SeCo-LDA: Mining Service Co-Occurrence Topics for Composition Recommendation

Zhenfeng Gao, Yushun Fan, Cheng Wu, Wei Tan, *Senior Member, IEEE*, Jia Zhang, *Senior Member, IEEE*, Yayu Ni, Bing Bai, and Shuhui Chen

Abstract—Service composition remains an important topic where recommendation is widely recognized as a core mechanism. Existing works on service recommendation typically examine either association rules from mashup-service usage records, or latent topics from service descriptions. This paper moves one step further, by studying latent topic models over service collaboration history. A concept of *service co-occurrence topic* is coined, equipped with a mechanism developed to construct *service co-occurrence documents*. The key idea is to treat each service as a document and its co-occurring services as the bag of words in that document. Four gauges are constructed to measure self-co-occurrence of a specific service. A theoretical approach, Service Co-occurrence LDA (SeCo-LDA), is developed to extract latent service co-occurrence topics, including representative services and words, temporal strength, and services' impact on topics. Such derived knowledge of topics will help to reveal the trend of service composition, understand collaboration behaviors among services and lead to better service recommendation. To verify the effectiveness and efficiency of our approach, experiments on a real-world data set were conducted. Compared with methods of Apriori, content matching based on service description, and LDA using mashup-service usage records, our experiments show that SeCo-LDA can recommend service composition more effectively, i.e., 5% better in terms of Mean Average Precision than baselines.

Index Terms—Topic model, service co-occurrence LDA, automatic service composition, service composition recommendation

1 Introduction

With the wide adoption of Service-Oriented Architecture (SOA) and Cloud Computing, the quantity of published web services on the Internet has been rapidly growing [1]. By reusing existing services (i.e., APIs), software developers are able to create service compositions (i.e., mashups) quickly to meet complex functional needs and offer additional commercial values [2]. However, the overwhelming amount of services makes it a challenge for developers to understand the latent service composition patterns or the current trend of mashup creation within certain domains. They also find it difficult to manually select proper service candidates to make compositions to meet specific functional requirements. Such challenges call for new techniques to help developers gain a better understanding of latent composition patterns of services in a service ecosystem, and to help select services intelligently and automatically.

In the research of automatic service composition, service association learning and probabilistic generative modeling are two commonly used techniques. On the one hand, leveraging service usage records, many existing models [3], [4], [5], [6], [7] are developed based on the Apriori algorithm to mine frequent service patterns and instruct mashup creation. Although association rules can be mined, we can not tell what the functionality of a specific rule (i.e., service association pattern) is and what exactly the impact of each containing service is. Furthermore, the popularity of these patterns may change over time, while these conventional approaches could not reveal such time-dependent property. On the other hand, leveraging word descriptions of services and mashups, recommendation with probabilistic generative models have become increasingly popular in recent years [8], [9], revealing latent semantic functional topics. However, one topic covers services of similar functionality, and each topic is usually described with a distribution over words. Thus, the identified topics could not reveal information about the composition patterns between services. Through these models, one can only compute the textual similarity between services and topics, which cannot represent what we call services' "impacts" on a topic.

Few researchers have considered using latent models such as topic model to mine frequent patterns from service usage records. In contrast to their focuses on identifying semantic or functional topics [8], [9], in this paper, we coin a new concept of "service co-occurrence topic" to reveal the latent patterns about service composition. One service co-occurrence topic contains services with complementary functionality, and each topic is described with a distribution over services in a service ecosystem, instead of distributions over words in semantic functional topics. During the evolution of the service ecosystem, service co-occurrence topics emerge gradually.

- *Z. Gao, Y. Fan, C. Wu, Y. Ni, B. Bai, and S. Chen are with the Department of Automation, Tsinghua University, Beijing 100084, China. E-mail: {gzf13, nyy07, bb13, chensh13}@mails.tsinghua.edu.cn, {fanyus, wuc}@tsinghua.edu.cn.*
- *W. Tan is with the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, NY 10598. E-mail: wtan@us.ibm.com.*
- *J. Zhang is with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Moffett Field, CA 15213. E-mail: jia.zhang@sv.cmu.edu.*

Manuscript received 13 Feb. 2017; revised 26 Feb. 2018; accepted 24 Mar. 2018. Date of publication 30 Mar. 2018; date of current version 7 June 2019. (Corresponding author: Zhenfeng Gao.)
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TSC.2018.2821149

1939-1374 © 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: SOUTHERN METHODIST UNIV. Downloaded on October 18,2024 at 15:56:48 UTC from IEEE Xplore. Restrictions apply.

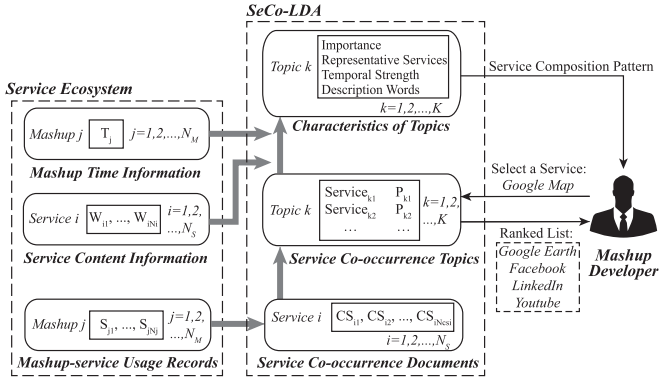


Fig. 1. Framework of Service Recommendation based on SeCo-LDA. Leveraging mashup-service usage records, we construct *service co-occurrence documents*, based on which SeCo-LDA is applied to reveal service co-occurrence topics. When mashup developer selects a service, then the system will suggest a ranked list of subsequent services together with latent service composition patterns, taking into consideration of the timestamps when related mashups were created and the content information of services. Explanations about the notations are presented in Section II in detail.

Services composited by mashups repeatedly reveal latent compositions patterns, and are more likely to belong to the same service co-occurrence topic, whose characteristics include representative services and words, temporal strength and services' impacts. For example, a service co-occurrence topic about location-aware social network contains location-aware APIs (e.g., *Google Maps* with impact factor *0.5*) and social network APIs (e.g., *Facebook* with impact factor *0.3*), as well as its description including words like *location*, *social*, *network* and *site*. Such a service co-occurrence topic first appeared in 2009 and became more popular around 2012. The more popular a service co-occurrence topic is at a time, the more frequently mashups composing corresponding services would appear around that time period. Such knowledge will provide developers with an overall view of service composition patterns in the service ecosystem. Information about the impact of services over their composition will also help to recommend services more effectively.

To the best of our knowledge, no algorithm exists to mine the latent service co-occurrence topics and their characteristics. Based on our previous work [10], in this paper we propose a novel model "Service Co-occurrence LDA" (SeCo-LDA) by extending conventional Latent Dirichlet Allocation (LDA) to identify latent service composition patterns in the service ecosystem. We create a term "service co-occurrences" to address the sparsity problem caused by directly using mashup-service usage records. In our model, each service is treated as a document with its co-occurring services as words, that is, we use co-occurring services to describe a specific service. Basically, the service co-occurrences of a specific service consist of two parts: 1) co-occurrences with other services in the service ecosystem, and 2) its self-co-occurrence. For the co-occurrences with other services, for example, if service s_1 has co-occurred with s_2 once and with s_3 twice among all the mashups, the first part of s_1's *service co-occurrence document* would be $d_{1\_other} = \{s_2, s_3, s_3\}$. For the self-co-occurrence, based on our previous work [10], we establish four different gauges to compute the d_{1_self} in this paper: Zero Self-co-occurrence (ZS), Self-co-occurrence based on Mashup Usage Records (SMUR), Self-co-occurrence based on Service Co-occurrence Pair Number (SSCPN), and Self-co-occurrence based on Average Service Co-occurrence Pair Number (SASCPN). Combining d_{1_other} and d_{1_self} we can get the service co-occurrence document for s_1 as $d_1 = \{d_{1\_other}, d_{1\_self}\}$.

Our rationale is that the service co-occurrence information is more objective than textual service description. Every mashup in a service ecosystem created by mashup developers (i.e., service users) invokes APIs to fulfill specific demands. Such usage scenarios are more convincing comparing to their textual definitions provided by service developers, which may suffer from problems like language difference or improper description. We directly model the generation of the co-occurring services to discover the latent "service co-occurrence topics." Afterwards, we calculate the characteristics of the topics and make recommendation for service composition. The overall framework of our approach is shown in Fig. 1. The main contributions are summarized as follows:

1) A concept of latent "service co-occurrence topic" is created, along with a mechanism of constructing service co-occurrence documents. Specially, we provide four gauges to calculate the self-co-occurrence of a specific service.
2) A novel method SeCo-LDA is developed to discover service co-occurrence topics and their characteristics (i.e., representative services and words, temporal strength and impacts of services). *Representative services and words* intuitively illustrate the functionality of a topic. *Temporal strength* describes the evolution of popularity of topics, which further reveals the trend of service composition. *Service impact* reveals the importance of services in a topic, shedding lights on more accurate depiction of collaboration behaviors among services and ultimately leading to better recommendation.
3) Potential application scenarios of SeCo-LDA are provided. Comprehensive experiments over real-world data set from ProgrammableWeb show that SeCo-LDA reaches a 5 percent higher Mean Average Precision (MAP) value than baselines, when recommending related services for a selected service.

The rest of this paper is organized as follows. Section II provides the definition of the problems and our SeCo-LDA model. Parameter learning and the calculation of the characteristics are described in Section III. Section IV shows experimental results on a real-world data set from ProgrammableWeb.com. Section V summarizes the related work and then Section VI concludes the paper.

