Learning to Build Accurate Service Representations and Visualization

Junqi Zhang, Yushun Fan, Jia Zhang, *Senior Member, IEEE*, Bing Bai

Abstract—With the boom of Web services, there is a growing need for visualizing service ecosystems to help people browse services and understand their functionalities and positions in the systems. One foundational step of building a proper visualization is to ensure accurate representations for the comprising services. However, it is not a trivial task as service profiles may not be sufficient for two significant reasons. Firstly, while the services themselves being used in various scenarios, their profiles may not always precisely reflect all of them. Secondly, service profiles usually comprise quite a few universal background terms that cannot distinguish services. To address these two issues, we apply machine learning techniques to incrementally learn service representations in a whole. A tailored topic model is developed, named Service Representation-Latent Dirichlet Allocation (SR-LDA). The core idea is to learn more comprehensive and up-to-date information about services from the profiles of the involved service compositions (i.e., mashup profiles), while introducing a global filter to identify and filter out background terms. Both quantitative and qualitative experiments on a real-world dataset demonstrate that the proposed SR-LDA builds higher-quality service representations comparing with baselines. We further generate a knowledge map to visualize a service ecosystem based on the learned service representations. Such a knowledge map directly leads to the detection of four typical functionality patterns of Web services and serves the purpose of mashup creation.

Index Terms—Service representation, service visualization, service ecosystem, topic model, knowledge map

1 Introduction

Web services are self-described programmable applications, offering interoperability and universal accessibility over the Internet [1]. Such a remote reusability allows software developers to efficiently integrate multiple functionalities offered by different Web services and quickly build value-added service compositions, also called mashups [2], [3]. As a matter of course, an increasing number of services published by their providers, together with mashups and corresponding developers, have formed several Web service ecosystems such as ProgrammableWeb.com [4].

However, the overload of information from the rapidly increasing number of services in a service ecosystem brings a considerable challenge for mashup developers as well as system managers: how to choose from the sea of services? In order to alleviate this problem, the research community has developed many service recommendation and discovery methods [5], [6], [7], [8]. However, most methods can only give references when developers have clear demands, whereas cannot help people with fuzzy problems like: "Are there any interesting services that can bring appealing functionalities to build my mashups?" "What are the functional relationships among various services?" "What domains of services are still in the early stages of market competition and have the space for new entries?" In such cases, visualizing a service ecosystem may provide a practical overview of all the services, and help in understanding their functionalities and complex relationships among them. For example, some services form a clique meaning that they bear similar functionalities, and a clique composed of abundant services indicates that there is not much chance for new services to succeed in that service domain. By studying such a visualization, people may also identify services that deserve further investigation [9].

Before forming a proper visualization of a service ecosystem, though, one foundational step is to make sure the representations of all its comprising services are accurate. A term *service profile*[1] was coined to describe and represent services, earlier built from WSDL files for SOAP services and later from textual descriptions for RESTful services. Relying on such original service profiles published by service providers, however, suffers from the following two significant matters.

First, service profiles may not always precisely reflect the various usage scenarios of corresponding services. Original service profiles are created by relevant service providers, whose subjective descriptions might not accurately depict the service functionality. In addition, mashup developers may find new ways, beyond service publishers' initial thoughts, to use the services. For example, an originally published "social" service is widely used in mashups focusing on "books."[2]

Second, service profiles typically comprise quite some background terms that do not distinguish services. For instance, the description of service *Facebook* says "*The Facebook API is a platform for building applications that are available*

- *Junqi Zhang and Yushun Fan are with the Department of Automation, Tsinghua University, Beijing, China. E-mail: jq-zhang15@mails.tsinghua.edu.cn, fanyus@tsinghua.edu.cn.*
- *Jia Zhang is with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Moffett Field, CA. E-mail: jia.zhang@sv.cmu.edu.*
- *Bing Bai is with the Cloud and Smart Industries Group, Tencent, Beijing, China. E-mail: icebai@tencent.com.*

Manuscript received September 1, 2019.

1. https://www.w3.org/Submission/OWL-S
2. Detailed examples are presented in Section 5.3.4.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2020.3001307, IEEE Transactions on Services Computing

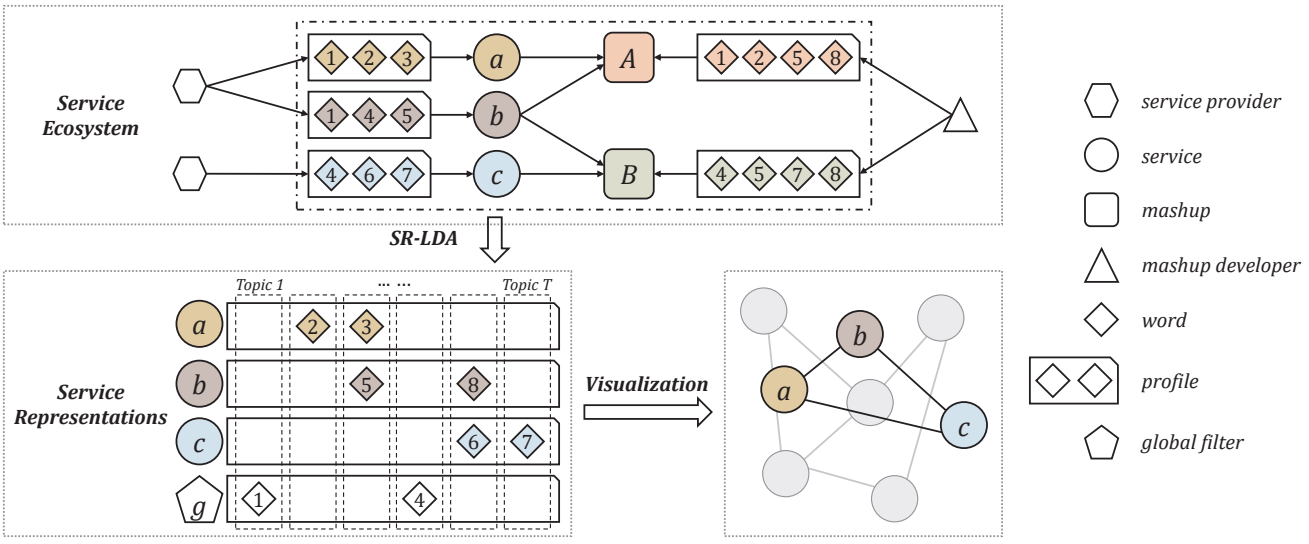


Fig. 1: The basic idea of the proposed SR-LDA and visualization. Service representations are constructed from profiles of both mashups and services, as well as the invocation records, in a service ecosystem. Additionally, a global filter is introduced to identify and down-weight background terms. A service knowledge map is generated for visualization.

to the members of the social network of Facebook..." In this profile description, the terms "API," "platform," and "applications" are service ecosystem-common background terms that possibly appear in various Web services. Such words describing shared features are irrelevant to services' specific characteristics, which means background terms bring an adverse impact to service representations.

To address the aforementioned two issues, our position is that, service representations shall be up-to-date, objective and distinct from each other. More specifically, we believe that service users' opinions should be taken into consideration and automatically incorporated into service representations, while background terms should be filtered out. We hypothesize that service users' opinions are hidden and reflected in service usage history, which may be recorded by service composition (i.e., mashup) profiles. In other words, the profiles of mashups carry the supplementary information of the comprising services invoked by the mashups. Note that service users' opinions may also be included in their comments or annotations, which however might not be verified thus will not be considered in this paper.

Based on our previous work [10], we propose to extend the ordinary Author-Topic Model [11], [12] into a tailored topic model, named Service Representation-Latent Dirichlet Allocation (SR-LDA), to build high-quality services representations through learning from the underneath service ecosystem and then generate a knowledge map of Web services. The core idea of this paper is illustrated in Fig. 1. A service ecosystem comprises a collection of services and mashups, as well as their invocation relationships. For example, mashup A invokes services a and b; mashup B invokes services b and c. The profiles of the constituent services and mashups altogether form a corpus comprising a collection of words (terms), i.e., word 1 to word 8.

Fig. 1 shows that the original profile of service b contains words 1, 4, and 5. However, our probabilistic model (that will be discussed in detail later) finds that both mashups A and B adopt word 8 to describe service b. Thus, after studying the service ecosystem, word 8 is incorporated into the representation of service b, even though it hasn't been used in service profiles. Meanwhile, since words 1 and 4 appear roughly uniformly in most service and mashup profiles, they are caught by an introduced concept of "global filter"[3] and identified as background terms to mitigate their adverse impact on service representations. Note that our model also categorizes the words into topics.

