2017 IEEE 14th International Conference on Services Computing

Service Recommendation from the Evolution of Composition Patterns

Zhenfeng Gao, Yushun Fan*, Cheng Wu
Tsinghua National Laboratory for Information Science and Technology
Department of Automation
Tsinghua University
Beijing 100084, China
gzf13@mails.tsinghua.edu.cn
{*fanyus, wuc*}*@tsinghua.edu.cn*

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

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

***Abstract*—A service ecosystem, consisting of various kinds of services and mashups, evolves over time. Existing works on the evolution of service systems focus on either evaluating the impacts of services' changes on the usage of services and the stability of the whole ecosystem, or discovering co-occurrences between services, but fail to disclose any knowledge about the evolution of service composition patterns. Based on our previous work of SeCo-LDA, through scrutinizing the dependencies between different service co-occurrence topics, this paper reveals the latent service composition trends in a service ecosystem. We derive topic dependencies and describe it as a directed topic evolution graph, where four topic evolution patterns are identified. A novel methodology, named Dependency Compensated Service Co-occurrence LDA (DC-SeCo-LDA), is developed to calculate the directed dependencies between different topics, build the topic evolution graph. The evolution trend of service composition could be disclosed by the graph intuitively, and dependency compensation could be adopted to improve the performance when making service recommendation. Experiments on ProgrammableWeb.com show that DC-SeCo-LDA can recommend service composition more effectively, i.e., 2% better in terms of Mean Average Precision compared with baseline approaches.**

***Keywords*-SeCo-LDA; topic evolution graph; service composition recommendation**

I. Introduction

As Service-Oriented Architecture (SOA) and Cloud Computing are widely adopted, the amount of published web services on the Internet has been rapidly growing [1]. By reusing existing services (i.e., APIs), software developers are able to quickly create service compositions (i.e., mashups) to meet complex function needs and offer additional business values [2] . However, users' demands on mashups could vary over time. As a consequence, developers' preference of combining certain domains of services may gradually change, making the trend of services composition patterns keep on evolving. The evolution of service system makes it challenging for developers to comprehend the trend of service composition patterns, or to manually select proper candidates to meet specific functional requirements. Such challenges call for new techniques to help developers gain a better understanding of the evolution characteristics of service ecosystem, and to help select services more effectively and intelligently.

In the research of service ecosystem evolution, most works analyze the impact of a single service's change, and try to deal with the version problem in order to maintain system stability [3]–[8]. Although evolution characteristics can be mined, these works study the evolution problem from the perspective of individual services or service dependencies, which could not expose any information about the trend of service composition, e.g., what kind of service composition patterns are becoming more popular recently, how they merge or branch into new ones, and so on.

Few works have considered mining evolution characteristics of service ecosystem from the perspective of service composition patterns. In our previous work [9], we introduced a concept of "service co-occurrence topic," which demonstrates the existence of latent service composition patterns described with the distribution over services in the ecosystem. For example, a topic on peoples social life is described by the distribution on services like ***Twitter***, ***Facebook***, ***Yahoo Blog*** and so on. In this paper, we further study the service ecosystem's evolution problem from the sight of service co-occurrence topics, and define the directed evolutional relationship between topics as "topic dependency." Some existing probabilistic topic models on text mining could reveal semantic topics [10]–[12], which are described with distributions over text words. Their results could lead to calculating topic similarity, which is an undirected correlation between topics. In contrast, we argue that "topic dependencies" are directed, reflecting the latent evolution trend of service composition patterns. For example, a topic on multimedia information and a topic on

*Corresponding Author

2474-2473/17 $31.00 © 2017 IEEE
DOI 10.1109/SCC.2017.21

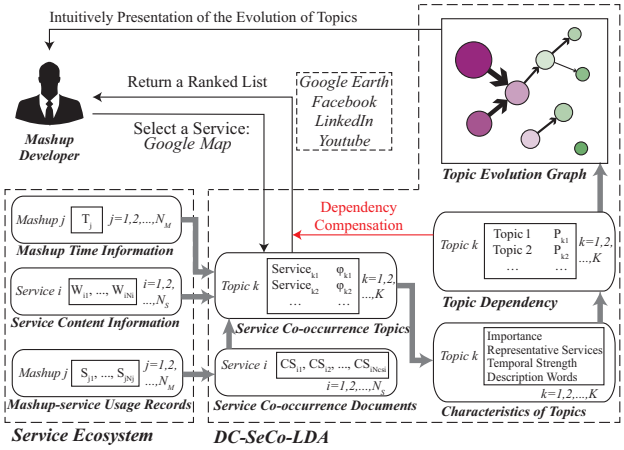

Figure 1. Framework of DC-SeCo-LDA. Based on SeCo-LDA, DC-SeCo-LDA discovers topic dependencies to make compensation for service recommendation, and builds topic evolution graph to illustrate service composition trend intuitively.

shopping guidance might gradually merge into a new topic on multimedia-based shopping recommendation.

