Service Recommendation for Mashup Creation based on Time-aware Collaborative Domain Regression

Bing Bai, Yushun Fan*
Tsinghua National Laboratory for Information Science and Technology
Department of Automation
Tsinghua University
Beijing 10084, China
bb13@mails.tsinghua.edu.cn
fanyus@tsinghua.edu.cn

Keman Huang
School of Computer Science and Technology
Tianjin University
Tianjin 300072, China
keman.huang@tju.edu.cn

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

Bofei Xia, Shuhui Chen
Tsinghua National Laboratory for Information Science and Technology
Department of Automation
Tsinghua University
Beijing 10084, China
xiabf11@mails.tsinghua.edu.cn
chensh13@mails.tsinghua.edu.cn

***Abstract*—Mashup has emerged as a promising way to compose web APIs and create value-added compositions. The increasing of APIs demands more accurate recommendation algorithms. However, service domain evolution, mashup-side cold-start and information evaporation are somehow overlooked by existing work. In this paper, by extending the *collaborative topic regression* (*CTR*) model, the procedure of service selection is modeled with a generative process, and the mashup-side cold-start problem that cannot be dealt with by naïve *CTR* is resolved. By learning the maximum a posteriori estimates of the whole generative process, both content information and historical usage are taken into consideration to extract service domains, thus the service domains can evolve with the evaluation of historical usage pattern. Meanwhile, information evaporation is also considered by giving time-related confidence levels to historical usage to track the evolution of service ecosystem. Experiments on the real-world ProgrammableWeb[1] data set show that compared with the state-of-the-art methods, our approach gains a 6.8% improvement in terms of recommendation accuracy.**

Keywords—Service recommendation; Mashup Creation; Time-aware; Topic modeling; LDA; Collaborative Topic Regression

I. INTRODUCTION

With the wide adoption of the *Service-Oriented Architecture* (*SOA*) and *Cloud Computing*, web services, usually coming with the form of web APIs, grow rapidly both in the quantity and diversity [1]. To create value-added software efficiently and effectively, reusing services to composite mashups becomes a promising solution [2]. Facing the large amount of services, service recommendation has been proved to be an expedient method when creating a mashup [3, 5, 10].

Some existing service recommendation approaches use content matching methods like key-words search [6]. As semantic-based methods become widely used, novel approaches are proposed and significantly improve the recommendation's performance [7, 8]. It extracts content-based features from WSDL documents and user queries to perform semantic-based recommendation approaches. Additionally, the evolution of service ecosystem has been put attention to, and time-aware recommendations improve the performance once more [3]. Apart from these function-related recommendation methods, other researches focus on non-functional properties of services, i.e., Quality of Service (QoS) [4]. Customized recommendations are proposed by predicting characteristics such as reliability, availability, response time, and so on [9, 17, 18].

However, the following issues, which are closely related to high quality service recommendation, are overlooked by most of the existing recommendation systems:

1) **Service Domain Evolution**: Service domains emerge gradually during the long-term collaboration and competition in the service ecosystem [10]. Therefore, the service domains shouldn't be simplified as the semantic topics, which are constructed based on the service/mashup descriptions given by the providers/developers. It's necessary to take historical usage into consideration when extracting service domains.

2) **Mashup-side Cold-start**: For service recommendation, the service usage of a mashup query is unknown to the recommendation system and the only information about the query is the descriptions in the form of content [3, 10]. As a result, service recommendation suffers from the mashup-side (i.e. user-side in other recommendation problems) cold-start problem, which makes it difficult to use algorithms like standard collaborative filtering or matrix factorization directly to make services recommendation.

3) **Information Evaporation**: Service ecosystem is constantly evolving over time, driven by new services emerging and existing services perishing [5], which leads to the phenomenon of information evaporation. The older usage history is less likely to be relevant for determining.

To the best of our knowledge, there isn't an existing recommendation algorithm that overcomes all the aforementioned limitations. Hence, the *Time-aware Collaborative Domain Regression* (*Time-aware CDR*) model is proposed in this paper by extending the *Collaborative Topic Regression* (*CTR*) model, to solve these issues and make better service recommendation.

Collaborative Topic Regression (*CTR*) has been successfully applied to article recommendation. It combines traditional collaborative filtering with topic modeling [20]. However, when modeling the users (i.e. mashups in service recommendation), *CTR* only uses the historical usage. Thus, naïve *CTR* can't deal with the mashup-side cold-start problem. We overcome this limitation by modifying the

*Communication Author
[1] www.ProgrammableWeb.com

978-1-4673-7272-5/15 $31.00 © 2015 IEEE
DOI 10.1109/ICWS.2015.37

generative process and modeling the mashups with both content information and historical usage. By learning the maximum a posteriori estimates of the whole generative process, both content information and historical usage are taken into consideration when extracting service domains, thus the service domains can evolve with the historical usage information enriching. Meanwhile, by given time-related confidence levels to historical usage, information evaporation is taken into consideration and the evolution of service ecosystem is tracked.

Therefore, the main contributions of this paper are summarized as follows:

1) By extending the *CTR* model, the mashup-side cold-start problem is overcome, and service domains are extracted with both content information and historical usage.

2) Information Evaporation is taken into consideration and the *Time-aware CDR* model is proposed.