The main contributions of this paper are three-fold:

- We have proposed a tailored machine learning model to build accurate service representations. The model can effectively extract supplementary information about services from service usage history, while automatically filtering out background terms.
- We have tested our model over the real-world ProgrammableWeb dataset. Both quantitative and qualitative analyses have demonstrated that our model builds higher-quality representations of services comparing with baselines.
- We have presented a proper visualization of a service ecosystem based on the learned service representations. Four typical functionality patterns of Web services are derived from a knowledge map. The assistance of this map in mashup creation is also illuminated.

The rest of this paper is organized as follows. Section 2 describes the model framework. Section 3 shows how to train the optimal parameters. Section 4 reports the experiments and case studies. Section 5 describes how the knowledge map can be generated and our findings. Section 6 discusses the related work and Section 7 draws conclusions.

3. Specifically defined in Section 2.2.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2020.3001307, IEEE Transactions on Services Computing

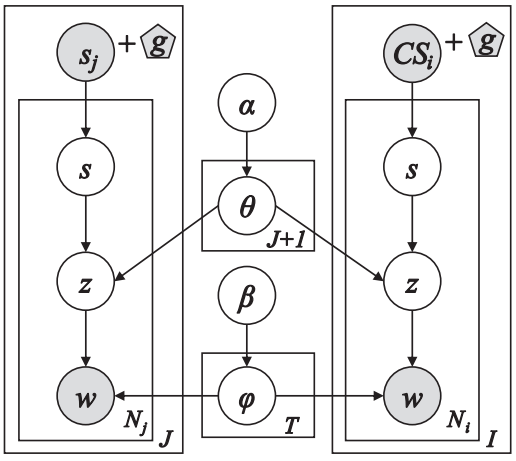


Fig. 2: Graphical model of SR-LDA. Words in the profiles of services and mashups are assigned to proper services or the global filter.

2 Model Framework

In this section, we will first introduce the definitions of the notations used for Web service ecosystems, then describe the framework of our model.

2.1 Background notations

The definitions of the notations used for mashups and services are defined as follows.

For services, in this paper, we use subscript "j" to denote that one notation is related to service s_j, and $j = 1 : J$. Service description $\mathcal{SD}_j = \{w_{j1}, w_{j2}, \ldots, w_{jn_j}\}$ is the set of n_j words (terms) used in the profile of service s_j.

For mashups, we use subscript "i" to denote that one notation is related to mashup m_i, and $i = 1 : I$. $\mathcal{CS}_i = \{cs_{i1}, cs_{i2}, \ldots, cs_{ih_i}\}$ is the set of h_i component services invoked by mashup m_i, and $\mathcal{MD}_i = \{w_{i1}, w_{i2}, \ldots, w_{in_i}\}$ is the set of n_i words used to describe mashup m_i.

As illustrated above, in this model we mainly use the invocation relationships between mashups and services, as well as the profiles of services and mashups. Note that the profiles include, but are not limited to, content descriptions, category information, tags and WSDL documents.

2.2 Generative process of SR-LDA

The original service profiles are static and may contain service ecosystem-common background terms. Thus the representations built directly from service profiles may not be effective. In this paper, we incorporate mashup profiles as additional sources of information about services and try to filter out the background words automatically. To achieve the goals, we define a tailored generative process Service Representation-Latent Dirichlet Allocation (SR-LDA), whose graphical model is shown in Fig. 2, to simulate how the profiles of a service ecosystem are formed in terms of words.

Figuratively speaking, we regard services as "authors" of the profiles of both services and mashups. On one hand, every service is the author of its own profile; on the other hand, every mashup profile is co-authored by its comprising services. Additionally, an introduced "global filter" is shared by all the profiles in a service ecosystem as the author of background terms. For conciseness, we use the subscript "$J + 1$" to denote that one notation is related to the global filter. For example, notation $\boldsymbol{\theta}_{J+1}$ represents the topic proportions of the global filter.

The probabilistic generative process of SR-LDA is defined as follows. Assuming that there are T topics,

1) For each topic $z = 1 : T$, draw word proportions $\boldsymbol{\phi}_z \sim \text{Dirichlet}(\beta)$.
2) For the global filter, draw topic proportions $\boldsymbol{\theta}_{J+1} \sim \text{Dirichlet}(\alpha)$.
3) For each service s_j, $j = 1 : J$,
   a) Draw topic proportions $\boldsymbol{\theta}_j \sim \text{Dirichlet}(\alpha)$
   b) For each word $w_{jn} \in \mathcal{SD}_j$,
      i) Draw a service assignment $s_{jn} \sim \text{Uniform}(\{j, J+1\})$
      ii) Conditioned on s_{jn}, draw a topic assignment $z_{jn} \sim \text{Mult}(\boldsymbol{\theta}_{s_{jn}})$.
      iii) Conditioned on z_{jn}, draw the word $w_{jn} \sim \text{Mult}(\boldsymbol{\phi}_{z_{jn}})$.
4) For each mashup m_i, $i = 1 : I$,
   a) For each word $w_{in} \in \mathcal{MD}_i$,
      i) Draw a service assignment $s_{in} \sim \text{Uniform}(\mathcal{CS}_i \bigcup \{J+1\})$
      ii) Conditioned on s_{in}, draw a topic assignment $z_{in} \sim \text{Mult}(\boldsymbol{\theta}_{s_{in}})$.
      iii) Conditioned on z_{in}, draw the word $w_{in} \sim \text{Mult}(\boldsymbol{\phi}_{z_{in}})$.

As shown in Fig. 1 and discussed in Section 1, when a mashup invokes multiple services (e.g., mashup A invokes services a and b), it is difficult yet necessary to identify to which service each word in the mashup profile is corresponding. For example, how to infer that word 8 is used to describe service b. Based on the generative process of SR-LDA, we can figure out the corresponding relations between words and services via maximizing the posterior probability in Section 3.3. In this way, words that are used to describe mashups can also fertilize the representations of services (i.e., the topic proportions $\boldsymbol{\theta}_{1:J}$). Furthermore, as the global filter corresponds to the profiles of all the services and mashups, background terms that appear in many profiles will be assigned to the global filter, thus the model can automatically identify background terms and then down-weight their adverse impact on the representations.

3 Parameter Learning

Effective service representations can be achieved by maximizing a posterior estimation of the generative SR-LDA model. Assuming that T topics exist, we are interested in learning two sets of variables: (1) service-topic proportions $\boldsymbol{\Theta} = \boldsymbol{\theta}_{1:J+1}$, and (2) topic-word proportions $\boldsymbol{\Phi} = \boldsymbol{\phi}_{1:T}$. Our aim is to maximize the posterior distribution on $\boldsymbol{\Theta}$ and $\boldsymbol{\Phi}$. In this section, firstly we will merge the data from the service part and the mashup part for conciseness, then we will

derive the probability of the generation of content and how to obtain the maximum a posteriori estimates, finally we will analyze the computational complexity of our approach.

3.1 Notation refinement

Since the data from service profiles and mashup profiles shares similar structure, for conciseness, we refine the notations by merging them for explaining the model training procedure. We define

$$\mathcal{D} = \{\mathcal{SD}_1, \ldots, \mathcal{SD}_J, \mathcal{MD}_1, \ldots, \mathcal{MD}_I\},$$

in other words, $\mathcal{D}$ contains all the profiles in a service ecosystem, and $|\mathcal{D}| = I + J$.

Similarly, we define the "authors" of all the profiles as follows,

$$\mathcal{S} = \Big\{\big\{1, J+1\big\}, \ldots, \big\{J, J+1\big\}, \big\{\mathcal{CS}_1 \bigcup \{J+1\}\big\}, \ldots, \big\{\mathcal{CS}_I \bigcup \{J+1\}\big\}\Big\},$$

in other words, S contains the global filter besides the corresponding services associated with $\mathcal{D}$. The subscript "k" denotes that the notation is related to the merged variables.

3.2 Probability of corpus generation

Under the generative process defined in Section 2.2, the words (terms) are drawn independently when conditioned on $\mathbf{\Theta}$ and $\mathbf{\Phi}$. Thus, the probability of the corpus $\mathbf{w}$ generated is

$$P(\mathbf{w}|\mathbf{\Theta}, \mathbf{\Phi}, \mathcal{S}) = \prod_{\mathbf{w}} P(w_{kn}|\mathbf{\Theta}, \mathbf{\Phi}, \mathcal{S}_k),$$

where w_{kn} is the nth word token in the kth merged profile, $\mathbf{w}$ is the vector recording all the word tokens, and $\mathcal{S}_k$ is the corresponding set for the kth merged profile.