To the best of our knowledge, no algorithm exists to mine such dependencies of service co-occurrence topics. In this paper, we propose a novel model, "Dependency Compensated Service Co-occurrence LDA" (DC-SeCo-LDA), by extending SeCo-LDA [9] to identify the evolution characteristic of service ecosystem at the topic level. The framework of our approach is shown in Fig. 1. We design an algorithm to calculate the topic dependencies based on *topic-service* distribution. Topic dependencies, which is directed, reveal the latent trend of service composition patterns in the service ecosystem. Inspired by [8], [13], we propose four "topic evolution patterns" (i.e., *Merge*, *Branch*,*Co-occur* and *Arise*) and construct a topic evolution graph, which provides information more concisely and intuitively and can help people understand the evolution of the topics from systemic aspect. An example of topic evolution graph is presented in Fig. 1. What's more, when making service recommendation, we design an algorithm to use topic dependencies to compensate the topics' distribution over services in order to improve the recommendation performance. The main contributions of this paper are summarized as follows:

1) A concept of "topic dependency" is created to describe the evolution relationship between different service co-occurrence topics. Meanwhile, four "topic evolution patterns" are proposed, i.e., *Merge*, *Branch*, *Co-occur* and *Arise*, to classify different forms of topic evolution trends.

2) A novel method DC-SeCo-LDA is developed to discover topic dependencies, build topic evolution graph and make dependency compensation to improve the recommendation performance. Topic evolution graph presents information concisely and intuitively, revealing latent trend of service composition patterns in a service ecosystem. Dependency compensation is adopted when making service recommendation to improve the results.

3) Comprehensive experiments over real-world data set ProgrammableWeb.com are conducted. Results show that DC-SeCo-LDA can discover significant topic dependencies to build evolution graph and achieve a higher MAP value than baselines when recommending related services for a selected one, approximately 2% better than the second-best baseline approach.

The rest of this paper is organized as follows. Section II provides the definition of background and DC-SeCo-LDA model. Parameter learning, calculation of topic dependencies and approaches to build topic evolution graph and make dependency compensation are illustrated in Section III. Section IV shows experimental results on a real-world data set from ProgrammableWeb.com, including discovering topic dependencies, building topic evolution graph and making recommendation. Section V summarizes the related work and then Section VI concludes the paper.

II. DC-SeCo-LDA Model

The key idea of DC-SeCo-LDA is to discover directed topic dependencies, build topic evolution graph, and promote the performance of recommendation. Analogous to SeCo-LDA [9], we first construct service co-occurrence documents and apply a probabilistic topic model. We make some changes when calculating temporal strength, which we will illustrate in Section III. Then we design algorithms to discover topic dependencies, build topic evolution graph and make compensation when recommending to improve the results.

In this section, firstly, we will describe the definitions about the background, then introduce the SeCo-LDA part in our model. At last, we will pose the three problems that DC-SeCo-LDA deals with.

A. Background

Definition 1: Topology of service ecosystem. The topology of a service ecosystem containing mashup-service citation records is modeled as an undirected graph $G = (M \cup S, E)$ in which: $M = \{m_1, m_2, ..., m_{N_M}\}$ is the set of mashups and $S = \{s_1, s_2, ..., s_{N_S}\}$ is the set of services; N_M is the number of mashups and N_S is the number of services; $E \subseteq M \times S$ is the historical usage records between mashups and services, i.e., if mashup j invokes service i, $E(j, i) = 1$.

Definition 2: Service Co-occurrence Topics. Service co-occurrence topics [9] describe latent composition patterns between services and are represented by the distribution over services in the ecosystem. For example, topic k is described by $\{\phi_{jk}, j = 1, ..., N_S\}$, in which ϕ_{jk} describes the impact of service j on topic k when making service composition, and $\sum_j \phi_{jk} = 1$. In this paper, we consider these characteristics of each service co-occurrence topic:

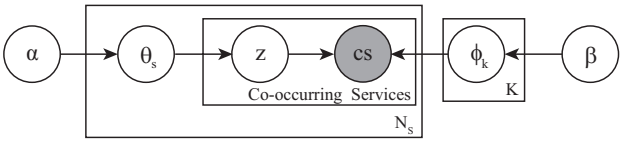

Figure 2. Graphic Model of SeCo-LDA part.

1) Topic Importance: different service co-occurrence topics reveal different composition patterns and have different importances in the service ecosystem.

2) Topic Representative Services: services that are the most popular in a service co-occurrence topic, from which we could tell the meaning of the composition pattern.

3) Topic Temporal Strength: an distribution over time, reflecting a service co-occurrence topic's lifecycle and revealing the change of its popularity.

4) Topic Time Expectation: time expectation could be calculated with topics temporal strength. With this, we could roughly distinguish new topics with old ones.

B. SeCo-LDA part

In DC-SeCo-LDA, the process of constructing service co-occurrence documents and applying a probabilistic topic model on the corpora are similar with those in our previous work [9]. Here is a brief overview.

1) Service Co-occurrence Documents

For each service $s_i \in S$, using its co-occurring services as word tokens, we represent s_i as a "bag of service co-occurrences" $d_i = \{\#(sc_j) = c_{i,j} | j \in S\}$ in which: $\#(sc_j) = c_{i,j}$ means service co-occurrence on s_j appears $c_{i,j}$ times in the description document of s_i, or service i and j are composited together for $c_{i,j}$ times by mashups.