3) Comprehensive experiments on the real-world data set from ProgrammableWeb.com show that the overall MAP of our *Time-aware CDR* is 6.8% better compared with the state-of-the-art method.

The rest of this paper is organized as follows. Section II formulates the recommendation problem formally and gives a brief introduction to *CTR*. Section III gives an overview of our recommendation model. Section IV describes the training and recommending algorithms in details. Section V reports the experimental results. Section VI summarizes the related work and Section VII offers some concluding remarks.

II. Background

In this section, the background of *Time-aware CDR* is introduced. Firstly, required definitions are presented and the problem of service recommendation for mashup creation is formulated, and then a brief introduction to *CTR* is presented.

A. Problem Definition

Definition 2.1: Service ecosystem. In this paper, the set $SE = (M, MD, MT, S, SD, R)$ is used to model the service ecosystem, in which M represents the set of mashups, S represents the set of services, and $|M| = I$, $|S| = J$ respectively. $MD_i = \{w_{i1}, w_{i2}, \cdots, w_{in_i}\}$ is the set of n_i words used to describe mashup $i \in M$, and similarly, $SD_j = \{w_{j1}, w_{j2}, \cdots, w_{jn_j}\}$ is the set of n_j words used to describe service $j \in S$. $MT = \{t_1, t_2, \cdots, t_I\}$ records the release time of mashups, and $R = (r_{ij})_{i=1,j=1}^{I \times J}$ records the historical usage between mashups and services, which means that $r_{ij} = 1$ when service j is involved by mashup i, and $r_{ij} = 0$ when not.

Definition 2.2: Service domain. Services with similar functionalities form service domains, and most services focus on not only one functionality [10]. As a result, a service can belong to several domains, and connections between the services and different domains have discrepant weights.

In this paper, assuming there are K domains, a non-negative vector $\theta_j = \{\theta_{j1}, \theta_{j2}, \cdots, \theta_{jK}\}$ is used to describe the domain proportions of service $j \in S$. For a specific service $j \in S$, there exists that $\sum_{k=1}^{K} \theta_{jk} = 1$. The same definition can also be used to describe mashups, i.e., $\eta_i = \{\eta_{i1}, \eta_{i2}, \cdots, \eta_{iK}\}$ describes the domain proportions of mashup $i \in M$ and $\sum_{k=1}^{K} \eta_{ik} = 1$.

With the definitions above, we can formally define the following problem:

Problem: Service recommendation for mashup creation. Given the service ecosystem *SE*, for a new mashup query $l \in Q$ described by $QD_l = \{w_{l1}, w_{l2}, \cdots, w_{ln_l}\}$, the goal is to recommend services in the form of a ranked list RL, in which a service with a higher rank has a higher probability to be adopted by query *l*.

B. Collaborative Topic Regression

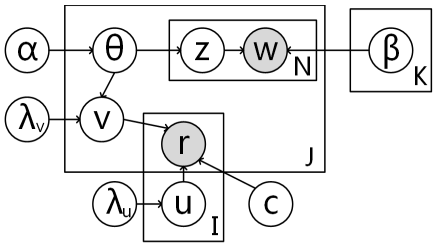

Fig. 1. The graphical model for the CTR model

Collaborative Topic Regression (*CTR*) model combines traditional collaborative filtering with topic modeling. It is proposed to recommend scientific articles to users [20]. Figure 1 shows the graphical model of *CTR*. Assume there are K topics $\mathrm{B} = \beta_{1:K}$, each of which is a distribution over a fixed vocabulary (assume there are W words in the vocabulary). The generative process is as follows,

1. For each user (mashup) *i*, draw user (mashup) latent vector $\eta_i \sim \mathrm{N}(0, \lambda_u^{-1} I_K)$.
2. For each item (service) *j*,
   a) Draw topic proportions $\theta_j \sim \mathrm{Dirichlet}(\alpha)$.
   b) Draw item latent offset $\varepsilon_j \sim \mathrm{N}(0, \lambda_v^{-1} I_K)$ and set the item latent vector as $v_j = \theta_j + \varepsilon_j$.
   c) For each word w_{jn},
      i. Draw topic assignment $z_{jn} \sim \mathrm{Mult}(\theta_j)$.
      ii. Draw word $w_{jn} \sim \mathrm{Mult}(\beta_{z_{jn}})$.
3. For each user-item (mashup-service) pair (*i*, *j*), draw the rating

$$r_{ij} \sim \mathrm{N}(\eta_i^T v_j, c_{ij}^{-1}),$$

where c_{ij} is the confidence parameter for r_{ij}, $c_{ij} = a$

when $r_{ij}=1$ and $c_{ij}=b$ when $r_{ij}=0$, $a>b>0$.

As presented above, *CTR* only uses historical usage to model users (mashups in service recommendation). Thus, it can't deal with the mashup-side cold-start problem. Meanwhile, the confidence parameters are fixed for all usage records.

III. Overview of Methodology

In this section, the *Time-aware Collaborative Domain Regression* model is illustrated in details, and some supplementary explanation is presented.

