Time-Aware Service Recommendation for Mashup Creation in an Evolving Service Ecosystem

Yang Zhong
Tsinghua National Laboratory for Information Science and Technology
Department of Automation
Tsinghua University
Beijing 10084, China
zhongy12@mails.tsinghua.edu.cn

Yushun Fan*
Tsinghua National Laboratory for Information Science and Technology
Department of Automation
Tsinghua University
Beijing 10084, China
fanyus@mail.tsinghua.edu.cn

Keman Huang
Tsinghua National Laboratory for Information Science and Technology
Department of Automation
Tsinghua University
Beijing 10084, China
hkm09@mails.tsinghua.edu.cn

Wei Tan
IBM Thomas J. Watson Research Center
Yorktown Heights, NY 10598, USA
wtan@us.ibm.com

Jia Zhang
Carnegie Mellon University
Silicon Valley
jia.zhang@sv.cmu.edu

***Abstract*—Web service recommendation has become a critical problem as services become increasingly prevalent on the Internet. Some existing methods focus on content matching techniques such as keyword search and semantic matching while others are based on Quality of Service (QoS) prediction. However, services and their mashups are evolving over time with publishing, perishing and changing of interfaces. Therefore, a practical service recommendation approach should take into account the evolution of a service ecosystem. In this paper, we present a method to extract service evolution patterns by exploiting Latent Dirichlet Allocation (LDA) and time series prediction. A time-aware service recommendation framework for mashup creation is presented combining service evolution, collaborative filtering and content matching. Experiments on real-world ProgrammableWeb data set show that our approach leads to a higher precision than traditional collaborative filtering and content matching methods.**

Keywords-service recommendation; LDA; service ecosystem; time-aware; mashup creation

I. INTRODUCTION

With the wide adoption of Service-Oriented Architecture and Cloud Computing, the number of web services (nowadays usually in the form of web APIs) published on the Internet has been rapidly growing [1]. Mashup, a web application created through service composition, has become a popular technique to reuse existing services and shorten development cycle [2]. As a consequence, several web service ecosystems (such as Bell lab's ProgrammableWeb[1] and myExperiment[2] by the universities of Southampton, Manchester and Oxford) have emerged in the recent years, continuously accumulating web services and their mashups [3, 4]. In spite of such encouraging facts, creating a mashup may take an inexperienced developer a great amount of time to search in the sea of available services in the repositories for service components. Therefore, service recommendation and discovery approach is essential to facilitate developers in locating desired services.

Most existing service recommendation approaches are content matching methods, mainly focusing on keyword search [6, 7] and semantic-based search [8]. However, keyword search is usually inefficient while semantic-based approach is expensive to construct in practice. A probabilistic approach for service discovery based on Latent Dirichlet Allocation (LDA) is proposed in [9] to address the challenge. It extracts features from WSDL documents and employs the LDA model to characterize the latent topics between services and user queries. In contrast to these service recommendation methods considering functional requirements, other methods focus on helping developers find services meeting expected Quality of Service (QoS). Non-functional properties of services under consideration include reliability, availability, and response time. In addition to formal QoS measurement, user-centric collaborative filtering [10, 11] mechanism has also been used to support service recommendation. For example, a hybrid approach that combines collaborative filtering and content matching is proposed in [12] to improve the performance of service recommendation. Recently, some researchers also apply social network analysis to service recommendation [13, 14, 23] and combine service ranking with service clustering [15].

One phenomenon that has usually been ignored in service discovery is that, services and their mashups evolve over time. Few existing methods consider or exploit temporal information for service recommendation. Our previous work [5, 16] proposed a method based on link prediction in a time-varying service network. Nevertheless, it is purely based on past service usage and does not take into account the functional requirements of individual mashups. Thus, its recommendation precision is not satisfactory.

In this paper, we address such limitations by conducting joint analysis on temporal information, topology and content in an evolving service ecosystem. Two assumptions are put forth. First, services with similar functions form a particular service *domain* that can be interpreted as a specific topic. Second, developers tend to adopt popular services in popular domains at the moment of request. Under these assumptions, service usage over time is modeled as a probabilistic generative model. Our key idea is to represent each sliced time interval as a "bag of services" and introduce the concept

*Communication Author

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

[2] http://www.myexperiment.org

of topic modeling to describe the relations between timestamps, topics and services. Through parameter estimation, our model is able to predict service usage at subsequent intervals. In addition, combining with past usage and text description of services and mashups, our model offers a comprehensive service recommendation technique taking into consideration of functional user requirements as well as peer experience. The main contributions of this paper are summarized as follows:

1) We propose a novel service activity prediction method based on Latent Dirichlet Allocation (LDA), which is capable of extracting a time sequence of topic activity and service-topic correlation matrix from service usage history. Applying our time series prediction method, we can forecast topic evolution and predict service activity in the near future.

2) Combining service activity prediction with mashup-description-based collaborative filtering and service-description-based content matching, we propose a time-aware service recommendation framework for mashup creation in an evolving service ecosystem.

