Time-Aware Service Recommendation for Mashup Creation

Yang Zhong, Yushun Fan, Keman Huang, *Member, IEEE*,
Wei Tan, *Senior Member, IEEE*, and Jia Zhang, *Member, IEEE*

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, while others are based on QoS measurement. However, service ecosystem is evolving over time with services publishing, prospering and perishing. Few existing methods consider or exploit the evolution of service ecosystem on service recommendation. This paper employs a probabilistic approach to predict the popularity of services to enhance the recommendation performance. A method is presented that extracts service evolution patterns by exploiting latent dirichlet allocation (LDA) and time series prediction. A time-aware service recommendation framework is established for mashup creation that conducts joint analysis of temporal information, content description and historical mashup-service usage in an evolving service ecosystem. Experiments on a real-world service repository, ProgrammableWeb.com, show that the proposed approach leads to a higher precision than traditional collaborative filtering and content matching methods, by taking into account temporal information.

Index Terms—Service recommendation, service ecosystem, time-aware, mashup creation, latent dirichlet allocation

1 Introduction

With the wide adoption of service-oriented architecture and cloud computing [28], 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 software development cycle [2]. As a consequence, several web service ecosystems (represented by ProgrammableWeb.com[1] and myExperiment.org[2]) have emerged in 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 appropriate service components. Therefore, service discovery and recommendation approach is essential to facilitate mashup developers in locating desired services.

Most existing service recommendation methods are based on content matching, mainly focusing on keyword search [6], [7] and semantic-based approach [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 methods considering content description, others focus on helping developers find services meeting expected quality of service (QoS) criteria. Non-functional properties of services under consideration include reliability, availability, and response time. In addition to formal QoS measurement, user-centric collaborative filtering mechanism [10], [11] 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 such temporal information (TI) for service recommendation. Our previous work [5], [16] proposed a method based on link prediction in a time-varying service network. This paper takes a step further to study the evolution pattern of service usage. Additionally, we conduct joint analysis on temporal information, topology and content in an evolving service ecosystem to combine their advantages to improve the recommendation precision.

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 two assumptions, service usage

1. http://www.programmableweb.com
2. http://www.myexperiment.org

- *Y. Zhong and Y. Fan are with the Department of Automation, Tsinghua University, Beijing, China. E-mail: zhongy12@mails.tsinghua.edu.cn, fanyus@mail.tsinghua.edu.cn.*
- *K. Huang is with the School of Computer Science and Technology, Tianjin University, Beijing, China. E-mail: victoryhkm@gmail.com.*
- *W. Tan is with the IBM Thomas J.WatsonResearch Center, Yorktown Heights, New York, NY. E-mail: wtan@us.ibm.com.*
- *J. Zhang is with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Moffett Field, CA. E-mail: jia.zhang@sv.cmu.edu.*

Manuscript received 13 Oct. 2014; accepted 2 Dec. 2014. Date of publication 17 Dec. 2014; date of current version 12 June 2015.
For information on obtaining reprints of this article, please send e-mail to: reprints.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TSC.2014.2381496

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 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 account functional 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, which is capable of extracting a time sequence of topic activities 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 (MDCF) and service-description-based content matching (SDCM), 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 into account temporal information.

The remainder of this paper is organized as follows. Section 2 introduces a model to describe an evolving service ecosystem and formulates the service recommendation problem. Section 3 describes model training methods. Section 4 presents our time-aware service recommendation framework. Section 5 reports the experimental results. Section 6 summarizes the related work and Section 7 concludes the paper.

2 Problem Definition

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

Definition 1 (Topology). *The topology of a service ecosystem is modeled in an undirected graph $G = (M \cup S, E)$ where M is the set of mashups and S is the set of services. $E \subseteq M \times S$ represents 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 comprises a collection of words MW(m) to describe its functions. Similarly, each service* s *is associated with a collection of words SW (s) to describe its functions.*

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

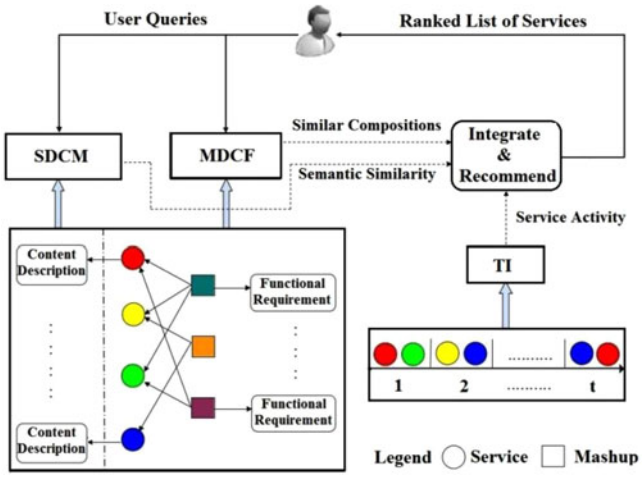

Fig. 1. Overview of time-aware service recommendation framework.

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

Problem 1 (Time-aware Service Recommendation for Mashup Creation). *Given past mashup-service usage with timestamps and text description about mashups and services in an evolving service ecosystem, for a new required mashup at the moment with user queries as a collection of words, a ranked list of services will be recommended to the requesting user. Services with higher rank in the list should have higher probability to be adopted by the user than others with lower ranks.*

The mashup creation problem is thus turned into the ranked recommendation list generation problem. Hence we propose a time-aware service recommendation framework that systematically considers temporal information, topology and content in an evolving service ecosystem. As shown in Fig. 1, the framework consists of three major components: (1) *temporal information* extraction, (2) *mashup-description-based collaborative filtering* and (3) *service-description-based content matching*. The three components offer advice on 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 functional requirements of the required mashup. Particularly, MDCF recommends services based on historical mashups with similar functional requirements; SDCM calculates content similarity between the functional requirements of the new required 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 the three components in details in the next section, and the time-aware service recommendation framework will be presented in Section 4.

TABLE 1
Notations Used in This Model

Symbol	Description
T	Number of topics
|S|	Number of services
|TG|	Number of timestamps
$\|ST_t\|$	Number of service tokens in timestamp t
ST	Vector form of all service tokens
s_{ti}	The ith service token in timestamp t
z_{ti}	Topic assigned to s_{ti}
θ_t	The parameters of multinomial distribution over topics specific to timestamp t
ϕ_z	The parameters of multinomial distribution over services specific to topic z
α, β	The parameters of Dirichlet priors to the multinomial distribution θ_t and ϕ_z

3 Model Framework

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

3.1 Temporal Information Extraction

The objective of TI is to predict service activity in the near future based on service usage history. Directly predicting service activity at the service level will encounter the sparseness problem, since the reuse rate of most services is very low [23]. Instead, we propose to predict service activity at the topic level to alliveate the sparseness issue.

3.1.1 Model of Service Usage History

One fundamental assumption to model service usage history is that users tend to consume popular services in popular service domains at the moment of request. Based on this assumption, we apply an idea similar 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 latent dirichlet allocation [17] can be employed to model the generative process of service usage over time. Table 1 summarizes the notations that we use in this model.

As a preliminary step, we retrieve a collection of service tokens that are consumed at timestamp t from H and denote it as ST_t. For example, if a mashup m is created at timestamp t that consists of two services s_1 and s_2, then we add s_1 and s_2 into ST_t.