A. Generative Process

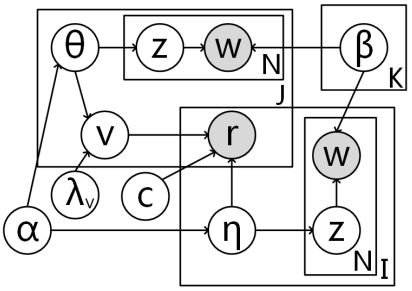

Fig. 2. The graphical model for the Time-aware CDR model

The graphical model of our *Time-aware CDR* is shown in Figure 2. Again assume that there are *K* domains $\mathrm{B}=\beta_{1:K}$. The generative process of *Time-aware CDR* is as follows,

1. For each mashup *i*,
   a) Draw topic proportions $\eta_i \sim \mathrm{Dirichlet}(\alpha)$.
   b) For each word w_{in},
      i. Draw topic assignment $z_{in} \sim \mathrm{Mult}(\eta_i)$.
      ii. Draw word $w_{in} \sim \mathrm{Mult}(\beta_{z_{in}})$
2. For each service *j*,
   a) Draw topic proportions $\theta_j \sim \mathrm{Dirichlet}(\alpha)$.
   b) Draw service latent offset $\varepsilon_j \sim \mathrm{N}(0, \lambda_v^{-1} I_K)$ and set the service latent vector as $v_j = \theta_j + \varepsilon_j$.
   c) For each word w_{jn},
      i. Draw topic assignment $z_{jn} \sim \mathrm{Mult}(\theta_j)$.
      ii. Draw word $w_{jn} \sim \mathrm{Mult}(\beta_{z_{jn}})$.
3. For each mashup-service pair (*i*, *j*), draw the rating
   $$r_{ij} \sim \mathrm{N}(\eta_i^T v_j, c_{ij}^{-1}),$$
   where c_{ij} is the confidence parameter for r_{ij}, and
   $$c_{ij} = \lambda_\eta \frac{e^{-\lambda_t (t_{current} - t_i)}}{\frac{1}{M}\sum_i e^{-\lambda_t (t_{current} - t_i)}} \quad (1)$$

B. Supplementary Explanation

Firstly, as shown above, the procedure of service selection is modeled with a generative process. By learning the maximum a posteriori estimates of the whole generative process, both content information and historical usage are taken into consideration when extracting service domains. Thus ***service domains can evolve with the historical usage enriching***.

Secondly, compared with the *CTR* model, *Time-aware CDR* models the mashups (in the same position of users in *CTR*) with both content information and historical usage, which helps to overcome the ***Mashup-side Cold-start Problem***.

Lastly, for the confidence parameter, the larger the c_{ij} is, the more trusted the r_{ij} is. λ_η is the regularization parameter, and λ_t is the time decay constant. To separate the effect of λ_η and λ_t, a normalizing factor is introduced so that changing λ_t will not change the average of c_{ij}. A usage record of earlier-released mashup has a smaller confidence parameter, thus the phenomenon of ***Information Evaporation*** is taken into account and the evolution of service ecosystem is tracked.

IV. Parameter Learning and Recommending

After the above introduction to the methodology, the problem remained is to learn the maximum a posteriori estimates from the given information of service ecosystem. In this section, how the optimal parameters are obtained, as well as how to use the well-trained model to make recommendation are explained in details.

A. Parameter Learning

An EM-style algorithm is developed to learn the maximum a posterior estimates. With a constant omitted and α set to 1, the complete log likelihood of $v_{1:J}$, $\eta_{1:I}$, $\theta_{1:J}$ given λ_v, c_{ij}, *R* and B is as follows:

$$L = -\frac{\lambda_v}{2}\sum_j (v_j - \theta_j)^T (v_j - \theta_j) - \sum_{i,j} \frac{c_{ij}}{2}(r_{ij} - \eta_i^T v_j)^2 + \sum_i \sum_n \log\left(\sum_k \eta_{ik} \beta_{k,w_{in}}\right) + \sum_j \sum_n \log\left(\sum_k \theta_{jk} \beta_{k,w_{jn}}\right) \quad (2)$$

Optimizing (2) directly can be rather intractable, so we first optimize this function by coordinate ascent with B fixed, which means v_j, η_i and θ_j are optimized iteratively.

First, optimal $v_{1:J}$ is obtained with $\eta_{1:I}$ and $\theta_{1:J}$ fixed. Gradient-based methods can be adopted, and there is also an analytical solution, namely

$$v_j \leftarrow \left(\mathrm{H} C_j \mathrm{H}^T + \lambda_v I_K\right)^{-1} \left(\mathrm{H} C_j R_j + \lambda_v \theta_j\right) \quad (3)$$

where $\mathrm{H} = (\eta_i)_{i=1}^I$, C_j is the diagonal matrix with c_{ij} as the diagonal elements, and $R_j = (r_{ij})_{i=1}^I$.