2 The SeCo-LDA Topic Model

In contrast to conventional content-based topic models, our model treats each service as a document, and its bag of co-occurring services as the bag of words in that document. We model these reconstructed *service co-occurrence documents* with a probabilistic generative model, which is analogous to the application of a probabilistic topic model to discover topics in text documents. However, a core difference is that the discovered topics with our model would be characterized by a multinomial distribution over services,

Authorized licensed use limited to: SOUTHERN METHODIST UNIV. Downloaded on October 18,2024 at 15:56:48 UTC from IEEE Xplore. Restrictions apply.

rather than over words, revealing the latent composition patterns of services. Thus we can discover the impacts of services on a topic according to the topic's distribution over services, which can be revealed by our model while it cannot be directly revealed by conventional association learning algorithms or content-based topic models.

In this section, we first lay out some background definitions and formulate the problems. Afterwards, we describe how to construct the service co-occurrence document of each service, including the co-occurrence with other services and its self-co-occurrences. Finally, we will introduce the topic model based on them.

2.1 Background

We provide definitions about a service ecosystem to expound the application scope of our fundamental theory. In general, our solution is suitable for any service ecosystem that satisfies Definitions 1, 2, 3. As shown in Fig. 1, the topology of a service ecosystem contains mashup-service usage records, based on which we construct service co-occurrence documents and apply SeCo-LDA. Service content and mashup time information will be exploited to discover the characteristics of latent service co-occurrence topics.

Definition 1 (Topology of Service Ecosystem). *The topology of a service ecosystem containing mashup-service usage 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 m_j invokes service s_i, $E(j, i) = 1$.*

Definition 2 (Service Content). *In a service ecosystem, every service $s \in S$ comprises a collection of words $SW(s) = \{w_{s1}, w_{s2}, \ldots, w_{sn_s}\}$ to describe its functionality, in which n_s is the number of description words for service s.*

Definition 3 (Mashup Time Information). *In a service ecosystem, each mashup is provided by developers with its publication time. Let T_j stands for the publication time of mashup m_j. That is to say, services used by mashup m_j were composited at time T_j to realize significant composition patterns.*

We now formulate the following two problems.

Problem 1 (Discovery of Service Co-occurrence Topics and their Characteristics). *During the long evolution of a service ecosystem, some certain services might have been composited and cooperated repeatedly to realize some certain functionality when developers create new mashups, revealing certain kinds of latent topics about service composition patterns. Different from semantic functional topics, service co-occurrence topics describe latent composition patterns between services and are represented by the distribution over services in the ecosystem. For example, topic z is described by $\{\phi_{zs}, s = 1, ..., N_S\}$, in which ϕ_{zs} describes the impact of service s_s on topic z when making service composition, and $\sum_s \phi_{zs} = 1$. The characteristics of service co-occurrence topics contain:*

1) Topic Importance: *Different service co-occurrence topics reveal different composition patterns and have different importance in the service ecosystem. The most important ones illustrate the popular trend of service composition.*
2) Topic Representative Services: *Some services have high impact on a service co-occurrence topic, from which we could tell the primary functionality of the composition pattern. The more impact a service has in one topic, the more probability that it will be chosen when creating compositions on this topic.*
3) Topic Representative Description Words: *word description that could describe a topic funcitionally.*
4) Topic Temporal Strength: *an distribution over time, reflecting a service co-occurrence topic's lifecycle and revealing the change of popularity of certain patterns, which further reveals the trend of service composition in the service ecosystem.*

It is significant to discover the characteristics of topics. From the perspective of knowledge understanding, they will help mashup developers understand the latent service composition patterns. From an application perspective, the information will help to mine deeper characteristics of a service ecosystem and make better recommendation for service composition.

Problem 2 (Recommendation for Service Composition). *In this paper, we consider such a situation as in [3] to make use of service co-occurrence topics: assuming a mashup-developer selects the first API from the original search results provided by an online service repository (e.g., ProgrammableWeb.com). Now he needs to find other APIs to create a new mashup. Different from scenarios in goal-based service composition [11], [12], he may not know exactly what kind of mashup he wants to make, and just hopes to find related services to make significant compositions. Here we refer to the selected service as s_l, and the result of recommendation for its service composition is provided as a ranked list $R_l = \{s_{l1}, s_{l2}, ...\}$.*

2.2 Construction of Service Co-occurrence Documents

If we use a matrix to represent mashup-service usage records, it could be very sparse, in that most mashups only contain less than five services in the service ecosystem (e.g., ProgrammableWeb.com). Probabilistic models based on such data could cause sparsity problem [13]. To address this issue, we create "service co-occurrences" in our model. Each service is treated as a document with its co-occurring services (other services that are composited together with the service in some mashups) as words. As mentioned earlier, the service co-occurrences of a specific service consist of two parts: 1) co-occurrences with other services in the service ecosystem and 2) its self-co-occurrence. Combining the two parts of service co-occurrences, we could form the service co-occurrence document for each service. We use $c(i, j)$ $(i, j \in [1...N_S])$ to represent the co-occurrence index between two services. Specially, when $i = j$, $c(i, i)$ stands for the self-co-occurrence of service s_i.

Definition 4 (Co-occurrence with Other Services). *For each service $s_i \in S$, if service $s_j \in S$ $(j \neq i)$ is co-occurred (composited together) with s_i by mashups in the service ecosystem for $c_{i,j}$ times in total, we define the service co-occurrence index $c(i, j) = c_{i,j}$. Thus, according to our definition, we have $c(i, j) = c(j, i)$ $(j \neq i)$.*

Authorized licensed use limited to: SOUTHERN METHODIST UNIV. Downloaded on October 18,2024 at 15:56:48 UTC from IEEE Xplore. Restrictions apply.

Definition 5 (Self-co-occurrence). *Our previous work [10] did not consider self-co-occurrence of each service. Intuitively, for lack of the service itself in its co-occurrence document (i.e., s_i wouldn't appear in s_i's document), it may result in information losses when being applied a probabilistic model. To test whether the problem exists, we go one step further and propose a collection of gauges to define self-co-occurrence of each service. Detail comparison and analysis of these gauges are provided in the experimental section. For each service s_i, the four gauges of calculating self-co-occurrence $c(i,i)$ are:*

1) Zero Self-co-occurrence: *As in our previous work [10], we do not consider each service's self-co-occurrence, i.e., $c(i,i)=0$.*
2) Self-co-occurrence based on Mashup Usage Records: *The more times a service is composited by mashups, the more important it could be. If s_i has been composited by mashups to co-operate with other services for ms_i times in total, we define the self-co-occurrence index $c(i,i)=ms_i$.*
3) Self-co-occurrence based on Service Co-occurrence Pair Numbers: *The more times a service has co-occurred with other services, the more important it could be, that is, the more times it should appear in its co-occurrence document. Here we define the self-co-occurrence index as the s_i's service co-occurrence pair number scp_i, which means s_i has been co-occurred with other services for scp_i times totally. Due to the definition, $c(i,i)$ could be calculated according to:*

$$c(i,i) = scp_i = \sum\nolimits_{j,j\neq i} c_{i,j}. \tag{1}$$

4) Self-co-occurrence based on Average Service Co-occurrence Pair Numbers: *Instead of considering the total number of co-occurrence pairs of s_i, in SASCPN, we use the average pair numbers $ascp_i$. Here $c(i,i)$ could be calculated according to:*

$$c(i,i) = ascp_i = < \frac{1}{N_{csi}} \sum\nolimits_{j,j\neq i} c_{i,j} >, \tag{2}$$

where N_{csi} is the number of other services that have co-occurrence with s_i, and $< \cdot >$ refers to rounding the result.

Definition 6 (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 represents s_i's co-occurring service s_j, and $\#(sc_j) = c(i,j)$ means s_j appears $c_{i,j}$ times in the document of s_i. All service co-occurrence documents are referred to as D.*

Based on the definitions, we are ready to build the corpus of service co-occurrence documents. We treat d_{i_other} as the first part of s_i's document derived from s_i's co-occurrence with other services, and d_{i_self} as the second part derived from its self-co-occurrence. The process of constructing all the documents is described in Algorithm 1.

Adopting different gauges of service self-co-occurrence, we could obtain different corpora of service co-occurrence documents. For convenience, we represent them as Documents with ZS, Documents with SMUR, Documents with SSCPN, and Document with SASCPN, respectively.

Algorithm 1. Construct Service Co-occurrence Documents

Input:
1) E: The historical mashup-service usage records
2) The gauge (one among ZS, SMUR, SSCPN and SASC- PN) selected to calculate the self-co-occurrence

Output:
1) D: Service Co-occurrence Documents

Procedure:
01. **For** each service $s_i \in S$
02. Calculate s_i's co-occurrence index with others $c(i,j)$ $(j \neq i)$ according to Def. 4
03. Get d_{i_other} according to Def. 6 with the previous result
04. Calculate s_i's self-co-occurrence index $c(i,i)$ according to Def. 5 using the selected gauge
05. Get d_{i_self} according to Def. 6 with the previous result
06. Combine d_{i_other} and d_{i_self} to get d_i
07. **End**
08. Aggregate the service co-occurrence documents to get D

2.3 Service Co-Occurrence LDA

Based on the service occurrence documents, we extend the basic Latent Dirichlet Allocation [14] model and name it "Service Co-occurrence LDA" (SeCo-LDA). The key intuition is that if some services are composited together repeatedly by mashups in a service ecosystem, they are likely to belong to a same service co-occurrence topic, indicating latent composition patterns among them.

