategies for tag-based recommendation. Those tag-based strategies mainly fall into the following three categories (Z.-K. Zhang, Zhou, & Zhang, 2011):

- Network-based model: A tag-based network is viewed as a tripartite graph, consisting of three integrated bipartite graphs or a hyper-graph. Zhang et al. proposed an integrated diffusion on tripartite graph for resource recommendation (Z.-K. Zhang, Zhou, & Zhang, 2010). And Deng et al. defined a social network-based method for service recommendation with user’s trust (Deng et al., 2014).
- Tensor-based model: The tensor factorization (TF)-based method is very popular when designing recommendation algorithms with social tags. TF is based on singular value decomposition (SVD), with
which the ternary relation can be used to reduce dimensions of features (Z.-K. Zhang et al., 2011).

- Topic-based model: The topic-based method exploits tags in a probabilistic framework. Each tag is a topic indicator and can be used to estimate the probability of annotating an item by summing up the transition probabilities through all tags (Peng et al., 2010). It supposes that if a user is interested in an item in a specific topic, he/she will also like other items in the same topic. The most popular topic-based methods include LSA (Latent Semantic Analysis), TF-IDF (term frequency-inverse document frequency), LDA (Latent Dirichlet Allocation), Ontology, etc.

### 2.2.2 Time-based Service Recommender

Time is another key factor for users’ preference. When introducing time information into recommendation, the main problem is how to find the correlation between time dynamics and users’ preference (Xiang et al., 2010). In general, more services are used recently, and more potential purchase time and item launch time should be considered in users’ interests. Some relevant works investigate the importance of time in recommendations (Lee, Park, & Park, 2008) to improve the accuracy of recommendations. Furthermore, they analyzed a variety of time information, including item launch time, users’ ordering time, and the difference between those two kinds of time information. The user’s rating time is taken into account in (Ding, Li, & Orlowska, 2006) to improve the precision of item-based collaborative filtering. Zheng and Li (N. Zheng & Li, 2011) presented a implicit feedback based recommender system with time information for social tagging. In detail, an exponential time decay function is designed to compute time weights for different items according to the corresponding bookmarking time and item launched time. Xiang et al. (Xiang et al., 2010) proposed a session-based temporal graph (STG) to simultaneously model user’s long-term and short-term preferences over time. In the latest related studies, Zhong et al. (Zhong, Fan, Huang, Tan, & Zhang, 2014) proposed a time-aware service recommendation to create a service ecosystem. Hu, et. al. (Hu, Peng, & H, 2014) introduced time information into service recommendation for data sparse. It seems to be an important cue that implicit feedback of time information is taken into account in service recommendation.

Those implicit feedback-based recommendation methods mentioned above intended to handle the temporal dynamics and tags in various ways. In addition, most of them are validated and evaluated in specific applications, rather than real-world dataset for service recommendation. Differently, this paper explicitly models the variation of user’s preferences by analyzing their behaviors in service bookmarking. Furthermore, we leverage users’ personalized implicit feedbacks in their watchlists to recommend services with an extended user-based CF method, which merges time and tag information into pseudo ratings.

### 3. Problem Definition

This section introduces the basic notations and definitions required in our approach to service recommendation. The authors define four different kinds of communities in service recommendation, i.e. users, services, tags and timestamps. In this paper, $U = \{u_1, u_2, ..., u_\mid U \}$ denotes a set of users, $S = \{s_1, s_2, ..., s_\mid S\}$ denotes a service set, $T = \{t_1, t_2, ..., t_\mid T\}$ denotes a tag set of a service, and $TS = \{t_s_1, t_s_2, ..., t_s_\mid TS\}$ denotes a timestamp set of a publishing service. The relationship among services, tags and timestamps is expressed as $CR = \{< s, t_1|t_2 ... |t_i, t_s, r_s >\}$, in which $s$ denotes the service name, $t_i$ denotes one of the tags of the service $s$, $t_s$ denotes the publishing time or updating time of the services $s$, and $r_s$ denotes the reputation rating of $s$.

In this paper, watchlist is defined to represent the bookmarking set of user’s favorite services.

**Definition 1. (Watchlist):** A service bookmarking set with a RSS Feed service, which can deliver regular changes of a service to any user who is interested in it. The watchlist of user $u$ can be expressed as $WL_u$, with a four tuple:

$$WL_u = \langle u, S_u, T_u, CR_u \rangle \quad (1)$$

where $u \in U$ denotes the owner of the watchlist, $S_u$ denotes the service set tracked by user $u$ in the watchlist ($S_u \subseteq S$), $T_u$ denotes the tag set of user $u$ ($T_u \subseteq T$), and $CR_u (CR_u \subseteq CR)$ denotes the relationship among $S_u, T_u$ and $TS_u$.

![Figure 2. A exemplary Watchlist in PWeb.](image)

Figure 2 shows an exemplary watchlist in PWeb. The left part of Figure 2 is a snapshot of a watchlist in PWeb and the right part is the corresponding abstract representation, as the followings illustrate.