Then we can obtain the probability of word token w_{kn} being generated, based on the probability assumptions defined in Section 2.2, i.e.,

$$\begin{aligned}
&P(w_{kn}|\mathbf{\Theta}, \mathbf{\Phi}, \mathcal{S}_k)\\
=&\sum_{j=1}^{J+1}\sum_{t=1}^{T} P(w_{kn}, z_{kn}=t, s_{kn}=j|\mathbf{\Theta}, \mathbf{\Phi}, \mathcal{S}_k)\\
=&\sum_{j=1}^{J+1}\sum_{t=1}^{T} P(w_{kn}|z_{kn}=t, \boldsymbol{\phi}_t)\\
&\qquad P(z_{kn}=t|s_{kn}=j, \boldsymbol{\theta}_j)P(s_{kn}=j|\mathcal{S}_k)\\
=&\frac{1}{|\mathcal{S}_k|}\sum_{j\in\mathcal{S}_k}\sum_{t=1}^{T}\phi_{w_{kn}t}\theta_{tj},
\end{aligned}$$

where $\phi_{w_{kn}t}$ is the probability of word token w_{kn} conditioned on t, θ_{tj} is the probability of topic t conditioned on service j, and $\mathcal{S}_k$ is the component service set of the kth profile, added with the global filter. According to the uniform assumption, $P(s_{kn}=j|\mathcal{S}_k) = \frac{1}{|\mathcal{S}_k|}$ when $j \in \mathcal{S}_k$, and 0 otherwise.

Based on the analyses above, the conditional probability of $\mathbf{w}$ becomes:

$$\begin{aligned}
&P(\mathbf{w}|\mathcal{S}, \alpha, \beta)\\
=&\iint P(\mathbf{w}|\mathbf{\Theta}, \mathbf{\Phi}, \mathcal{S})p(\mathbf{\Theta}, \mathbf{\Phi}|\alpha, \beta)\mathrm{d}\mathbf{\Theta}\mathrm{d}\mathbf{\Phi}\\
=&\iint \Big(\prod_{w_{kn}\in\mathbf{w}}\frac{1}{|\mathcal{S}_k|}\sum_{j\in\mathcal{S}_k}\sum_{t=1}^{T}\phi_{w_{kn}t}\theta_{tj}\Big)p(\mathbf{\Theta}|\alpha)p(\mathbf{\Phi}|\beta)\mathrm{d}\mathbf{\Theta}\mathrm{d}\mathbf{\Phi},
\end{aligned}$$

and according to the probability distribution assumption, $p(\mathbf{\Theta}|\alpha)$ and $p(\mathbf{\Phi}|\beta)$ are the Dirichlet priors on $\mathbf{\Theta}$ and $\mathbf{\Phi}$, respectively.

3.3 Maximum a posteriori estimation

Our inference scheme is based upon maximizing the observation that

$$\begin{aligned}
&p(\mathbf{\Theta}, \mathbf{\Phi}|\mathcal{D}, \mathcal{S}, \alpha, \beta)\\
&\qquad = \sum_{\mathbf{z},\mathbf{s}} p(\mathbf{\Theta}, \mathbf{\Phi}|\mathbf{z}, \mathbf{s}, \mathcal{D}, \mathcal{S}, \alpha, \beta)P(\mathbf{z}, \mathbf{s}|\mathcal{D}, \mathcal{S}, \alpha, \beta),
\end{aligned}$$

where $\mathbf{z} = \{z_{kn}\}$ is the topic assignments for the words in D, and $\mathbf{s} = \{s_{kn}\}$ is the service assignments.

An approximate posterior on $\mathbf{\Theta}$ and $\mathbf{\Phi}$ can be obtained by applying variational inference, expectation propagation or Markov-chain Monte Carlo (MCMC) schemes. In this paper, we adopt a MCMC method, i.e., Gibbs Sampling, to approximate our SR-LDA model. First an empirical sample-based estimate of $P(\mathbf{z}, \mathbf{s}|\mathcal{D}, \mathcal{S}, \alpha, \beta)$ can be obtained. Afterwards we can get the expectation of $\mathbf{\Theta}$ and $\mathbf{\Phi}$ directly by exploiting the conjugation of Dirichlet distribution and multinomial distribution.

The Gibbs sampler corresponding for $P(\mathbf{z}, \mathbf{s}|\mathcal{D}, \mathcal{S}, \alpha, \beta)$ is expressed in the following basic equation:

$$\begin{aligned}
&P(s_{kn}=j, z_{kn}=t|w_{kn}=w, \mathbf{z}^{\neg kn}, \mathbf{s}^{\neg kn}, \mathcal{D}^{\neg kn}, \mathcal{S}, \alpha, \beta)\\
\propto&\frac{g_{jt}^{\neg kn}+\alpha}{\sum_{t'} g_{jt'}^{\neg kn}+T\alpha}\times\frac{c_{tw}^{\neg kn}+\beta}{\sum_{w'} c_{tw'}^{\neg kn}+W\beta},
\end{aligned} \tag{1}$$

where g_{jt} is the number of word tokens assigned to topic t and service j at the same time, i.e., $g_{jt} = \sum I(z_{kn}=t, s_{kn}=j)$, c_{tw} is the frequency at which word w is assigned to topic t, i.e., $c_{tw} = \sum I(w_{kn}=w, z_{kn}=t)$, and $I(\cdot)$ is the indicator function. The superscript $\neg$ denotes a quantity excluding the current instance.

Starting with random initialization on the service assignments $\mathbf{s}$ and topic assignments $\mathbf{z}$, we can use the Gibbs sampler for sampling. After sampling a sufficient number of burn-in iterations, the sampler will converge and approach the stationary distribution. We can then accumulate the results for several iterations, average them and compute the expectation of the true posterior. Given $\mathbf{z}$, $\mathbf{s}$, $\mathcal{D}$, α and β, we can get that $\mathbf{\Theta}$ and $\mathbf{\Phi}$ follow simple Dirichlet distributions based on the assumptions in Section 2.2 and the fact that the Dirichlet distribution is conjugate prior for the multinomial distribution, i.e.,

$$\boldsymbol{\phi}_t|\mathbf{z}, \mathcal{D}, \beta \sim \text{Dirichlet}(\boldsymbol{c}_{t\cdot} + \beta)$$

$$\boldsymbol{\theta}_j|\mathbf{z}, \mathbf{s}, \alpha \sim \text{Dirichlet}(\boldsymbol{g}_{j\cdot} + \alpha),$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2020.3001307, IEEE Transactions on Services Computing

TABLE 1: Notations Used in SR-LDA

Symbol / Type	Description
i / Scalar	Subscript for mashups
j / Scalar	Subscript for services
k / Scalar	Subscript for merged variables
$\mathcal{MD}_i$ / Set	Profile content for mashup i
$\mathcal{SD}_j$ / Set	Profile content for service j
$\mathcal{D} = \{\mathcal{D}_k\}$ / Set	$\mathcal{D}_k$ is the kth merged profile content
$\mathbf{w} = \{w_{kn}\}$ / $\lvert\mathbf{w}\rvert$-dimensional vector	w_{kn} is the word for nth word token in $\mathcal{D}_k$
W / Scalar	Vocabulary size
$\mathcal{CS}_i$ / Set	Component services of mashup i
$\mathcal{S} = \{\mathcal{S}_k\}$ / Set	S_k is the corresponding service set for $\mathcal{D}_k$
T / Scalar	Number of topics
$\mathbf{\Phi} = \boldsymbol{\phi}_{1:T}$ / $T \times W$ matrix	Topic-word proportions. ϕ_{tw} indicates the probability of word w given topic t
$\mathbf{\Theta} = \boldsymbol{\theta}_{1:J+1}$ / $(J+1) \times T$ matrix	Service-topic proportions, and $\boldsymbol{\theta}_{J+1}$ is the topic proportions for the global filter. θ_{jt} indicates the probability of topic t given service j
α / Scalar	Dirichlet prior for service-topic proportions
β / Scalar	Dirichlet prior for topic-word proportions
$\mathbf{z} = \{z_{kn}\}$ / $\lvert\mathbf{w}\rvert$-dimensional vector	z_{kn} is the topic assignment for w_{kn}
$\mathbf{s} = \{s_{kn}\}$ / $\lvert\mathbf{w}\rvert$-dimensional vector	s_{kn} is the service assignment for w_{kn}
g_{jt} / Scalar	Number of word tokens assigned to topic t and service j at the same time
c_{tw} / Scalar	Frequency at which word w is assigned to topic t
N_{burn} / Scalar	Number of burn-in iterations
N_{acc} / Scalar	Number of accumulation iterations

where $\boldsymbol{c}_{t\cdot}$ is the vector containing $c_{tw}, w = 1, \ldots, W$, and $\boldsymbol{g}_{j\cdot}$ is the vector containing $g_{jt}, t = 1, \ldots, T$.