Recall that we have a corpus containing N_S service co-occurrence documents, the number of which equals to the amount of services in the service ecosystem. Assume that there were K different service co-occurrence topics expressed over N_S unique services in the service co-occurrence vocabulary. We set $z = 1:K$ as the topic indicator variable. The topic distribution of service co-occurrence documents (i.e., $P(z|d)$) can be represented by an $N_S \times K$ matrix $\boldsymbol{\Theta}$, each row of which, $\boldsymbol{\theta}_i$, being a $K-$dimension multinomial distribution for document s_i with $\theta_{iz} = P(z|i)$ and $\sum_{z=1}^{K} \theta_{iz} = 1$. The service co-occurrence distribution for topics (i.e., $P(s|z)$) can be represented by a $K \times N_S$ matrix $\boldsymbol{\Phi}$, each row of which, $\boldsymbol{\phi}z$, being an N_S-dimension multinomial distribution for topic z with $\phi_{zs} = P(s|z)$ and $\sum_{s=1}^{N_S} \phi_{zs} = 1$.

Following the convention of LDA [14], in SeCo-LDA, we use symmetric Dirichlet priors for $\boldsymbol{\Theta}$ and $\boldsymbol{\Phi}$ with hyperparameters α and β, respectively. We treat every individual service as a document, and its co-occurring services as the bag of "words" in that document. The graphical model for SeCo-LDA is shown in Fig. 2. The generative process can be described as follows.

1) For each topic z,
   Draw $\boldsymbol{\phi}_z \sim Dirichlet(\beta)$;
2) For each service s_i in the service ecosystem,
   a) Draw topic proportions $\theta_i \sim Dirichlet(\alpha)$.
   b) For each co-occurring service cs_{in} in service co-occurrence document d_i,

Authorized licensed use limited to: SOUTHERN METHODIST UNIV. Downloaded on October 18,2024 at 15:56:48 UTC from IEEE Xplore. Restrictions apply.

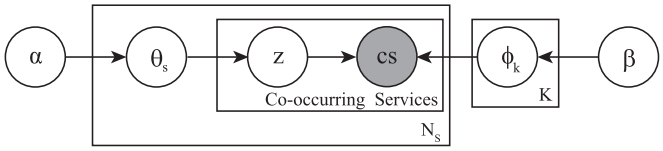


Fig. 2. Graphic Model for Service Co-occurrence LDA (SeCo-LDA).

i) Draw topic assignment $z_{in} \sim Multi(\theta_i)$.
ii) Draw s_i's co-occurring service $cs_{in} \sim Multi(\phi_{z_{in}})$.

Since we have four different corpora, for convenience, we represent SeCo-LDA based on these different corpora as SeCo-LDA with ZS, SeCo-LDA with SMUR, SeCo-LDA with SSCPN and SeCo-LDA with SASCPN, respectively.

3 Learn the SeCo-LDA Model

In this section, we first introduce the parameter learning of the SeCo-LDA model and then provide algorithms to extract characteristics of service co-occurrence topics (i.e., topic importance, representative services, representative description words and topic temporal strength).

3.1 Parameter Learning

According to the generative process, we define the joint probability of co-occurring services CS and the set of corresponding topics Z as follows:

$$P(CS, Z|\mathbf{\Theta}, \mathbf{\Phi}) = \prod_{i=1}^{N_S} \prod_{z=1}^{K} \prod_{s=1}^{N_S} \theta_{iz}^{n_{iz}} \phi_{zs}^{n_{zs}}, \tag{3}$$

where n_{iz} is the number of times that topic z has been associated with service co-occurrence document d_i, and n_{zs} is the number of times that co-occurring service s_s has been generated from topic z.

By placing the Dirichlet priors α and β, we can obtain the following equation of the joint probability:

$$\begin{aligned} & P(CS, Z|\alpha, \beta) \\ &= \int P(CS, Z, \theta, \phi|\alpha, \beta) d\theta d\phi \\ &= \int P(CS, Z, \theta, \phi) P(\theta|\alpha) P(\phi|\beta) \\ &= \prod_{i=1}^{N_S} \frac{\Gamma(\sum_z \alpha_z)}{\prod_z \Gamma(\sum_z \alpha_z)} \cdot \frac{\prod_z \Gamma(n_{iz} + \alpha_z)}{\Gamma(\sum_z n_{iz} + \alpha_z)} \\ &\quad \times \prod_{z=1}^{K} \frac{\Gamma(\sum_s \beta_s)}{\prod_s \Gamma(\sum_s \beta_s)} \cdot \frac{\prod_s \Gamma(n_{zs} + \beta_s)}{\Gamma(\sum_s n_{zs} + \beta_s)}. \end{aligned} \tag{4}$$

Like [14], we use the collapsed Gibbs sampling to make inferences with SeCo-LDA. The sampling is initialized by assigning random topic labels Z, and then updates them iteratively until reaching the setting number of iteration. In particular, for the tth co-occurring service s_s in the service co-occurrence document of s_i, the topic assignment is updated according to:

$$\begin{aligned} & P(z_{it}|cs_{it} = s, Z^{\neg(i,t)}, D^{\neg(i,t)}) \\ & \propto (\alpha_z + n_{iz}^{\neg it}) \times \frac{n_{zs}^{\neg it} + \beta_s}{\sum_s (n_{zs}^{\neg it} + \beta_s)}. \end{aligned} \tag{5}$$

After the burn-in stage, the sampling converges to the true posterior distribution. Posterior expectation of θ_{iz} and ϕ_{zs} is described as:

$$\theta_{iz} = \frac{n_{iz} + \alpha_z}{\sum_z (n_{iz} + \alpha_z)} \tag{6}$$

$$\phi_{zs} = \frac{n_{zs} + \beta_s}{\sum_s (n_{zs} + \beta_s)}. \tag{7}$$

We represent the setting number of Gibbs sampling iteration as N_G. In summary, the algorithm of applying Gibbs sampling to estimate parameters is listed as Algorithm 2:

Algorithm 2. Gibbs Sampling

Input:
1) The hyper-parameters α and β
2) Co-occurring service CS in the documents
3) The setting number of iteration N

Output:
1) Estimated parameters θ and ϕ

Procedure:
01. assigning random topic labels Z
02. **For** $iter = 1 : N_G$
03. **For** each co-occurring service cs_{it}
04. Sample topic assignment for the t-th co-occuring service in document d_i, z_{it}, according to Eqn. (5)
05. **End**
06. **End**
07. Get θ according to Eqn. (6)
08. Get ϕ according to Eqn. (7)

Further, the empirical posterior distribution of topics, which reflects the importance or popularity of topics in the service ecosystem, is given by:

$$P(z|D) = \frac{n_z}{\sum_z n_z}. \tag{8}$$

where n_z is the times that the topic of a co-occurring service has been assigned to z.

Intuitively, $\mathbf{\Theta}$ describes the topic distribution of services in the system (*service-topic* distribution), while $\mathbf{\Phi}$ indicates the distribution on co-occurring services of topics (*topic-service* distribution). They are the two key components in the model that will help to discover interesting knowledge about service co-occurrence topics.

3.2 Discovery of Service Co-occurrence Topics

From the results obtained from Eqns. (6), (7), (8), along with services' content and mashups' time information, we can discover knowledge about individual service co-occurrence topics, which will be helpful for people to have a better understanding of latent service composition patterns.

3.2.1 Topic Importance

Given by Eqn. (8), $\{P(z), z = 1 : K\}$ indicates the probability that services in one topic get composited. We can refer to the value of the distribution as the importance of different topics in the service ecosystem. Topics with higher importance will have a higher probability to be chosen by developers when

Authorized licensed use limited to: SOUTHERN METHODIST UNIV. Downloaded on October 18,2024 at 15:56:48 UTC from IEEE Xplore. Restrictions apply.

creating new mashups. The top topics reflect the major composition theme or trend of composing services.

3.2.2 Topic Representative Services

When creating mashups, developers usually prefer the most popular services in certain service co-occurrence topics to make significant service composition. Ranked value of *topic-service* distribution $\{\phi_{zs}\}$ reflects services' impact on topic z. The higher impact a service has in a topic, the higher probability it will be chosen when creating service composition patterns of this topic. The top representative services reflect the major composition functionality of this topic.

Another way to find representative services is to rank the services in each topic according to mashup-service usage records directly. However, it usually may not be accurate, for there may be situations that in some domains developers tend to composite more services than those from other areas. But with the help of *topic-service* distribution, we can address this problem and find more significant representation.

In summary, services' impacts on the topics shed lights on more accurate depiction of services' collaboration behaviors and ultimately lead to better recommendation, which we will show in the experimental section.

3.2.3 Topic Representative Description Words

The original results of SeCo-LDA only provide each service co-occurrence topic with a distribution over services. However, it is oftentimes desirable to extract representative words to describe a topic properly [15]. Along with the information about representative services, the representative description words will enable developers to have a brief and intuitive idea about the functionality of this service co-occurrence topic. Based on service content and *topic-service* distribution, we can extract the words with high *occurrence expectation* for a topic. The *occurrence expectation* of word w in topic z, $E(n_{zw})$, is calculated as:

$$\begin{aligned} E(n_{zw}) &= \sum_s n_{sw} P(service_s|z) \\ &= \sum_s n_{sw}\phi_{zs}, \end{aligned} \tag{9}$$

where n_{sw} stands for the occurrence counts of word w in service s_s.

After ranking the *occurrence expectation*, representative description words are extracted. We will show in the experimental section that the representative description words are intuitive and useful for developers to recognize and distinguish between different service co-occurrence topics. Note that services and mashups in a service ecosystem are usually described using the same language.