- **user name:** $u = \{‘Haineszhang’\}$
- **service set:** $S_u = \{s_1, s_2, ..., s_{\mid S_u}\}$
- **tag set:** $T_u = \{t_1, t_2, ..., t_{\mid T_u}\}$
- **correlation records:** $WL_u = \{s_1, t_1|t_2 ... |t_{\mid T_u}, 20130316,5\}$
  $\{s_1, t_1|t_2 ... |t_{\mid T_u}, 20130801,4\}$
  $\{s_1, t_6|t_1, 20131226,3\}$
  $\{s_1, t_6|t_1, 20131218,4\}$
  $\{s_1, t_6|t_1, 20130316,5\}$
Definition 2. (Preference Drift Function, PDF): A kind of user preference drift model, describing the features of change tendency associated with user’s preference over time. In Eq.(2), the preference drift function is defined as an adaptive exponential forgetting function, which is similar to Newton’s law of cooling:

\[ w_{\text{pref}}(u,s) = \exp(-\alpha \times \text{time}(u,s)) \]  

(2)

where \( w_{\text{pref}}(u,s) \) denotes the weight of user’s interest to identify the declined degree of a user’s preference. \( \alpha \) denotes the time attenuation parameter which can be used to adjust the preference drift speed. \( \text{time}(u,s) \) is a non-negative integer \( (\text{time}(u,s) \geq 0) \), denoting the number of services in user \( u \)’s watchlist at that time. The latest published or updated service in watchlist is recorded as 0, and the penultimate service is 1. The time weight strategy is based on the hypotheses that people’s interest is always drifting over time. The preference drift function curve with different \( \alpha \) is shown in Figure 3, in which \( \alpha \) is an adjusted parameter for user’s preference drift speed. Obviously, as shown in Figure 3, the preference drift speed will drift steeply with the increment of \( \alpha \).

![Figure 3. Preference Drift Function Curve with different \( \alpha \) ](image)

4. APPROACH

4.1 Overview of Service Recommendation

The main goal of our approach is to recommend services in a precise and personalized manner. Figure 4 illustrates the overall structure of the recommendation model proposed in this paper. Our recommendation model mainly consists of three parts, i.e. the implicit feedback database, the pseudo rating matrix generator and the recommender system. The pseudo rating generator is the key component in the proposed recommendation model. It’s designed to transform implicit feedbacks into explicit ratings by merging time information and tag preference. Then a more accurate pseudo rating can be generated for service recommendation. More specifically, the pseudo rating generator contains a binary rating matrix generator, a reputation rating matrix generator, a time weight rating matrix generator, a tag weight rating matrix generator and a hybrid rating matrix generator.

![Figure 4. Overview of our recommendation model](image)

4.2 Computing Pseudo Ratings

Compare to the implicit binary rating matrix, Algorithm 1 explains how to leverage time and tag information as positive factors to improve the accuracy of service recommendation.

![Algorithm 1. Service recommendation procedure](image)

The inputs of Algorithm 1 include the user set \( U (|U| = m) \), the service set \( S (|S| = n) \), the watchlist set \( WL(|WL| = m) \), the time attenuation parameter \( \alpha \), the weight balance parameter \( \lambda \) that is used to adjust the weight of time and tag, and the case amplification power \( \rho \) (Breese, Heckerman, & Kadie, 1998) that emphasizes high similarity and punishes low similarity. The corresponding output of Algorithm 1 is the Top-K recommended services list \( L \). The detailed explanation of the case amplification will be discussed in section 4.4.
Considering how to generate the pseudo rating matrix, the sample data is listed in Table I. There are three users and three services. The value in the square bracket means the reputation rating of a service. Time information of a service is defined as $time(u,s)$ in braces. The publishing time of a service is represented as ‘YYYYMMDD’. Several service tags can be split with ‘$|$’, and the symbol ‘#’ means no interaction between users and services.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_1$</td>
<td>t:</td>
<td>b</td>
</tr>
<tr>
<td>$u_2$</td>
<td>t:</td>
<td>b</td>
</tr>
<tr>
<td>$u_3$</td>
<td>#</td>
<td>t:</td>
</tr>
</tbody>
</table>

Furthermore, the following five steps should be taken to generate the pseudo rating matrix of services. The outcome of each step is a user-service matrix, which will be assigned with pseudo rating values by a specific mechanism.

### 4.2.1 Pseudo Rating Matrix of Binary

Suppose that if a user collects a service, the basic rating of the service given by the user is 1; if not, it is 0; If service $i$ exists in user $j$’s watchlist, then the rating $a_{ij} = 1$; otherwise, $a_{ij} = 0$. The binary 0-1 rating matrix can be annotated as $A_{m \times n}$, where $m$ and $n$ denote the numbers of users and services respectively. For instance, the baseline matrix can be the following one, which illustrates that $u_1$ adds both $s_1$ and $s_3$ in his/her watchlist, $u_2$ adds $s_1$ and $s_2$ in his/her watchlist, and $u_3$ only adds $s_3$.

$$A_{3 \times 3} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

### 4.2.2 Pseudo Rating Matrix of Reputa

In this paper, the reputation score is mainly used to relieve the lack of individualized rating data. It is often expressed as one to five stars from in PWWeb. Then the reputation matrix can be constructed by replacing ‘1’ in the baseline matrix with the service reputation scores listed in Table 1. Then the reputation matrix $\mathbb{B}$ of the case in 4.2.1 is as follows:

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

### 4.2.3 Pseudo Rating Matrix of Time

Generally, preference drifting adopts either time window or forgetting functions to learn and track the changes of users’ behaviors over time. Unfortunately, most of the time window methods completely ignore the historic information (N. Zheng & Li, 2011). In this paper, 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. In Newton’s law of cooling, the temperature will gradually decrease until it’s equal to the environment temperature. Similarly, in preference drift, the newly published services in the watchlist have higher impact than that of the existent services in the watchlist. Eq. (2) in section 3 can be used to measure the preference drifts over time. The time weighted matrix is expressed as $C_{m \times n}$, and the following shows results when the time attenuate parameter $\alpha = 0.5$.

$$C_{3 \times 3} = \begin{bmatrix} 0.6065 & 0 & 1 \\ 1 & 0.6065 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