Finally we can reach the expectation of $\mathbf{\Theta}$ and $\mathbf{\Phi}$ for any given instances $\mathbf{z}$ and $\mathbf{s}$, i.e.,

$$E[\phi_{tw}|\mathbf{z}, \mathcal{D}, \beta] = \frac{c_{tw} + \beta}{\sum_{w'} c_{tw'} + W\beta} \tag{2}$$

$$E[\theta_{jt}|\mathbf{z}, \mathbf{s}, \alpha] = \frac{g_{jt} + \alpha}{\sum_{t'} g_{jt'} + T\alpha}. \tag{3}$$

The notations used here are summarized in Table 1. Assuming that the number of burn-in iterations is N_{burn} and the number of accumulation iterations is N_{acc}, the pseudo code of parameter learning is listed in Algorithm 1.

3.4 Computational complexity analysis

For each step of Gibbs sampling, we have to use g_{jt} and c_{tw}. However, as $\mathbf{z}$ and $\mathbf{s}$ only change one instance each sampling, g_{jt}, we decide to cache $\sum_{t'} g_{jt'}$, c_{tw} and $\sum_{w'} c_{tw'}$, and update them during a constant time period. The algorithm thus can become more efficient.

Algorithm 1: Parameter learning of SR-LDA

Input: α, β, $\mathcal{D}$, $\mathcal{S}$, T, N_{burn} and N_{acc}

Output: Optimal $\mathbf{\Theta}$ and $\mathbf{\Phi}$

Procedure:

```
01  Initialize z and s randomly
02  Calculate and cache g_jt, Σ_t' g_jt', c_tw and Σ_w' c_tw'
03  For iter = 1 : N_burn + N_acc
04    For each word token w_kn
05      Sample z_kn and s_kn according to Eq. (1)
06      Update g_jt, Σ_t' g_jt', c_tw and Σ_w' c_tw'
07    End
08    If iter ≥ N_burn + 1
09      Record the sampling results of z_kn and s_kn
10    End
11  End
12  Calculate the average of recorded z and s
13  Calculate expectational Φ according to Eq. (2)
14  Calculate expectational Θ according to Eq. (3)
```

The time complexity of initialization for Line 01 of Algorithm 1 is bounded by $O(|\mathbf{w}|)$, where $|\mathbf{w}|$ is the total number of word tokens, for we need to give every word token an assignment of the topic and an assignment of the service. The time complexity for Line 02, i.e., building the cache, is bounded by $O(|\mathbf{w}| + J \cdot T + T \cdot W)$, for we need to go through all words' assignment and count g_{jt}, $\sum_{t'} g_{jt'}$, c_{tw} and $\sum_{w'} c_{tw'}$ generated in Line 01. If we assume that each mashup invoke $\overline{h}$ services on average, for every word token, we need to go through all the possible topic and service assignments, so the time complexity of every Gibbs sampling iteration is bounded by $O(|\mathbf{w}|T\overline{h})$, thus the time complexity of Line 03 to Line 11 is $O\big((N_{burn}+N_{acc})|\mathbf{w}|T\overline{h}\big)$. For Line 12 to Line 14, we calculate the expectations for $\mathbf{\Theta}$ and $\mathbf{\Phi}$, and the time complexity is $O(N_{acc}|\mathbf{w}|+|\mathbf{w}|T+J\cdot T)$.

Considering that $|\mathbf{w}|$ is the number of word tokens in the corpus, it is safe to say that $|\mathbf{w}|$ is far greater than J and W. Therefore, the major time complexity of Algorithm 1 comes from the Gibbs sampler. We can thus conclude that the total time complexity of SR-LDA is bounded by $O\big((N_{burn} + N_{acc})|\mathbf{w}|T\overline{h}\big)$, and $|\mathbf{w}|$ is roughly in proportion to $(I + J)$.

4 Experiments

We have designed and conducted a collection of experiments to evaluate the effectiveness of our approach[4]. In this section, we will first introduce the dataset we used for evaluation, the evaluation metric and the baseline methods. Then we will discuss the quantitative and qualitative results, comparing with the baselines. Additionally, we will show case studies on the quality of word-service assignment.

4. The dataset and results are published at http://www.simflow.net/team/baibing/sr-lda.zip

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2020.3001307, IEEE Transactions on Services Computing

TABLE 2: Statistical Information about ProgrammableWeb Dataset

Total # of services	13,931
Total # of services invoked by mashups	1,241
Total # of mashups	6,295
Vocabulary Size	9,555
Average # of services invoked by mashups	2.06
Average # of word tokens in service and mashup profiles	34.82

4.1 Dataset

ProgrammableWeb.com is to-date the largest online repository of public Web services (APIs) and their mashups [13]. Since it provides APIs for people to fetch its stored data, ProgrammableWeb.com has been used as a testbed in many Web service researches [13], [14], [15], [16]. The profiles of services and mashups include their textual descriptions as well as category tags. After preprocessing the profiles by stemming and removing stop words, we obtained a vocabulary size of 9,555. Detailed statistical information about the dataset is summarized in Table 2.

4.2 Evaluation scheme

Recall that the mission of SR-LDA is to build accurate representations for services from the service ecosystem. However, the accuracy of service representations is hard to measure directly. Intuitively, high-quality representations should be able to: (1) reflect the similarities of services in the same domain; and (2) reflect the differences among services in different domains. Based on such a perspective, we designed a three-phase method as below to evaluate the quality of service representations generated by SR-LDA.

First, similar with [17], we perform a K-means clustering method over the generated service representations, to divide them into clusters. Second, we calculate the Davies-Bouldin Index (DBI) [18] between each pair of service clusters resulted. DBI@K is defined as follows:

$$\text{DBI@}K = \frac{1}{K}\sum_{i=1}^{K}\max_{j\neq i}\Big(\frac{\text{avg}(\mathcal{C}_i)+\text{avg}(\mathcal{C}_j)}{d_{cen}(\boldsymbol{\mu}_i,\boldsymbol{\mu}_j)}\Big),$$

where

$$\text{avg}(\mathcal{C}) = \frac{2}{|\mathcal{C}|(|\mathcal{C}|-1)}\sum_{1\leq i<j\leq|\mathcal{C}|}\text{dist}(\boldsymbol{x}_i,\boldsymbol{x}_j)$$

is the average distance within cluster $\mathcal{C}$, K is the number of clusters, and $d_{cen}(\boldsymbol{\mu}_i,\boldsymbol{\mu}_j)$ indicates the distance between the centers of two clusters $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}_j$. Third, the Calinski-Harabasz Index (CHI) [19] is also calculated for double-checking the quality of generated service representations. CHI@K is defined as:

$$\text{CHI@}K = \frac{\text{tr}(\mathbf{B}_K)}{\text{tr}(\mathbf{W}_K)}\frac{J-K}{K-1},$$

where

$$\mathbf{B}_K = \sum_{i=1}^{K}|\mathcal{C}_i|(\boldsymbol{\mu}_i-\bar{\boldsymbol{x}})(\boldsymbol{\mu}_i-\bar{\boldsymbol{x}})^{\mathrm{T}}$$

is the inter-cluster dispersion, $\bar{\boldsymbol{x}}$ represents the global centroid of services, and

$$\mathbf{W}_K = \sum_{i=1}^{K}\sum_{\boldsymbol{x}\in\mathcal{C}_i}(\boldsymbol{x}-\boldsymbol{\mu}_i)(\boldsymbol{x}-\boldsymbol{\mu}_i)^{\mathrm{T}}$$

is the intra-cluster dispersion.

A lower value of DBI or a higher value of CHI indicates more effective service representations, meaning that services within a cluster are more similar to each other, and the clusters are separated better, which corresponds to the aforementioned intuition.