3.2.4 Topic Temporal Strength

Temporal strength describes the evolution lifecycle of a topic, and a higher value of temporal strength indicates that there was a higher probability that service compositions of this topic would appear at that time period. Intuitively, the lifecycle of a service co-occurrence topic usually includes *beginning period*, *boom period* and *fading period*, roughly paralleling the shape of a normal distribution. During the *beginning period*, a few services of this service co-occurrence topic appear and developers begin to composite them to create mashups. During a *boom period*, the most representative services of the topic emerge and a large number of mashups invoking these services are created by developers. During the *fading period*, developers' interest on this topic gradually decrease, and there would be less service composition of this topic.

Different service co-occurrence topics may have different lifecycles. Some may be long lasting while some topics' *boom period* may be very short and fade quickly. Some topics may have several *boom periods* while some may have only one.

To sufficiently describe topics' lifecycle, topic temporal distribution should be studied. For topic z in the service ecosystem, we use services' impact distribution over time to describe its temporal characteristics. Instead of leveraging services' publication time as [10], in this paper, we take the services' composited time into consideration, revealed from mashups' publication time information. The key intuition here is that if one service of a topic is composited at t_0, it makes contribution to the temporal strength of this topic at time t_0, no matter when this service is published. Note that we do not consider the services' invoked time by users. In this paper, we focus on discovering the service composition patterns. Mashups' publication time exactly record the time when services were composited to create significant patterns. Therefore, it is more suitable to consider services' composited time instead of their invoked time to discover service composition patterns.

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

$$\begin{aligned} \Pr(time \leq t_0|z) &= \sum_s \sum_{j,t_j^{(s)} \leq t_0} \frac{P(s|z)}{\sum_s co_s \cdot P(s|z)} \\ &= \sum_s \sum_{j,t_j^{(s)} \leq t_0} \frac{\phi_{zs}}{\sum_s co_s \cdot \phi_{zs}}. \end{aligned} \tag{10}$$

In a typical service ecosystem, the unit of time is *day*, which is discrete. The probability mass function (PMF) for temporal distribution of z is:

$$\Pr(time = t_0|z) = \sum_s \sum_{j,t_j^{(s)} = t_0} \frac{\phi_{zs}}{\sum_s co_s \cdot \phi_{zs}}. \tag{11}$$

With such information, the expectation time of topic z, with which we could distinguish new topics with old ones, can be calculated as follows:

$$E_z[time(s)] = \sum_s \sum_j t_j^{(s)} \cdot \frac{\phi_{zs}}{\sum_s co_s \cdot \phi_{zs}}. \tag{12}$$

3.3 Recommendation for Service Composition

As mentioned earlier in Problem 2 in Section 2.1, in this paper, we consider a situation of recommending related services as in [3], which is one of the application scenarios of

Authorized licensed use limited to: SOUTHERN METHODIST UNIV. Downloaded on October 18,2024 at 15:56:48 UTC from IEEE Xplore. Restrictions apply.

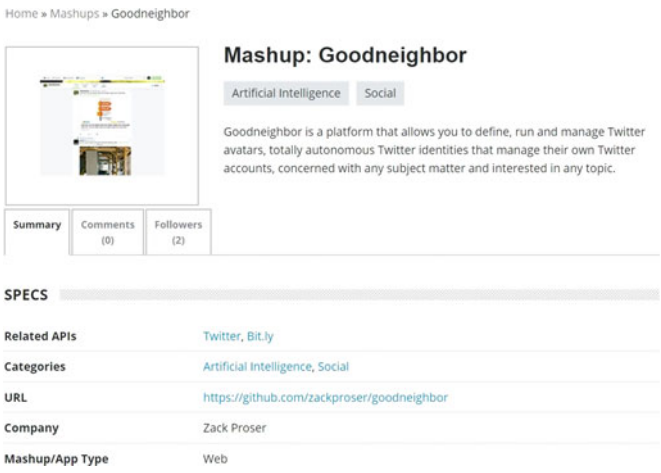


Fig. 3. A Real Mashup Example on ProgrammableWeb.com.

our technique. Assume that a developer has selected service $s_l \in S$ from the original search results provided by an online service repository (e.g., ProgrammableWeb.com), then we mean to provide a ranked service list to inspire him to make significant compositions. We could calculate s_l' co-occurrence expectation with another service $s_k \in S$ as follows:

$$\begin{aligned} & c^*(l,k) \\ &= \sum_z \Pr(cs = k|topic = z) \cdot \Pr(topic = z|s_l) \\ &= \sum_z \phi_{z,k} \cdot \theta_{l,z}. \end{aligned} \tag{13}$$

Services with high co-occurrence expectation are preferred to be recommended to developers, inspiring him to make service compositions. Finally, the recommended service list could be described as $RL = \{s_{l1}, s_{l2}, ...|c^*(l, s_{l1}) \geq c^*(l, s_{l2}) \geq ...\}$.

4 Experiments

In this section, we first present an overview of the data set from ProgrammableWeb as our test bed. Second, we show the parameter settings and experimental results on discovery of the service co-occurrence topics and recommendation for service composition. Last but not least, we discuss four ways to measure the self-co-occurrence and the complexity of our approach.

4.1 Data Set

In general, for any service ecosystem that satisfies Definitions 1, 2, 3, our theoretical approach SeCo-LDA can be applied to discover service co-occurrence topics as well as their characteristics.

In our experiments, we focus on the ProgrammableWeb.com,[1] which has been accumulating a variety of services and mashups since its inception in 2005 [16], [17]. A real mashup example *Goodneighbor*[2] available on the ProgrammableWeb is presented in Fig. 3, containing its name, description, related APIs, etc. To evaluate our methodology, we crawled the information of all services and mashups from ProgrammableWeb's inception (September 2005) to November 2015, extracting their descriptions, mashups' publication time and mashup-service usage records. The overview information of the data set from ProgrammableWeb is presented in Table 1. The data set used in this paper is available on our github repository.[3]

TABLE 1
Data Set from ProgrammabelWeb.com

Total # of services	12,711
Total # of services having co-occurrence with others	975
Total # of mashups	6,239
Total # of mashups containing more than one service	2,513
Total # of word vocabulary	21,328

4.2 Parameter Setting

4.2.1 Other Parameters

When adopting SeCo-LDA, we set the hyper-parameters $\alpha = 50/K$ and $\beta = 0.01$ according to the empirical formula [18], and the iteration number N_G as 1000.

4.2.2 Number of Topics K

We apply SeCo-LDA on the corpora (i.e., Documents with ZS, SMUR, SSCPN and SASCPN) to discover their latent service co-occurrence topics respectively. Similar with topic models based on words, when training, the documents in the corpus are treated as unlabeled. Note that there is a close connection between the number of topics, K, and the models' evaluation. Thus, our goal is to achieve high likelihood of the corpus as in [14]. In particular, we computed the $perplexity$ of the corpus to evaluate the models.

The perplexity, used by conventional language modeling, is monotonically decreasing in the likelihood of the test data, and is algebraicly equivalent to the inverse of the geometric mean per-word likelihood. Generally, a lower perplexity score indicates better generation performance. For a set of Mdocuments, the perplexity is described as:

$$perplexity(D) = \exp\left\{-\frac{\sum_{d=1}^{M} \log p(\boldsymbol{w}_d)}{\sum_{d=1}^{M} N_d}\right\}, \tag{14}$$

where $\boldsymbol{w}_d$ is the word vector of document d, and the probability of the word vector $p(\boldsymbol{w}_d)$ could be calculated as $p(\boldsymbol{w}_d) = \prod_n \sum_z p(w_{dn}|topic = z)p(topic = z|d)$.

To illustrate how we choose the numbers of topics detailedly, we take SeCo-LDA with ZS as an example. With other parameters fixed, the perplexity with different numbers of topics for SeCo-LDA with ZS is presented in Fig. 4. The lowest perplexity is obtained when the number of topics is 35. So when conducting experiments afterwards, we set the number of topics for SeCo-LDA with ZS as 35. Similarly, we found the optimal number of topics for SeCo-LDA with SMUR, SSCPN and SASCPN as 45, 50 and 40, respectively.

4.3 Results on Service Co-occurrence Topics

Representatively, applying SeCo-LDA with ZS, we present experimental results on discovery of service co-occurrence topics. We illustrate that our approach could reveal latent service co-occurrence topics as well as their characteristics, which would help us understand the service composition

1. https://www.programmableweb.com
2. https://www.programmableweb.com/mashup/goodneighbor
3. https://github.com/gzf09/Data-Set

Authorized licensed use limited to: SOUTHERN METHODIST UNIV. Downloaded on October 18,2024 at 15:56:48 UTC from IEEE Xplore. Restrictions apply.

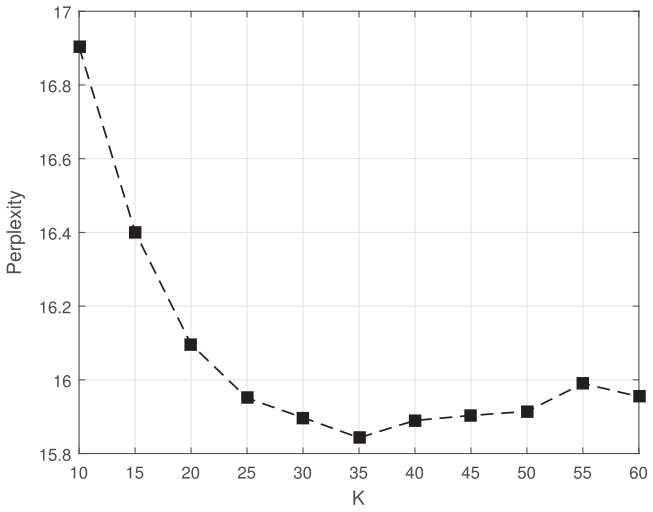


Fig. 4. Perplexity with Different Numbers of Topics K for SeCo-LDA with ZS.

patterns better. The parameters of SeCo-LDA with ZS are set as discussed before.