### 4.2.4 Pseudo Rating Matrix of Tag

Tags are usually given by users, service publishers, and domain experts. This paper only focuses on the service tags annotated by service publishers and the domain experts. Therefore, it’s possible for each user to collect same tags when linking their preferences to the same service. The tag weight of a service is defined as Eq.(3) (N. Zheng & Li, 2011) to measures a user’s preference for a specified service, i.e. the degree of user $u$’s interests in service $s$:

$$w_{tud}(u,s) = \sum_{t \in T(u,s)} w_{ut}$$

where $T(u,s)$ denotes a tag set of services in user $u$’s watchlist, $w_{ut}$ denotes the score of each tag $t$ in $T(u,s)$. More specifically, the tag score $w_{ut}$ can be generated with Eq.(4):

$$w_{ut} = \frac{\text{frequency}(u,t)}{\sum_{i=1}^{s} \text{frequency}(u,t)}$$

where $\text{frequency}(u,t)$ denotes how many times the tag $t$ occurs in the watchlist of user $u$. $\sum_{i=1}^{s} \text{frequency}(u,t_i)$ denotes how many times all the tags occurs in user $u$’s watchlist. Tag score is a non-negative real number between 0 and 1 ($w_{ut} \in [0,1]$), and $\sum_{i=1}^{s} w_{ut} = 1$. With Eq.(3) and Eq.(4), the tag-weighted matrix $\mathbb{D}_{m \times n}$ can be generated as follow:

$$\mathbb{D}_{3 \times 3} = \begin{bmatrix} 0.8 & 0 & 0.6 \\ 0.8 & 0.6 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

### 4.2.5 Pseudo Rating Matrix of Hybrid

Finally, the time weighted matrix, the tag weighted matrix and the reputation weighted matrix are combined as the hybrid rating matrix $\mathbb{E}$, as Eq.(5) defines.

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

where parameter $\lambda$ is introduced to adjust the weights of time-weighted matrix and tag-weighted matrix. For instance, $\lambda = 0.5$ means that time weight and tag weight share the same importance degree. The corresponding hybrid pseudo rating matrix for the sample data is generated as follow.

$$\mathbb{E}_{3 \times 3} = \begin{bmatrix} 2.813 & 0 & 4 \\ 3.6 & 1.8098 & 0 \\ 0 & 0 & 5 \end{bmatrix}$$

Note that, the tag matrix, time matrix and reputation matrix either can be used alone or work together, which depends on the implicit feedback dataset.
4.3 Finding Neighbors

Based on the generated pseudo rating matrix \( \mathbf{E} \), the traditional user-based CF method can be adopted to get the recommendation list of services. Although there are many methods to compute 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:

\[
\text{sim}(u,v) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\| \|\mathbf{v}\|} = \frac{\sum_{x \in X(u,v)} M_{u,x} \times M_{v,x}}{\sqrt{\sum_{x \in X(u,v)} M_{u,x}^2} \times \sqrt{\sum_{x \in X(u,v)} M_{v,x}^2}}
\]  

(6)

where \( X(u,v) \) is a set of common services in the watchlists of both user \( u \) and user \( v \), and \( M_{u,x} \) is the pseudo rating of service \( x \) specified by user \( v \). After computing the similarity between users, the nearest neighborhood of a certain user will be selected.

4.4 Predicting User’s Preference

The pseudo ratings of the target user \( u \) for a service \( s \) is estimated by combining the ratings of the top K neighbors of user \( u \) and that of the target users for the service \( s \), seen in Eq.(7). After confirming the neighbors of neighborhoods, we can predict the preference score of each user to each service. The services which are of higher preference scores and not been tracked by any target user will be recommended. Eq.(7) is defined to measure and predict whether user \( u \) prefers service \( s \) or not.

\[
\text{preference}_\text{score}(u,s) = \frac{\sum_{v \in \text{Neighbor}(u)} M_{v,x} \times \text{sim}(u,v)}{\sum_{v \in \text{Neighbor}(u)} \text{sim}(u,v)}
\]  

(7)

where \( \text{Neighbor}(u) \) denotes the neighborhoods of user \( u \).

In our proposed method, case amplification (Breese et al., 1998) is also introduced to emphasize higher similarities and punish lower similarities. The amplification function is defined in Eq. (8).

\[
\text{sim}' = \text{sim} \times |\text{sim}|^{\rho - 1}
\]  

(8)

where \( \text{sim} \) is the similarity before case amplification, \( \text{sim}' \) is the similarity after case amplification, and \( \rho \) is the case amplification power and \( \rho \geq 1 \), which is typically assigned as \( \rho = 2.5 \) (Lemire, 2005). Case amplification is mainly used to reduce noise in the dataset. It intends to favor high weights as small values through a power become negligible. The preference score based on amplification similarity is measured by Eq. (9).

\[
\text{preference}_\text{score}_\text{amp}(u,s) = \frac{\sum_{v \in \text{Neighbor}(u)} M_{v,x} \times \text{sim}'(u,v)}{\sum_{v \in \text{Neighbor}(u)} \text{sim}'(u,v)}
\]  

(9)

where \( \text{preference}_\text{score}_\text{amp} \) is the preference score predicted by the amplified similarity \( \text{sim}'(u,v) \).

5. Experiments and Evaluation

In order to evaluate the efficiency of our approach from different views, a series of experiments are designed to find out whether publishing time and tags of services will impact users’ preference or not. For this purpose, several group experiments are performed on a real-world dataset from PW from different aspects, including hit-ratio, hit-rank, and diversity of the recommendation with various parameters respectively. All the experiments are executed in MatLab 2010a with SQL Server 2005 and performed in a PC with Windows 7 Professional, Intel Core (TM) i7 cpu @2.80G (4 core), and memory of 4G RAM.