The graphical model of temporal information extraction is shown in Fig. 2. The generative process of service usage history can be described as follows:

1) For each topic $z = 1{:}T$
   Draw $\phi_z \sim$ Dirichlet (β)
2) For each timestamp t in TG
   a) Draw $\theta_t \sim$ Dirichlet (α)
   b) For each service token s_{ti} in ST_t
      Draw a topic $z_{ti} \sim$ Multinomial (θ_t)
      Draw a service $s_{ti} \sim$ Multinomial (ϕ_{zti}).

According to the generative process, we define the joint probability of service tokens ST and the set of corresponding topics Z as follows:

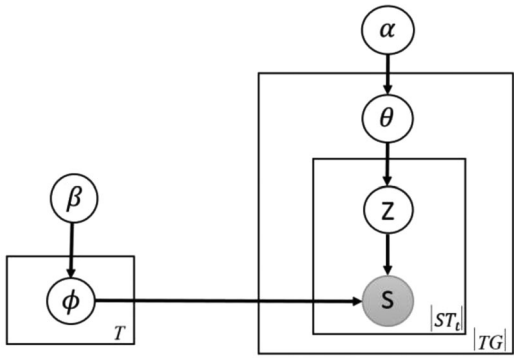

Fig. 2. Graphical model of temporal information extraction.

$$P(ST, Z|\theta, \phi) = \prod_{t=1}^{|TG|} \prod_{z=1}^{T} \prod_{s=1}^{|S|} \theta_{tz}^{n_{tz}} \phi_{zs}^{n_{zs}}, \tag{1}$$

where n_{tz} is the number of times that topic z has been associated with timestamp t and n_{zs} is the number of times that service s has been generated from topic z.

By placing a Dirichlet prior α over θ and another prior β over ϕ, we can obtain the following equation:

$$\begin{aligned} P(ST, Z|\alpha, \beta) &= \int P(ST, Z, \theta, \phi|\alpha, \beta) d\theta d\phi \\ &= \int P(ST, Z|\theta, \phi) P(\theta|\alpha) P(\phi|\beta) d\theta d\phi \\ &= \prod_{t=1}^{|TG|} \frac{\Gamma(\sum_z \alpha_z)}{\prod_z \Gamma(\alpha_z)} \frac{\prod_z \Gamma(n_{tz} + \alpha_z)}{\Gamma(\sum_z n_{tz} + \alpha_z)} \\ &\quad \times \prod_{z=1}^{T} \frac{\Gamma(\sum_s \beta_s)}{\prod_s \Gamma(\beta_s)} \frac{\prod_s \Gamma(n_{zs} + \beta_s)}{\Gamma(\sum_s n_{zs} + \beta_s)}. \end{aligned} \tag{2}$$

3.1.2 Learning Algorithm

A variety of algorithms have been developed to estimate parameters of topic models. In this paper, we apply the Gibbs sampling [18] to infer unknown parameters $\{\theta, \phi\}$ mainly due to its ease of implementation. More specifically, we begin with the posterior probability for sampling the latent topic for each service token with flat priors

$$P(z_{ti}|Z^{\neg ti}, ST) \propto \frac{n_{tz_{ti}}^{\neg ti} + \alpha}{\sum_z \left(n_{tz_{ti}}^{\neg ti} + \alpha\right)} \times \frac{n_{z_{ti}s_{ti}}^{\neg ti} + \beta}{\sum_s \left(n_{z_{ti}s}^{\neg ti} + \beta\right)}, \tag{3}$$

where the superscript $\neg$ denotes a quantity excluding the current instance.

We then estimate the parameters by the sampling results. Through a similar deduction with LDA, we update the parameters as follows:

$$\theta_{tz} = \frac{n_{tz} + \alpha}{\sum_z (n_{tz} + \alpha)} \tag{4}$$

$$\phi_{zs} = \frac{n_{zs} + \beta}{\sum_s (n_{zs} + \beta)}, \tag{5}$$

where θ_{tz} can be interpreted as activity of topic z at timestamp t and ϕ_{zs} represents the correlation strength between service s and topic z. The algorithm of applying Gibbs sampling to estimate parameters is listed below

Algorithm 1. Gibbs sampling

Input:
1) Hyper-parameters α and β
2) Service tokens ST
3) Iteration number N

Output
1) Parameters estimates $\{\theta, \phi\}$

Procedure
01. Initialize Z randomly
02. **For** $iter = 1{:}N$
03. **For** each service token s_{ti}
04. sample z_{ti} according to equation (3)
05. **End**
06. **End**
07. Read out θ according to equation (4)
08. Read out ϕ according to equation (5)

3.1.3 Service Activity Prediction

With the model learned, we can further predict service activity through topic evolution. More specifically, $\{\theta_{tz}\}_{t \in TG}$ constitute a time series for each fixed topic z. By applying a time series prediction method to the set, activity of topic z at time $t+1$ can be forecasted. Several methods exist to solve this problem, such as linear weighted moving average (LWMV) [19] and auto regression [20]. In this paper, we choose to adopt the linear weighted moving average because of its efficiency and simplicity. Some advanced methods such as auto regression may not be suitable here due to the sparseness problem, since the reuse rate of most services is very low [42].

Given a time window length l, linear weighted moving average predicts the activity of topic z at time $t+1$ through the following equation:

$$\theta_{(t+1)z} = \sum_{i=1}^{l} \lambda_i \theta_{(t+1-i)z}, \tag{6}$$

where λ_i are positive real numbers subject to the constraint $\sum_i \lambda_i = 1$. 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.

With topic evolution as a bridge between timestamps and services, activity of service s at time $t+1$ (popularity score) can be calculated as follows:

$$p_{TI}(s, t+1) = \sum_{z=1}^{T} \theta_{(t+1)z} \phi_{zs}. \tag{7}$$

3.2 Mashup-Description-Based Collaborative Filtering

Collaborative filtering is one of the state-of-the-art methods in the recommendation community [27]. Its basic idea is that similar users are likely to consume similar items. Previous works such as [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 support collaborative filtering.

We propose to recommend services for a new required mashup based on composite services in similar historical mashups. For example, if a new required mashup m is similar to 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. Thus, the key to this component is the similarity measurement. Since we leverage mashup description to calculate the similarities between mashups, this component is named as mashup-description-based collaborative filtering.

3.2.1 Similarity Computation

Traditional methods typically model the words descriptions using the Vector Space Model (VSM), and then adopt cosine similarity between vectors as measurement. However, the VSM-based similarity calculation method has an inherent drawback since it cannot capture the semantic similarity between different words. For example, "map" and "geology" are two frequent words used to describe mapping-related service compositions. Clearly, the two words are perceived to have close semantic association, but the VSM-based method fails to catch the semantic similarity, treating them as two independent and different words.

To address the challenge and enhance the similarity measurement, we employ the LDA model to calculate the similarity among mashups based on their functional requirements. LDA can capture the semantic association between words, by introducing latent topics as a bridge between mashup and its functional requirements. Experiments on our data set with collaborative filtering show that LDA-based method is relatively 20 percent higher than VSM-based method in terms of precision.

Specifically, we model the generation of functional requirements of mashups by LDA. Thus we can feed word tokens of all mashups' functional requirements MW into Algorithm 1 (instead of service tokens) to get θ_{mz} (distribution of mashup m over topics) and ϕ_{zw} (distribution of topic z over words).