4.3.1 Important Topics & Representative Services

Calculated by Eqn. 8, topics with higher $P(z|D)$ are considered more important or popular in the service ecosystem, and will be more likely to be selected by developers when composing services. The more important a topic is, the higher probability there would be service composition of this topic.

The top five important topics are Topics 5(4.07 percent), 6 (3.93 percent), 35(3.77 percent), 19(3.63 percent) and 4(3.56 percent). The top 5 representative services for them are presented in Table 2. The *topic-service* probability ϕ_{zs} and mashup examples are also listed. Obviously, the representative services are sufficient to provide an overall view about the service co-occurrence topics' major functionalities. We could roughly identify that in the ProgrammableWeb.com service ecosystem, the most popular service co-occurrences topics are those about location-aware service composition, online multi-media community and social network.

In some service co-occurrence topics, the top representative services may occupy a dominant position, and we can regard such topics as *core-service-oriented* service co-occurrence topics. In Topic 6, service *YouTube* and *Amazon Product Advertising* show more than 93 percent impact among all the services. Topic 6 is about *YouTube*-centered video-based product advertising, in which *YouTube* provides video resources and *Amazon Product Advertising* allows developers to leverage the Amazon Product Discovery features that Amazon uses to advertise products and power their own business. A mashup example of this topic is *Find Best Three*, which is a buying suggestion and product comparison tool presenting the three best goods with the help of video reviews. Similarly, in Topic 4, service *Google Maps* and *Wikipedia* occupy nearly 90 percent impact among all the services. Topic 4 is about *Google Maps*-centered location-aware knowledge searching and resource sharing, where *Google Maps* provides language localization, region localization and geocoding, while *Wikipedia* provide direct, high-level access to the data contained in the MediaWiki databases. A mashup example of this topic is *TravelOxi.com*, which provides a new convenient way to explore over 400,000 travel destinations around the globe. Users could get instant information of photos, videos, maps, news and more.

In other topics, however, top representative services are not so dominant, such as Topics 5, 35 and 19. Topic 5 is about enriching maps or information where location is an important factor. In Topic 5, *Geonames* is a geographical database with useful information about different places. *Panoramio*, *Eventful* and *Yahoo Local Search* provide geolocated resources to enrich information about the locations. What's more, *Microsoft Bing Maps* allows people to build maps which can include routes and traffic info. A mashup example is *Distances Calculator*. It is a free web-based tool that allows users to easily calculate the distance between any two cities, showing the route and providing all kinds of information (e.g., photos, news and weather information) about the origin and destination. Topic 35 is about music information searching and recommending, in which *Last.fm* provides world's largest online music catalogue, *Amazon Product Advertising* allows developers to leverage the Amazon product features, *MusicBrainz* contains a database with a huge amount of music metadata, *Spotify Echo Nest* allows developers to analyze tracks and to add rich artist and song metadata to their applications, and *LyricWiki* provides lyrics. An example mashup of this topic is *Musikki*, a music search engine that gives users all the information in a single

TABLE 2
Top 5 Representative Services for Topics 5, 6, 35, 19 & 4

Topic 5 (4.07%)		Topic 6 (3.93%)		Topic 35 (3.77%)		Topic 19 (3.63%)		Topic 4 (3.56%)	
ϕ_{zs}	*Service Name*	ϕ_{zs}	*Service Name*	ϕ_{zs}	*Service Name*	ϕ_{zs}	*Service Name*	ϕ_{zs}	*Service Name*
0.15413	GeoNames	0.72705	YouTube	0.28517	Last.fm	0.13145	Google Geocoding	0.79285	Google Maps
0.09794	Panoramio	0.20321	Amazon Product Advertising	0.09795	Amazon Product Advertising	0.08644	Google Earth	0.10187	Wikipedia
0.09152	Eventful	0.01666	Yahoo Query Language	0.09708	MusicBrainz	0.08374	Google App Engine	0.03488	Vimeo
0.08991	Microsoft Bing Maps	0.01333	WebThumb	0.05201	Spotify Echo Nest	0.08194	Panoramio	0.01653	Google Ajax Feeds
0.06262	Yahoo Local Search	0.00667	Mobypicture	0.05115	LyricWiki	0.06213	Google Visualization	0.01102	Zazzle
Mashup Ex.	Distances Calculator	***Mashup Ex.***	Find Best Three	***Mashup Ex.***	youbeQ-Maps with Life	***Mashup Ex.***	Musikki	***Mashup Ex.***	TravelOxi.com

Authorized licensed use limited to: SOUTHERN METHODIST UNIV. Downloaded on October 18,2024 at 15:56:48 UTC from IEEE Xplore. Restrictions apply.

TABLE 3
Representative Description Words for Topics 29 & 33

	Topic 29	Topic 33
Representative Services	LinkedIn, Last.fm, Tumblr	Yahoo Weather, Blogger, Instapaper
Part of Extracted Words	easy available simple text webpage triporia hotel search location expedia movie	market data property chart search rate content list trend sale network fresh category

page with a single search. Topic 19 is a service composition group based on all kinds of Google services. A mashup example is *youbeQ-Maps with Life*, which allows people to explore the Earth, meet new people, discover new places and share information.

4.3.2 Topic Representative Description Words

With words' *occurrence expectation* calculated by Eqn. (9), we can extract a collection of words with high probability to represent for each topic. We use the Porter stemming algorithm and remove meaningless word stems.

Similar topics may have similar functionalities, but differences usually do exist between them. Results about Topics 29 and 33 are shown in Table 3. From representative services, we could identify that they are similar topics about social life on the Internet. But leveraging representative description words, we could also tell the tiny differences between them. Topic 29 concentrates more on people's daily life in business social network (*LinkedIn*), sharing multi-media resources (*Last.fm*, "text," "webpage," "movie") and travelling places ("triporia," "hotel," "expedia"). Information here is usually presented with short text conveniently (*Tumblr*, "easy," "simple"), making it easy to post and share multi-media resources. However, Topic 33 emphasizes on using a blog essay or long articles (*Blogger*, *Instapaper*, "list," "category") to record something ("content") or discuss about some problems (*Yahoo Weather*, "property," "trend").

In general, with representative services and description words together, we could have an overall view about the concrete meaning of a service co-occurrence topic.

4.3.3 Topic Temporal Strength

In our data set, the services' publication time ranges from September 4, 2005 to October 15, 2015. We take a *day* as the unit when calculating topics' temporal distribution over services. Higher temporal strength indicates that more service compositions of this topic would appear at that time period.

In Fig. 5, we present the topic temporal distribution over services of Topics 8 and 15. Topic 8, which contains representative services like *Instagram*, *Netflix*, *Reddit* and *Vimeo*, is about social multi-media sharing and discussion. Topic 15, whose representative services are *BTC-e*, *Coinbase*, *BitStamp* and *Mt Gox*, is about the transaction of BitCoin. From the expectiation times of the two topics, we can identify that Topic 15 is a relatively newer topic, while Topic 8 is an older one. Furthermore, from Fig. 5, we could conclude that Topic 8 became more popular since 2009 and only have one *boom period*, while Topic 15 has two *boom periods*, one is around 2009 and another is in 2014. The social media can be defined as a collection of online services that support to share opinions, thoughts and experiences [19]. In fact, most of these services sprang up around 2010, and more and more developers used them to create mashups. In addition, BitCoin was first proposed by Satoshi Nakamoto in concept in 2009 [20], arousing more and more studies on it [21]. The most popular trading platforms first arose around 2012, and many mashups were developed around 2014 to support the BitCoin transaction, invoking hot BitCoin services. There were about 12.5 million bitcoins in mid-2014, valued at approximately$7 billion [22].

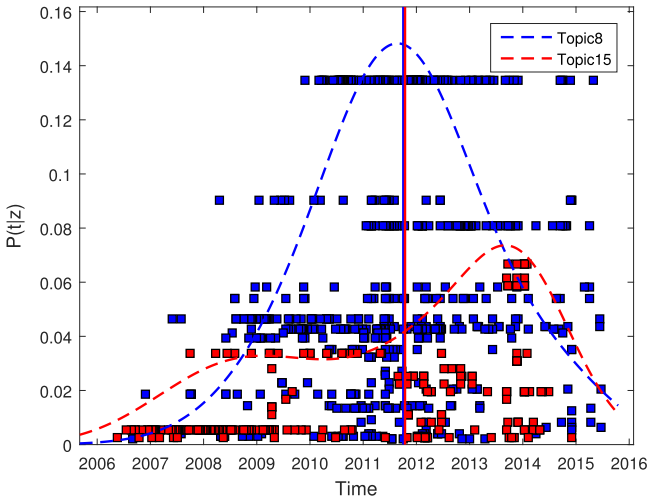


Fig. 5. Distribution Over Service for Topics 8 & 15. Blue stands for Topic 8, and red for Topic 15. Each square symbolizes a representative services invoked at that time, the heights of which stands for the representative services' impact on this topic. The dotted lines describe the fitted curves of topics' temporal strength. The vertical lines show the expectation time of each topic.

Above all, information provided by topics' temporal strength consists of external empirical knowledge basically. Thus, it could help to mine deeper characteristics of service ecosystems, such as topic dependencies and make evolution analysis from the perspective of the systems.

4.4 Comparison Results on Service Composition

To demonstrate that topics revealed by our approach shed lights on more accurate depiction of services collaboration behavior, we compare the results of recommendation for service composition using our model and baselines. Recommendation is one of the application scenarios for service co-occurrence topics.

4.4.1 Baselines

In order to compare with SeCo-LDA with ZS, SMUR, SSCPN and SASCPN, three baselines of recommending for service composition are chosen as comparisons.