3) Comprehensive experiments on a real-world data set from ProgrammableWeb.com show that our approach yields better precision by taking temporal information into account.

The rest of this paper is organized as follows. Section II introduces a model to describe an evolving service ecosystem and formulates the service recommendation problem. Section III describes model training methods. Section IV presents our time-aware service recommendation framework. Section V reports the experimental results. Section VI summarizes the related work and Section VII concludes the paper.

II. Problem Definition

We propose to model an evolving service ecosystem along three dimensions: *topology*, *content* and *temporal information*.

Definition 1: Topology. The topology of a service ecosystem is modeled with an undirected graph $G=(M \cup S, E)$ in which: $M=\{m_1, m_2, ..., m_n\}$ is the set of mashups and $S=\{s_1, s_2, ..., s_k\}$ is the set of services; $E \subseteq M \times S$ is the historical composition relation between mashups and services, i.e., if a mashup invokes a service, there exists a relation between them.

Definition 2: Content. Every mashup $m \in M$ comprises a collection of words $MW(m)=\{w_1, w_2, ..., w_{n_m}\}$ to describe its functional requirements. Similarly, each service $s \in S$ is associated with a collection of words $SW(s)=\{w_1, w_2, ..., w_{n_s}\}$ to describe its functions.

Definition 3: Temporal information. Given a sequence of timestamps with a particular time granularity (e.g., day, week, month) $TG=\{1,2,...,t\}$, the service usage history in an evolving service ecosystem is described in a set of ordered pairs by $H=\{(s,m,t) | s \in S, m \in M, t \in TG\}$ where (s,m,t) indicates that service s is invoked by mashup m at timestamp t.

A service ecosystem is dynamic in nature, i.e., with interactions between mashups and services evolving over time. Compared with the static topological view of G, H takes such evolution into account. Based on the 3-dimensional service ecosystem definition, we formulate the problem of service recommendation for mashup creation as follows:

Definition 4: Time-aware Service Recommendation for Mashup Creation. Given G and H regarding passed time points $\{1,2,...,t\}$, for a new mashup m required at time $(t+1)$ with user requirements in a collection of words $Q=\{q_1, q_2, ..., q_p\}$, a ranked list of services denoted by $R(m)$ will be recommended. A service with higher rank in $R(m)$ has a higher probability to be adopted by m.

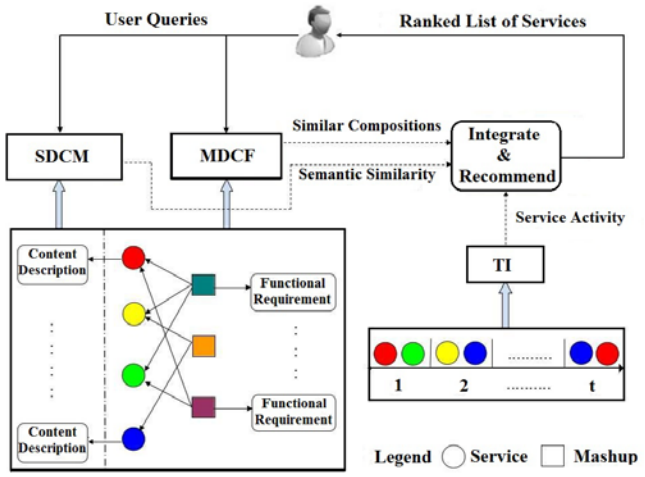

Figure 1. Time-aware service recommendation framework.

The mashup creation problem is thus turned into finding the ranked recommendation list $R(m)$. We propose a time-aware service recommendation approach that systematically considers temporal information, topology and content of services in an evolving service ecosystem.

As shown in Fig. 1, our method consists of three components: *temporal information* (TI) extraction, *mashup-description-based collaborative filtering* (MDCF) and *service-description-based content matching* (SDCM). The three components suggest service recommendation from different perspectives. TI exploits service usage history to predict service activity in the near future. It offers popularity scores of services in recent time frame regardless of functional requirements of individual mashups. Complementary to TI, MDCF and SDCM score the relevance of services against the functional requirements of a user required mashup. MDCF recommends services based on historical mashups with similar functional requirements; SDCM calculates semantic similarity between the functional requirements of the mashup and the content description of

services. All scores will be integrated to generate the recommended list of services for the required mashup. We will discuss in detail the three components in the next section and the recommendation framework in section IV.

III. MODEL TRAINING

In this section, we will introduce the construction of the three components in our approach: TI, MDCF and SDCM.