Second, optimal $\eta_{1:I}$ are obtained with $v_{1:J}$ and $\theta_{1:J}$ fixed. However, maximizing the formula of $\log\left(\sum(\cdot)\right)$ is difficult because it couples too many parameters. So we first define $q(z_{in} = k) = \phi_{ink}$ as that in [19, 20], then separate the items

with η_i, apply Jensen's inequality and finally get

$$\begin{aligned} L(\eta_i) \geq & -\sum_j \frac{c_{ij}}{2}\left(r_{ij} - \eta_i^T v_j\right)^2 \\ & + \sum_n \sum_k \phi_{ink}\left(\log\left(\eta_{ik}\beta_{k,w_{in}}\right) - \log\left(\phi_{ink}\right)\right) \\ & = L_1\left(\eta_i, \phi_i\right) \end{aligned} \quad (4)$$

where $\phi_i = \left(\phi_{ink}\right)_{n=1,k=1}^{N\times K}$ and there exists $\sum_k \phi_{ink} = 1$. $L_1\left(\eta_i, \phi_i\right)$ gives a tight lower bound of $L(\eta_i)$, and by optimizing $L_1\left(\eta_i, \phi_i\right)$, optimal parameter of η_i can be obtained. The *Sequential Maximal Optimization* (*SMO*, see [21] for details) algorithm is adopted to optimize η_i. On the other hand, the optimal ϕ_{ink} satisfies

$$\phi_{ink} \propto \eta_{ik}\beta_{k,w_{in}} \quad (5)$$

Third, optimal $\theta_{1:J}$ is obtained with $v_{1:J}$ and $\eta_{1:I}$ fixed. Similar with the steps when optimizing η_i, we define $q\left(z_{jn} = k\right) = \phi_{jnk}$, apply Jensen's inequality and get

$$\begin{aligned} L\left(\theta_j\right) \geq & -\frac{\lambda_v}{2}\left(v_j - \theta_j\right)^T\left(v_j - \theta_j\right) \\ & + \sum_n \sum_k \phi_{jnk}\left(\log\left(\theta_{jk}\beta_{k,w_{jn}}\right) - \log\left(\phi_{jnk}\right)\right) \\ & = L_2\left(\theta_j, \phi_j\right) \end{aligned} \quad (6)$$

Then *SMO* is applied to optimize θ_j and the optimal ϕ_{jnk} satisfies

$$\phi_{jnk} \propto \theta_{jk}\beta_{k,w_{jn}} \quad (7)$$

At last, with the optimal $v_{1:J}$, $\eta_{1:I}$, $\phi_{1:I}$, $\theta_{1:J}$ and $\phi_{1:J}$, B is updated with the similar M-step as in *LDA* [19],

$$\beta_{kw} \propto \sum_i \sum_n \phi_{ink} 1\left[w_{in} = w\right] + \sum_j \sum_n \phi_{jnk} 1\left[w_{jn} = w\right] \quad (8)$$

From the above equations, we can see how historical usage influences the optimal η_i and θ_j, thus influences the ϕ_i and ϕ_j, and finally influences the B (the distributions of each service domains over the fixed vocabulary).

The summarization of parameter learning stage is as follows.

Algorithm 1: Parameter learning for Time-aware CDR

Input: *R*, *MD*, *SD*, *MT*, λ_v, λ_η, λ_ϕ, *K*

Output: Optimal parameters of $v_{1:J}$, $\eta_{1:I}$, $\theta_{1:J}$ and B

Procedure:

01. Calculate c_{ij} by Eq. (1)
02. Initialize $\phi_{ink} = 1/K$ and $\eta_{ik} = 1/K$ for all *i, n* and *k*
03. Initialize $\phi_{jnk} = 1/K$ and $\theta_{jk} = 1/K$ for all *j, n* and *k*
04. Initialize $\beta_{kw} = 1/W$ for all words in the vocabulary
05. **Repeat**
06. Optimize v_j for all *j* by Eq. (3)
07. Optimize η_i for all *i* by maximizing $L_1\left(\eta_i, \phi_i\right)$
08. Update and normalize ϕ_i for all *i* by Eq. (5)
09. Optimize θ_j for all *j* by maximizing $L_2\left(\theta_j, \phi_j\right)$
10. Update and normalize ϕ_j for all *j* by Eq. (7)
11. Update and normalize B by Eq. (8)
12. **Until** convergence

When adopting in practice, the resulting topics and proportions of *LDA* can be used to initialize η_i, θ_j and B to fasten the parameter learning stage and avoid converging to odd local maximum.

B. Recommending

When a mashup query arrives, the *Time-aware CDR* can be adopted to make recommendation with the (local) optimal parameters *V* and B learnt. First, domain proportions ζ_l are extracted by solving the following optimization problem

$$L\left(\zeta_l\right) = \sum_n \log\left(\sum_k \zeta_{lk}\beta_{k,w_{ln}}\right) \quad (9)$$

Again we define $q\left(z_{ln} = k\right) = \phi_{lnk}$ and apply Jensen's inequality and get

$$\begin{aligned} L\left(\zeta_l\right) \geq & \sum_n \sum_k \phi_{lnk}\left(\log\left(\zeta_{lk}\beta_{k,w_{ln}}\right) - \log\left(\phi_{lnk}\right)\right) \\ & = L_3\left(\zeta_l, \phi_l\right) \end{aligned} \quad (10)$$

Similar with the steps in the parameter learning stage, ζ_l and ϕ_l are optimized iteratively. *SMO* is applied to optimize ζ_l, and the optimal ϕ_{lnk} satisfies

$$\phi_{lnk} \propto \zeta_{lk}\beta_{k,w_{ln}} \quad (11)$$

After the domain proportions of mashup query are extracted, the recommendation rating of services to the query can be calculated by

$$r_{lj}^* = \zeta_l^T v_j \quad (12)$$

finally the ranked list RL of services in descending order is calculated and output to the mashup developer for reference.