Baseline Method 1: AA. Apriori Algorithm (AA) [5] is a commonly-used technique to mine association rules. In this approach, each mashup is represented as the union of annotated tags of its component services. Apriori could mine positive rules of tags from the transactions of mashups[23]. The probability of composition of any two services $s_1 \in S$ and $s_2 \in S$ is estimated as:

Authorized licensed use limited to: SOUTHERN METHODIST UNIV. Downloaded on October 18,2024 at 15:56:48 UTC from IEEE Xplore. Restrictions apply.

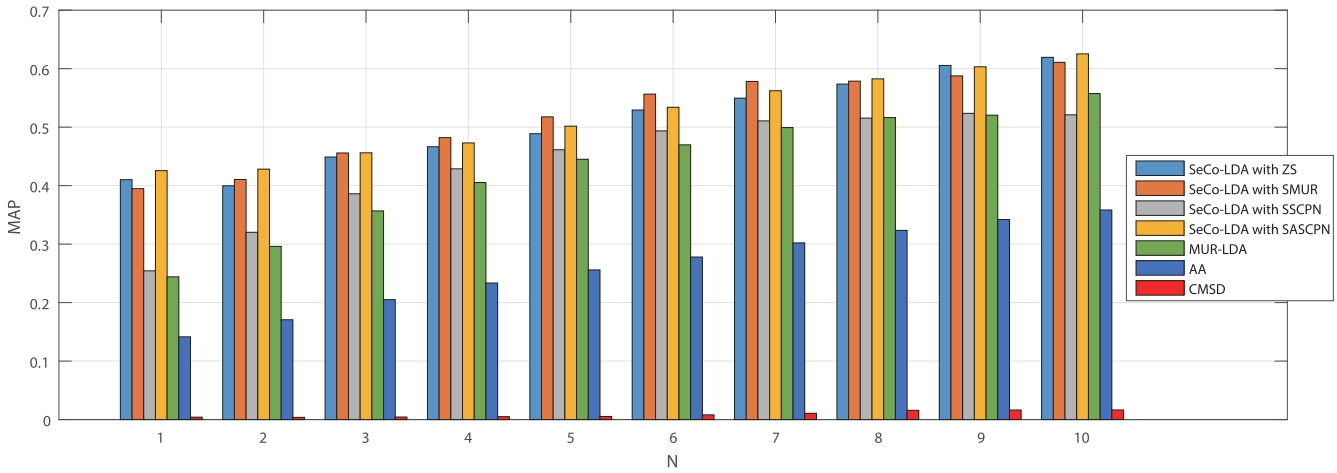


Fig. 6. The MAP for SeCo-LDA Models (i.e., with ZS, SMUR, SSCPN and SASCPN), MUR-LDA, AA and CMSD with Different Number of N.

$$F(s_1, s_2) = \sum_{r \in R(s_1, s_2)} \beta \cdot supprort(r) + (1-\beta) \cdot confidence(r), \quad (15)$$

where $R(s_1, s_2)$ denotes the rules that are satisfied by $< s_1, s_2 >$, which means in the rules s_1 and s_2 appear together. β is a weighting coefficient. The higher $F(s_1, s_2)$ is, the more possiblely that a composition of s_1 and s_2 would appear.

For a selected service s_l, its recommended service list can be described as $RL_{AA} = \{s_{l1}, s_{l2}, ...|F(s_l, s_{l1}) \geq F(s_l, s_{l2}) \geq ...\}$.

Baseline Method 2: CMSD. Using Content Matching based on Service Description (CMSD) [8], [24], based on service word descriptions, we apply the LDA model to calculate the semantic similarities between the selected service and others. We also run Gibbs sampling to get probability distribution of services over topics $p_2(z|s)$ and topics over words $p_2(w|z)$.

When a selected service s_l comes up, CMSD calculates the semantic similarity between s_l and another service $s_i \in S$, which is described as $SW_i = \{w_{i1}, w_{i2}, ..., w_{in_i}\}$, as follows:

$$p_{CM}(s_i|s_l) = \sum_{w \in SW_i} \sum_{z=1}^{K} p_2(w|z) \cdot p_2(z|s_l). \quad (16)$$

Services with higher semantic similarities are preferred to be recommended. The recommendation list is: $RL_{CMSD} = \{s_{l1}, s_{l2}, ...|p_{CM}(s_{l1}|s_l) \geq p_{CM}(s_{l2}|s_l) \geq ...\}$.

Baseline Method 3: MUR-LDA. We apply LDA directly using Mashup-service Usage Records (MUR-LDA). That is, we regard mashups as documents and services as word tokens, which is similar with [25]. We model the generation of services in mashups with a probabilistic topic model. Using the collapsed Gibbs sampling, we can get the probability distribution of mashups over topics $p_3(z|m)$ and topics over services $p_3(s|z)$. For a selected service s_l, we could calculate its expected co-occurrences with another service $s_i \in S$:

$$c_3(l, i) = \sum_m \sum_z p_3(s = i|z) \cdot p_3(z|m) \cdot p(m|s_l), \quad (17)$$

where $p(m|s_l)$ stands for the probability of mashup m_m given service s_l. It could be calculated using Bayes theorem. The recommended service list is ranked by the value of $c_3(l, i)$: $RL_{MUR} = \{s_{l1}, s_{l2}, ...|c_3(l, s_{l1}) \geq c_3(l, s_{l2}) \geq ...\}$.

4.4.2 Experimental Settings

The parameter setting of SeCo-LDA with ZS, SMUR, SSCPN and SASCPN has been discussed in Section 4.2. For AA, we set the weight coefficient $\beta = 0.5$. For CMSD and MUR-LDA, we set the hyper-parameters as the SeCo-LDA models. According to perplexity, we design experiments and find the optimal number of topics for CMSD is 15, and for MUR-LDA it is 10.

We train the models on the data set from ProgrammableWeb.com, and test the performance of recommendation on services that have co-occurrences with others in the service ecosystem. For each selected service, we recommend the most related N candidates for it, and compare the list with the original ranked service co-occurrence relationship revealed from mashup-service usage information.

4.4.3 Evaluation Metric

Mean Average Precision [26] is used as evaluation metric in this part, which is a widely used measure in recommendation systems:

$$MAP = \frac{1}{|S|} \sum_{i \in S} \frac{1}{N} \sum_{s \in CS_i} \frac{n(s)}{r(s)}, \quad (18)$$

where S denotes the set of testing services; N represents the recommended number of services; CS_i denotes the co-occurring services of service s_i; for each $s \in CS_i$, $r(s)$ refers to the ranking position of s in service-composition recommendation list and $n(s)$ represents the number of co-occurring services in CS_i that rank higher than or equal to s in recommendation list.

MAP is a real number between 0 and 1. Higher MAP indicates better performance of the recommendation method.

4.4.4 Results of Recommendation

The results of MAP of SeCo-LDA and baselines with different numbers of N are shown in Fig. 6.

Authorized licensed use limited to: SOUTHERN METHODIST UNIV. Downloaded on October 18,2024 at 15:56:48 UTC from IEEE Xplore. Restrictions apply.

CMSD only uses the word description of services to reveal latent semantic topics and make recommendation. Experiments show that its MAP value is the lowest among the four methods, indicating that the description of services might not be a good indicator of service composition patterns individually. The other three methods take the usage records of services into consideration. AA is the most popular model to draw association rules, which has a higher MAP value than CMSD as shown in Fig. 6. Using AA we could get the probability of popular service association rules, but could not know what kind of a specific service association is or the concrete functionality of it. MUR-LDA and SeCo-LDA models both use topic model to analyze the association relationship of services in the system. But the core difference is that MUR-LDA regards mashups as documents containing services as word tokens, while SeCo-LDA models regard services as documents and their co-occurring services as word tokens. MUR-LDA may come up with the sparsity problem for that most mashups usually contain less than five services.

Overall, our SeCo-LDA models perform better than the baselines. Among them, SeCo-LDA with SSCPN uses service co-occurrence pair numbers to describe a service's self-co-occurrence. This could result in bad results, for that the aggregate number of a service's all co-occurrence pairs may be much larger than the number of a single pair, which would bring many more "words" in this service's co-occurrence document, but reduce its quality at the same time. As a result, SeCo-LDA with SSCPN gets the lowest MAP among the four SeCo-LDA models, a little better than MUR-LDA in average and worse than MUR-LDA when N is large. The others do not consider services' self-co-occurrence (SeCo-LDA with ZS), or take self-co-occurrence into account in a more suitable way (SeCo-LDA with SMUR and SASCPN). They get the highest MAP value, approximately 5 percent better than MUR-LDA. Detailed comparison will be discussed in the next subsection.

From the experiments shown above, we can conclude that by analyzing *service co-occurrence documents*, SeCo-LDA models could shed lights on more accurate depiction of services collaboration behaviors, and perform better than the baselines when recommending for service composition.

4.5 Further Comparison among the Four Gauges

4.5.1 Discussion by Intuition

First of all, intuitively, we discuss the documents derived from the four gauges. Ultimately, our goal is to use these documents to discover the latent composition patterns between services. ZS does not take self-co-occurrence into consideration. Among the three that considering services' self-co-occurrence, SMUR concentrate on the mashup-side information, while SSCPN and SASCPN pay more attention to information about the number of service co-occurrence pairs. As mentioned before in Section 4.4.4, in SSCPN, too many s_i may appear in its document d_i, reducing its quality and making it sometimes poorer than ZS. For SMUR and SASCPN, it is easy to find that $ms_i \geq ascp_i$, and the equality holds only when each mashup invoked every service in the service ecosystem. Actually, for the data set from ProgrammableWeb, the average of $ascp_i$ is much less than that of ms_i. With a further insight into the implication of s_i's document, $c(i,i)$ indicates that in d_i, the other services may co-occur with each other with the help of $c(i,i)$ "words" of s_i, and we intent to apply SeCo-LDA to mine the latent relationship between services in d_i. The definition of SMUR just fits this. As a result, SMUR might be a better choice.