4.3 Baselines and hyperparameter settings

In the experiments, we designed two baseline methods for comparison:

- **Vanilla LDA**. For this baseline, we used the original profiles of services and applied the vanilla Latent Dirichlet Allocation (LDA) [20] to extract topics of each service. This baseline can provide an evidence for how well the vanilla LDA works in service ecosystems.
- **DSR-LDA (Degenerated SR-LDA)**. For this method, we abandoned the global filter in the proposed SR-LDA. This method is equivalent to a customized application of the Author-Topic Model [11]. A comparison with this baseline can provide evidence for both the adverse impact on building service representations brought by background terms and the effectiveness of the global filter.

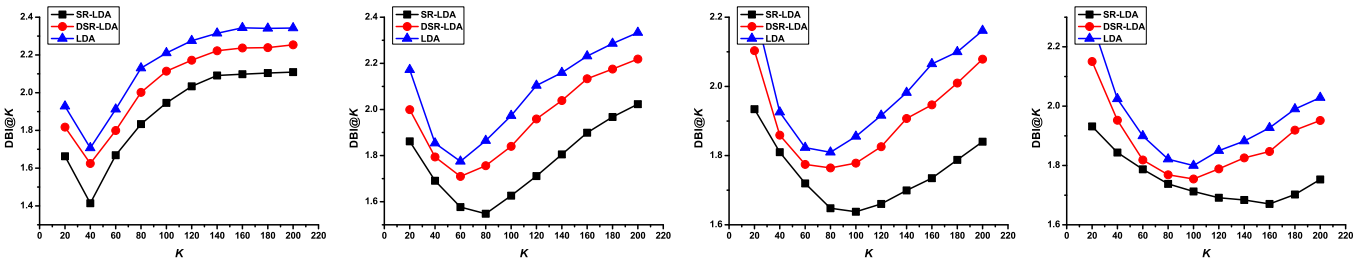
For all runs of our SR-LDA model and baselines, we adopted $\alpha = 50/T$ and $\beta = 0.01$ according to the empirical formula [21]. We computed the results when the topic number T equals to 40, 80, 120 and 160. For all values of T, we ran five different Gibbs sampling chains over the whole dataset, discarding the first 8,000 iterations for burn-in, and then took the average of the next 2,000 iterations as final results. For the quantitative evaluation, the DBI and CHI results reported are the average of these five chains, and the K-means clustering is performed when K equals to 20, 40, 60, $\cdots$, 180, and 200. For qualitative evaluation and knowledge map generation, we set T to 120 based on the inflection point of perplexity curve [20].

4.4 Quantitative comparisons

The DBI results of the SR-LDA and baselines with different topic numbers T are shown in Fig. 3, and Table 3 presents the average CHI results over different cluster numbers K. It can be found that at all tested topic numbers T and cluster numbers K, the baseline LDA receives the highest DBI and SR-LDA performs the best. The proposed SR-LDA also achieves the best performance according to the average CHI results. Comparing the results of baseline LDA and DSR-LDA, it can be found that considering the mashup profiles can increase the quality of service representations. While comparing the results of the proposed SR-LDA and the baselines, another two interesting findings were exposed.

Firstly, the gaps of both the DBI and CHI between SR-LDA and DSR-LDA are generally more significant than

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2020.3001307, IEEE Transactions on Services Computing



(a) $T = 40$ (b) $T = 80$ (c) $T = 120$ (d) $T = 160$

Fig. 3: DBI results for the proposed SR-LDA and baselines. At all tested topic numbers T and cluster numbers K, SR-LDA shows superior performance.

TABLE 3: Average CHI Results for SR-LDA and Baselines

	$T = 40$	$T = 80$	$T = 120$	$T = 160$
SR-LDA	**566.64**	**352.54**	**255.80**	**209.65**
DSR-LDA	394.60	282.44	231.17	196.45
LDA	364.57	271.27	226.96	192.21

those between DSR-LDA and LDA, indicating that background terms are bringing more adverse impact on building high-quality service representations. The supplementary information from mashup profiles won't bring much improvement in the quality of service representations unless the background terms are filtered out by the global filter.

Secondly, Fig. 3 reveals that the DBI results of SR-LDA show a different trend compared with the results of the two baselines, especially when T is set to be large. For example, when T is set to 120, the optimal DBI for SR-LDA is obtained when $K = 100$, while optimal DBI values for LDA and DSR-LDA are obtained when $K = 80$. If we enlarge T to 160, the difference becomes more significant. This phenomenon shows that without filtering out background terms, LDA and DSR-LDA cannot guarantee that all topics learned from data are effective for identifying the functionalities of services.

As a conclusion, SR-LDA achieves lower DBI values and higher CHI values at a variety of topic numbers T and cluster numbers K, indicating that service representations resulted from SR-LDA are more accurate.

4.5 Qualitative analyses

We also conducted qualitative analyses on the topics given by different methods. Table 4 reports top words and services in three example topics. The left part is the results given by SR-LDA, and the right part is the information of the corresponding topics given by DSR-LDA. Here, we report the results of SR-LDA and DSR-LDA, by comparing which can provide information about the role of the global filter.

As illustrated in Table 4, for SR-LDA, words in Topic #43 are general background terms used in service ecosystems, and words in Topic #96 are technical background terms. SR-LDA can assign them to the global filter, thus down-weighting the impact of those words. As for DSR-LDA, Topic #23 and #29 are the most similar topics to the corresponding topics. As DSR-LDA cannot filter these background terms, the quality of service representations suffers from the adverse impact of them.

For the last topic pair reported in the table, we can conclude that the topics are weather-related. As weather information is seldom used without location, in practice, Web services like *Google Maps* are also related to this topic. Both methods can discover this phenomenon from the profiles of mashups. As "Mapping" category are centralized, and "Weather" category is relatively evenly, given this topic, the probability of *Google Maps* would be even larger than any specific weather-related Web service. However, DSR-LDA cannot automatically filter background terms. Thus there are words like "data" and "provide" listed in this topic, and some environmentally-friendly energy-related services like *Clean Power SolarAnywhere* and *AMEE* also show up on this topic. This shows that without filtering background terms, the quality of function-related topics will also suffer.

As a conclusion, SR-LDA is able to learn more precise topic distributions with the help of the global filter.

4.6 Case studies: corresponding service inference

Since SR-LDA assigns words from mashup's profiles to component services, we also conducted case studies on the quality of word-service assignment. Results are listed in Table 5.

4.6.1 *Case 1: ShopTalk*

ShopTalk is a mashup helping users to track orders from any phone. It invokes two services, i.e., *Cloudvox* that offers Web-based telephone services, and *Shopify* that gives access to the orders. For the words like "application," "shop," "phone" and "order," the model is quite sure about to which service they are corresponding. However, for words like "call," it could mean "call through the phone," while it could also mean "call the API." So the model is not 100% sure about which service the word should be assigned to, but it also gives a 74.01% confidence to *Cloudvox*, which is the right choice in this case.

4.6.2 *Case 2: Episkeptis*

Episkeptis is a mashup offering restaurant searching and sharing functionalities. It invokes two services, i.e., *Facebook* that offers social functionalities, and *Google Maps* that helps searching restaurants and bars. Interestingly, although there are no words like "restaurant" or "bar" in the profile of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2020.3001307, IEEE Transactions on Services Computing