The summarization of recommending stage is as follows.

Algorithm 2: Recommending with Time-aware CDR

Input: *QD*, $v_{1:J}$, B

Output: *RL*

Procedure:

01. Initialize $\phi_{lnk} = 1/K$ and $\zeta_{lk} = 1/K$ for all *l, n* and *k*
02. **Repeat**
03. Optimize ζ_l for all *l* by maximizing $L_3\left(\zeta_l, \phi_l\right)$
04. Update and normalize ϕ_l for all *l* by Eq. (11)
05. **Until** convergence
06. Calculate the recommendation ratings for all *l* and *j* by Eq. (12)
07. Calculate the ranked list in descending order and get *RL*

V. EXPERIMENTS

Data from ProgrammableWeb.com is used to test our methodology. In this section, information of the data set is presented first, then evaluation metrics, baseline methods and results are also introduced.

A. Data Set

To test our methodology, we crawled the data of services and mashups from September 2005 to July 2014 from ProgrammableWeb.com. Information of the data set is presented in the following table. When testing the algorithms, services that have never been used are removed.

TABLE I. DATA SET ON PROGRAMMABLEWEB.COM

Total # of services	10306
Total # of services used in at least one mashup	1267
Total # of mashups	5855
Average # of services in one mashup	2.12

B. Evaluation Metric

To evaluate the performance of our recommendation system, the widely accepted metric, *Mean Average Precision @ top N* (MAP@N), is used. And it is defined as follows:

$$\text{MAP@N} = \frac{1}{|Q|}\sum_{q\in Q}\frac{1}{N_q}\sum_{s\in S_q}\left(\frac{n(q,s)}{r(q,s)}\cdot I(q,s)\right) \quad (13)$$

Where $|Q|$ denotes the number of queries, $N_q = \min\left(N,|S_q|\right)$ and S_q is the set of actually used services in query q. For each service $s \in S_q$, $r(q,s)$ denotes the ranking position of s in the recommendation list and $n(q,s)$ denotes the ranking position of s in the list that only contains the services in both S_q and the recommendation list, and $I(q,s)=1$ when $r(q,s)\le N$ and $I(q,s)=0$ when not. Besides, the overall MAP means MAP@J.

MAP@N summarizes effectiveness of precision, recall, and ranking in a single metric [11]. A higher MAP@N means a better recommendation performance that the algorithm can offer.

C. Baseline Methods

Five baseline recommendation methods are chosen as comparisons.

1) ***Baseline Method 1: Service-description-based Matching (SDM)***

For *Service-description-based Matching*, services whose descriptions are semantically similar with the query for a new mashup are recommended. First the *LDA* model is applied to extract the topic proportions of description to services and user query1, and then cosine similarities between the topic proportions are calculated. The recommendation rating is defined as follows:

$$SDM(q,s_j) = sim(q,s_j) \quad (14)$$

where $sim(q,s_j)$ calculates the cosine similarity of the topic proportions of mashup query and service *j*.

2) ***Baseline Method 2: Mashup-description-based Collaborative Filtering (MDCF)***

Mashup-description-based Collaborative Filtering is a traditional neighborhood-based collaborative filtering method. It recommends services under the assumption that similar mashups have a higher possibility to use a certain service at the same time. The recommendation rating is defined as follows:

$$MDCF(q,s_j) = \frac{\sum_{m_i\in U(N,q)} sim(q,m_i)\,r_{ij}}{\sum_{m_i\in U(N,q)} sim(q,m_i)} \quad (15)$$

where $U(N,q)$ contains the Top N similar mashups with q, and $sim(q,m_i)$ calculates the cosine similarity of the topic proportions of mashup query and mashup *i*.

3) ***Baseline Method 3: LDA + Matrix Factorization (MF)***

For this method, first *LDA* is applied to extract the topic proportions of mashups' description and user query, then the matrix R is factorized by solving the optimization problem

$$\min_{v_j}\left(\sum_i\left(\sum_j\left(r_{ij}-\eta_i^T v_j\right)^2+\lambda\left\|v_j\right\|^2\right)\right) \quad (16)$$

and the recommendation rating is defined as follows

$$MF(q,s_j) = \zeta^T v_j \quad (17)$$

Note that the η_i and ζ here are the resulting proportions of *LDA*.

4) ***Baseline Method 4: Time-Aware Service Recommendation (TSR)***

The method proposed in [3] is also tested. *TSR* promotes the performance of service recommendation by predicting service activity in the near future to make recommendation.

5) ***Baseline Method 5: Service Recommendation based on Collaborative Domain Regression (CDR)***

Time decay constant λ_t is set to 0 to test the performance of *CDR*, which means for this baseline method, information evaporation is ignored.

D. Experiment Result

The experiment results are presented in this part. Firstly the settings of experiments are listed, then the performance of our method and baseline methods is presented, finally the effects of some parameters are studied.

1) Experiment settings

In our experiment, the time granularity is set to one month. For a certain month, data before it is used as the training set and data within the month is used as the testing set. The performance of our method and baseline methods is tested from August 2012 to July 2014, and 24 MAP@N results are obtained. The final metric is the weighted average number of the 24 MAP@N results with the number of mashups in the testing sets as weights. In other words, we test the methods in two years period month by month and get the weighted average of the above results as the final metric to tell which method outperforms the other ones.