2) Service Co-occurrence LDA

Assume that there were K different service co-occurrence topics expressed over N_S unique services in service ecosystem. We set $z = 1 : K$ as the topic indicator variable. The topic distribution for service co-occurrence documents (i.e., $P(z|d)$) can be represented by a $N_S \times K$ matrix $\mathbf{\Theta}$, each row of which, $\boldsymbol{\theta}_i$, being a $K-$dimensional multinomial distribution for document s_i. The distribution over services for topics (i.e., $P(s|z)$) can be represented by a $K \times N_S$ matrix $\mathbf{\Phi}$, each column of which, $\boldsymbol{\phi}_z$, being a N_S-dimensional multinomial distribution for topic z.

We use symmetric Dirichlet priors for $\mathbf{\Theta}$ and $\mathbf{\Phi}$ with hyper-parameters α and β [14], respectively. Graphical model is shown in Fig. 2. The generative process of SeCo-LDA part is described as follows. For each service i,

1) Draw topic proportions $\theta_i \sim Dirichlet(\alpha)$.
2) For each co-occurring service sc_{in} of service i,
   a) Draw topic assignment $z_{in} \sim Multi(\theta_i)$.
   b) Draw co-occurring service $sc_{in} \sim Multi(\phi_{z_{in}})$.

C. Problem Definition

1) Discovering Topic Dependencies

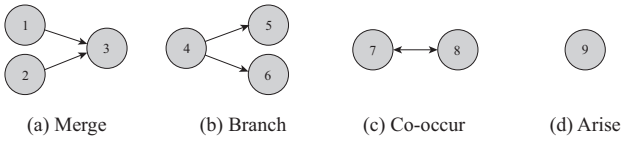

Figure 3. Examples of the Four Topic Evolution Patterns.

Definition 3: Topic Dependency, Child & Parent Topic. We regard the probability of occurrence of one topic conditioned on another topic as the topic dependency between them. Specifically, we represent $\Pr(k_j \rightarrow k_i | k_i)$ as the topic dependency from topic j to topic i, or the degree of topic j's evolutional influence on topic i. We define topic i as topic j's child topic, and topic j as topic i's parent topic. We use a $K \times K$ matrix $\boldsymbol{TD}$ to describe all the topic dependencies, where $td(i,j) = \Pr(k_j \rightarrow k_i | k_i)$.

Different from topic similarity [10], [11] describing undirected relationship, topic dependency refers to directed relationship between two topics, revealing the evolutional influence of one topic on another. The key to discover topic dependencies is to obtain the topics distributions over services, while traditional word-based topic model [12] could not achieve this. With SeCo-LDA part in our model, we can discover topic dependencies, whose algorithm will be introduced in Section III.

2) Building Topic Evolution Graph

Understanding topic dependency, we could tell whether a topic is generated or derived from another. Accumulating related topic dependencies, a topic evolution graph could be created. As the example shown in Fig.1, topic evolution graph contains information about topics importance and time information, as well as the dependencies between them. It provides developers an overall view of the evolution trend of service composition patterns.

Definition 4: Topic Evolution Graph. A topic evolution graph is modeled as a directed graph $G_E = (T, \boldsymbol{TD})$ in which: $T = \{t_1, t_2, ..., t_K\}$ is the nodes set of service co-occurrence topics. Each node represents a topic; $\boldsymbol{TD}$ is the edges set of directed topic dependencies, i.e., if $td(i,j) > 0$, there exists an directed edge from topic j pointing to i.

Definition 5: Topic Evolution Patterns (Functionally). Evolution patterns could be discovered as shown in Fig. 3.

Merge: two or more topics merge into one new topic.

Branch: one topic branches into two or more new topics.

Co-occur: if a topics child topic is its parent topic at the same time, we call this a co-occur pattern, that is, the two topics have influence on each other. Notice that their time expectations must be close.

Arise: if a topic just appeared in the topic evolution graph without any parents, we call this an arise pattern, that is, this topic just arises without other old topics contributing to it.

3) Recommending for Service Composition

We consider such a situation as in [9], [15]: assuming a mashup-developer selects the first API, and wants to

find other APIs to create a new mashup. He might not know exactly what kind of mashup he wants to make, and just hope to find related services to make significant compositions. Referring to the selected service as s_l, the result of recommendation is represented as a ranked list $R_l = \{s_{l1}, s_{l2}, ...\}$.

III. Learn the DC-SeCo-LDA Model

In this section, we first introduce the parameter learning and discovery of service co-occurrence topics along with their characteristics. Then We will present algorithms to calculate topic dependencies, construct topic evolution graph and make dependency compensation for recommendation.

A. Parameter Learning

Like [14], we use the collapsed Gibbs sampling to make inferences with the SeCo-LDA part in our model. Posterior expectation of θ_{ik} and ϕ_{jk} is described as:

$$\theta_{ik} = \frac{\#(d = i, z = k) + \alpha_k}{\sum_k \#(d = i, z = k) + \alpha_k} \tag{1}$$

$$\phi_{jk} = \frac{\#(z = k, sc = j) + \beta_j}{\sum_k \#(z = k, sc = j) + \beta_j} \tag{2}$$

Intuitively, $\mathbf{\Theta}$ describes *service-topic* distribution, while $\mathbf{\Phi}$ indicates *topic-service* distribution.