A. Temporal Information Extraction

One fundamental assumption is that users tend to consume popular services in prevalent service domains at a given time. We apply a similar idea to topic modeling and analyze the service usage history in a probabilistic manner. Specifically, service domains are viewed as latent topics thus the concepts of LDA [17] can be employed to model the generative process of service usage over time. As a preliminary step, we retrieve a collection of services that were consumed at timestamp t from H for every $t \in TG$ and denote it by $ST(t) = \{s_1, s_2, ..., s_{n_t}\}$. The generation process of $ST(t)$ can be modeled as follows:

1) For each topic $k = 1, 2, ..., T$:
   Draw $\phi_k \sim Dirichlet(\beta)$
2) For each timestamp $t \in TG$:
   a) Draw $\theta_t \sim Dirichlet(\alpha)$
   b) For each $s \in ST(t)$
      i. Draw a topic $z \sim Multinomial(\theta_t)$
      ii. Draw a service $s \sim Mutinomial(\phi_z)$

where T is the number of topics and ϕ_k is the multinomial distribution over services specific to topic k and θ_t is the multinomial distribution over topics specific to timestamp t. α and β are the prior parameters of Dirichlet distribution for θ_t and ϕ_k, respectively.

We then apply the Gibbs sampling [18] to infer the desired parameters θ_t and ϕ_k. θ_t is a $1 \times T$ vector $(\theta_t^1, \theta_t^2, ..., \theta_t^T)$ for every $t \in TG$ which can be interpreted as topics activity at time t. For every topic k, $\theta_t^k (t \in TG)$ constitute a time series. The activity of topic k at time $t+1$ can be forecasted by applying a time series prediction method to $\theta_t^k (t \in TG)$. Several methods exist to solve the time series prediction problem, such as linear weighted moving average [19] and auto regression [20]. In this paper, we choose to adopt the linear weighted moving average because of its efficiency and simplicity.

Given a time window length l, for every topic $k \in \{1, 2, ..., T\}$ we can predict the activity of topic k at time $t+1$ through the following equation:

$$\theta_{t+1}^k = \sum_{i=1}^{l} \lambda_i \theta_{t+1-i}^k \quad (1)$$

where $\lambda_i (i = 1, 2, ..., l)$ are positive real numbers subject to the constraint $\sum_{i=1}^{l} \lambda_i = 1$. By tuning λ_i, we can adjust the impact of topic activities in different past time intervals on that of future. A reasonable policy is to place more weight on more recent time intervals.

Similarly, we have ϕ_k as a $1 \times |S|$ vector $(\phi_k^1, \phi_k^2, ..., \phi_k^{|S|})$ for every $k \in \{1, 2, ..., T\}$. $|S|$ is the cardinality of the set of services S. ϕ_k represents correlation strength of services with topic k. $\phi_k (k = 1, 2, ...T)$ constitute a $T \times |S|$ service-topic correlation matrix. With latent topics as a bridge between timestamps and services, the activity of service $s \in S$ indexed by n in the matrix at time $t+1$ can be calculated as follows:

$$p_{TI}(s|t+1) = \sum_{k=1}^{T} \theta_{t+1}^k \phi_k^n \quad (2)$$

$p_{TI}(s|t+1)$ carries service usage evolution and will be incorporated into our time-aware service recommendation framework.

B. Mashup-description-based Collaborative Filtering

Collaborative filtering is one of the state-of-the-art methods in the recommendation community [10]. Its basic idea is that similar users are likely to consume similar items. Previous works [10, 12] focus on application of collaborative filtering to QoS-aware service recommendation. However, it is often hard to obtain QoS data in reality. In contrast to existing work, we collect objective description data about mashups and services. To apply the idea of collaborative filtering to our setting, we propose to recommend services for a new mashup based on historical service compositions of similar mashups. For example, if a new mashup m is similar with a historical mashup m' and m' constitutes services s_1 and s_2, then we believe m is also likely to consume s_1 and s_2. Since we leverage mashup description to calculate the similarities between mashups, this component is named as mashup-description-based collaborative filtering (MDCF).

We apply the LDA model to calculate the similarity among mashups using mashup descriptions. Specifically, we input the set of mashups M and the associated sets of words $MW(m)$ for all $m \in M$ into the LDA model. Afterwards we run Gibbs Sampling to get posterior distribution of mashup over topics $p(k|m)$ and topic over words $p(w|k)$. When a new mashup m comes up with user queries $Q = \{q_1, q_2, ..., q_p\}$, its similarity with a past mashup $m_j \in M$ can be calculated using the following equation:

$$sim(m, m_j) = \sum_{q \in Q} \sum_{k=1}^{T} p(q|k)\, p(k|m_j) \quad (3)$$

After obtaining similarities between mashups, the probability of a service $s \in S$ to be used by m at time $t+1$ can be evaluated as follows:

$$p_{CF}\left(s|m\right) = \sum_{m_j \in U(K,m)} sim\left(m, m_j\right) y(m_j, s) \quad (4)$$

where $U\left(K,m\right)$ contains the Top K similar mashups with m ; $y\left(m_j, s\right)=1$ if $\left(m_j, s\right) \in E$ and $y\left(m_j, s\right)=0$ if $\left(m_j, s\right) \notin E$. $p_{CF}\left(s|m\right)$ will become an important factor in the subsequent recommendation framework.

C. Service-description-based Content Matching

Content matching recommends services for a new mashup through semantic matching between user queries of the new mashup and content descriptions of services. Since we employ service description to calculate the semantic similarities between mashups and services, this component is named as service-description-based content matching (SDCM).