When a new required mashup m comes up with user query Q, its similarity with a historical mashup m_i can be calculated using the following equation:

$$sim(m, m_i) = \sum_{w \in Q} \sum_{z=1}^{T} \theta_{m_i z} \phi_{zw}. \tag{8}$$

The intuition behind the above equation is that, the similarity between the new required mashup and historical mashup is calculated as the likelihood of generating the current user query according to the estimated language model of the historical mashup, i.e., the topic distribution of the historical mashup along with words distribution of topics.

3.2.2 Collaborative Filtering

With similarities among mashups in hand, further filtering can be conducted.

Given a positive integer K, historical mashups whose similarity rankings are higher than K are to be retained while others will be filtered out. The value of K can be set empirically and will be discussed in detail in the experimental section.

Once the set of most similar historical mashups are obtained, the relevance score of services with respect to the new required mashup m can be evaluated as follows:

$$r_{CF}(s,m) = \sum_{m_i \in U(K,m)} sim(m,m_i) I(m_i,s), \tag{9}$$

where *U (K,m)* contains the Top-*K* similar mashups with m; the indicator function $I(m_i,s)$ is 1 if $(m_i,s) \in E$ and 0 otherwise.

3.3 Service-Description-Based Content Matching

Different from MDCF, which recommends services based on similar mashups, content matching directly calculates the content similarity between the new required mashup and services. Since we employ service description data to calculate the content similarities between user queries and services, this component is named as service-description-based content matching.

Similar with [9], we model the generation of content descriptions of services by LDA. Thus we can feed word tokens of all services' content descriptions *SW* into Algorithm 1 to get θ_{sz} (distribution of service s over topics) and ϕ_{zw} (distribution of topic z over words).

Similar with MDCF, we measure the content similarity as the likelihood of generating the current user query according to the estimated language model of the service. Accordingly, when a new required mashup m comes up with user queries as a collection of word tokens, SDCM calculates the content similarity between m and service s as follows:

$$r_{CM}(s,m) = \sum_{w \in Q} \sum_{z=1}^{T} \theta_{sz} \phi_{zw}. \tag{10}$$

Note that our SDCM is slightly different from [9], in that we apply summation over queries in equation (10) instead of multiplication in [9]. Our hypothesis is that over a large candidate service pool, extracting feature union may yield higher performance comparing to feature intersection. Experimental results on our data set have demonstrated our hypothesis. SDCM works significantly better with MDCF than [9] with MDCF. Details will be discussed in the experimental section about Fig. 8.

4 Recommendation Algorithm

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

4.1 Component Integration

When a new mashup m is requested with user queries at time $t+1$, we employ the three components to provide advice on suitability of services from different perspectives. TI is responsible for popularity scores while MDCF and SDCM are invoked to obtain relevance scores. By integrating the three kinds of scores through multiplication, we can generate a new kind of score to measure the suitability of service s as follows:

$$pr(s,m) = p_{TI}(s,t+1) r_{CF}(s,m) r_{CM}(s,m). \tag{11}$$

The rationale behind integration by multiplication, which is equivalent to geometric mean, is that it is less sensitive to extreme values than traditional arithmetic mean. Services with higher integrated scores have a higher probability to be adopted by mashup m. Therefore, we can rank and recommend a list of services for the new required mashup in a descending order of the integrated scores.

4.2 Algorithm Design

Now we present the details of our time-aware service recommendation algorithm for mashup creation in the following table:

Algorithm 2. Time-aware service recommendation

Input:
1) *G*: Topology model
2) *ST*: Service tokens of all timestamps
3) ***MW***: Word tokens of all functional requirement
4) ***SW***: Word tokens of all content descriptions
5) *T*: The number of latent topics in LDA model
6) *N*: The number of iterations in Gibbs sampling
7) α and β: The prior parameters in LDA model
8) *l* and λ_i: Window length and weights in LWMV
9) *K*: Top-K similar mashups in MDCF
10 *Q*: User queries for new mashup m

Output:
1) *LS(m)*: Ranked list of services for m

Procedure:
01. $\{\theta_{tz}, \phi_{zs}\} = GibbsSampling(\alpha, \beta, N, ST)$
02. **For** topic $z = 1{:}T$
03. predict topic evolution by equation (6)
04. **End**
05. **For** each service *s*
06. get $p_{\mathrm{TI}}(s, t+1)$ by equation (7)
07. **End**
08. $\{\theta_{mz}, \phi_{zw}\} = GibbsSampling(\alpha, \beta, \mathrm{N}, \boldsymbol{MW})$
09. $\{\theta_{sz}, \phi_{zw}\} = GibbsSampling(\alpha, \beta, \mathrm{N}, \boldsymbol{SW})$
10. **for** each historical mashup m_i
11 get *sim(m, m_i)* by equation (8)
12. **end**
13. **for** each service *s*
14. get r_{CF} *(s, m)* by equation (9)
15. **end**
16. **for** each service *s*
17. get r_{CM} *(s, m)* by equation (10)
18. **end**
19. **for** each service *s*
20. get *pr(s, m)* by equation (11)
21. **end**
22. return *LS(m)* in descending order w.r.t. *pr(s, m)*

The algorithm can be divided into two stages: offline stage (Lines 01 $\sim$ 09) and online stage (Lines 10 $\sim$ 22). The offline part only needs to be conducted once at the start of each time interval. The online part performs every time when receiving a query. Lines 01 $\sim$ 07 are the implementation of TI that predicts service activity in the near future. Lines 08 and 10 $\sim$ 15 describe the construction of MDCF. Lines 09 and 16 $\sim$ 18 complete the calculation of SDCM. Lines 19 $\sim$ 22 integrate the three components and generate a recommended list of services to the user for mashup creation.

4.3 Computational Complexity

This section discusses the upper bound on the computational complexity of the proposed algorithm. In the following discussion, we assume the online query has a number of P word tokens on average.

4.3.1 Complexity of TI

The complexity of Gibbs sampling to estimate the parameters in TI (Line 01) is bounded by $O(N|H|T)$, where $|H|$ is the number of service tokens in ST. From equation (6), we know the complexity of topic activity prediction (Lines $02 \sim 04$) is $O(lT)$. Similarly, the complexity of service activity prediction (Lines $05 \sim 07$) is $O(T|S|)$.

4.3.2 Complexity of MDCF

MDCF involves both offline (Line 08) and online (Lines $10 \sim 15$) computation. The complexity of LDA in Line 08 is $O(NVT)$, where V is the number of word tokens in MW. For the online part, the time complexity of similarities computation among mashups is $O(P|M|T)$ according to equation (8), where $|M|$ is the number of historical mashups. Lines $13 \sim 15$ involves sorting of mashups with a complexity of $O(|M|\log|M| + K|S|)$.

4.3.3 Complexity of SDCM

Similarly, for the offline part of SDCM (Line 09), the computational complexity is $O(NWT)$ where W is the number of word tokens in SW. By equation (10), we know the computational complexity of Lines $16 \sim 18$ is $O(P|S|T)$.

4.3.4 Overall Computational Complexity

The complexity of the integration (Lines $19 \sim 21$) is $O(|S|)$ and the ranking of services (Line 22) is $O(|S|\log|S|)$.

In practice, it is usually the case that $|S|$ and l are negligible compared with NW, K is much greater than $\log|S|$, and PT is much greater than $\log|M|$. Therefore, the overall complexity of offline computing is $O(NT(V + |H| + W))$ and the online part is $O(PT(|M| + |S|) + K|S|)$. The complexity analysis demonstrates that the proposed algorithm is computationally feasible in practice.