B. Discovery of Service Co-occurrence Topics

For topic importance, we refer to the value of the topics' posterior distribution [9]. For representative services, the ranked value of services based on *topic-service* distribution $\{\phi_{jk}\}$ reflects services' popularity, or impact, on topic k.

When considering topics temporal strength, we use services' impact distribution over time to describe topics' temporal characteristics. Instead of counting the publication timestamps of service as in our previous work on SeCo-LDA [9], in this paper, we take into account the invocation time of services. The key intuition here is that if one service of a topic is invoked at t_0, it makes actual contribution to the temporal strength of the topic at time t_0, no matter when this service is published.

We represent the set of service s's invoked time as $TIN_s = \{t_j^{(s)} | j = 1 : in_s, t_1^{(s)} \leq ... \leq t_{in_s}^{(s)}\}$, in which $t_j^{(s)}$ is the date (*day* as the unit) that service s is invoked by mashups for the j-th time; in_s is the total times that s has been invoked. The accumulated temporal contribution of all services to topic z until time t_0 forms the cumulative distribution function (CDF) of topic z as follows:

$$\Pr(time \leq t_0 | z) \quad = \quad \sum_s \sum_{j, t_j^{(s)} \leq t_0} \frac{\phi_{sz}}{\sum_s in_s \cdot \phi_{sz}} \tag{3}$$

The time expectation of topic z can be calculated as follows:

$$T_z = E_z[time(s)] = \sum_s \sum_j t_j^{(s)} \cdot \frac{\phi_{sz}}{\sum_s in_s \cdot \phi_{sz}} \tag{4}$$

With topics' time expectations, we can distinguish new topics with old ones, which will be helpful when calculating topic dependencies and building topic evolution graph.

C. Discovery of Topic Dependency

The key of DC-SeCo-LDA is to discover dependencies between different service co-occurrence topics. According to Def. 3, the probability of the occurrence of topic j conditioned on topic i is calculated by applying total probability formula as follows :

$$\begin{aligned} & \Pr(k_j \to k_i | k_i) = E_s(\Pr(topic_s = k_j | k_i)) \\ = \quad & \sum_s P(topic_s = k_j | service_s) \cdot P(service_s | k_i) \\ = \quad & \sum_s \theta_{sk_j} \cdot \phi_{sk_i} \end{aligned} \tag{5}$$

An intuitive explanation of the formula above is: whenever randomly drawing a service co-occurrence document of service s which contains topic i, and then generating a co-occurring service in this document(for service s), $\Pr(k_j \to k_i | k_i)$ is the probability that this co-occurring service is assigned with latent service co-occurrence topic j. In other words, $\Pr(k_j \to k_i | k_i)$ reflects the degree of topic j's evolutional influence on topic i.

Nevertheless, on the one hand, topic dependencies whose values are relatively small might be noisy information and hinder us from further analysis. On the other hand, as investigated in [16], we treat topic dependencies of relatively "new" topics on "old" ones as noisy information too. This is reasonable: a new topic may have little influence on an old one. To address these two types of situations, we design a two-step pruning process:

Step 1: Threshold Cutting-off. We set a threshold ξ and remove all the topic dependencies less than ξ in order to wipe out noisy information.

Step 2: Temporal Regularization. After threshold cutting-off, we set a tolerance number of days ε and make temporal regularization. For dependency $td(i, j) > 0$ from topic j to topic i, we prune it only if $T_j - T_i > \varepsilon$. This means, if topic i is newer than topic j, we keep the dependency from j to i; if topic i is older than topic j, we only keep the dependency if the difference between topic i's time expectation and j's is not larger than ε. Here we set ε to tolerate some degree of deviation for calculation.

After pruning, the topic dependencies left in $\boldsymbol{TD}$ represent significant and meaningful directed dependency relationship between topics.

D. Construction of Topic Evolution Graph

According to Def. 4, we design the process of building topic evolution graph as follows:

Step 1: Drawing the Nodes. Each node in the graph stands for a specific service co-occurrence topic. We set the size of these nodes in direct proportion to the topics' importance. Similarly, we use a series of gradually varied colors to express the differences of topics' time expectations.

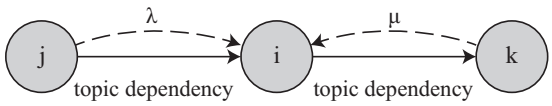

Figure 4. Example for Dependency Compensation for Topic i. We introduce λ to weaken the features inherited from its parent Topic j, and μ to strength the features that Topic i influences its child Topic k.

Step 2: Drawing the Edges. Each edge in the graph stands for a meaningful topic dependency. We draw the directed edges according to the pruned $\boldsymbol{TD}$, whose thickness being in direct proportion to the value of topic dependencies.

Step 3: Integrated Layout. To make the graph more concise and intuitive, we apply Fruchterman-Reingold Algorithm [17] to the topic evolution graph's integrated layout.