5.1 Dataset

Generally speaking, service user’s behavior is slightly different to other users due to their expertise and task-orientation (X. Zhang et al., 2013). The main objective of our approach is to recommend top-N services more accurately and practically. Our datasets are crawled from a public well-known Web service registry center named Programmableweb (PWeb), releasing “who bookmarked what” by its open API. Until Jan. 2015, PWeb contains 69,384 users, 15,000 services (including API services and mashup services), and 280,890 user-service pairs with bookmarks. In addition, the user profiles and service profile can also be released through its own APIs. With these APIs, we can get the profile of registered users and service profiles, and the later includes service reputation ratings, publishing time and service tags. The selected attributes of the dataset are listed in Table II. The first column is the watchlist id that can be used to identify the unique watchlist records. The second column is the registration name of users on PWeb. The third one is the service invocation address annotated with URL, in which the last word of URL is the unique service name on PWeb. The fourth column records the publishing time or updating time of services. The User ID and Service ID in the fifth and sixth column are used to identify different users and services with a unique code respectively. The column Rate means the service reputation rate, denoting the average rating of each service. The range of Rate is from zero to five. The last column is the service tag annotated by service publishers. Usually, same services have same reputation ratings, publishing or updating time and tags.

<table>
<thead>
<tr>
<th>ID</th>
<th>User Name</th>
<th>Service URL</th>
<th>Time</th>
<th>User ID</th>
<th>Service ID</th>
<th>Rate</th>
<th>Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>anand12345</td>
<td><a href="http://www.programmableweb.com/mashup/citystrides">http://www.programmableweb.com/mashup/citystrides</a></td>
<td>28 Jul 2013 09:02:59</td>
<td>1151</td>
<td>4455</td>
<td>5</td>
<td>fun, games, geocoding, health, running, social</td>
</tr>
</tbody>
</table>

Table II. Dataset Statistic Description
5.2 Processing

In the raw data crawled from PWeb, only a few users have bookmarked services frequently, and most of them collected only one or two services in his/her watchlist. In the worst case, some services might never be bookmarked by any user, and most of the users may never bookmark any service. Undoubtedly, the recommendation method will lead to the severe data sparse and cold-start problem if it is used in raw dataset. Therefore, the authors select 4568 users who have collected more than 20 services to construct the user set \( U \), and the services bookmarked by these users will be in the service set \( S \). To make sure that the experimental results are independent with the specified dataset, 1000 users are randomly selected from \( U \) using resampling technique. Each dataset consists of many entries, and each entry follows the formatting shown in TABLE II, which can be formalized as \([u, s, t_1, t_2, \ldots, t_h, ts] \). After grouping, each subset can be divided into two parts: the testing set covers 20% of the most recent published services in the watchlist of each user, and the rest is the training set. Table III lists the statistical description of the whole dataset and each subset. The Density column denotes the ratio between watchlist number and the dimensions of the user service matrix. In the data set, many services are annotated with the same service tags.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>User</th>
<th>Service</th>
<th>Tags</th>
<th>Density</th>
<th>Watchlist</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall</td>
<td>4568</td>
<td>6362</td>
<td>1869</td>
<td>0.0040</td>
<td>116,589</td>
</tr>
<tr>
<td>Subset1</td>
<td>1000</td>
<td>3138</td>
<td>1338</td>
<td>0.0082</td>
<td>25,795</td>
</tr>
<tr>
<td>Subset2</td>
<td>1000</td>
<td>3252</td>
<td>1373</td>
<td>0.0079</td>
<td>25,724</td>
</tr>
<tr>
<td>Subset3</td>
<td>1000</td>
<td>3113</td>
<td>1354</td>
<td>0.0081</td>
<td>25,529</td>
</tr>
<tr>
<td>Subset4</td>
<td>1000</td>
<td>3228</td>
<td>1352</td>
<td>0.0080</td>
<td>25,670</td>
</tr>
<tr>
<td>Subset5</td>
<td>1000</td>
<td>3372</td>
<td>1389</td>
<td>0.0077</td>
<td>25,838</td>
</tr>
<tr>
<td>Subset6</td>
<td>1000</td>
<td>3734</td>
<td>1395</td>
<td>0.0069</td>
<td>25,622</td>
</tr>
<tr>
<td>Subset7</td>
<td>1000</td>
<td>3627</td>
<td>1355</td>
<td>0.0070</td>
<td>25,329</td>
</tr>
<tr>
<td>Subset8</td>
<td>1000</td>
<td>3476</td>
<td>1353</td>
<td>0.0074</td>
<td>25,670</td>
</tr>
<tr>
<td>Subset9</td>
<td>1000</td>
<td>3551</td>
<td>1327</td>
<td>0.0072</td>
<td>25,548</td>
</tr>
<tr>
<td>Subset10</td>
<td>1000</td>
<td>3676</td>
<td>1364</td>
<td>0.0069</td>
<td>25,497</td>
</tr>
</tbody>
</table>

The authors perform all the experiments by generating a top-10 service recommendation list for each user. Since the size of user’s neighborhoods is an important factor for the quality of recommendation, it is varied from 10 to 80 by an interval of 1 in this paper. The baseline is computed by the log-based matrix, using the binary matrix. Specifically, cosine correlation is used in this section to compute all the similarities between different users.

5.3 Evaluation Metrics

The authors select hit-ratio, hit-rank and diversity as three metrics to evaluate our method, based on the average results of those ten subsets. The experiments are performed on each subset to produce a fixed number of services and evaluate its accuracy.