5 Experiments

In this section, we explain how we 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.

5.1 Data Set Preparation

To the best of our knowledge, ProgrammableWeb.com 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 metadata such as name, creation date, description and the list of services used. Table 2 summarizes the basic properties of our data set.

TABLE 2
Basic Properties of ProgrammableWeb Data Set

Number of services	7,077
Number of mashups	6,594
Number of unique words	13,648

5.2 Preprocessing

For each service in the data set, there is a description consisting of a bag of words that describe the functionality of the service. Before the description can be used as the underlying service's associated collection of words in our model, several nature language preprocessing tasks have to be conducted.

In this work, we applied the four-step data preprocessing method similar in [21] to extract meaningful words from the original descriptions:

1) *Original words generating*. First of all, we extract all original words contained in the descriptions.
2) *Pruning*. Secondly, we filter out 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*; and some adverbs such as *where, when, quite,* 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 can be directly derived from the service list of a mashup. Given a time granularity, we can obtain the sequence of timestamps and service usage history according to the creation dates and the service list of the 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.

Without losing generality, in our experiment, we adopted a time granularity as one month. To examine the performance of our approach, we divided the data set into training sets and testing sets, with a moving cutoff timestamp. We use the data before the cutoff timestamp as the training set and the data with exactly that cutoff timestamp as testing set. As shown in Fig. 3, we move the cutoff timestamp from September 2011 to August 2012, obtaining twelve corresponding training data sets and testing data sets. For each mashup appeared in the testing month, we use its description as a user query and its composite services as the ground truth.

5.3 Evaluation Metric

The evaluation metric that we used in this experiment is mean average precision (MAP) [22], which is a widely used measure in recommender system

$$MAP = \frac{1}{|M_c|}\sum_{m \in M_c}\frac{1}{|CS_m|}\sum_{s \in CS_m}\frac{top(s,m)}{rank(s,m)}, \qquad (12)$$

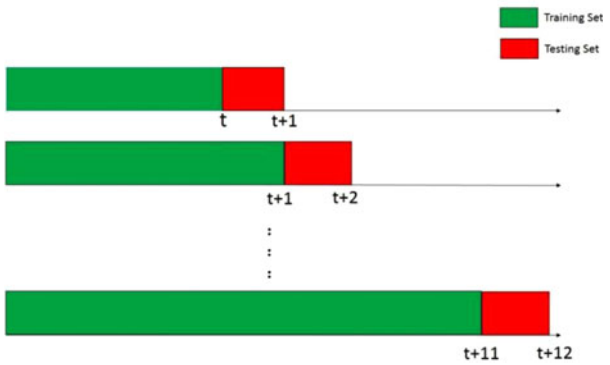

Fig. 3. Generaton of training and testing data sets.

where M_c represents the set of mashups in the testing month and $|M_c|$ is the cardinality of M_c; CS_m represents the set of component services of mashup m and $|CS_m|$ is the cardinality of CS_m; *rank*(*s*, *m*) is the ranking position of service s in the recommended list of services for a testing mashup m; *top*(*s*, *m*) is the number of composite services of mashup m whose ranking position is higher or equal to *rank*(*s*, *m*).

MAP is a real number between 0 and 1. The higher MAP indicates a better recommendation performance. 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.

5.4 Comparison Methods

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

To make our comparison more complete, we have introduced an alternative method to calculate popularity scores named as FR. It 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 integrated score of a service *s* for a new required mashup *m* with user queries *Q* as follows:

$$pr_{CF*FR}(s,m) = r_{CF}(s,m)f(s), \tag{13}$$

where *f* (*s*) is equal to the usage times of service *s* divided by the sum of usage times for all services. Finally, MDCF*FR returns a ranked list of services for *m* in a descending order of the integrated scores. 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.

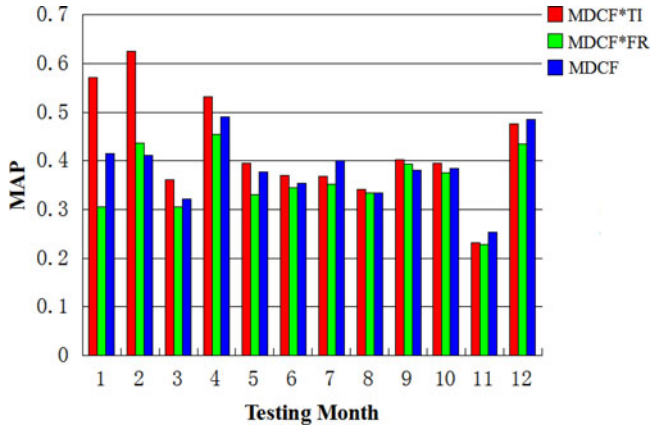

Fig. 4. MAP for MDCF*TI, MDCF*FR and MDCF in 12 testing months. TI helps boost the performance of MDCF.

5.5 Experimental Results and Analysis

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, we set $\lambda_2 = 0.1$, $\lambda_1 = 0.9$ and $l = 2$. The selection of *T* and *K* is explained in the following sections and other parameters are set empirically by experiments.

Fig. 4 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. 5 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 over SDCM with TI included, and ranks the highest in all testing months except one. SDCM*FR wins in only one month.

Fig. 6 demonstrates the MAP of MDCF*SDCM, MDCF*SDCM*TI and MDCF*SDCM*FR in the experiment. MDCF*SDCM*TI surpasses others in 11 of the 12

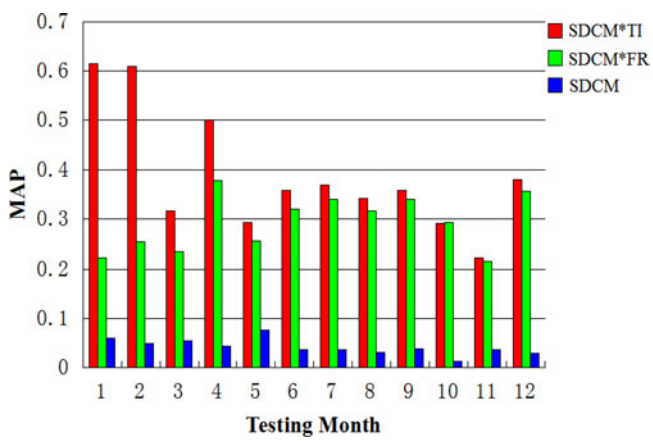

Fig. 5. MAP for SDCM*TI, SDCM*FR and SDCM in 12 testing months.TI helps boost the performance of SDCM.

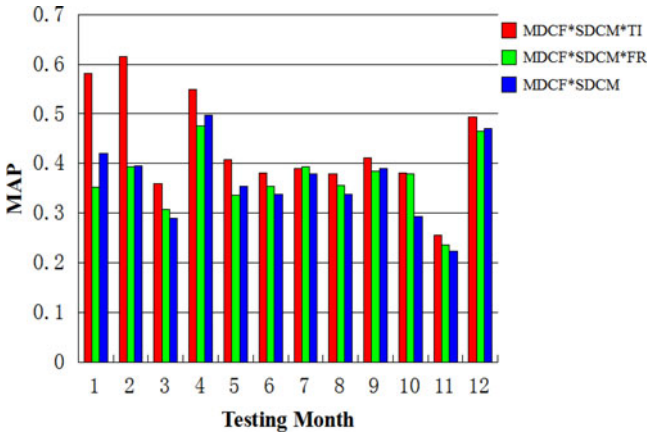

Fig. 6. MAP for MDCF*SDCM*TI, MDCF*SDCM*FR and MDCF*SDCM in 12 testing months. TI helps boost the performance of MDCF*SDCM.

testing months. MDCF*SDCM*FR gets the highest MAP in one month, and there is not much difference between the overall performance of MDCF*SDCM and that of MDCF*SDCM*FR.