The parameters are set as follows. When adopting the *LDA* model, we set topic number K = 40, α = 1.25 and β = 0.01. For *MDCF*, N is set to 250. For *MF*, the regularizing parameter λ is set to 0.01. When testing *TSR*, we follow the parameters in [3].

When adopting *Time-aware CDR*, the parameters are set

as follows. λ_v is set to 10, λ_η is set to 1, and λ_t is set to 0.08. When testing the Baseline Method 5, the parameters are the same with *Time-aware CDR* except that λ_t is set to 0.

2) Performance comparison

Figure 3 illustrates the MAP@N of different recommendation algorithm on different size of recommendation list. *MDCF* outperforms *SDM*, for *MDCF* takes historical usage into consideration. Unlike the neighborhood-based method *MDCF*, *MF* uses all the training set to model services and can capture the totality of weak signals encompassed in all of a mashup's historical usage [22], thus improves the performance. *TSR* takes temporal information into consideration and comes up with a slightly better performance than *MF*. Among the baseline methods, our *CDR* models the whole process of service selection and extracts the service domains with both content information and historical usage, and gets the highest MAP.

After taking information evaporation into *CDR*, *Time-aware CDR* promotes the accuracy of service recommendation significantly again, and outperforms all the baseline methods.

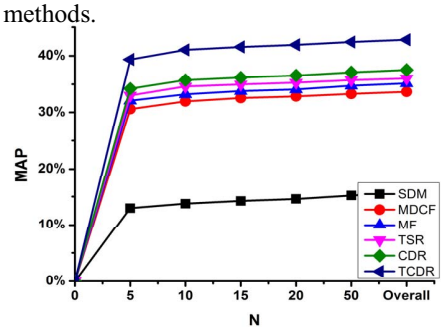

Fig. 3. MAP@N of the methods on different sizes of recommendation list

The detailed performance of different algorithms is summarized in Table II. Comparing the performance of *MF* and *CDR*, it can be inferred that it brings a promotion of about 2.4% by taking service domain evaluation into consideration. And by comparing the performance of *Time-aware CDR* and *CDR*, it can be inferred that taking information evaporation into consideration can promote the performance with more than 5% again.

TABLE II. PERFORMANCE COMPARISON OF DIFFERENT ALGORITHMS ON MAP@20 AND OVERALL MAP

Recommendation Algorithm	MAP@20	Overall MAP
SDM	14.59%	15.80%
MDCF	32.85%	33.68%
MF	34.08%	35.18%
TSR	35.31%	36.04%
CDR	36.57%	37.54%
Time-aware CDR	**41.91%**	**42.81%**

3) Effect of time decay constant λ_t

λ_t is a key parameter in our algorithm, which refers to how rapidly the service ecosystem is evolving. For a more rapidly evolving service ecosystem, the information is being evaporated faster, and λ_t should be larger. A proper λ_t can enhance the performance of the algorithm.

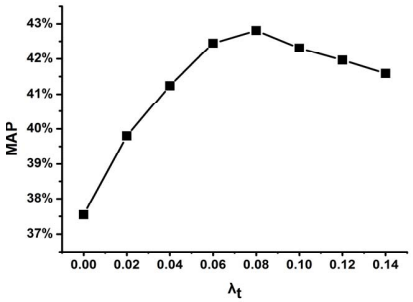

Fig. 4. The effect of time decay constant λ_t

Figure 4 shows the overall MAP values with different time decay constant. As presented, 0.08 is the best choice for time decay constant λ_t with our tested dataset. Which means, for ProgrammableWeb service ecosystem, information is being evaporated at a speed of 7.69% per month, or 61.71% per year.

4) Effect of λ_η and λ_v

Here the effects of parameters λ_η and λ_v are studied. These two parameters influence the performance of *Time-aware CDR* mainly in the following two ways. Firstly, when extracting service domains, λ_η and λ_v balance the influence of historical usage and content information (recall the Eq. (4), Eq. (6) and the definition of c_{ij}), a larger λ_η or λ_v means the historical usage influence the service domains more. Secondly, when modeling the services (i.e., optimizing v_j for all j), λ_η / λ_v balances the influence of historical usage and content information (recall the Eq. (3) and the definition of c_{ij}), if λ_η / λ_v is large, *Time-aware CDR* behaves more like matrix factorization with information evaporation considered, while if λ_η / λ_v is too small, *Time-aware CDR* behaves more like *SDM*.

Figure 5 shows the performance when changing λ_η and λ_v respectively. It's interesting to note that increasing λ_η or decreasing λ_v lead to a similar trend. As a conclusion, the ratio of λ_η and λ_v effect the performance significantly (i.e., the second type of influence when changing λ_η and λ_v as analyzed above), and the performance of recommendation is more sensitive to the change of λ_η.