Hit-ratio (Deshpande & Karypis, 2004) 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.(10):

\[
\text{hit-ratio} = \frac{\sum_{u \in U} \text{number}_u \ _\text{of} \ _\text{hit}}{m \cdot |L|} \tag{10}
\]

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 recommend any appropriate service to users. However, the hit-ratio-based method treats all the hits equally, regardless of their occurrence frequency in the top-K list.

Hit-rank (Deshpande & Karypis, 2004) considers the position of each service in the hits of the recommendation list. The result can be used to effectively investigate the time impact in the watchlist. So this paper uses hit-rank as the supplement metric of hit-ratio. It is defined in Eq.(11):

\[
\text{hit-rank} = \frac{1}{m \cdot |L|} \sum_{u \in U} \sum_{i=1}^{h} \frac{1}{p_i} \tag{11}
\]

where \( h \) is the number of hits occurring at the positions \( p_1, p_2, \ldots, p_h \) within the recommendation list.

Diversity (Z.-K. Zhang et al., 2010) identifies the uniqueness of the recommendation lists provided by different users. As Eq. (12) states, it can be treated as the inter-user diversification. Different values of diversification represent various personalization in users’ recommendation lists, and can be used to exploit whether the diversification degree varies in each method or not.

\[
diversity = \frac{2}{m(m-1)} \sum_{u \in U} (1 - \frac{|L_i \cap L|}{|L|}) \tag{12}
\]

where \( L_i \) denotes the recommendation list of user \( u_i \), \( |L| \) is the length of the recommendation list, and \( m \) is the number of users. In our experiments, \( |L| = 10 \) and \( m = 1000 \).
5.4 Result and Analysis

In order to evaluate the impact of different parameters, five experiments are designed and discussed in detail. The first group is responsible to measure the dataset sensitivity. The second group intends to exploit the impact of time attenuation parameter $\alpha$, which is helpful to find out the best decline speeds of users’ interests. The third one concentrates on the impact of tags, while the fourth group tries to address the effect of weight balance parameter $\lambda$. Finally, the fifth group experiment compares the results of different methods by changing the size of different neighborhood. Table IV lists the abbreviations of our method and other ones used in this section, in which ‘B’ means the traditional user-based CF method with specific ratings, and ‘E’ presents the extended CF method with case amplification. Table IV. Abbreviation of methods

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>B-LOG</td>
<td>Basic log-based method</td>
</tr>
<tr>
<td>E-LOG</td>
<td>Enhanced log-based method</td>
</tr>
<tr>
<td>B-REP</td>
<td>Basic reputation-based method</td>
</tr>
<tr>
<td>E-REP</td>
<td>Enhanced reputation-based method</td>
</tr>
<tr>
<td>B-TIME</td>
<td>Basic time-based method</td>
</tr>
<tr>
<td>E-TIME</td>
<td>Enhanced time-based method</td>
</tr>
<tr>
<td>B-TAG</td>
<td>Basic tag-based method</td>
</tr>
<tr>
<td>E-TAG</td>
<td>Enhanced tag-based method</td>
</tr>
<tr>
<td>B-HYBRID</td>
<td>Basic hybrid-based method</td>
</tr>
<tr>
<td>E-HYBRID</td>
<td>Enhanced hybrid-based method</td>
</tr>
</tbody>
</table>

5.4.1 Dataset Sensitivity Analysis

To validate the data sensitivity in our approach, both the whole dataset and each subset are evaluated with the fixed parameter. Here, the time attenuation parameter is assigned as $\alpha = 0.001$, the weight balance parameter $\lambda = 0.1$, the neighborhood size $n_s = 60$ and the amplification power $\rho = 2.5$. The corresponding experimental results are shown in Table V. The result of each data group is split with ‘|’, i.e. the left is based on traditional CF and the right the case amplification result. It can be concluded that the case amplification method works better than the corresponding traditional CF approach. The bold hit-ratio data in Table V refers to the ones that are lower than the corresponding log-based method. Our proposed method can be evaluated in each subset and get a relatively stable performance. Only tag-based method and reputation method have slightly sensitive for datasets. With the enhanced CF methods, the highest improvement on hit-ratio is 0.0523 on Subset 10 and lowest improvement of hit-ratio is 0.067. The greatest improvement between B-HYBRID and B-LOG is 0.035 on subset 2. The E-HYBRID method can get an average of 1.53% and 1.85% improvement over B-LOG and E-LOG methods respectively.