Table 3 summarizes the average MAP in twelve testing months for all nine methods considered in this paper as follows:

Three conclusions can be drawn from the experimental results shown in Table 3

1) The proposed approach, MDCF*SDCM*TI, achieves the best performance as expected. Our explanation is that it employs comprehensive information (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, TI is much more effective than FR to help improve the performance of MDCF, SDCM and MDCF*SDCM. The reason for the different performance is that TI is able to capture recent usage trends of services at the moment of request while FR fails to do so by simply normalizing service usage frequency.

5.5.1 Component Contribution Analysis

We integrate three different components in our model: TI, MDCF and SDCM. Here we further examine the individual contribution of different components to the recommendation performance.

We first rank the individual components by their predictive power. We respectively remove each particular component from MDCF*SDCM*TI and evaluate the decrease of MAP according to Table 3. A larger decrease means a higher predictive power. Afterwards we remove the three components one by one in descending order of predictive power (TI > MDCF > SDCM). Finally we show in Fig. 7 the MAP of the three generated methods: MDCF*SDCM*TI, MDCF*SDCM and SDCM.

TABLE 3
The Average MAP for Different Methods

	Alone	TI	FR
MDCF	*38%*	***41%***	*35%*
SDCM	*4%*	***38%***	*29%*
MDCF*SDCM	*36%*	***42%***	*36%*

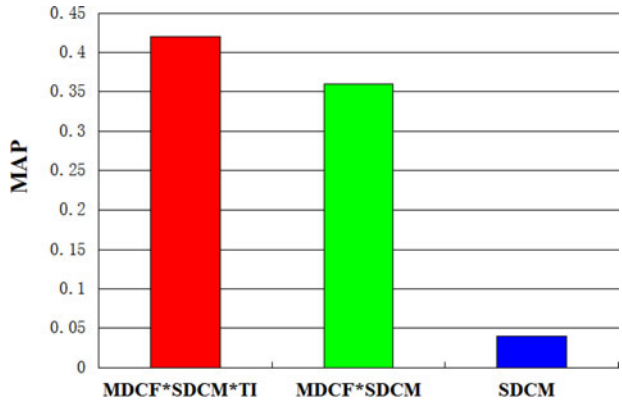

Fig. 7. MAP of three methods for component contribution analysis.

We can observe a clear drop on MAP when ignoring each of the components. This demonstrates that our approach works well by combining the different components and each component contributes improvement to the recommendation performance.

We also designed an experiment to test our hypothesis described in Section 3.3, to compare our approach with the method proposed in [9] (denoted by LCM). We examine the recommendation performance of using the two methods alone, combining with MDCF, and with MDCF*TI, respectively.

Fig. 8 reports the average MAP of SDCM, SDCM*MDCF, SDCM*MDCF*TI and their counterpart methods while SDCM being replaced by LCM. As individual methods, LCM is better than SDCM in recommendation performance. However, when combined with other components MDCF and TI, the integrated methods with SDCM significantly outperform those with LCM, which demonstrates the comtribution of SDCM. Fig. 8 also shows that our proposed integrated method (SDCM*MDCF*TI) significantly outperforms LCM.

5.5.2 Impact of K

MDCF selects only Top-*K* similar historical mashups for collaborative filtering in equation (9). Mashups whose

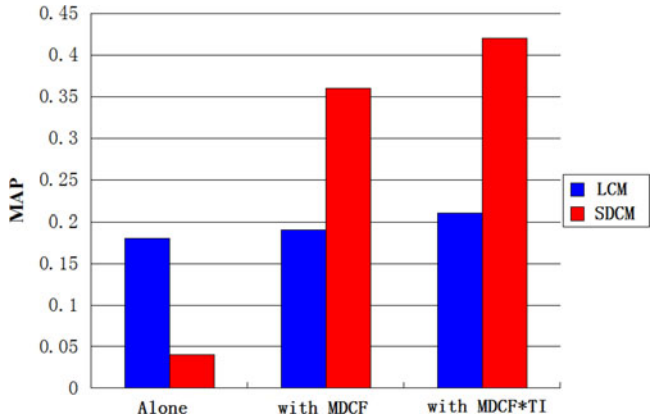

Fig. 8. Average MAP for SDCM, LCM and their combinations with MDCF and MDCF*TI, respectively.

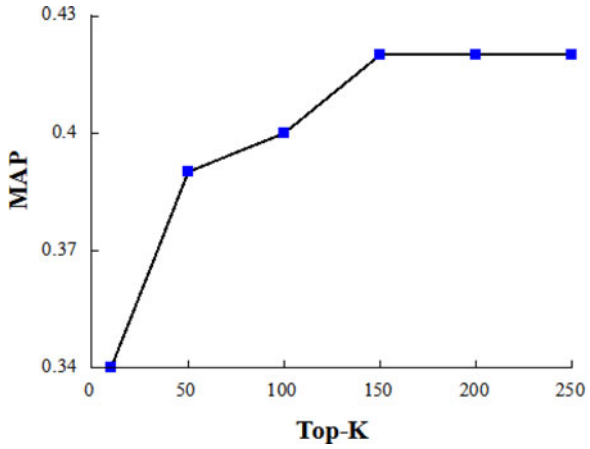

Fig. 9. Impact of K on MAP of proposed approach.

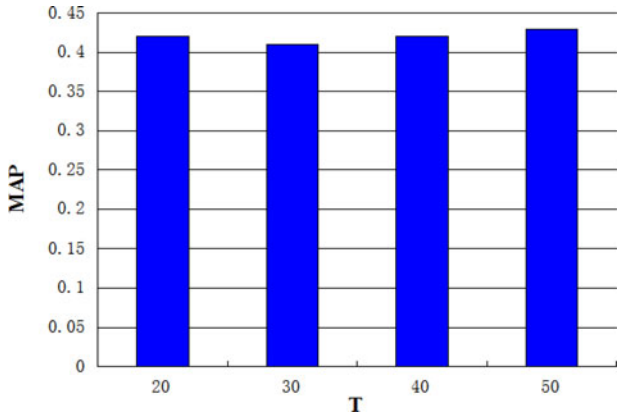

Fig. 11. MAP of the proposed approach with the number of topics varied.

similarity ranking is lower than K are considered as dissimilar ones, and are not included in the computation process. To study the impact of K on recommendation performance, we perform an analysis by varying K from 10 to 250 with a step value of 50. Fig. 9 shows the MAP of our approach with K varied under the experimental setting of other parameters unchanged as before.

Observing from Fig. 9, we can draw the conclusion that K impacts the recommendation performance significantly. When K is small (<150), increasing the number often receives a performance improvement. The trend becomes stable when K is up to 150. This explains why we choose $K = 150$ in our experimental setting.