Similar with [9], we apply the LDA model to calculate the similarities. The input of the component is the set of services S and associated collections of words $SW\left(s\right)$ for all $s \in S$. We then run Gibbs Sampling to get probability distribution of service over topics $p\left(k|s\right)$ and topic over words $p\left(w|k\right)$. When a new mashup m comes up with user queries as a collection of words $Q=\left\{q_1, q_2, ..., q_p\right\}$, SDCM calculates the semantic similarity between m and a service s as follows:

$$p_{CM}\left(s|m\right) = \sum_{q \in Q} \sum_{k=1}^{T} p\left(q|k\right) p\left(k|s\right) \quad (5)$$

Note that our SDCM is slightly different from [9], in that summation over queries is replaced by multiplication. Our hypothesis is that over a large candidate service pool, extracting feature union shall yield higher performance comparing to feature intersection. Experimental results on our data set show that the two methods are comparable in performance. So we can view SDCM as a baseline method from existing content matching approaches in performance comparison of our experiment. $p_{CM}\left(s|m\right)$ will be integrated into our service recommendation framework in the following section.

IV. Time-Aware Service Recommendation Framework

Based on previously introduced TI, MDCF and SDCM, in this section, we show how to integrate them to support time-aware service recommendation.

When a new mashup m is requested at time $t+1$ with user queries as a collection of words $Q=\left\{q_1, q_2, ..., q_p\right\}$, we first use MDCF to calculate $p_{CF}\left(s|m\right)$ through equation (4). Then we use SDCM to obtain semantic similarity $p_{CM}\left(s|m\right)$ through equation (5). Together with service activity $p_{TI}\left(s|t+1\right)$, we can obtain the probability of a service s to be consumed by m as follows:

$$p\left(s|m\right) = p_{CF}\left(s|m\right) p_{CM}\left(s|m\right) p_{TI}\left(s|t+1\right) \quad (6)$$

Finally, we offer a ranked list of services $R\left(m\right)$ for m in a descending order w.r.t. $p\left(s|m\right)$.

The detailed recommendation algorithm is as follows:

Algorithm 1: time-aware service recommendation

Input:

1) $G=\left(M \cup S, E\right)$: The topology model
2) $MW\left(m_i\right)$: The functional requirements of $m_i \in M$
3) $SW\left(s_j\right)$: The content description of $s_j \in S$
4) H : The service usage history
5) $ST\left(t\right)$: services consumed at timestamp t
6) TG : The sequence of timestamps
7) T : The number of latent topics in LDA model
8) α and β: The prior parameters in LDA model
9) l : The window length
10) $\lambda_i\left(i=1,2,...,l\right)$: Weights on different time intervals
11) K : Top-K similar mashups in MDCF
12) $Q=\left\{q_1, q_2, ..., q_p\right\}$: User queries for new mashup m

Output:

1) $R\left(m\right)$: recommended list of services for m

Procedure:

01. $\left\{\theta_t, \phi_k\right\} = GibbsSampling\left(TG, \left\{ST\left(t\right)\right\}\right)$
02. Predict topic activity at time $t+1$ by equation (1)
03. Calculate service activity at time $t+1$ by equation (2)
04. $\left\{p\left(k|m\right), p\left(w|k\right)\right\} = GibbsSampling\left(M, \left\{MW\left(m\right)\right\}\right)$
05. $\left\{p\left(k|s\right), p\left(w|k\right)\right\} = GibbsSampling\left(S, \left\{SW\left(s\right)\right\}\right)$
06. Calculate similarities among mashups by equation (3)
07. Get $p_{CF}\left(s|m\right)$ by equation (4)
08. Get $p_{CM}\left(s|m\right)$ by equation (5)
09. Calculate integrated result $p\left(s|m\right)$ by equation (6)
10. Return $R\left(m\right)$ in descending order of $p\left(s|m\right)$

Lines 01~03 are the implementation of TI that predicts service activity in the near future; Lines 04, 06 and 07 describe the construction of MDCF; Lines 05 and 08 complete the calculation of SDCM; Lines 09~10 integrate the three components and generate a recommended list of services for the new mashup.

We now discuss the computational complexity of Algorithm 1. We can divide it into two stages: offline stage (lines 01~05) and online stage (lines 06~10). The offline part

only needs to be conducted once at the start of each time interval while the online part performs when receiving a query. We assume that the online query has a number of P word tokens and N is the number of iterations in Gibbs sampling.

The complexity of Gibbs sampling to estimate the parameters in TI (Line 01) is bounded by $O(N \cdot |H| \cdot T)$. From equation (1), we know the complexity of topic activity prediction (Line 02) is $O(l \cdot T)$. Similarly, the complexity of service activity prediction (Line 03) is $O(|S| \cdot T)$. So the overall computational complexity of TI is $O((N \cdot |H| + |S| + l) \cdot T)$.