Table V. Result Comparison with Different Datasets and Parameter

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Metrics</th>
<th>$n_s = 60$</th>
<th>$L = 10$</th>
<th>$\alpha = 0.001$</th>
<th>$\lambda = 0.1$</th>
<th>$\rho = 2.5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall</td>
<td>hit-ratio</td>
<td>0.0964</td>
<td>0.0986</td>
<td>0.1309</td>
<td>0.0991</td>
<td>0.1298</td>
</tr>
<tr>
<td></td>
<td>hit-rank diversity</td>
<td>0.0632</td>
<td>0.0616</td>
<td>0.0493</td>
<td>0.0621</td>
<td>0.0623</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.1246</td>
<td>0.0621</td>
<td>0.0577</td>
<td>0.0578</td>
<td>0.0601</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.1246</td>
<td>0.0965</td>
<td>0.1244</td>
<td>0.0959</td>
<td>0.1257</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.1081</td>
<td>0.1488</td>
<td>0.1348</td>
<td>0.1568</td>
<td>0.1177</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.0544</td>
<td>0.0707</td>
<td>0.0656</td>
<td>0.0739</td>
<td>0.0544</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.1044</td>
<td>0.0845</td>
<td>0.1045</td>
<td>0.0819</td>
<td>0.1031</td>
</tr>
<tr>
<td>Subset1</td>
<td>hit-ratio</td>
<td>0.1218</td>
<td>0.1399</td>
<td>0.1115</td>
<td>0.1450</td>
<td>0.1364</td>
</tr>
<tr>
<td></td>
<td>hit-rank diversity</td>
<td>0.0632</td>
<td>0.0697</td>
<td>0.0557</td>
<td>0.0685</td>
<td>0.0685</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.1246</td>
<td>0.0965</td>
<td>0.1244</td>
<td>0.0959</td>
<td>0.1257</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.1081</td>
<td>0.1488</td>
<td>0.1348</td>
<td>0.1568</td>
<td>0.1177</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.0544</td>
<td>0.0707</td>
<td>0.0656</td>
<td>0.0739</td>
<td>0.0544</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.1044</td>
<td>0.0845</td>
<td>0.1045</td>
<td>0.0819</td>
<td>0.1031</td>
</tr>
<tr>
<td>Subset2</td>
<td>hit-ratio</td>
<td>0.1226</td>
<td>0.1587</td>
<td>0.1311</td>
<td>0.1511</td>
<td>0.1319</td>
</tr>
<tr>
<td></td>
<td>hit-rank diversity</td>
<td>0.0696</td>
<td>0.0801</td>
<td>0.0648</td>
<td>0.0704</td>
<td>0.0695</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.0978</td>
<td>0.0855</td>
<td>0.1193</td>
<td>0.0937</td>
<td>0.0967</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.1201</td>
<td>0.1421</td>
<td>0.1282</td>
<td>0.1500</td>
<td>0.1351</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.0654</td>
<td>0.0745</td>
<td>0.0657</td>
<td>0.0728</td>
<td>0.0712</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.0975</td>
<td>0.0758</td>
<td>0.0946</td>
<td>0.0730</td>
<td>0.0983</td>
</tr>
<tr>
<td>Subset3</td>
<td>hit-ratio</td>
<td>0.1350</td>
<td>0.1593</td>
<td>0.1353</td>
<td>0.1681</td>
<td>0.1442</td>
</tr>
<tr>
<td></td>
<td>hit-rank diversity</td>
<td>0.0699</td>
<td>0.0794</td>
<td>0.0687</td>
<td>0.0787</td>
<td>0.0700</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.1293</td>
<td>0.0984</td>
<td>0.1300</td>
<td>0.0936</td>
<td>0.1294</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.1263</td>
<td>0.1394</td>
<td>0.1212</td>
<td>0.1460</td>
<td>0.1344</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.0605</td>
<td>0.0675</td>
<td>0.0611</td>
<td>0.0653</td>
<td>0.0616</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.1234</td>
<td>0.0825</td>
<td>0.1205</td>
<td>0.0836</td>
<td>0.1175</td>
</tr>
<tr>
<td>Subset4</td>
<td>hit-ratio</td>
<td>0.1226</td>
<td>0.1587</td>
<td>0.1311</td>
<td>0.1511</td>
<td>0.1319</td>
</tr>
<tr>
<td></td>
<td>hit-rank diversity</td>
<td>0.0696</td>
<td>0.0810</td>
<td>0.0648</td>
<td>0.0704</td>
<td>0.0695</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.0978</td>
<td>0.0855</td>
<td>0.1193</td>
<td>0.0938</td>
<td>0.0967</td>
</tr>
<tr>
<td>Subset5</td>
<td>hit-ratio</td>
<td>0.1201</td>
<td>0.1421</td>
<td>0.1282</td>
<td>0.1500</td>
<td>0.1351</td>
</tr>
<tr>
<td></td>
<td>hit-rank diversity</td>
<td>0.0654</td>
<td>0.0745</td>
<td>0.0658</td>
<td>0.0728</td>
<td>0.0712</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.0975</td>
<td>0.0773</td>
<td>0.0946</td>
<td>0.0744</td>
<td>0.0983</td>
</tr>
<tr>
<td>Subset6</td>
<td>hit-ratio</td>
<td>0.1263</td>
<td>0.1394</td>
<td>0.1212</td>
<td>0.1460</td>
<td>0.1344</td>
</tr>
<tr>
<td></td>
<td>hit-rank diversity</td>
<td>0.0605</td>
<td>0.0675</td>
<td>0.0611</td>
<td>0.0653</td>
<td>0.0616</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.1234</td>
<td>0.0825</td>
<td>0.1205</td>
<td>0.0836</td>
<td>0.1175</td>
</tr>
<tr>
<td>Subset7</td>
<td>hit-ratio</td>
<td>0.1201</td>
<td>0.1421</td>
<td>0.1282</td>
<td>0.1500</td>
<td>0.1351</td>
</tr>
<tr>
<td></td>
<td>hit-rank diversity</td>
<td>0.0654</td>
<td>0.0745</td>
<td>0.0658</td>
<td>0.0728</td>
<td>0.0712</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.0975</td>
<td>0.0773</td>
<td>0.0946</td>
<td>0.0744</td>
<td>0.0983</td>
</tr>
</tbody>
</table>
We also discuss the difference between the overall dataset and the subsets. The results show that with overall dataset, the values of hit-ratio and hit-rank is usually lower than that of the subsets. The main reason is that the data density of overall dataset is seriously lower than that of the subsets. Compared to the subsets, however, the overall dataset has the same variation trends. To avoid the data sparse problem, it is reasonable and practical to use the average result of those ten subsets to evaluate our method, rather than that of the whole dataset.