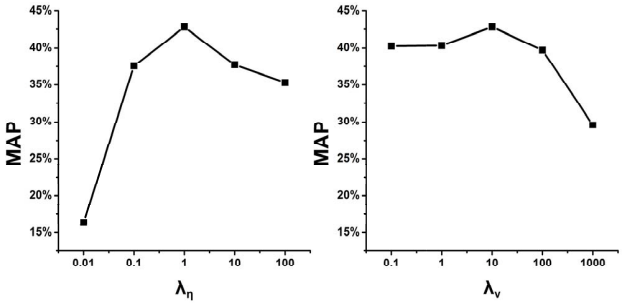

Fig. 5. The effect of λ_η and λ_v

E. A Glimpse into the Evolution of Service Ecosystem

Driven by new services emerging and existing services perishing, the service ecosystem is evolving constantly. By taking the information evaporation into account, the evolution of service ecosystem is tracked.

Google Maps is a popular service of the "Mapping" category and has been widely used in many mashups of different functional request. According to the information of ProgrammableWeb.com, *Google Maps* was released at Dec. 5th in 2005, and has been used in 2043 mashups by the end of July 2014. We study the resulting model of *Google Maps* given by *Time-aware CDR* in July 2014 and July 2012, to see what changes have happened to *Google Maps* over the two years.

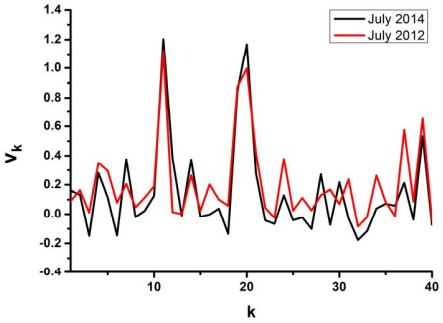

Fig. 6. The model of *Google Maps* in different time

Figure 6 shows the latent vectors of *Google Maps* in the different years (Note that for a certain service j, $\sum_k v_{jk}$ doesn't need to be 1 and v_{jk} can be smaller than 0).From the figure, we can witness the change of the model of *Google Maps* over the two years (Strictly speaking, the domains evolve in the meantime. Here we ignore this effect to focus on the evolution of *Google Maps*). Table III reports some of the details of typical changes for reference.

TABLE III. WEIGHTS CHANGE OF GOOGLE MAPS IN DIFFERENT TIME

Index of domain	v_k in 2014	v_k in 2012	Keywords to describe the domain
11	1.202	1.122	neighborhood, street, nearby, geolocation, location
24	0.130	0.378	restaurant, cook, food, menu, dish
28	0.272	0.131	science, research, biology, experiment
30	0.219	0.070	news, review, latest, newspaper, article
39	0.532	0.653	weather, forecast, wind, temperature

From the table above, we can get that v_{11} and v_{39} remain large over the two years (although v_{39} drops a bit), for that developers still tend to use *Google Maps* to offer geolocation-related or weather-related functionalities. However, v_{24} decreases a lot over the two years, for that specialized services like *Ordr.in* (released in Sep. 2011) and *Zomato* (released in Sep. 2011) become more and more popular recently. Additionally, *Google Maps* has become more popular in more domains besides geolocation and weather, since the collaborations with "science" and "news" have been strengthened.

As a conclusion, taking information evaporation into consideration can enable the model to track the evolution of service ecosystem, and thus improve the quality of recommendation.

VI. RELATED WORK

A. Service Recommendation

With the rising importance of service sector and the widely adaption of the Service-oriented Architecture (SOA), more and more enterprises and organizations are turning to publish their capability as services over the internet [4], and as a result, web service recommendation and selection become a fundamental issue.

Early work often use keyword-based search engines combining with information from *Web Services Description Language* (*WSDL*), or *Universal Description, Discovery and Integration* (*UDDI*) [6]. However, keyword-based methods can only deliver limited performance, and to protect business interests of service providers and to prevent information leakage, the implementation details of services are usually invisible to service consumers now [12]. Under this situation, *LDA* model is widely used in recent work to characterize the latent topics between services and user queries, and description content takes the role of WSDL documents [10, 3]. On the other hand, network-based methods explore the interactions among services, service developers and users. [13] constructed an evolution network model from historical usage of the services and used a link prediction approach to perform recommendation. [14] came up with a new method to mine latent negative association rules to promote recommendation performance.

On the other hand, a lot of research work centers on Qos-based web service recommendation. Neighborhood-based collaborative filtering algorithms were the major approaches to predict unknown Qos values, including user-based [15] and hybrid approaches [16]. In addition, location information was also taken into consideration in [17]. However, Qos information is hard to get, which limits the adoption of Qos-based recommendation.

Nowadays, it is well-accepted that service ecosystem is rapidly evolving over time. [3] presented a recommendation method by predicting service activity in the near future. [18] proposed a time-aware and data sparsity tolerant approach for Qos prediction with considering temporal information and employing the random walk algorithm.

B. Collaborative Topic Modeling

Collaborative Topic Regression is first presented by [20] to recommend scientific articles for readers. It combined traditional collaborative filtering with topic modeling. [23] combined collaborative topic regression with probabilistic matrix factorization of social networks to recommend tags. [24] took the limitation of human attention into account, and proposed a *Limited-attention CTR* to predict voting on news items.

In this paper, we go a step further by extending the *Collaborative Topic Regression* model to make service

recommendation. We model the procedure of service selection with a generative process and extract the service domains with both content information and usage history. Moreover, information evaporation is taken into consideration and the performance of service recommendation is promoted again.

VII. CONCLUSIONS

Service recommendation has been proved as a promising way to solve the overloaded information of the large amount of services. However, service domain evolution, mashup-side cold-start problem and information evaporation are overlooked by most of the existing recommendation systems.