We take a test instance as an example to further explore the causes of trends between K and MAP. We calculate the similarity between the test instance and all historical mashups. Afterwards we obtain a ranked list of mashups in a descending order of their similarities.

Fig. 10 shows the relationship between similarity and ranking position. The similarity drops dramatically as ranking gets lower. Similarity of the 1st mahup is nearly 10 times as that of the *1,000*th mashup in the list, and the value tends to become zero as ranking position extends beyond 4,000. Therefore, we can conclude that only a small portion of historical mahups can be perceived similar with the test instance and be included in collaborative filtering for service recommendation. The remaining mashups' similarity values are too low to have significant contributions to the recommendation performance according to equation (8). These dissimilar mashups can be neglected to save computation cost without harming precision.

5.5.3 Impact of Number of Topics

We studied how the number of topics in the LDA model influences the recommendation performance.

With prior knowledge that there are about 30 service domains in ProgrammableWeb.com, we show the MAP of the proposed method with T varied from 20 to 50 (with all other parameters fixed) in Fig. 11.

It shows that although the performance changes with T varied, the largest difference is less than 0.02. This demonstrates that our time-aware service recommendation approach is not sensitive to the number of topics.

5.5.4 Convergence Property

We take testing month 1 for example to see the effect of the number of Gibbs sampling iterations on recommendation performance.

Fig. 12 shows the MAP of our proposed algorithm with N varied. We see that the algorithm can converge in less than 60 iterations and the recommendation performance becomes stable after that. This suggests that our algorithm is efficient and has a good convergence property.

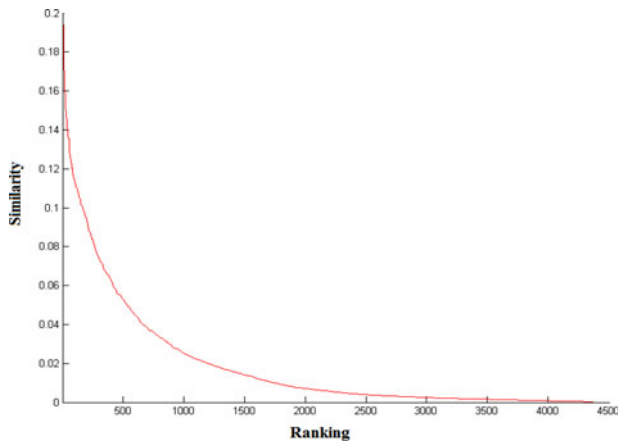

Fig. 10. Example to show the relationship between similarity and ranking position in MDCF.

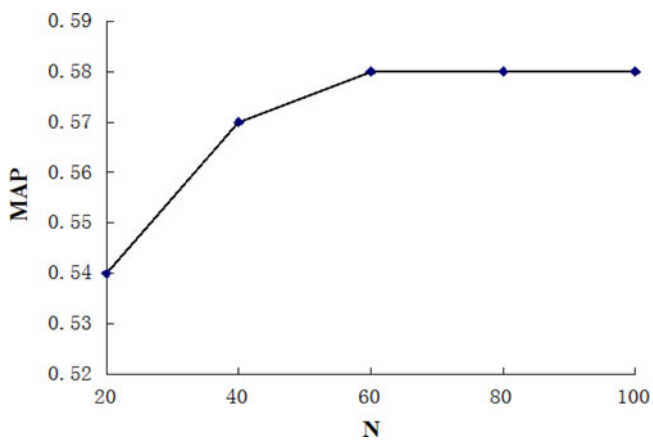

Fig. 12. MAP of the proposed approach with number of iterations in Gibbs sampling varied.

TABLE 4
Ranking Of Ground Truth By Different Methods

	MDCF	MDCF*TI	MDCF*SDCM*TI
G	***1***	***1***	***1***
T	***24***	***9***	***3***
A	***33***	***16***	***12***

5.5.6 Qualitative Case Study

We now present a case study to illustrate the effectiveness of our approach.

The testing mashup for illustration consists of three services: Google Maps (G), Twilio (T) and Google Apps Engine (A). The corresponding functional requirement (i.e., user query) is: *"Geospeaker is a web application providing a virtual loudspeaker to communicate with people around you."* Table 4 reports the ranking of ground truth in the recommended list of services given by MDCF, MDCF*TI and MDCF*TI*SDCM, respectively.

As shown in the table, MDCF successfully puts Google Maps in the first place, but the ranking of the rest two services are not satisfactory. By MDCF alone, which leverages similar compositions for recommendation, we can only get a MAP of 39 percent for this instance. Armed with TI, MDCF*TI improves the ranking of both Twilio and Google Apps Engines at least 50 percent, by taking popularity scores into consideration while keeping Google Maps unchanged. With the help of SDCM, our approach further boosts the ranking of Twilio and Google Apps Engine, by adding content matching into the model and the final MAP is increased to 55 percent.

6 Related Work

Service discovery and recommendation has been acknowledged as a key problem since the dawn of web service technologies.

6.1 Semantic-Aware Recommendation

Early works usually applied techniques from the information retrieval (IR) community, such as TF/IDF and Vector Space Model, on WSDL documents of services [6], [7]. Meng et al. [24] proposed a user-based collaborative filtering algorithm to recommend services. However, these keyword search-based methods typically suffer from poor performance in practice.

Several methods take into consideration the semantic compatibility between services and the query. In MashupAdvisor, the AI planner and the semantic matcher are used to recommend services for composition [31]. Zhao et al. [32] constructed a semantic Bayesian network based on the semi-supervised learning method for the recommendation. A recent work [8] focused on services described in semantic languages to automate the process of service discovery. A hybrid approach was proposed in [12], which combined semantic-based content matching and QoS prediction. However, it is always difficult to acquire semantic information and the construction of ontology is trapped in expensive running time and high complexity.

Different from traditional content-based methods, [9] proposed a probabilistic approach for service discovery based on LDA. It extracts features from WSDL documents and exploits LDA model to characterize the latent topics between services and user queries. It then recommends related services based on topic relevance. Chen et al. [33] developed a service clustering method, in which WSDL documents and services tags are both utilized to cluster services to facilitate the service recommendation. However, it has become difficult to get WSDL documents since RESTful services have been widely used. Even worse, due to business interests and the privacy protection requirement, service providers tend not to offer the WSDL documents [34].

In our mind, the service's description can be viewed as its source of information. LDA can be used to learn the latent functional topics of services based on their descriptions [21], [44]. Hence, in this work, based on the service-s'decription, we extract the topic-level semantic vector to characterize functionality of services.

6.2 QoS-Aware Recommendation

A number of research work center on QoS-based web service selection and recommendation [10], [11], [26]. For example, collaborative filtering has been introduced into QoS prediction recently [10]. Cao et al. [35] presented a hybrid collaborative filtering algorithm based on QoS to provide bidirectional recommendation for both providers and consumers. Zheng et al. [36] proposed two personalized QoS ranking prediction approaches to calculate the QoS of the services for different users, and then recommended the services with a higher QoS quality for consumers. Chen et al. [25] provided instant recommendation for partially composed composite service while meeting QoS requirements. W. Ahmed et al. [37] proposed a novel hidden Markov models (HMM) method for QoS metrification, which measures and predicts the behavior of web services in terms of response time.