5.4.2 Impact of Time

In order to evaluate the effect of time, time-weight ratings are conducted in the time-weight model with different time attenuate parameter ($\alpha$). The test range of $\alpha$ is $\alpha \in [0.001, 0.002, 0.005, 0.015, 0.02, 0.025, 0.03, 0.035, 0.04, 0.05]$. The experimental results are shown in Figure 6. When time weight is introduced, the performance of hit-ratio increases significantly when $\alpha < 0.04$. The value of the time attenuation parameter is expected to be less than 0.01 ($\alpha < 0.01$). If the parameter is out of the specific ranges, the recommendation accuracy will sharply decrease, even lower than that of the log-based methods.

Additionally, the authors also compare the time weighted methods ($\alpha=0.001$) to the log-based methods with different neighborhood sizes of users. The comparison results are shown in Figure 7 and Figure 8. Statistically, performance improvement of performances is significant on hit-ratio with different neighborhood sizes. The biggest variance of hit-ratio between B-TIME and B-LOG is 0.0123 when neighbor size ns=80, and the biggest variance of hit-rank between B-TIME and B-LOG is 0.004 when neighbor size ns=80. The biggest variance of hit-ratio between E-TIME and E-LOG is 0.0136, and the biggest variance of hit-rank between B-TIME and B-LOG is 0.004 when neighbor size ns=80. Similarly, the biggest variance of diversity appears as 0.0004 when ns=10. The experimental result illustrates that time weighted methods enable recommendation with higher accuracy in most cases. With the increasing hit-ratio and hit-rank, the diversity of service recommendation is deceased. Therefore, we can conclude that the time-weighted model with the appropriate time attenuate parameter brings a stable improvement on hit-ratio and hit-rank. If the parameter is out of the range, it will lead to negative impacts on our dataset. In addition, the neighborhood size can also affect the proposed time-based method. It can facilitate users to find more accurate neighbor for an active user in the recommender systems.

<table>
<thead>
<tr>
<th>Subset9</th>
<th>hit-ratio</th>
<th>hit-rank</th>
<th>diversity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.1041</td>
<td>0.0546</td>
<td>0.1084</td>
</tr>
<tr>
<td></td>
<td>0.1272</td>
<td>0.0655</td>
<td>0.0901</td>
</tr>
<tr>
<td></td>
<td>0.1355</td>
<td>0.0564</td>
<td>0.1183</td>
</tr>
<tr>
<td></td>
<td>0.1366</td>
<td>0.0611</td>
<td>0.0865</td>
</tr>
<tr>
<td>Subset10</td>
<td>hit-ratio</td>
<td>hit-rank</td>
<td>diversity</td>
</tr>
<tr>
<td></td>
<td>0.1019</td>
<td>0.0551</td>
<td>0.0979</td>
</tr>
<tr>
<td></td>
<td>0.1269</td>
<td>0.0651</td>
<td>0.0868</td>
</tr>
<tr>
<td></td>
<td>0.1121</td>
<td>0.0577</td>
<td>0.1036</td>
</tr>
<tr>
<td></td>
<td>0.1342</td>
<td>0.0647</td>
<td>0.0807</td>
</tr>
<tr>
<td>Avg(1-10)</td>
<td>hit-ratio</td>
<td>hit-rank</td>
<td>diversity</td>
</tr>
<tr>
<td></td>
<td>0.1183</td>
<td>0.0628</td>
<td>0.1079</td>
</tr>
<tr>
<td></td>
<td>0.1439</td>
<td>0.0728</td>
<td>0.0863</td>
</tr>
<tr>
<td></td>
<td>0.1246</td>
<td>0.0626</td>
<td>0.1129</td>
</tr>
<tr>
<td></td>
<td>0.1488</td>
<td>0.0699</td>
<td>0.0857</td>
</tr>
</tbody>
</table>

Figure 6 The Plot of B-TIME, E-TIME, B-LOG and E-Log with different Time Attenuate Parameter ($\alpha$)
5.4.3 Impact of Tags

This part mainly focuses on tags’ impact. The tag-weight ratings are used to compare the proposed tag-weight model with the log-based model by the experiments on 10 subsets with neighborhood sizes from 10 to 80. As the results in Figure 8 show, for all the neighborhoods, the average hit-ratio based on our tag-weight model is better than that of the log-based model. It indicates that the recommendation accuracy can be improved by simply adding tag information into the service recommendation model. The impact of tags is evaluated by comparing the performance of B-TAG vs. B-LOG and that of E-TAG vs. E-LOG. The biggest variance of hit-ratio between B-TAG vs B-LOG is 0.0261 and the maximum variance of hit-rank between them is 0.0018. It implies that neighborhood size will impact the stability of the tag-based method. Moreover, the tag-based method has a better performance on diversity. It means that it’s possible to introduce some novel recommendations with better performance than that of the log-based method. We can find that the service tags can not only link the users to services, but present the interests of users. The more frequently a tag occurs in a user’s watchlist, the higher interests of he/she expresses on the corresponding services annotated with this tag.

5.4.4 Impact of Weight

Considering the impact of time and tag independently, or combing the time-based method and the tag-based method as a hybrid recommendation method, which is of effective performance? Before the corresponding experiment, subset 1 is taken as an example to plot the distribution of 1000 users’ non-zero time ratings and tag ratings, following the sequence in the rating matrix. The corresponding ratings distribution is shown in Figure 9.