TABLE 4: Top Words and Services in Example Topics

SR-LDA					DSR-LDA				
t	w	$P(w\|t)$	s	$P(s\|t)$	t	w	$P(w\|t)$	s	$P(s\|t)$
#43	web	13.89%	#global filter#	75.35%	#23	web	9.99%	*Google Maps*	0.55%
	website	10.38%	*NationBuilder*	0.10%		base	6.13%	*Flickr*	0.52%
	tool	7.74%	*Amazon S3*	0.08%		interact	5.89%	*OpenLayers*	0.30%
	enable	6.06%	*Fitbit*	0.06%		javascript	4.95%	*Twitter*	0.25%
	help	5.54%	*Google Analytics Managment*	0.06%		design	4.40%	*Google Maps Flash*	0.20%
	interface	4.20%	*Bing*	0.05%		feature	3.06%	*AnyChart*	0.18%
	software	3.31%	*PayPal*	0.03%		build	3.01%	*Yahoo Maps*	0.17%
#96	data	10.29%	#global filter#	90.04%	#29	format	18.80%	*Google Maps*	0.14%
	RESTful	6.13%	*PriceGrabber*	0.01%		response	14.81%	*Yandex Bar*	0.10%
	JSON	4.31%	*UniGraph*	0.00%		RESTful	14.47%	*Yandex Webmaster*	0.08%
	format	4.12%	*iCasework UsefulFeedback*	0.00%		call	14.13%	*Bank of Russia Daily Info*	0.07%
	online	3.42%	*eRail.in Indian Railways*	0.00%		JSON	13.94%	*Mail.Ru*	0.07%
	return	3.19%	*BigCommerce*	0.00%		xml	11.15%	*Yandex Money*	0.07%
	response	2.67%	*FlightAware*	0.00%		let	4.12%	*Yandex Metrica*	0.07%
#110	weather	21.67%	*Google Maps*	2.71%	#61	weather	14.13%	*Google Maps*	3.99%
	forecast	4.90%	*Weather Underground*	0.79%		energy	4.90%	*Weather Underground*	0.69%
	science	4.30%	*NOAA National Weather Service*	0.70%		data	4.47%	*WeatherBug*	0.61%
	condition	3.09%	*WeatherBug*	0.68%		environment	4.37%	*NOAA National Weather Service*	0.58%
	astronomy	3.08%	*Microsoft Bing Maps*	0.37%		forecast	3.20%	*Clean Power SolarAnywhere*	0.37%
	nasa	2.26%	*OpenWeatherMap*	0.37%		provide	2.77%	*AMEE*	0.37%
	earth	2.06%	*Weather Channel*	0.35%		condition	2.44%	*OpenWeatherMap*	0.35%

TABLE 5: Example Mashup Descriptions and the Results about Corresponding Service Inference

Mashup		*ShopTalk*						
Description		A *Shopify* shop owner adds the *ShopTalk* application to their shop, then can call into their shop from any phone via *Cloudvox*. After identifying using a PIN, shopkeeper can hear total orders. Runs on *Heroku*.						
Words		application	shop	call	phone	PIN	order	run
Component services	*Cloudvox*	4.88%	3.70%	74.01%	95.65%	77.27%	6.09%	19.62%
	Shopify	3.89%	96.24%	3.67%	4.26%	22.33%	91.73%	17.01%
	#global filer#	91.23%	0.06%	22.33%	0.09%	0.40%	2.18%	63.37%
Mashup		*Episkeptis*						
Description		*Episkeptis* helps users identify the best restaurants, bars, clubs and cafes in their city, with the help of their *Facebook* friends. Users are able to rate, recommend and share their favorites. Site is in Greek.						
Words		help	restaurant	bar	friend	recommend	share	site
Component services	*Facebook*	2.61%	13.87%	4.94%	95.83%	91.08%	88.90%	17.54%
	Google Maps	0.12%	86.04%	94.84%	4.17%	8.83%	1.27%	2.27%
	#global filer#	97.26%	0.09%	0.22%	0.00%	0.09%	9.83%	80.19%
Mashup		*Music Enthusiast*						
Description		Search for your favorite artist. Be able to visually see locations of their upcoming concerts and events. Check out their hottest videos on *YouTube*.						
Words		artist	able	visual	location	upcome	concert	video
Component services	*Eventful*	34.27%	4.57%	17.72%	16.85%	92.32%	89.09%	0.37%
	Google Maps	1.18%	0.78%	53.56%	70.40%	1.37%	2.30%	1.18%
	YouTube	64.55%	0.40%	28.68%	0.37%	6.25%	8.58%	98.41%
	#global filer#	0.00%	94.25%	0.03%	12.38%	0.06%	0.03%	0.03%

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2020.3001307, IEEE Transactions on Services Computing

Google Maps, the model can still be pretty sure that these words are related to the service *Google Maps*. In fact, the model has learned such characteristics from other mashups' profiles, indicating that bringing mashups' profiles is quite effective in helping to build services' representations. A similar phenomenon about *Facebook* can also be witnessed, for word "recommend" does not show up in *Facebook*'s profile, but our model is quite sure that they are related to each other.

4.6.3 Case 3: Music Enthusiast

Music Enthusiast is a mashup designed for music fans. It invokes three services, i.e., *Eventful* and *Google Maps* for visualization of upcoming concerts and events, and *YouTube* for the hottest videos of artists. This case is relatively more complex than Case 1 and Case 2. For the unambiguous words like "able," "upcome," and "video," SR-LDA are very confident and can assign them to the right services or the global filter. However, words like "artist" and "location" can be related to multiple services. For example, *Eventful* is "the world's largest collection of events, taking place in local markets throughout the world, from concerts and sports to singles events and political rallies," so "artist" can be related to *Eventful*. Meanwhile, since *YouTube* can also provide videos of famous artists, so it could also be related to "artist." Finally, SR-LDA assigns 34.27% to *Eventful* and 64.55% to *YouTube*, corresponding to such an ambiguity.

As a conclusion, SR-LDA can assign words in mashups' profiles to appropriate services or the global filter, which helps to improve the quality of service representations.

5 Generating Knowledge Maps

After building accurate service representations by SR-LDA, our next step is to create a proper visualization of service ecosystems. At a higher level, such a visualization should help users gain an overview of the ecosystem, such as general categorizations. At a lower level, such a visualization could help users perceive popular services in each category together with surrounding similar services.

In this section, first we will present how to generate knowledge maps for visualizing a service ecosystem, then we will demonstrate the resulting map and present its effect for mashup creation with our further findings.

5.1 Knowledge map generation process

A knowledge map of a service ecosystem is an undirected graph comprising Web services as nodes, which may be connected through the similarity relationships between them.

The diameter of a service node reflects its popularity in the ecosystem, i.e., the diameter is proportional to the logarithm of the number of times that the service is invoked by mashups,

$$\text{diameter}_j \propto \log(\text{pop}_j),$$

where $\text{pop}_j = \text{cardinality}\big(\{i|s_j \in \mathcal{CS}_i\}\big)$.

The weight of an edge is set to the cosine similarity between the two services at its ends, i.e.,

$$\text{weight}_{j,j'} = \text{cosine}(\langle \boldsymbol{\theta}_j, \boldsymbol{\theta}_{j'} \rangle) = \frac{\boldsymbol{\theta}_j^{\mathrm{T}} \boldsymbol{\theta}_{j'}}{\|\boldsymbol{\theta}_j\| \|\boldsymbol{\theta}_{j'}\|},$$

In a displayed knowledge map, we intentionally ignore the relationship between services, whose similarity score is less than 0.2. In other words, we only present the edges whose weights are larger than 0.2. However, when we are playing layout, all the edges are kept at first. After reaching a stationary state, some of the edges below the threshold will be dropped to fine-tune the knowledge map for better visualization.

The color of a service node is set to the primary category labeled by the service's provider. For example, all the service nodes mainly representing the "Social" services share the same color.

Equipped with such settings, we run the ForceAtlas2 [22] algorithm and Gephi [23] to generate service ecosystem knowledge maps over all service representations. The major reason why we selected ForceAtlas2 is because it is a force-directed layout algorithm, which can effectively simulate the gravity and repulsion among nodes to reach an optimal layout state of a graph.

5.2 Overview of a knowledge map

Following the aforementioned methodology, we have generated a knowledge map for ProgrammableWeb.com service ecosystem[5], which is shown in Fig. 4.

One core phenomenon is that the visualization clearly reveals the existence of a collection of service clusters, each representing a topic (category) dominated by a few giant services. Example giant services are the *Google Maps* and *Microsoft Bing Maps* in the "Mapping" category, *YouTube* and *Netflix* in the "Video" category, *Last.fm* in the "Music" category, and *Facebook* and *Twitter* in the "Social" category. Note that each category is assigned a distinctive color code, which helps users to gain an overview of all available services in the ecosystem.

In the center of the map, a few services stand out which provide infrastructural functionalities, like *Google AJAX Libraries*, *Freebase*, *Yahoo Answers* and so on. Such services can collaborate with a variety of services, so they are placed in the center of the map, and do not form distinct service clusters.

Just outside of the center are some relatively active service categories. For example, dominated by *Facebook* and *Twitter*, the cluster of "Social" services occupies in the lower part of the map. Usually, services in the "social" category are collaborating with content providers and providing sharing or recommending functionalities. As a result, it is shown that around the cluster of "Social" category are "Video"-related services represented by *YouTube*, "Photo"-related services represented by *Flickr*, as well as "Music"-related services represented by *Last.fm*. However, compared with the "Video" category and "Photo" category, the "Music" category is noticeably farther away from the "Social" category, indicating that the collaboration between the two categories is weaker.