MDCF involves offline (Line 04) and online (Line 06~07) computation. The complexity of LDA in Line 04 is $O(N \cdot V \cdot T)$ where V is the total number of word tokens appeared in the functional requirements of all existing mashups. For the online part, the time complexity of similarities computation among mashups is $O(P \cdot |M| \cdot T)$ according to equation (3). Line 07 involves sorting of mashups with a complexity of $O(|M| \log |M| + K|S|)$.

Similarly, for the offline part of SDCM (Line 05), the computational complexity is $O(N \cdot W \cdot T)$ where W is the total number of word tokens appeared in the content description of all services. By equation (5), we know the computational complexity of Line 08 is $O(P \cdot |S| \cdot T)$.

The complexity of the integration (Line 09) is $O(|S|)$ and the ranking of services (Line 10) is $O(|S| \log |S|)$.

In practice, usually $N \cdot W \gg |S| \gg l$, $K \gg \log |S|$ and $PT \gg \log |M|$, and therefore the overall complexity of offline computing is $O(N \cdot T(|H| + V + W))$ and the online part is $O(P \cdot T(|M| + |S|) + K \cdot |S|)$.

V. EXPERIMENTS

In this section, we explain how we have applied our time-aware service recommendation approach for mashup creation to a real data set, crawled from ProgrammableWeb.com, to evaluate its performance. A collection of experiments were designed to compare our approach with state-of-the-art methods.

A. Data Set Preparation

To the best of our knowledge, ProgrammableWeb is by far the largest online repository of web services and their mashups. Through RESTful APIs, we crawled the metadata of services and mashups from the web with timestamps ranging from September 2005 to August 2012. Each service contains metadata such as name, summary and description. Every mashup contains the information such as name, creation date, description and the list of services used. Table I summarizes the basic properties of our data set.

TABLE I. BASIC PROPERTIES OF PROGRAMMABLEWEB DATA SET

Number of services	7,077
Number of mashups	6,594
Size of vocabulary	13,648

B. Preprocessing

For each service in our data set, there is a description consisting of a bag of words that describe the functionality of the service. Before we can view the description as the underlying service's associated collection of words in our model, however, several nature language preprocessing tasks have to be done. In this work, we applied the four-step data preprocessing method similar in [21] to extract meaningful words from the original description:

1) **Original Words Generating**. First of all, we extract all original words contained in the descriptions.
2) **Pruning**. Secondly, we filter words that are not meaningful for recognizing the service. Some examples include: some articles such as *a, an, the*; some prepositions such as *in, on, with, by, for, at, about, from, etc*; some adverbs such as *where, when, quite,* etc; and general words such as *api, service, mashup*, etc.
3) **Suffix Striping**. In the third step, we perform suffix stripping to obtain stem words. For example, *map*, *mapping*, *maps*, and *mappings* will be replaced with the same stem *map.*
4) **Spell Correcting**. In the last step, we use spell correct tool to adjust misspelled word. For example, *websit* will be corrected as *website*.

The topology model $G = (M \cup S, E)$ can be directly derived from the list of services of a mashup from our data set. Given a time granularity, we can obtain a sequence of timestamps TG and service usage history H according to the creation dates and the list of services of a mashup. Moreover, we use the processed description data of a service as its associated collection of words in our model. Similar actions are performed on mashups.

In our experiment, we adopted a time granularity of one month. To examine the performance of our approach, we divided the data set into training and testing sets, with a moving cutoff timestamp. We use the data before it as the training set and data with exactly that timestamp as testing set. We move the cutoff timestamp from September 2011 to August 2012, obtaining twelve corresponding training data and testing data sets. In other words, we tested our method in one year period month by month. For each mashup appeared in the testing month, we use its description as a user query and its services as the ground truth.

C. Evaluation Metric

The evaluation metric that we used in this experiment is Mean Average Precision (MAP) [22], which is a widely used measure in recommendation system:

$$MAP = \frac{1}{|M_c|} \sum_{m \in M_c} \frac{1}{|S_m|} \sum_{s \in S_m} \frac{n(s)}{r(s)} \qquad (7)$$

where M_c denotes the set of mashups in the testing month and S_m represents the set of services consumed by m for every $m \in M_c$; for each $s \in S_m$, $r(s)$ denotes the ranking position of s in $R(m)$ and $n(s)$ refers to the number of services in S_m that rank higher than or equal to s in $R(m)$.

MAP is a real number between 0 and 1. The higher MAP indicates a better accuracy of the recommendation method. By moving the cutoff timestamp, we can calculate the MAP for each testing month and use the average value of MAP for all testing months as the evaluation metric to compare different methods.

D. Comparison Methods

To study the performance of our time-aware service recommendation approach, we compare our method with others generated from a combination of three components. MDCF alone can return a ranked list of services in a descending order of $p_{CF}(s|m)$ and is exactly the well-known collaborative filtering adapted to our setting. SDCM alone generates the list of services based on $p_{CM}(s|m)$ and can be viewed as the representative of content matching approaches. MDCF*TI recommends services in a descending order of $p_{CF}(s|m)\,p_{TI}(s|t+1)$. Similarly, we can define SDCM*TI, MDCF*SDCM and MDCF*SDCM*TI. Note that our time-aware service recommendation approach can be viewed as MDCF*SDCM*TI.