Services are deployed geo-distributed and delivered to users located in different geographic locations over the Internet. Hence, the network condition will affect the user-experience of the services. Klein et al. [38] evaluated the latency between any two network locations and then proposed a generic algorithm to achieve the near-optimal composition with low latency. Wang et al. [39] employed the historical network latency records and the IP address information to predict the missing values of network delay. Then simulated annealing algorithm was used to recommend the near-optimal composition solutions to the users. However, the latencies between all the location pairs are dynamic over time and it is time-consuming to evaluate. Based on the assumption that users with near-by location may have similar service experience, Lo et al. [40] incorporated the local connectivity between users to identify the neighborhood and then developed a location-aware matrix factorization model to predict the missing QoS values. Chen et al. [41] grouped the users based on the IP address, clustered the services based on QoS similarity and then applied the collaborated filtering method.

However, QoS information is not always available. Therefore, instead of using QoS attributes, we employ descriptions of mashups to calculate similarities in our work.

6.3 Network-Aware Recommendation

Another group of researchers try to introduce social network analysis into service recommendation. Zhang et al. [42] modeled past service usage behaviors into social networks and leveraged social network analysis to facilitate service reuse. Tan et al. [43] studied the usage patterns of services in the service-workflow system and then a GPS-like assistance tool, ServiceMap, is developed to recommend the service operation chains for users. In [13], a service recommendation algorithm is presented, which takes into consideration users' interest and social relationship between mashups. [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. Zhang et al. [46] presented the correction policy and the precision policy for computing user similarity to improve the accuracy of the recommendation. Maaradji et al. [47] introduced the SoCo framework, where a social network was built from the interactions between users and services, as well as services compositions. Wang et al. [45] emphasized on mining mashup community from users' perspective. Huang et al. [30] exploited the data dependency, similarity and usage to construct a component layered graph and a Steiner-Tree-Search-based algorithm was introduced to recommend service compositions.

However, we observed that services and their mashups evolve over time such as publishing, prospering and perishing [4]. Few existing methods take into account the evolution of service usage over time. Our previous work [5], [16] proposed a service recommendation method based on link prediction in a dynamic service network. However, it is purely based on past service usage and does not consider the functional requirements of individual mashups. [29] presents a Dynamic Topic Model (DTM), which explicitly models the evolution of topics by introducing nature parameters of topic distribution for each time slice. Applied in the context of this paper, however, DTM suffers from data sparseness due to substantially increased complexity. Instead, we have presented a time-aware service recommendation approach based on LDA, which seamlessly combines topology, content, and temporal information in an evolving service ecosystem.

7 Conclusion

We have presented a model that combines network structure, content description and service usage history to describe an evolving service ecosystem. Based on our model, we have developed a time-aware service recommendation framework for mashup creation based on LDA, consisting of three components: temporal information extraction, mashup-description-based collaborative filtering and service-description-based content matching. The three components exploit temporal information, topology and content of an evolving service ecosystem, respectively. Experimental results on a real-world data set from ProgrammableWeb.com show that our approach is relatively 10 percent better than collaborative filtering (38–>42 percent) and much better than content matching (4–>42 percent) in terms of mean average precision.

In the future 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 the model to make our recommendation framework more personalized.

Acknowledgments

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

References

[1] V. Andrikopoulos, S. Benbernou, and M. P. Papazoglou, "On the evolution of services," *IEEE Trans. Softw. Eng.*, vol. 38, no. 3, pp. 609–628, May/Jun. 2012.

[2] X. Liu, Y. Hui, W. Sun, and H. Liang, "Towards service composition based on mashup," in *Proc. IEEE World Congr. Serv.*, 2007, pp. 332–339.

[3] A. P. Barros and M. Dumas, "The rise of web service ecosystems," *IEEE IT Prof.*, vol. 8, no. 5, pp. 31–37, Sep./Oct. 2006.

[4] E. Al-Masri and Q. H. Mahmoud, "Investigating web services on the world wide web," in *Proc. 17th Int. Conf. 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 Trans. Autom. Sci. Eng.*, vol. 11, no, 3, pp. 906–920, Jul. 2014.

[6] X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang, "Similarity search for web services," in *Proc. 13th Int. Conf. Very Large Data Bases*, 2004, vol. 30, pp. 372–383.

[7] C. Platzer and S. Dustdar, "A vector space search engine for web services," in *Proc. 3rd IEEE Eur. Conf. Serv. Comput.*, 2005, pp. 62–71.

[8] G. C. Hobold and F. Siqueira, "Discovery of semantic web services compositions based on SAWSDL annotations," in *Proc. IEEE 19th Int. Conf. Web Serv.*, 2012, pp. 280–287.

[9] C. Li, R. Zhang, J. Huai, X. Guo, and H. Sun, "A probabilistic approach for web service discovery," in *Proc. IEEE Int. Conf. Serv. Comput.*, 2013, pp. 49–56.

[10] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "QoS-aware web service recommendation by collaborative filtering," *IEEE Trans. Serv. Comput.*, vol. 4, no. 2, pp. 140–152, Apr.–Jun. 2011.

[11] X. Chen, X. Liu, Z. Huang, and H. Sun, "RegionKNN: A scalable hybrid collaborative filtering algorithm for personalized web service recommendation," in *Proc. 17th IEEE Int. Conf. Web Serv.*, 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 *Proc. IEEE 20th Int. Conf. Web Serv.*, 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," *Int. J. Web Serv. Res.*, 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 *Proc. IEEE 20th Int. Conf. Web Serv.*, 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 *Proc. IEEE 20th Int. Conf. Web Serv.*, 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 *Proc. IEEE 20th Int. Conf. Web Serv.*, 2013, pp. 507–514.