For some more independent functionalities, they locate at peripheral positions in the map. For example, "Bitcoin"-related services are on the lower right corner of the map.

5. This paper only reports a portion of the knowledge map for demonstration. The full version is published on the web.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2020.3001307, IEEE Transactions on Services Computing

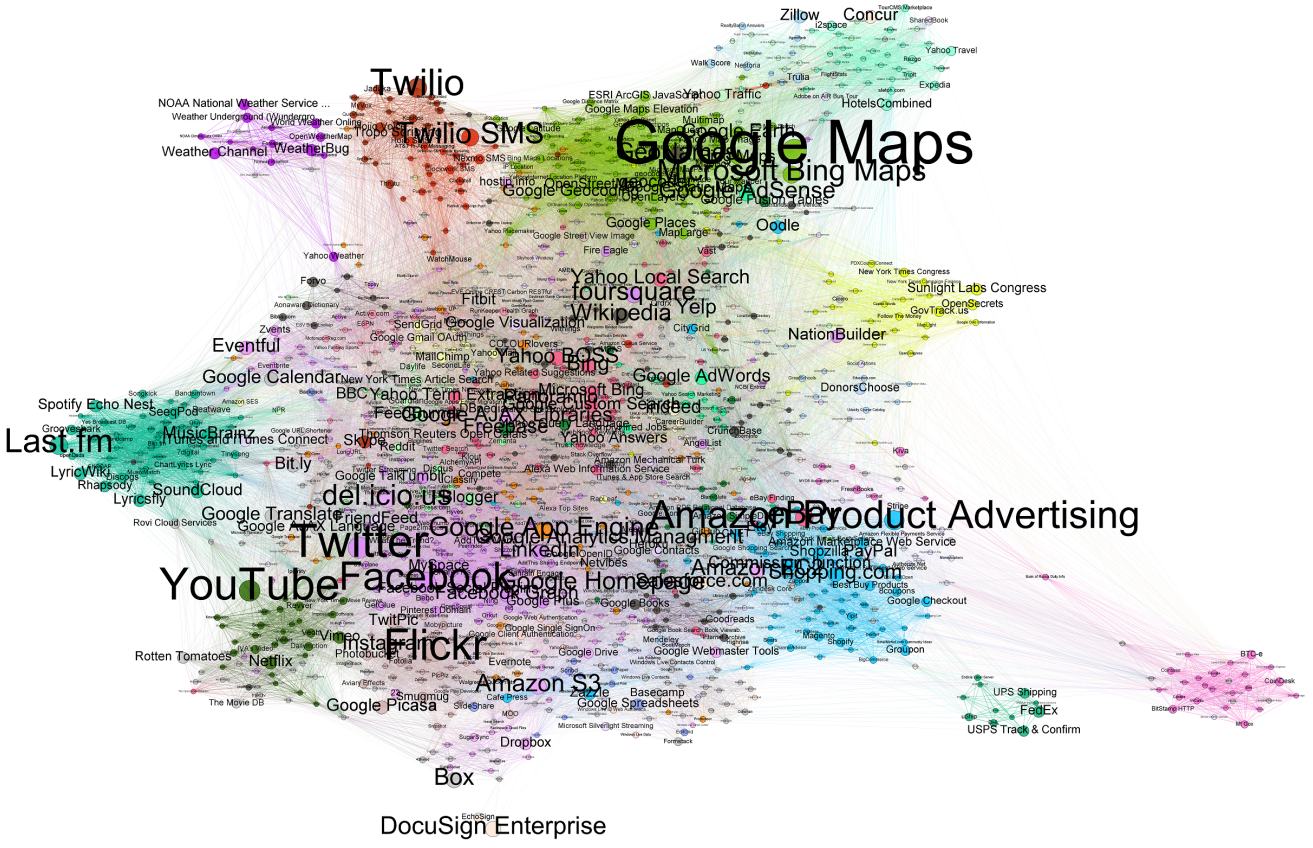


Fig. 4: Overview of the knowledge map generated based on services representations learned by SR-LDA.

Such location-related examples demonstrate that an objective knowledge map is capable of providing an intuitive overview of the massive services and their complex relationships. Such perception will help mashup developers gain a better understanding of the service ecosystems and their comprising services.

5.3 Detection of functionality patterns

By scrutinizing the generated knowledge map, focusing on the services with a conspicuous diameter or color and services located around the edge of service clusters, we have further discovered that services are of distinct functionality patterns. Here we will discuss four patterns illustrated in Fig. 5: dominating, inclining, balancing, and realigning.

5.3.1 Dominating

The services with the "dominating" functionality pattern are usually well-known ones and attract the vast majority of invocations by mashups in one service domain. Such services are easy to locate in a service knowledge map. Fig. 5a shows the neighbors of the service *Google Maps*, which is one of the most popular Web services in the ProgrammableWeb service ecosystem and has been used in more than 2,000 mashups. It stands out as a representative service of the "Mapping" category. In the knowledge map, it can be seen that the node representing the *Google Maps* is quite large and surrounded by other services of the "Mapping" category. Similar situation can be found with the service *Last.fm* in the "Music" category in Fig. 5b.

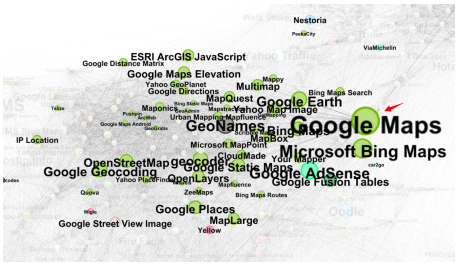
5.3.2 Inclining

The functionality pattern "inclining" means that a service is majoring in one domain but also provides auxiliary functionalities. The services showing such a functionality pattern are using a differentiated competition strategy compared with the dominating services. Fig. 5c illustrates the neighbors of an inclining service *LinkedIn*. As we can see, *LinkedIn* is very close to *Facebook* as a representative of "Social" services, but it also has relationships with *SimplyHired Jobs*, *CareerBuilder* and *indeed*, coinciding with that *LinkedIn* is "*a business social networking hub*." For another inclining service *Google Places* in Fig. 5d, it mainly deals with place search requests, but it also provides details of places, which makes *Google Places* gain neighbors like *Yelp*, *foursquare* and *Bing*.

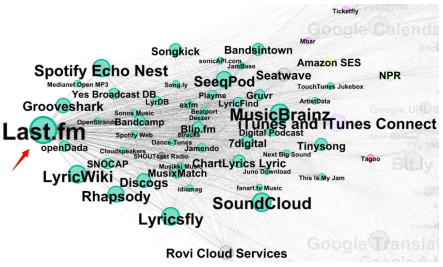
5.3.3 Balancing

The functionality of services with the "balancing" pattern is comparable in two domains. Service *GetGlue* is "*a social networking service where users 'check-in' to share what movies, videos, or TV shows,*" thus we can witness that *GetGlue* is serving as the connection of the "Social" category and the "Video" category in Fig. 5e. If developers want to make compositions relating to both functionalities, *GetGlue* would be a potential choice. Note that the service *Pinterest Domain* in Fig. 5f is "*a virtual pinboard allowing users to select photos from web pages to share with their friends*," quite similar with *GetGlue* while connecting the domains of "Social" category and "Photos" category.