E. Dependency Compensation for Recommendation

Intuitively, we prefer the reveled topics to be more independent so that the revealed topics could have their own features and provide a better description of service co-occurrence documents. Consider the worst case, if all the revealed topics have the same distribution, they would make no contribution to service recommendation, that is, the revealed topics were insignificant. On the contrary, if the revealed topics have mutually independent distribution, each topic would have its own features, and these topics could provide a more comprehensive description of service ecosystem, resulting in better performance in recommendation. Actually, due to the existence of dependency relationship, some topics may have partial common features in distribution. This may lead to poor performance on recommendation.

The key idea of dependency compensation is that if we could weaken these inherited features between topics, the performance of service recommendation might be improved. Few have considered this aspect because topic dependency relationship could not be calculated through traditional models. We leverage a topics' parent and child topics to compensate the distribution of this topic, highlighting the characteristics of its own. As shown in Fig. 4, we introduce coefficient λ for parent-side dependency compensation, and coefficient μ for child-side. When considering topic i's distribution, we intend to weaken the features inherited from its parent topics, and strengthen the features on which it affects its child topics (which might be topic i's particular features). So, intuitively, λ might be a negative value, and μ be positive. The algorithm of dependency compensation for recommendation is provided as Algorithm 1. The complexity of the compensation part is $O(K)$.

Note that some of the compensated *topic-service* distributions may dissatisfy $\sum_i {\phi_{ik}}' = 1$. It is caused by the compensation process, which, in a sense, makes adjustment of the topics importance in service ecosystem. Also, there may occur negative value in $\phi'(i)$, indicating the topic's degree of rejecting specific service.

Algorithm 1: Topic Dependency Compensation for Recommendation

Input:
1) $\boldsymbol{\Phi}$ & $\boldsymbol{\Theta}$: *topic-service* and *service-topic* distribution
2) **TD**: topic dependency matrix
3) s_l: a service selected by the developer

Output:
1) R_l: recommended service list

Procedure:
01. Initialize compensated *topic-service* distribution $\boldsymbol{\Phi}' = \boldsymbol{\Phi}$
02. **For** each topic i $(i = 1, 2, ..., K)$
03. **For** topic i's each parent topic j
04. Use parent-side dependency to compensate i's distribution $\phi'(i) = \phi'(i) + \lambda \cdot TD(i, j) \cdot \phi(j)$
05. **End**
06. **For** topic i's each child topic k
07. Use child-side dependency to compensate i's distribution $\phi'(i) = \phi'(i) + \mu \cdot TD(k, i) \cdot \phi(k)$
08. **End**
09. **End**
10. Get the compensated *topic-service* distribution $\boldsymbol{\Phi}'$
11. Calculate s_l's expected co-occurrence with another service s_m $c^*(l, m) = \sum_z \Pr(sc = m|topic = z) \cdot \Pr(topic = z|s_l)$ $= \sum_z \phi'_{m,z} \cdot \theta_{l,z}$
12. Return the recommended list for s_l: $R_l = \{s_{l1}, s_{l2}, ...|c^*(l, s_{l1}) \geq c^*(l, s_{l2}) \geq ...\}$

IV. Experiments

In this section, we will firstly introduce the ProgrammableWeb.com data set on which we apply DC-SeCo-LDA. We will not present detail results of individual topics and their characteristics, which are analogous to those in our previous work [9]. Instead, we will focus on discussing the results of discovering topic dependencies. Afterwards, we will provide results of building topic evolution graph and finding topic evolution patterns. At last, we will present the experiments conducted to compare DC-SeCo-LDA with baselines for recommendation.

A. Data Set

ProgrammableWeb.com has been accumulating a variety of services and mashups since established in 2005 [18], [19]. To evaluate our methodology, we crawled the information of all service APIs and mashups from its inception (September 2005) to August 2016, including their descriptions and mashup-service usage records. Details of the dataset received is presented in Table I.

Table I
Data Set of ProgrammabelWeb.com

Total # of services	13,931
Total # of services that have been used by mashups	1,241
Total # of mashups	6,295
Average # of services in the mashups	2.06

To make comparison with SeCo-LDA, which doesn't consider topic dependency compensation, we set $K = 35$, hyper-parameters $\alpha = 50/K$ and $\beta = 0.01$ as [9].

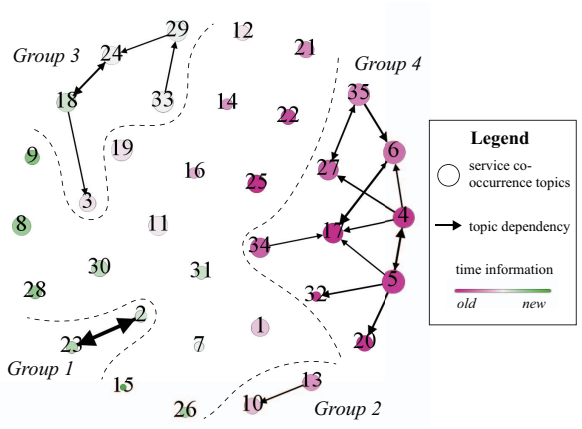

Figure 5. Topic Evolution Graph of ProgrammableWeb.com with $K = 35$.