Figure 7. The Plot of E-TIME, B-TIME, E-LOG and B-LOG with Different Neighborhood Size

Figure 8. The Plot of E-TAG, B-TAG, E-LOG and B-LOG with Different Neighborhood Size

Figure 9. The Distribution of Time Ratings and Tag Ratings
From the plot of rating distribution, it’s easy to find that the values of the time rating are usually higher than that of the tag rating when the time attenuation parameter $\alpha=0.001$. Most of tag ratings are valued from 0 to 0.3, and the time ratings are varied from 0.9 to 1. In addition, the plot shows that the number of non-zero tag rating is less than time rating, which implies that there exists service tag missing in our dataset.

After analyzing the rating distribution, a set of hybrid-based method experiments are conducted with different weight balance parameter $\lambda$ varying from 0 to 1, a stepwise of 0.1, and neighborhood size $ns=60$. In the hybrid method, the time attenuation parameter of a time-weight model is set as $\alpha=0.001$. Figure 10 illustrates the hit-ratio, hit-rank and diversity of recommendations with different weight balance parameter $\lambda$ respectively. The results show that different values of parameter $\lambda$ will influence the performance in service recommendation. The performance of the hybrid method is better than the ones containing either time factor or tag. As for the metrics changes from Figure 10 and Figure 11, it can be found that the best value of hit-ratio, hit-rank and diversity can be produced when $\lambda = 0.1$ in B-HYBRID and E-HYBRID. The value of weight $\lambda = 0.1$ means the proportion of the time ratings is 10% and the tag ratings is 90%. However, the hit-rank and diversity of hybrid-based method is even lower than the log-based method when $\lambda > 0.2$. Consequently, it’s needed to merge these two kinds of ratings with right balance parameter. Otherwise it will lead to negative effects.

![Figure 10. The Plot of Result Comparison on B-LOG, B-TIME, B-TAG, B-HYBRID with Different Neighborhood Size](image)

![Figure 11. The Plot of Result Comparison on E-LOG, E-TIME, E-TAG, E-HYBRID with Different Neighborhood Size](image)

### 5.5 Discussion

The computational complexity of our method considers user similarity and preference prediction, and the cost of either one is $O(mn)$, where $m$ denotes the number of users and $n$ is the number of services. In our recommender system, similarities between users are calculated offline and updated periodically. In order to shorten the calculation time of the similarity between users, some data are processed and stored in 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 methods for service recommendations, our proposed method makes a better performance. However, some issues still need to be
addressed, such as: (1) suffering from the cold start problem; (2) handling with data sparse problem; and (3) lacking of scalability with the increment of users.

6. CONCLUSIONS AND FUTURE WORK

Recommender systems are very popular on the Web to deliver the right products or services at the right time and in the right place to the right customers (Lamb, Randles, & Al-Jumeily, 2013). (Sunikka & Bragge, 2012), (Oard & Kim, 1998), and have been widely used in applied in many cases, such as commercial products, scientific workflows, quality methods and instruments (Manouselis & Costopoulou, 2008). This paper exploits how to merge time and tag information to construct a more effective recommendation model. One of the key issues is to discuss whether the tag and time information can be used to capture the users’ preferences or not. To avoid the bottleneck of rating data collection, implicit feedback techniques are introduced by inferring valuable information that is accessible for both users and the recommendation systems.

PWeb provides the service bookmarking information of a large number of registered users. We leverage the valuable information in PWeb to provide a personalized service recommendation method to help users find interested services. More specifically, a preference drift function is defined to capture the drifting of user’s interest, and the tag frequency is used to reflect user’s long-term interests. Based on time and tag information, the authors provide an effective mechanism to transform implicit feedbacks to explicit ratings for CF-based service recommendation, especially for the case without explicit ratings and QoS information. Moreover, the case amplification method is introduced to improve the recommendation accuracy of services. Finally, a series of experiments was designed to evaluate our method with a real-world dataset from PWeb. The results illustrate that our approach can provide better performance than traditional log-based methods.

This paper addresses the start point of merging time and tag information as a hybrid recommendation method for services. However, there are still some issues to be explored in the near future, including (1) to evaluate the proposed hybrid method with other dataset to explore the correlation between users’ preference and implicit data; (2) to deploy our recommender system in a parallel distributed platform or cloud platform; and (3) to merge other recommendation strategies, such as matrix factorization (Koren, Bell, & Volinsky, 2009), to relieve the problems caused by cold start and data sparse.

7. ACKNOWLEDGMENT

This paper was supported by the National Basic Research and Development Program of China under Grant No.2014CB340404, the National Natural Science Foundation of China under Grant No.61373037 and No. 61202031, the Central Grant Funded Cloud Computing Demonstration Project of China undertaken by Kingdee Software (China) co., ltd., and the Science and Technology Innovation Program of Hubei Province under Grant No.2013AAA02.

8. REFERENCES


Author

Xiuwei Zhang is an engineer of No.94005 Troops of PLA. He received his Master degree in Wuhan University of Technology in 2009 and Ph.D. degree from Wuhan University in 2014. His current research interests focus on service computing, big data and personalization.

Chong Wang (corresponding author), IEEE Member, is a Lecturer of State Key Lab of Software Engineering, Computer School, Wuhan University, China. Her research area covers business process management and service computing.

Jianxiao Liu is a lecturer in the College of Informatics at Huazhong Agricultural University, China. His research interests include service engineering, bioinformatics, etc.

Tian Gang, IEEE Student Member, is a PhD candidate of State Key Lab of Software Engineering, Computer School, Wuhan University, China. His main research area is service recommendation.

Keqing He, IEEE Member, is a Full Professor of State Key Lab of Software Engineering, Computer School, Wuhan University, China. His research area is service-oriented software engineering.