To overcome the aforementioned restriction, in this paper, by extending the *CTR* model, we proposed the framework which considers the procedure of service selection as a generative process. By modeling mashup with both content information and historical usage, the mashup-side cold-start problem is settled. By solving the maximum a posteriori problem, the latent domains and the domain proportions are leant with both content information and historical usage. Meanwhile, information evaporation is also taken into consideration by giving different confidence levels to historical usage. Experiments on a real-world data set show that comparing the state-of-the-art, our method can gain a 6.8%~9.1% improvement.

In the future, we plan to take release time of services and the competition between cold-start services and services that have been used into consideration, and refine the algorithm proposed in this paper to deal with the service-side cold-start problem, for that service ecosystem is evolving constantly and new services emerge every month.

ACKNOWLEDGMENT

This work is partially supported by the National Natural Science Foundation of China (No.61174169), the National High Tech. R&G Program (No.2012AA02A613) and the Specialized Research Fund for the Doctoral Program of Higher Education (No. 20120002110034).

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] 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 Proceedings of IEEE International Conference on Services Computing (SCC), 2011, pp.48-55.

[3] Y. Zhong, Y. Fan, K. Huang, W. Tan and J. Zhang, "Time-aware service recommendation for mashup creation in an evolving service ecosystem,", in Proceedings of IEEE International Conference on Web Services (ICWS), 2014, pp. 25-32.

[4] Z. Wang, J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," IEEE Journal on Selected Areas in Communications, Vol.14, pp. 1228-1234, 1996.

[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, Vol. 11, pp. 906-920, 2014.

[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] T. Qiu, L. Li and P. Lin, " Web Service Discovery with UDDI Based on Semantic Similarity of Service Properties," in Proceedings of the 3rd International Conference on Semantics, Knowledge and Grid (SKG), 2007, pp. 454-457.

[8] D. Kourtesis and I. Paraskakis, "Combining SAWSDL, OWL-DL and UDDI for Semantically Enhanced Web Service Discovery," in Proceedings of the 5th European Semantic Web Conference (ESWC), 2008, pp. 614-628.

[9] Z. Zheng, H. Ma, M. R. Lyu and I. King, "Qos-aware web service recommendation by collaborative filtering," IEEE Transactions on Services Computing, Vol. 4, pp. 140-152, 2011.

[10] B. Xia, Y. Fan, C. Wu, K. Huang, W. Tan, et al, "A Domain-aware Service Recommendation Method for Service Composition," in Proceedings of IEEE International Conference on Web Services (ICWS), 2014, pp. 439-446.

[11] Z. Zhao, X. Wang, Q. Xiang, et al, "Large-scale music tag recommendation with explicit multiple attributes," in Proceedings of the International Conference on Multimedia. ACM, 2010, pp. 401-410.

[12] C. Ye and H. A. Jacobsen, "Whitening SOA Testing via Event Exposure," IEEE Transactions on Software Engineering, Vol. 39, pp. 1444-1465, 2013.

[13] K. Huang, Y. Fan, W. Tan and X. Li, "Service Recommendation in an Evolving Ecosystem: A Link Prediction Approach," in Proceedings, of IEEE International Conference on Web Services (ICWS), 2013, pp. 507-514.

[14] Y. Ni, Y. Fan, K. Huang, et al, "Negative-Connection-Aware Tag-Based Association Mining and Service Recommendation" in Proceedings of International Conference on Service Oriented Computing (ICSOC), 2014, pp. 419-428.

[15] L. Shao, J. Zhang, Y. Wei, et al, "Personalized qos prediction forweb services via collaborative filtering," in Proceedings, of IEEE International Conference on Web Services (ICWS), 2007, pp. 439-446.

[16] 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 (ICWS), 2009, pp. 437-444.

[17] M. Tang, Y. Jiang, J. Liu, et al, "Location-aware Collaborative Filtering for QoS-based Service Recommendation," in Proceedings, of IEEE International Conference on Web Services (ICWS), 2012, pp. 202-209.

[18] Y. Hu, Q. Peng, X. Hu, "A Time-Aware and Data Sparsity Tolerant Approach for Web Service Recommendation," in Proceedings, of IEEE International Conference on Web Services (ICWS), 2014, pp. 33-40.

[19] D. M. Blei, A. Y. Ng and M. I. Jordan, "Latent Dirichlet Allocation," the Journal of Machine Learning Research, Vol. 3, pp. 993-1022, 2003.

[20] C. Wang, D. M. Blei, "Collaborative topic modeling for recommending scientific articles" in Proceedings, of ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), 2011, pp. 448-456.

[21] J. Platt, “Fast training of support vector machines using sequential minimal optimization,” Advances in kernel methods—support vector learning, 1999, 3.

[22] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in Proceedings, of ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), 2008, pp. 426-434.

[23] H. Wang, B. Chen, W. Li, "Collaborative topic regression with social regularization for tag recommendation," in Proceedings, of the Twenty-Third international joint conference on Artificial Intelligence, 2013, pp. 2719-2725.

[24] J.H. Kang, K. Lerman, "LA-CTR: A limited attention collaborative topic regression for social media," arXiv preprint arXiv:1311.1247, 2013.