B. Discovery of Topic Dependency

After obtaining *topic-service* distributions, we calculated the probability of occurrence of any topic conditioned on the other one according to Eqn. 6, and obtained the original topic dependency matrix $\boldsymbol{TD}_0$. To get significant results, a pruning process was performed.

We set the tolerance number of days $\varepsilon = 365$, that is to say, if the difference of time expectations of two topics is less than one year, we keep the dependency from the newer one to the older one if it exists. This setting is reasonable for that there might be deviation when calculating topics time expectation and it allows the existence of "*Co-occur*" evolution pattern.

We let ξ vary from 0.01 to 0.1 with interval 0.01, and counted the number of topic dependencies after the pruning process. According to experiments, empirically, $\xi = 0.07$ is an appropriate choice. If $\xi < 0.07$, there were too many dependencies left after pruning, containing noisy information; if $\xi > 0.07$, the number of topic dependencies may be very small, losing some important information about topic evolutional characteristics. Therefore, we set $\xi = 0.07$ and got 22 significant topic dependencies in service ecosystem.

C. Building Topic Evolution Graph and Finding Topic Evolution Patterns

The topic evolution graph for ProgrammableWeb.com is shown in Fig. 5. Specifically, we use a series of gradient colors from peach to green to express the topics time information. The green nodes are relatively new topics, while the peach ones are old. There're 4 major topic groups.

In **Group 1**, Topic 23 and Topic 2 constitute a "*Co-occur*" pattern. Topic 23 is ***Twilio***-centered and Topic 2 is ***Twilio SMS***-centered. After checking the original data crawled from website, we found that ***Twilio*** and ***Twilio SMS*** are actually the same service on voice and SMS message delivering. So it's obvious that these two topics have strong dependency relationship on each other.

Group 2 contains Topic 10 and Topic 13. Topic 13, whose representative services are ***Google AdSense***, ***Google AdWords***, ***Google Earth***, etc., is a Google service group that mainly focuses on online advertising. To improve the performance, tracking website visitors and evaluating the ads' effects should be necessary. This is what ***Google Analytics Management*** solves. Since it appeared, developers began to invoke it with other Google services to promote the performance of advertisement, which makes it easy to understand Topic 13 has influence on (or generates) Topic 10, a ***Google Analytics Management***-centered Google service group.

Topics in **Group 3** are relatively new, revealing the trend of service composition on social network sharing with text or multimedia information. Topic 33 provides a set of tools that are invoked together often, e.g., ***Yahoo Weather***, ***Blogger***, ***Instapaper***, etc. Developers used them to create mashups. Then, new kinds of social network platforms were created and co-occurred more frequently, illustrated by Topic 29. Afterwards, developers began to combine ***Twitter*** and ***Facebook*** with other services to enrich social network's functionality, generating ***Twitter***-centered (Topics 3 & 24) and ***Facebook***-centered (Topic 18) topics.

Group 4 is the biggest group in the graph, composed of relatively older topics and revealing the trend of location-aware information storing, sharing, searching and recommending. Two core topics of this group are Topics 4 & 5. Topic 4 is a ***Google Maps***-centered topic, and Topic 5 reflects service composition about geographical databases, containing all kinds of information about each site. An obvious "*Merge*" evolution pattern shown here is that Topic 17 is generated from Topics 4, 5, 6 and 34. Topic 6 is about ***YouTube***-centered product advertising. Topic 34, whose representative services are ***del.icio.us*** and ***eBay***, is about the organization of online information. Together with Topics 4 and 5, the four topics generated Topic 17, which is ***Flickr***-centered. ***Flickr*** can organize photos according to interpersonal relationship or content relation, and provide some functionality of social network. In other words, its function is a combination of Topics 4, 5, 6 and 34, which shows an example of service composition about location-aware information sharing and recommending. A representative "*Branch*" evolution pattern is that Topic 5 branched into Topic 17, 32 and 20. Topic 32 reveals service composition about property business based on location information; and Topic 20 reveals web tools that are frequently used together with ***Google Maps***. As shown in Fig. 5, except the four groups identified, other topics are individual ones without significant dependencies with other topics.

From the topic evolution graph, we could in general identify the trend of service composition patterns of ProgrammableWeb.com from 2005 till now. Supported by services like ***Google Maps***, location-aware information storing, sharing, searching and recommending (**Group 4**) was a

popular service composition trend in earlier days. In recent years, however, developers prefer to make service compositions to realize sharing text or multimedia information in social networks (**Group 3**).

D. Dependency Compensation for Recommendation

To demonstrate that dependency compensation is significant, we compared the results of recommendation for service composition using DC-SeCo-LDA model and four baselines. For each service, we recommended the most related N services for it, and compared the results with the original service co-occurrence relationships revealed by the dataset.