To make our comparison more complete, we have introduced an alternative method to calculate popularity scores of services (named as FR). FR simply gives the popularity scores of services by normalizing service usage frequency. We can also combine FR with MDCF, SDCM and MDCF*SDCM, respectively. Taking MDCF as an example, the newly formed method MDCF*FR calculates the probability of a service $s \in S$ to be used by a new mashup m at time $t+1$ with user queries $Q = \{q_1, q_2, ..., q_n\}$ as follows:

$$p_{CF*FR}(s|m) = p_{CF}(s|m)\, f(s) \tag{8}$$

where $f(s)$ is equal to the service usage times for service $s \in S$ divided by the sum of service usage times for all services. Finally, MDCF*FR returns a ranked list of services for m in a descending order of $p_{CF*FR}(s|m)$. Similarly, we can define SDCM*FR and MDCF*SDCM*FR by analogy with MDCF*FR and their descriptions are omitted due to space limitation. In summary, we consider nine methods: MDCF, MDCF*TI, MDCF*FR, SDCM, SDCM*TI, SDCM*FR, MDCF*SDCM, MDCF*SDCM*TI and MDCF*SDCM*FR.

E. Experimental Results

Next, we set up the parameters used in this experiment. As to parameters in the LDA model, we set $T = 40$, $\alpha = 1.25$ and $\beta = 0.01$ for all components. For MDCF, we set $K = 150$; with respect to TI extraction, we set $l = 2$, $\lambda_1 = 0.9$ and $\lambda_2 = 0.1$.

Fig. 2 reports the MAP of MDCF, MDCF*TI and MDCF*FR in the twelve testing months. MDCF employs functional requirements of mashups and mashup-service past usage to predict the relevance scores of services and its performance is moderate. With the help of popularity scores offered by TI, MDCF*TI gets the highest MAP among the three methods in nine of the twelve testing months. On the other hand, MDCF*FR is poorer in overall performance than MDCF.

Fig. 3 depicts the MAP of SDCM, SDCM*TI and SDCM*FR in the twelve testing months. SDCM only employs content description of services and its performance is unsatisfactory. SDCM*TI gets a significant improvement than SDCM with TI included and ranks the highest in all testing months except one. SDCM*FR wins in only one month.

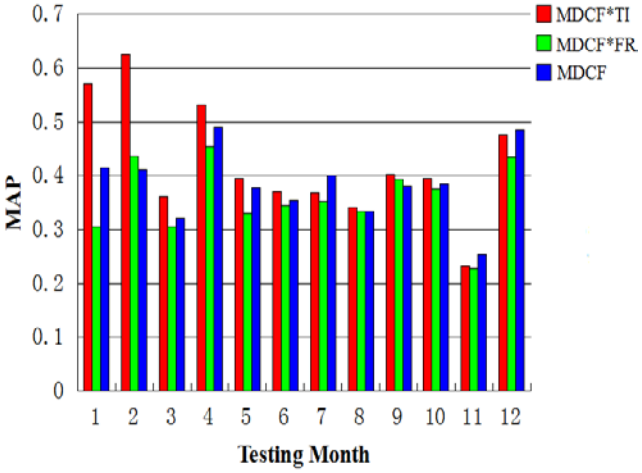

Figure 2. The MAP for MDCF*TI, MDCF*FR and MDCF in 12 testing months. TI helps boost the performance of MDCF.

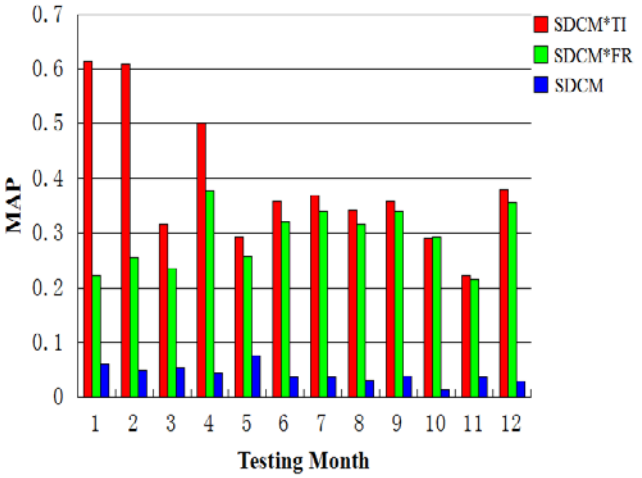

Figure 3. The MAP for SDCM*TI, SDCM*FR and SDCM in 12 testing months. TI helps boost the performance of SDCM.