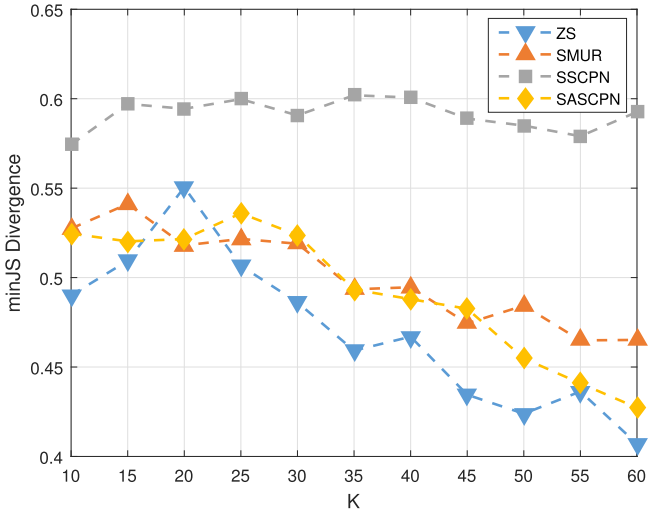


Fig. 7. The Average Minimal JS Divergence for SeCo-LDA with ZS, SMUR, SSCPN and SASCPN with Different Number of K.

4.5.2 Discussion on JS Divergence

On the one hand, we use the average minimal Jensen-Shannon (JS) divergence [13] as metric to compare the quality of topics learned with different documents. The JS divergence measures similarity of two pairs of distributions. The measure is 0 only for identical distributions and approaches infinity as the two differ more and more. Formally, it is defined as the average of the KL divergence of the two distributions (i.e., P and Q) to the average of the two distributions (i.e., $R = 1/2(P + Q)$):

$$D_{JS} = \frac{1}{2} D_{KL}(P||R) + \frac{1}{2} D_{KL}(Q||R), \tag{19}$$

where $D_{KL}(\cdot)$ represents the KL divergence.

For the *topic-service* distributions learned with each kind of service co-occurrence documents, we first calculate the JS divergence between any two different topics and find the most "similar" one for each topic, then we calculate the average of minimal JS divergence over all the topics. Adopting different numbers of K, the results are shown in Fig. 7.

In most cases, higher minimal JS divergence indicates the topics are more different. With K fixed, Fig. 7 shows that SSCPN gets the highest result among the four, but it is caused by regarding the aggregate number of co-occurrence pairs as $c(i,i)$, making each document d_i be composed mainly by words of s_i. So it does not mean topics learned by SSCPN have higher quality. For the other three gauges, generally, SMUR and SASCPN receive similar results, revealing topics of higher quality than ZS.

4.5.3 Discussion on MAP for Recommendation

On the other hand, as shown in Fig. 6, we compare the MAP results when making recommendation for service composition. As discussed before, SeCo-LDA with SSCPN gets the lowest MAP among the four SeCo-LDA models. All the left

Authorized licensed use limited to: SOUTHERN METHODIST UNIV. Downloaded on October 18,2024 at 15:56:48 UTC from IEEE Xplore. Restrictions apply.

TABLE 4
Running Time for SeCo-LDA with ZS with Different Data Scale

Multiple of Data Scale	N_W	Time (s) for *Part I*	Time (s) for *Part II*	Total Time (s)	Multiple of Total Time
1	30,238	0.021	1.827	1.848	1
2	60,476	0.035	3.741	3.776	2.04
5	151,190	0.221	10.971	11.192	6.06
10	302,380	0.903	27.067	27.970	15.14

three SeCo-LDA models achieve better performance than baselines, approximately 5 percent better than MUR-LDA. In Problem 2, we would not recommend itself for a service. So even without consideration of self-co-occurrence, ZS could reach relatively good results. Leveraging information about self-co-occurrence properly, SMUR and SASCPN get higher MAP value than ZS in average. SeCo-LDA with SMUR has the highest MAP value, especially when we need to recommend 4 to 7 candidate services. These results consist of the intuitive discussion and minimal JS divergence discussion we made before. In summary, SeCo-LDA with SMUR is most preferred when addressing Problem 2.

4.6 Discussion about Complexity

In order to show the time-cost tolerance of our approach when dealing with large-scale data sets, we make discussion about complexity.

We can divide the complexity of our appraoch into two parts: *Part I:* constructing service co-occurrence documents and *Part II:* inferring parameters in the probabilistic topic model. The complexity of *Part I* is bounded by $O(N_{maxs}^2 \cdot N_M)$, in which N_{maxs} stands for the maximum number of services in one mashup. The complexity of *Part II* is $O(N_G \cdot K \cdot N_W)$, where N_G stands for the number of iterations and K is the number of topics. N_W represents the total word tokens in the corpus of service co-occurrence documents. So the complexity of our approach is bounded by $O(N_{maxs}^2 \cdot N_M + N_G \cdot K \cdot N_W)$.

To test our conclusion, we manually expand the scale of mashups and services in ProgrammableWeb.com for 2, 5 and 10 times, keep the average number of services in one mashup unchanged basically, and record the running time of SeCo-LDA with ZS representatively. We set the unit of time as *second*, and conduct the experiments on a PC with 16GB memory and Core-i7 CPU. The results are recorded in Table 4. With data set's other characteristics fixed, we could conclude that N_W increases along with the multiple of data scale, and the multiple of total running time increases linearly as the scale increases approximately. So when dealing with data sets containing more services and mashups, the time-cost of our approach is usually tolerable.

5 Related Work

5.1 Recommendation Based on Composition Records

In the research of automatic service composition, most existing studies about service association learning with service usage data are based on the Apriori algorithm [23], [27], [28]. A global Co-utilization service ranking (gCAR) strategy, inferred from historical API usage metadata, is proposed in [3] based on Apriori to discover association rules, reducing the search time for comparable degree of completeness. A three-level model of service usage data (i.e., user request level, template level and instance level) is defined in [4] to model service usage patterns. Besides service usage data, other information is integrated to achieve better performance. In [5], Apriori is used to find association rules of social tags and predict future mashups based on mined rules. Service logs are taken into consideration in [6] to find the best composition among possible compositions to improve quality and performance of web service composition. Association rules are mined with Apriori to build a knowledge repository to instruct mashup creation. [29] consider negative and positive relation between services simultaneously to make recommendation.

5.2 Recommendation based on Functionality

Based on service description, function-based recommendation is the most commonly-used techniquef. Early works are based on semantic service languages [30], or use keyword-based search engines combining with information from Web Service Description Language (WSDL), or Universal Description, Discovery and Integration (UDDI) [31], [32], [33]. However, they can only deliver limited performance, sensitive to the words given by users.

On the one hand, as LDA [14] came up with a probabilistic approach for human language understanding, novel approaches for service recommendation appeared. To protect business interests of service providers and information leakage, implementation details of services are usually invisible to consumers [34]. Thus, word description became more popular than conventional service languages. A time-aware service recommendation framework is established in [35] for mashup creation, using latent topic models to model services' and mashups' word description as well as timestamps. [36] proposed a service recommendation algorithm by clustering and recommending candidate ones by different domains. [9] extends collaborative topic regression model, considering content information and usage data together to improve performance of recommendation. Above all, however, topics revealed in these works are built on word co-occurrence basically, different from "service co-occurrence topics" that we mean to reveal. [37] proposed a Citation-LDA, which uses paper citation as word tokens. But using method in [37] with mashup-service usage records would cause data sparsity problem [13], for most mashups only invoke less than five services.

On the other hand, in order to discover topics' temporal property, we take time information into consideration. Existing probabilistic topic models considering time information has been studied such as Topic Over Time model [38], Dynamic Topic Model [39] and Correlated Topic Model [40]. Through the models mentioned above, however, we can only compute the text similarity between document and topic, which is quite different from the documents' "impact" on a topic that we mean to discover.

Different from the works above, we use latent probabilistic topic model to mine frequent patterns from service usage records. In general, our solution to mine service co-occurrence topics and their characteristics is suitable for any service ecosystem that satisfies Definitions 1, 2, 3. The mashup-

Authorized licensed use limited to: SOUTHERN METHODIST UNIV. Downloaded on October 18,2024 at 15:56:48 UTC from IEEE Xplore. Restrictions apply.

service usage records are very important information, without which our solution would not work but traditional methods would do if WSDL files are provided.

6 Conclusions

In this paper, we present a novel approach for identifying the latent service composition patterns in service ecosystems. The key idea is to represent each service as a document with its co-occurring services as words.

Applying our SeCo-LDA technique, a service composition development process will contain the following three steps: (1) Developers understand the trend of service composition in the service ecosystem. The characteristics of topics would help people gain an overall view about service composition patterns in the service ecosystem. (2) Developers propose their demands to create mashups and get the recommendation list. In this paper we consider the scenario of recommendation for a selected service. Comparison with baseline approaches demonstrates that SeCo-LDA performs 5 percent better in terms of MAP when recommending service composition. (3) Developers choose services from the list to make final compositions.

This paper is our first attempt to apply probability generation model to mashup-service composition records in service ecosystems. Our future work would focus on two aspects: (1) addressing problems about terminology to extract more significant representative words for topics, and (2) utilizing information about service co-occurrence topics to design an effective recommendation framework for automatic mashup creation.