1) Evaluation Metric

MAP (Mean Average Precision) [20] was used as evaluation metric in this part:

$$MAP = \frac{1}{|S|}\sum_{i\in S}\frac{1}{N}\sum_{s\in SC_i}\frac{n(s)}{r(s)} \tag{6}$$

where S denotes the set of testing services; N represents the recommended number of services; SC_i denotes the co-occurring services of service i. For each $s \in SC_i$, $r(s)$ refers to the ranking position of s in recommended list and $n(s)$ represents the number of co-occurring services in SC_i that rank higher than or equal to s in recommendation list. In reality, most mashups in data set contain less than five services, so we make N vary from 1 to 8 when doing experiments on recommendation.

MAP is a real number between 0 and 1. The higher MAP indicates a better accuracy of the recommendation method.

2) Determining Coefficients for Dependency Compensation

To determine the sign of parent-side coefficient λ, setting $\mu = 0$, we made λ vary from -4 to 4 with interval 0.5 to make dependency compensation and recommend services. We calculated the mean MAP value with different numberd of N to evaluate the performance. With experiments we concluded that when λ is set a proper negative value, recommendation performance could be improved. The highest mean MAP value 0.4900 is reached when $\lambda = -2$.

Similar experiments show that only when μ is set a small positive value between 0 and 0.5, child-side dependency could contribute to recommendation. The highest mean MAP 0.4809 is reached when $\mu = 0.5$.

In accordance with our conjecture in Section III, the proper sign for λ is negative and positive for μ. It is reasonable for that as in Fig. 4, in order to highlight Topic i's own features, we have to erase the features inheriting from parent Topic j ($\lambda < 0$) and strength the features it affects its child Topic k ($\mu > 0$).

To consider parent-side and child-side dependency compensation at the same time, we made λ vary from -4 to 0 with interval 0.5 and μ vary from 0 to 1 with interval 0.1 to realize dependency compensation and record the mean MAP results. Ultimately, we found that when $\lambda = -1$ and $\mu = 0.5$, the highest mean MAP value ($N = 1...8$) 0.4907 can be achieved, which is better than either of using one direction of topic dependencies to make compensation. Furthermore, the result is approximately 2% better than SeCo-LDA, whose highest mean MAP is 0.4807.

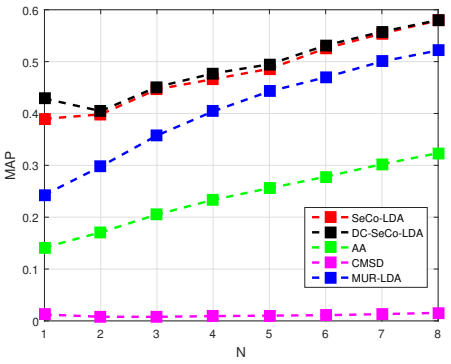

Figure 6. The MAP for DC-SeCo-LDA, SeCo-LDA, AA, CMSD and MUR-LDA with different numbers of N. AA is the convention method in association rule learning. CMSD and MUR-LDA use different information to apply topic model and make recommendation. β is set to 0.5 in AA. In other three probabilistic topic models, we set the number of topics $K = 35$, $\alpha = 50/K$ and $\beta = 0.01$. In DC-SeCo-LDA, we set $\lambda = -1$ and $\mu = 0.5$

3) Baselines

Baseline Method 1: SeCo-LDA. In our previous work [9], for a selected service s_l, we calculate its expected co-occurrence with other services as step 11 in Algorithm 1, using $\mathbf{\Phi}$ instead of $\mathbf{\Phi}'$. In other words, there is no dependency compensation. We give the results of recommendation for service composition as a service list $R'_l = \{s_{l1}, s_{l2}, ...|c^*(l, s_{l1}) \geq c^*(l, s_{l2}) \geq ...\}$.

We choose **Baselines** 2∼4 (**AA**, **CMSD** and **MUR-LDA**) and set their parameters as those in our previous work [9].

4) Results of Recommendation

The MAP results of DC-SeCo-LDA and baselines with different numbers of N are shown in Fig. 6, along with the parameter settings.

SeCo-LDA gets a higher MAP value than AA, CMSD and MUR-LDA with different numbers of N, approximately 5% better than MUR-LDA. DC-SeCo-LDA achieves the highest MAP, 2% better than the second-best baseline SeCo-LDA. Extending SeCo-LDA, DC-SeCo-LDA calculates topic dependencies and makes dependency compensation to topic distributions when recommending services. Dependency compensation weakens evolutional relationship between different service co-occurrence topics so that their particular features are relatively highlighted, thus improving the recommendation performance.

In summary, we can conclude that by making dependency compensation, DC-SeCo-LDA could perform better than the baselines when recommending service composition.

V. Related Work

Most researches on service evolution focus on analyzing the impact of single services changes and how to deal with

the version problem to ensure system stability. Different service changes have been examined in [1] to construct a unifying theoretical framework for controlling the evolution of services. Usage Profile has been used to evaluate service changes' impact [3], [4]. Changes to the WSDL specification of a service interface have been considered in [5] and [6]. An impact analysis model based on service dependency is proposed in [7] to discovery the way in which the change affects the services. Four service evolution patterns (i.e., compatibility, transition, split-map, and merge-map) are proposed in [8] to estimate the impact changes to services. However, few have considered to discover the latent trend of service composition in service system and find out topic dependencies and topic evolution patterns.