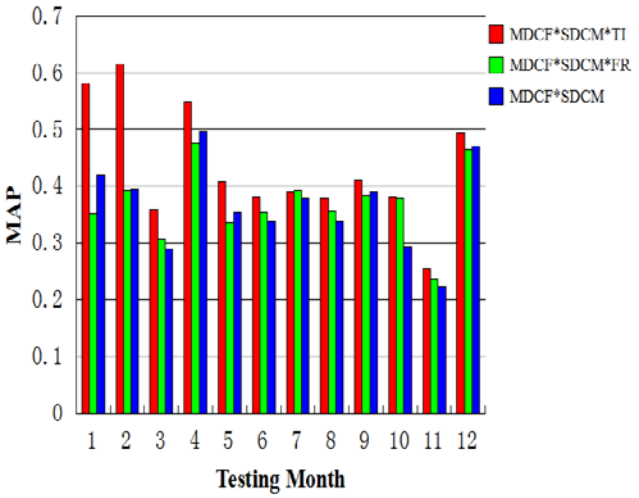

Figure 4. The MAP for MDCF*SDCM*TI, MDCF*SDCM*FR and MDCF*SDCM in 12 testing months. TI helps boost the performance of MDCF*SDCM.

Fig. 4 demonstrates the MAP of MDCF*SDCM, MDCF*SDCM*TI and MDCF*SDCM*FR in the experiment. MDCF*SDCM*TI surpasses others in MAP in eleven of the twelve testing months. MDCF*SDCM*FR gets the highest MAP in one month and there is no much difference between the overall performance of MDCF*SDCM and that of MDCF*SDCM*FR.

Table II summarizes the average MAP in twelve testing months for all nine methods considered in this paper as follows:

TABLE II. THE AVERAGE MAP FOR DIFFERENT METHODS.

	Alone	TI	FR
MDCF	38.4%	***42.2%***	35.8%
SDCM	4.2%	***38.8%***	29.4%
MDCF*SDCM	36.6%	***43.4%***	36.9%

The experimental results of Table II show that: (1) Our time-aware service recommendation approach, MDCF*SDCM*TI, achieves the best performance among all nine methods as expected since it employs content, topology and temporal information of an evolving service ecosystem. (2) As to relevance scores calculation, MDCF gets much better MAP than SDCM and their combination also presents good performance. (3) With respect to popularity scores, our proposed component TI is much more effective than FR to help improve the prediction accuracy of MDCF, SDCM and MDCF*SDCM.

From the experiments shown above, we can conclude that taking evolution into consideration can lead to a higher precision in service recommendation for mashup creation. Combining topology, content and evolution together can improve the recommendation performance in the evolving service ecosystem.

VI. RELATED WORK

Service discovery and recommendation has been acknowledged as a key problem since the dawn of Web service technologies. Early works employed techniques from information retrieval such as TF/IDF and VSM on WSDL documents of services [6, 7]. These keyword search-based methods suffer from poor performance in practice. A recent work [8] focused on services described in semantic languages to automate the process of service discovery. But it is always difficult to acquire semantic information and construction of ontology is trapped in expensive running time and high complexity. Different from above content-based methods, [9] proposed a probabilistic approach for service discovery based on LDA. It extracts features from WSDL documents and exploit LDA model to characterize the latent topics between services and user queries. Finally it recommends related services based on topic relevance. However, restful services have been widely used which make it hard to get WSDL documents of services. So in this work we view the description of a service as its source of content information.

Furthermore, a number of research work center on QoS based web service selection and recommendation. For example, collaborative filtering, which originates from the idea that similar users are likely to consume similar items, has also been introduced into service recommendation recently [10, 11]. A hybrid approach was proposed in [12] which combines content-based matching and collaborative filtering. However, QoS information is not always available. So instead of using QoS attributes, we employ description of a mashup to calculate similarities in our paper.

Another group of researchers try to introduce social network analysis into service recommendation. In [13], a service recommendation algorithm which takes users' interest and social relationship between mashups into consideration is proposed. [14] proposed a matrix model where multi-dimensional social relationships among users, topics, mashups, and services are described. A recent work [15] tries to perform services ranking and clustering mutually in a heterogeneous service network to improve the performance of service ranking.

However, we observed that services and their mashups evolve over time such as publishing, perishing and changing of interfaces [4]. Few existing methods aforementioned take evolution of service usage over time into account. Our previous work [5, 16] proposed a service recommendation method in an evolving ecosystem based on link prediction in a dynamic service network. It is purely based on past service usage and does not take the functional requirements of individual mashup into account. To move a step further, we have proposed a time-aware service recommendation approach for mashup creation based on LDA in this paper that combines together topology, content, and temporal information in an evolving service ecosystem.

VII. CONCLUSIONS

We have proposed a model that combines network structure, content description and service usage history to

describe an evolving service ecosystem. Based on our model, we have further developed a time-aware service recommendation approach for mashup creation based on LDA, consisting of three components: temporal information, mashup-description-based collaborative filtering and service-description-based content matching. These three components exploit temporal information, topology and content, respectively. Experimental results on a real-world data set from ProgrammableWeb.com show that our approach is 5% better than collaborative filtering and nearly 40% better than content matching in terms of mean average precision.