Acknowledgments

This research has been partially supported by the National Natural Science Foundation of China (No.61673230). 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] B. Tapia, R. Torres, H. Astudillo, and P. Ortega, "Recommending APIs for mashup completion using association rules mined from real usage data," in *Proc. IEEE Int. Conf. Chilean Comput. Sci. Soc.*, 2011, pp. 83–89.

[4] Q. A. Liang, J.-Y. Chung, S. Miller, and Y. Ouyang, "Service pattern discovery of web service mining in web service registry-repository," in *Proc. IEEE Int. Conf. e-Business Eng.*, 2006, pp. 286–293.

[5] K. Goarany, G. Kulczycki, and M. B. Blake, "Mining social tags to predict mashup patterns," in *Proc. ACM Int. Workshop Search Mining User-Generated Contents*, 2010, pp. 71–78.

[6] S. Bayati, A. F. Nejad, S. Kharazmi, and A. Bahreininejad, "Using association rule mining to improve Semantic Web services composition performance," in *Proc. IEEE Int. Conf. Comput. Control Commun.*, 2009, pp. 1–5.

[7] Y. Ni, Y. Fan, K. Huang, J. Bi, and W. Tan, "Negative-connection-aware tag-based association mining and service recommendation," in *Proc. Int. Conf. Serv.-Oriented Comput.*, 2014, pp. 419–428.

[8] Y. Zhong, Y. Fan, K. Huang, W. Tan, and J. Zhang, "Time-aware service recommendation for mashup creation in an evolving service ecosystem," in *Proc. IEEE Int. Conf. Web Serv.*, 2014, pp. 25–32.

[9] B. Bai, Y. Fan, K. Huang, W. Tan, B. Xia, and S. Chen, "Service recommendation for mashup creation based on time-aware collaborative domain regression," in *Proc. IEEE Int. Conf. Web Serv.*, 2015, pp. 209–216.

[10] 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 *Proc. IEEE Int. Conf. Web Serv.*, 2016, pp. 25–32.

[11] L. O. B. da Silva Santos, E. G. da Silva, L. F. Pires, and M. J. van Sinderen, "Towards a goal-based service framework for dynamic service discovery and composition," in *Proc. IEEE Int. Conf. Inf. Technol.: New Generations*, 2009, pp. 302–307.

[12] Z. D, "A Petri net-based approach for automated goal-driven web service composition," *Simul.*, vol. 83, no. 1, pp. 33–63, 2007.

[13] L. Hong and B. D. Davison, "Empirical study of topic modeling in Twitter," in *Proc. 1st Workshop Soc. Media Analytics*, 2010, pp. 80–88.

[14] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, 2003.

[15] J. Chang, S. Gerrish, C. Wang, J. L. Boyd-Graber , and D. M. Blei, "Reading tea leaves: How humans interpret topic models," in *Adv. Neural Inf. Process. Syst.*, 2009, pp. 288–296.

[16] A. P. Barros and M. Dumas, "The rise of web service ecosystems," *IT Prof.*, no. 5, pp. 31–37, 2006.

[17] E. Al-Masri and Q. H. Mahmoud, "Investigating web services on the world wide web," in *Proc. ACM Int. Conf. World Wide Web*, 2008, pp. 795–804.

[18] T. L. Griffiths and M. Steyvers, "Finding scientific topics," *Proc. Nat. Academy Sci.*, vol. 101, no. suppl 1, pp. 5228–5235, 2004.

[19] J. T. Kim, J. H. Lee, H. K. Lee, and E. H. Paik, "Design and implementation of the location-based personalized social media service," in *Proc. Int. Conf. Internet Web Appl. Serv.*, 2010, pp. 116–121.

[20] D. Ron and A. Shamir, "Quantitative analysis of the full Bitcoin transaction graph," in *Financial Cryptography and Data Security*, 2013, pp. 6–24.

[21] F. Reid and M. Harrigan, *An Analysis of Anonymity in the Bitcoin System*. New York, NY, USA: Springer, 2011.

[22] M. A. Cusumano, "The Bitcoin ecosystem," *Commun. ACM*, vol. 57, no. 10, pp. 22–24, 2014.

[23] R. Agrawal, R. Srikant, et al., "Fast algorithms for mining association rules," in *Proc. 20th Int. Conf. Very Large Data Base*, 1994, vol. 1215, pp. 487–499.

[24] 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.

[25] Y. Zhang, T. Lei, and Y. Wang, "A service recommendation algorithm based on modeling of implicit demands," in *Proc. IEEE Int. Conf. Web Serv.*, 2016, pp. 17–24.

[26] 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. Serv.-Oriented Comput.*, 2013, pp. 343–357.

[27] H. Toivonen, "Apriori algorithm," *Encyclopedia of Machine Learning*, pp. 39–40, 2010, doi: 10.1007/978-0-387-30164-8_27.

[28] M. Hegland, "The apriori algorithm - A tutorial," *Math. Comput. Imaging Sci. Inf. Process.*, vol. 67, no. 5, pp. 1–12, 2015.

[29] Y. Ni, Y. Fan, W. Tan, K. Huang, and J. Bi, "NCSR: Negative-connection-aware service recommendation for large sparse service network," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 2, pp. 579–590, Apr. 2016.

[30] A. A. Sycara K, Paolucci M, "Automated discovery, interaction and composition of Semantic Web services," *Web Semantics Sci. Serv. Agents World Wide Web*, vol. 1, no. 1, pp. 27–46, 2004.

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

[32] C. Platzer and S. Dustdar, "A vector space search engine for web services," in *Proc. 3rd Eur. Conf. Web Serv.*, 2005, Art. no. 9.

[33] S. Meng, W. Dou, X. Zhang, and J. Chen, "KASR: A keyword-aware service recommendation method on MapReduce for big data applications," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 12, pp. 3221–3231, Dec. 2014.

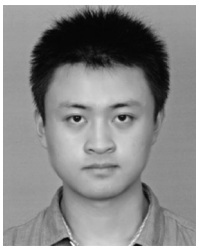
[34] C. Ye and H.-A. Jacobsen, "Whitening SOA testing via event exposure," *IEEE Trans. Softw. Eng.*, vol. 39, no. 10, pp. 1444–1465, Oct. 2013.

[35] Y. Zhong, Y. Fan, K. Huang, W. Tan, and J. Zhang, "Time-aware service recommendation for mashup creation," *IEEE Trans. Serv. Comput.*, vol. 8, no. 3, pp. 356–368, May/Jun. 2015.

[36] B. Xia, Y. Fan, W. Tan, K. Huang, J. Zhang, and C. Wu, "Category-aware API clustering and distributed recommendation for automatic mashup creation," *IEEE Trans. Serv. Comput.*, vol. 8, no. 5, pp. 674–687, Sep./Oct. 2015.

Authorized licensed use limited to: SOUTHERN METHODIST UNIV. Downloaded on October 18,2024 at 15:56:48 UTC from IEEE Xplore. Restrictions apply.

[37] X. Wang, C. Zhai, and D. Roth, "Understanding evolution of research themes: A probabilistic generative model for citations," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2013, pp. 1115–1123.
[38] X. Wang and A. McCallum, "Topics over time: A non-Markov continuous-time model of topical trends," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2006, pp. 424–433.
[39] D. M. Blei and J. D. Lafferty, "Dynamic topic models," in *Proc. ACM Int. Conf. Mach. Learn*, 2006, pp. 113–120.
[40] D. Blei and J. Lafferty, "A correlated topic model of science," *Ann. Appl. Statist.*, vol. 1, no. 1, pp. 17–35, 2007.



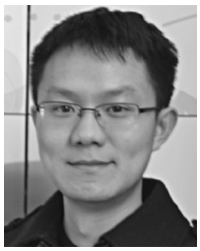
Zhenfeng Gao received the BS degree in control theory and application from Tsinghua University, China, in 2013. He is currently working toward the PhD degree in the Department of Automation, Tsinghua University. His research interests include services computing, service recommendation, and big data.

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. 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.

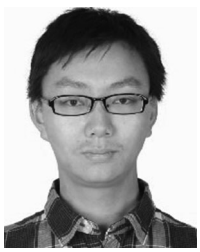
Cheng Wu received the BS and MS degrees in electrical engineering from Tsinghua University, Beijing, China. He is currently a fellow with Chinese Academy of Engineering. His research interests include complex system modeling and optimization, and modeling and scheduling in supply chains.

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, New York. 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. 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 BS and MS 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 with the Department of Electrical and Computer Engineering, Carnegie Mellon University. Her recent research interests include service oriented computing, with a focus on scientific workflow, cyberinfrastructure and intelligent system, AI in big data, and scientific collaboration. She is currently an associate editor of the *IEEE Transactions on Services Computing (TSC)*. She is a senior member of the IEEE.



Yayu Ni received the BS degree from the Department of Automation, Tsinghua University, Beijing, China, in 2011. He is currently working towards the PhD degree in the Department of Automation, Tsinghua University. His research interests include services computing, natural language processing, social network analysis, machine learning, and distributed computation.



Bing Bai received the BS degree in control theory and application from Tsinghua University, China, in 2013. He is currently working toward the PhD degree in the Department of Automation, Tsinghua University. His research interests include services computing, service recommendation, and big data.

Shuhui Chen received the BS degree from the Department of Power Engineering, Southeast University, China in 2001, and the MS degree from the Department of Automation, Nankai University, China, in 2009. He is working toward the PhD degree in the Department of Automation, Tsinghua University. His research interests include services computing, service recommendation, and big data.

Authorized licensed use limited to: SOUTHERN METHODIST UNIV. Downloaded on October 18,2024 at 15:56:48 UTC from IEEE Xplore. Restrictions apply.