Some popular topic models could revel semantic topics considering time information, such as Dynamic Topic Model [10] and Correlated Topic Model [11]. However, with their results, we can only get the undirected semantic similarities of topics, which is quite different from the directed "topic dependency" defined in this paper.

VI. Conclusions

In this paper, we have extended our previous work and proposed a novel approach to make service recommendation from the evolution of composition patterns. The key idea is to define and calculate "topic dependencies" with *topic-service* and *service-topic* distributions. Our work in this paper includes three parts: (1) defining "topic dependencies" and calculating them with DC-SeCo-LDA model; (2) drawing topic evolution graph and finding topic evolution patterns; (3) designing the algorithm to make dependency compensation when recommending to improve the performance.

Topic dependencies would help developers to understand the trend of service composition patterns in a service ecosystem. Comparison with baseline approaches also demonstrate that due to dependency compensation, DC-SeCo-LDA performs 2% better than the second-best baseline approach in terms of MAP when recommending for service composition.

In the future, leveraging information about service co-occurrence topics, we plan to design a framework to deal with service-side cold-start problem.

Acknowledgment

This research has been partially supported by the National Nature Science Foundation of China (No.61673230).

References

[1] V. Andrikopoulos, S. Benbernou, and M. P. Papazoglou, "On the Evolution of Services," *IEEE Transactions on Software Engineering*, vol. 38, no. 3, pp. 609–628, 2012.

[2] X. Liu, Y. Hui, W. Sun, and H. Liang, "Towards Service Composition based on Mashup," in *Proceedings of IEEE World Congress on Services (SERVICES)*, 2007, pp. 332–339.

[3] M. Yamashita, K. Becker, and R. Galante, "Service Evolution Management based on Usage Profile," in *Proceedings of IEEE International Conference on Web Services (ICWS)*, 2011, pp. 746–747.

[4] M. Yamashita, B. Vollino, K. Becker, and R. Galante, "Measuring Change Impact based on Usage Profiles," in *Proceedings of IEEE International Conference on Web Services*, 2012, pp. 226–233.

[5] M. Fokaefs, R. Mikhaiel, N. Tsantalis, E. Stroulia, and A. Lau, "An Empirical Study on Web Service Evolution," in *Proceedings of IEEE International Conference on Web Services*, 2011, pp. 49–56.

[6] D. Romano and M. Pinzger, "Analyzing the Evolution of Web Services Using Fine-grained Changes," in *Proceedings of IEEE International Conference on Web Services*, 2012, pp. 392–399.

[7] S. Wang and M. A. Capretz, "A Dependency Impact Analysis Model for Web Services Evolution," in *Proceedings of International Conference on Web Services*, 2009, pp. 359–365.

[8] S. Wang, W. A. Higashino, M. Hayes, and M. A. M. Capretz, "Service Evolution Patterns," in *Proceedings of IEEE International Conference on Web Services*, 2014, pp. 201–208.

[9] Z. Gao, Y. Fan, C. Wu, W. Tan, J. Zhang, Y. Ni, B. Bai, and S. Chen, "SeCo-LDA: Mining Service Co-occurrence Topics for Recommendation," in *Proceedings of IEEE International Conference on Web Services*, 2016, pp. 25–32.

[10] D. M. Blei and J. D. Lafferty, "Dynamic Topic Models," in *Proceedings of ACM International Conference on Machine Learning (ICML)*, 2006, pp. 113–120.

[11] D. Blei and J. Lafferty, "A Correlated Topic Model of Science," *The Annals of Applied Statistics*, no. 1.1, pp. 17–35, 2007.

[12] 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*, 2014, pp. 25–32.

[13] X. Wang, C. Zhai, and D. Roth, "Understanding Evolution of Research Themes: a Probabilistic Generative Model for Citations," in *Proceedings of the ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2013, pp. 1115–1123.

[14] 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.

[15] B. Tapia, R. Torres, H. Astudillo, and P. Ortega, "Recommending APIs for Mashup Completion Using Association Rules Mined from Real Usage Data," in *Computer Science Society*, 2011, pp. 83–89.

[16] Y. Jo, J. E. Hopcroft, and C. Lagoze, "The web of topics: discovering the topology of topic evolution in a corpus," in *Proceedings of International Conference on World Wide Web*, 2011, pp. 257–266.

[17] T. M. J. Fruchterman and E. M. Reingold, "Graph drawing by force-directed placement," *Software Practice and Experience*, vol. 21, no. 11, pp. 1129–1164, 1991.

[18] A. P. Barros and M. Dumas, "The Rise of Web Service Ecosystems," *IT professional*, no. 5, pp. 31–37, 2006.

[19] E. Al-Masri and Q. H. Mahmoud, "Investigating Web Services on the World Wide Web," in *Proceedings of International Conference on World Wide Web (WWW)*, 2008, pp. 795–804.

[20] Y. Yue, T. Finley, F. Radlinski, and T. Joachims, "A Support Vector Method for Optimizing Average Precision," in *Proceedings of the 30th International Conference on Research and Development in Information Retrieval*, 2007, pp. 271–278.