In the following work, we plan to refine the algorithm of forecasting topic activity to improve the short-term prediction of service activity. Moreover, we plan to incorporate user behavior into our model to make our time-aware service recommendation framework more personalized and study the evolution of interactions among users, mashups and services.

ACKNOWLEDGMENT

This work is partially supported by the National Natural Science Foundation of China (No.61033005 and No. 61174169), the National Science and Technology Support Program of China (2012BAF15G01) and the Independent Research Program of Tsinghua University (20111080998).

REFERENCES

[1] V. Andrikopoulos, S. Benbernou and M. P. Papazoglou, "On the evolution of services," IEEE Transactions on Software Engineering, Vol. 38, pp. 609-628, 2012.

[2] X. Liu, Y. Hui, W. Sun, and H. Liang, "Towards service composition based on mashup," in Proceedings of IEEE Congress on Services, 2007, pp. 332-339.

[3] A. P. Barros and M. Dumas, "The rise of web service ecosystems," IT professional, Vol. 8, pp. 31-37, 2006.

[4] E. Al-Masri and Q. H. Mahmoud, "Investigating web services on the world wide web," in Proceedings of the 17th International Conference on World Wide Web, 2008, pp. 795-804.

[5] K. Huang, Y. Fan, and W. Tan, "Recommendation in an evolving service ecosystem based on network prediction," IEEE Transactions on Automation Science and Engineering, accepted in 2013.

[6] X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang, "Similarity search for web services," in Proceedings of the 13th International Conference on Very Large Data Bases, Vol. 30, 2004, pp. 372-383.

[7] C. Platzer and S. Dustdar, "A vector space search engine for web services," in Proceedings of the 3rd IEEE European Conference on Services Computing, 2005, pp. 62-71.

[8] G. C. Hobold and F. Siqueira, "Discovery of semantic web services compositions based on SAWSDL annotations," in IEEE 19th International Conference on Web Services, 2012, pp. 280-287.

[9] C. Li, R. Zhang, J. Huai, X. Guo, and H. Sun, "A probabilistic approach for web service discovery," in Proceedings of the IEEE International Conference on Services Computing, 2013, pp. 49-56.

[10] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Wsrec: A collaborative filtering based web service recommender system," in Proceedings of IEEE International Conference on Web Services, 2009, pp. 437-444.

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

[12] L. Yao, Q. Z. Sheng, A. Segev, and J. Yu, "Recommending Web Services via Combining Collaborative Filtering with Content-Based Features," in Proceedings of IEEE 20th International Conference on Web Services, 2013, pp. 42-49.

[13] J. Cao, W. Xu, L. Hu, J. Wang, and M. Li, "A social-aware service recommendation approach for mashup creation," International Journal of Web Services Research (IJWSR), Vol. 10, pp. 53-72, 2013.

[14] B. Cao, J. Liu, M. Tang, Z. Zheng, and G. Wang, "Mashup service recommendation based on user interest and social network," in Proceedings of IEEE 20th International Conference on Web Services, 2013, pp. 99-106.

[15] Y. Zhou, L. Liu, C. Perng, A. Sailer, I. Silva-Lepe, and Z. Su, "Ranking services by service network structure and service attributes," in Proceedings of IEEE 20th International Conference on Web Services, 2013, pp. 26-33.

[16] K. Huang, Y. Fan, W. Tan, and X. Li, "Service recommendation in an evolving ecosystem: a link prediction approach," in Proceedings of IEEE 20th International Conference on Web Services, 2013, pp. 507-514.

[17] D. M. Blei, A. Y. Ng and M. I. Jordan, "Latent dirichlet allocation," Journal of Machine Learning Research, Vol. 3, pp. 993-1022, 2003.

[18] I. Porteous, D. Newman, A. Ihler, A. Asuncion, P. Smyth, and M. Welling, "Fast collapsed gibbs sampling for latent dirichlet allocation," in Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2008, pp. 569-577.

[19] R. Winter, J. H. Schiller, N. Nikaein, and C. Bonnet, "CrossTalk: cross-layer decision support based on global knowledge," IEEE Communications Magazine, Vol. 44, pp. 93-99, 2006.

[20] Y. Matsubara, Y. Sakurai, C. Faloutsos, T. Iwata, and M. Yoshikawa, "Fast mining and forecasting of complex time-stamped events," in Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2012, pp. 271-279.

[21] K. Huang, J. Yao, Y. Fan, W. Tan, S. Nepal, Y. Ni, and S. Chen, "Mirror, Mirror, on the web, which is the most reputable service of them all?" in Service-Oriented Computing: Springer, 2013, pp. 343-357.

[22] Y. Yue, T. Finley, F. Radlinski, and T. Joachims, "A support vector method for optimizing average precision," in Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2007, pp. 271-278.

[23] W. Tan, J. Zhang and I. Foster, "Network analysis of scientific workflows: a gateway to reuse," IEEE Computer, Vol. 43, pp. 54-61, 2010.