[17] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *J. Mach. Learn. Res.*, 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 *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery 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 Commun. Mag.*, vol. 44, no. 1, pp. 93–99, Jan. 2006.
[20] Y. Matsubara, Y. Sakurai, C. Faloutsos, T. Iwata, and M. Yoshikawa, "Fast mining and forecasting of complex time-stamped events," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery 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 *Proc. 11th Int. Conf. Serv.-Oriented Comput.*, 2013, pp. 343–357.
[22] Y. Yue, T. Finley, F. Radlinski, and T. Joachims, "A support vector method for optimizing average precision," in *Proc. 30th ACM SIGIR Int. Conf. Res. Develop. Inf. Retrieval*, 2007, pp. 271–278.
[23] W. Tan, J. Zhang, and I. Foster, "Network analysis of scientific workflows: A gateway to reuse," *IEEE Comput.*, vol. 43, no. 9, pp. 54–61, Sep. 2010.
[24] S. Meng, W. Dou, X. Zhang, and J. Chen, "KASR: A keyword-aware service recommendation method on MapReduce for big data application," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 12, pp. 3221–3231, Dec. 2014.
[25] L. Chen, J. Wu, H. Jian, H. Deng, and Z. Wu, "Instant recommendation for web services composition," *IEEE Trans. Serv. Comput.*, vol. 7, no. 4, pp. 586–598, May 2013.
[26] L. Zeng, B. Benatallah, A. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-aware middleware for web services composition," *IEEE Trans. Softw. Eng.*, vol. 30, no. 5, pp. 311–327, May 2004.
[27] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 6, pp. 734–749, Jun. 2005.
[28] D. Agrawal, S. Das, and A. El Abbadi, "Big data and cloud computing: New wine or just new bottles?" *Proc. VLDB Endowment*, vol. 3, pp. 1647–1648, 2010.
[29] D. M. Blei and J. D. Lafferty, "Dynamic topic models," in *Proc. 23rd ACM Int. Conf. Mach. Learn.*, 2006, pp. 113–120.
[30] G. Huang, Y. Ma, X. Liu, Y. Luo, X. Lu, and B. Blake, "Assisting navigation and complementary composition of complex service mashups," *IEEE Trans. Serv. Comput.*.
[31] H. Elmeleegy, A. Ivan, R. Akkiraju, and R. Goodwin, "Mashup advisor: A recommendation tool for mashup development," in *Proc. IEEE Int. Conf. Web Serv.*, 2008, pp. 337–344.
[32] C. Zhou, H. Chen, Z. Peng, Y. Ni, and G. Xie, "A semantic Bayesian network for web mashup network construction," in *Proc. IEEE/ACM Int. Conf. Green Comput. Commun. Int. Conf. Cyber, Phys. Soc. Comput.*, 2010, pp. 645–652.
[33] L. Chen, Y. Wang, Q. Yu, Z. Zheng, and J. Wu, "WT-LDA: User tagging augmented LDA for web service clustering," in *Proc. Int. Conf. Serv.-Oriented Comput.*, 2013, pp. 162–176.
[34] C. Ye and H. Jacobsen, "Whitening SOA testing via event exposure," *IEEE Trans. Softw. Eng.*, vol. 39, no. 10, pp. 1444–1465, Oct. 2013.
[35] J. Cao, Z. Wu, Y. Wang, and Y. Zhuang, "Hybrid collaborative filtering algorithm for bidirectional web service recommendation," *Knowl. Inf. Syst.*, vol. 36, pp. 607–627, 2012.
[36] Z. Zheng, Y. Zhang and M. R. Lyu, "Cloudrank: A qos-driven component ranking framework for cloud computing," in *Proc. 29th Int. Symp. Reliable Distrib. Syst.*, 2010, pp. 184–193.
[37] W. Ahmed, Y. Wu, and W. Zheng, "Response time based optimal web service selection," *IEEE Trans. Parallel Distrib. Syst.*.
[38] A. Klein, F. Ishikawa, and S. Honiden, "Towards network-aware service composition in the cloud," in *Proc. 21st Int. Conf. World Wide Web*, 2012, pp. 959–968.
[39] X. Wang, J. Zhu, and Y. Shen, "Network-aware QoS prediction for service composition using geolocation," *IEEE Trans. Serv. Comput.*.
[40] W. Lo, J. Yin, S. Deng, Y. Li, and Z. Wu, "Collaborative web service QoS prediction with location-based regularization," in *Proc. IEEE 19th Int. Conf. Web Serv.*, 2012, pp. 464–471.
[41] X. Chen, Z. Zheng, Q. Yu, and M. Lyu, "Web service recommendation via exploiting location and QoS information," accepted by *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 7, pp. 1913–1924, Jul. 2014.
[42] J. Zhang, W. Tan, J. Alexander, I. Foster, and R. Madduri, "Recommend-as-you-go: A novel approach supporting services-oriented scientific workflow reuse," in *Proc. IEEE Int. Conf. Serv. Comput.*, 2011, pp. 48–55.
[43] W. Tan, J. Zhang, R. Madduri, I. Foster, D. De Roure, and C. Goble, "ServiceMap: Providing Map and GPS assistance to service composition in bioinformatics," in *Proc. IEEE Int. Conf. Serv. Comput.*, 2011, pp. 632–639.
[44] B. Xia, Y. Fan, C. Wu, K. Huang, W. Tan, J. Zhang, and B. Bai, "A Domain-aware service recommendation method for service composition," in *Proc. 21st Int. Conf. Web Serv.*, 2014, pp. 439–446.
[45] J. Wang, H. Chen, and Y. Zhang, "Mining user behavior pattern in mashup community," in *Proc. IEEE Int. Conf. Inf. Reuse Integr.*, 2009, pp. 126–131.
[46] C. Zhang, X. Zhao, and J. Wang, "An item-targeted user similarity method for data service recommendation," in *Proc. IEEE 16th Int. Enterprise Distrib. Object Comput. Conf. Workshops*, 2012, pp. 172–178.
[47] A. Maaradji, H. Hacid, J. Daigremont, and N. Crespi, "Towards a social network based approach for services composition," in *Proc. IEEE Int. Conf. Commun.*, 2010, pp. 1–5.

Yang Zhong received the BS degree in control theory and application in 2012 from Tsinghua University, China. He is currently working toward the PhD degree at the Department of Automation, Tsinghua University. His research interests include services computing, service recommendation and big data.

Yushun Fan received the PhD degree in control theory and application from Tsinghua University, China, in 1990. He is currently a professor with the Department of Automation, Director of the System Integration Institute, and Director of the Networking Manufacturing Laboratory, Tsinghua University. From September 1993 to 1995, he was a visiting scientist, supported by Alexander von Humboldt Stiftung, with the Fraunhofer Institute for Production System and Design Technology (FHG/IPK), Germany. He has authored 10 books in enterprise modeling, workflow technology, intelligent agent, object-oriented complex system analysis, and computer integrated manufacturing. He has published more than 300 research papers in journals and conferences. His research interests include enterprise modeling methods and optimization analysis, business process reengineering, workflow management, system integration, object-oriented technologies and flexible software systems, petri nets modeling and analysis, and workshop management and control.

Keman Huang received the BS degree in automation and another BS degree in economics from Tsinghua University, China, in 2014 and 2009, respectively and the PhD degree in control theory and application. He is currently an assistant professor with the School of Computer Science and Technology, Tianjin University, China. His research interests include services computing, web service composition, social network analysis, data mining, and service recommendation. He is a member of the ACM and the IEEE.

Wei Tan received the BS and PhD degrees from the Department of Automation, Tsinghua University, China in 2002 and 2008, respectively. He is currently a research staff member with the IBM T. J. Watson Research Center, NY. From 2008 to 2010, he was a researcher at the Computation Institute, University of Chicago and Argonne National Laboratory. At that time, he was the technical lead of the caBIG workflow system. His research interests include NoSQL, big data, cloud computing, service-oriented architecture, business and scientific workflows, and petri nets. He has published more than 50 journal and conference papers, and a monograph "Business and Scientific Workflows: A Web Service-Oriented Approach" (272 pages, Wiley-IEEE Press). He received the Best Paper Award from the IEEE International Conference on Services Computing (2011), the Pacesetter Award from the Argonne National Laboratory (2010), and caBIG Teamwork Award from the National Institute of Health (2008). He is an associate editor of the *IEEE Transactions on Automation, Science and Engineering*. He was in the program committees of many conferences and has co-chaired several workshops. He is a member of the ACM and a senior member of the IEEE.

Jia Zhang received the MS and BS degrees in computer science from Nanjing University, China and the PhD degree in computer science from the University of Illinois at Chicago. She is currently an associate professor at the Department of Electrical and Computer Engineering, Carnegie Mellon University. Her recent research interests center on service oriented computing, with a focus on collaborative scientific workflows, Internet of Things, cloud computing, and big data management. She has co-authored one textbook titled "Services Computing" and has published more than 130 refereed journal papers, book chapters, and conference papers. She is currently an associate editor of the *IEEE Transactions on Services Computing (TSC)* and of *International Journal of Web Services Research (JWSR)*, and editor-in-chief of *International Journal of Services Computing (IJSC)*. She is a member of the IEEE.