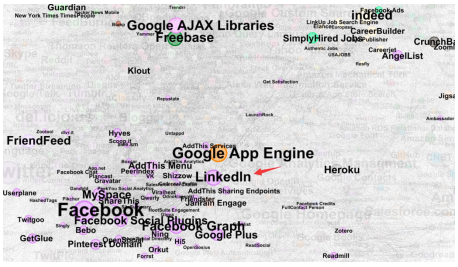
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2020.3001307, IEEE Transactions on Services Computing



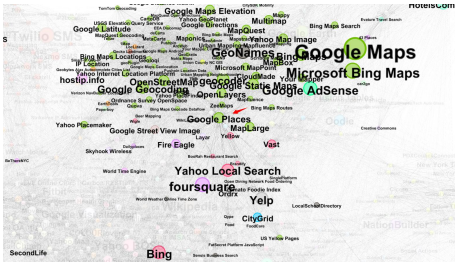
(a) Dominating: *Google Maps*



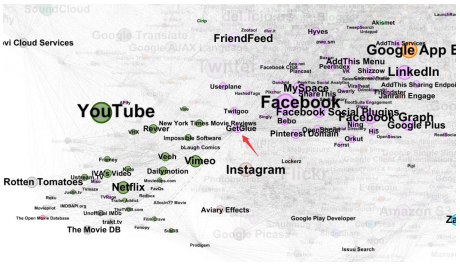
(b) Dominating: *Last.fm*



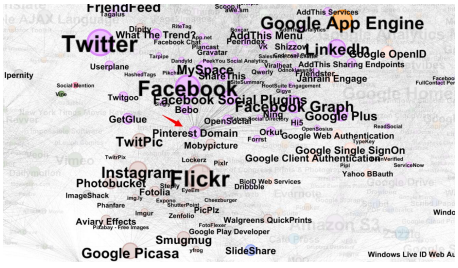
(c) Inclining: *LinkedIn*



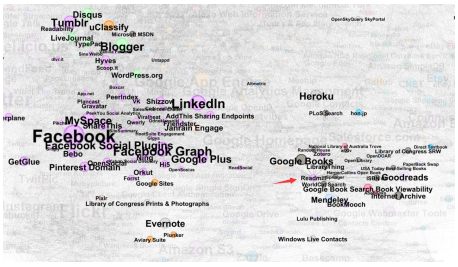
(d) Inclining: *Google Places*



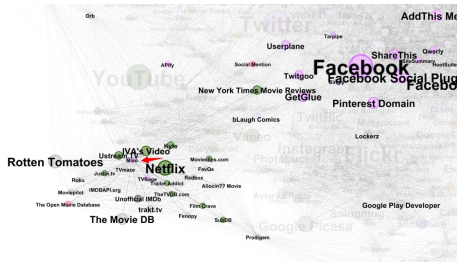
(e) Balancing: *GetGlue*



(f) Balancing: *Pinterest Domain*



(g) Realigning: *Readmill*



(h) Realigning: *Miso*

Fig. 5: Example Web services and their neighbors, revealing distinct functionality patterns.

5.3.4 Realigning

The "realigning" functionality pattern is interesting for investigation. Realigning services are usually some long-tail ones that get invoked by few mashups, such as *Readmill* and *Miso*. Such services are evidence of influence on service functionality from user usage. Service *Readmill* in Fig. 5g, published as a "Social" service, whose profile says that "*Readmill is an online and mobile platform for readers to share information about what eBooks they are reading, allowing them to highlight and discuss sections of eBooks with other users. . .*" However, mashups that invoke *Readmill* generally do not focus on "Social" properties. For example, mashup *ReadTracker* is "*an app for keeping track of the books you read*," mashup *Readmap* "*lets users plot a map of all the places they have been reading*," and mashup *The Book Report* makes users "*one step away from uncovering your timeline of books*." As a result, the social feature of *Readmill* degenerates and it becomes more related to the "Books" category than "Social." The service *Miso* in Fig. 5h shares a similar property. The publisher of *Miso* claims that *Miso* is designed to "*let people share what they are watching on TV in a fun and easy way*." While in practice, developers tend to view it as a simple source for information aggregation. For example, the mashup *Journamatic*, which invokes *Miso*, is to "*connect your social media sites and create a daily journal automatically from your check-ins, photos, tweets, and status updates*." As a result, we find *Miso* located near *Rotten Tomatoes* and *Netflix* instead of services in the "Social" category like *Facebook*.

These four patterns, revealed by our generated knowledge map, are some insights into the functionality of Web services, which can hardly be discovered directly from service profiles. Such insights will help to understand how the services have been used by existing mashups.

5.4 Assisting in mashup creation

Assuming that we are seeking for services to build a travel-related mashup, by which people can "*search for a destination and navigate their way, as well as share photos with their friends*." One straightforward strategy is to identify the services dominating the categories "Search," "Mapping," "Photo" and "Social." With the assistance of the generated knowledge map in Fig. 4, we can efficiently target on *Bing*, *Google Maps*, *Flickr* and *Facebook*, instead of looking up by category through the descriptions of the APIs.

In addition, the functionality patterns discovered in the generated knowledge map could offer mashup developers another advantage. The inclining service *Google Places* in Fig. 5d, as we stated above, is a mapping service that can provide details of a searched destination. The balancing service *Pinterest Domain* in Fig. 5f offers functionalities of both "Social" and "Photo." In other words, we may build the intended travel mashup with these two services, instead

of four. On the one hand, comprising fewer services means less cost when developing mashups and less potential risk in operation and maintenance. On the other hand, the knowledge map supplies a way for developers to bring out the value of other services but not only the dominating ones.

As a conclusion, the visualization of a service ecosystem based on accurate representations could help mashup developers fast locate relevant service and also be assistance for developers to better-selecting services.

6 Related Work

In this section, we will categorize representative related works in two sections and differentiate them from our approach accordingly.

6.1 Topic modeling and representation learning

In Web service field, building service representations is a fundamental step for most automatic methods, including service discovery and recommendation. Topic modeling has become a popular approach for service representation learning since the Latent Dirichlet Allocation (LDA) method was proposed in [20], which mines topic proportions from texts. LDA assumes that each word in a document is independently drawn from a multinomial distribution, depending on to which topic that the word is assigned. Based on LDA, a variety of topic models have emerged. Correlated Topic Models [24], [25] extend LDA and model the relationship between topics. It overcomes the limitation of LDA, which assumes that all the topics are independent from each other. Author-Topic Models (ATM) [11], [12] model authors and documents at the same time. Dynamic Topic Models [26], [27] capture the topic changing over time, thus can analyze the evolution of unobserved topics for a collection of documents. However, how to determine the number of topics remained a problem, so in [28] the Hierarchical Dirichlet Processes (HDP) infer the number of topics directly from the data. All such generative topic models benefit from the explicit hypotheses on probability distributions, and can provide interpretable results and require relatively less data for training.

Besides topic modeling, there are some other methods for representation learning, including Principal Component Analysis (PCA) [29], Independence Component Analysis (ICA) [30], Singular Value Decomposition (SVD) [31], as well as some nonlinear methods like Local Linear Transformation Embedding (LLTE) [32] and Stacked Denoising Autoencoders (SDAE) [33]. However, these methods, which lack interpretability, cannot give as many intuitions as generative topic models. SDAE also requires much more training data.

Concerning building service representations in Web service ecosystems, the relationship of mashup-service-word are more complex and becomes the primary issue that we consider in this paper. Zhong et al. [34] firstly proposed to extract service description from mashup profiles. To make better recommendations, they directly applied the ATM to extract words describing services from mashup profiles, so-called reconstructing service profiles. Due to ignoring the original service profiles, they cannot reconstruct profiles for never-used services. Different from them, we have designed a tailored generative model, processing the profiles of both mashups and services at the same time, also introduced a global filter for background terms.

6.2 Visualization for Web service ecosystem

Although academia has conducted in-depth research on service ecosystems, there is not much work on visualization. Lu et al. [35] presented the global structure of a service repository with parallel coordinates. Kumara et al. [36] used the Spherical Associated Keyword Space (SAKS) algorithm to visualize Web service clusters on a 2D spherical surface. Olayinka at al. [37] constructed and visualized a Web service network using the invocation records between mashups and services. Some others [38], [39], [40] proposed visualization tools or techniques to ease the process of service discovery or making service compositions.

Different from above all, we have visualized a service system considering both the functionality and invocation relation, based on the service representations learned by SR-LDA. We aim to help people navigate through a service ecosystem, as well as to provide insights into the relationships and functionalities of services.

Beyond the Web service field, knowledge visualization remains an active topic for research. Shou et al. [41] used a model-free method to analyze component stock corporations and generated knowledge maps of financial data. Hao et al. [42] proposed a knowledge map-based method to reduce information overload during browsing domain knowledge for new knowledge users. A knowledge map for question and answer archives was constructed by exploiting question–answer pair characteristics [43]. Gao et al. [44] presented how to build a large-scale, accurate and fresh knowledge graph.

7 Conclusions

As service-oriented software engineering becoming mainstream, more and more Web services are published into the service ecosystem on a daily basis. It becomes increasingly critical to help software developers understand the functionalities of the massive amount of services and their relationships in a service ecosystem. One core challenge is how to generate up-to-date and accurate representations of services.

Applying machine learning techniques, this paper presents SR-LDA, a model that incorporates service users' perceptions into service profiles to form more comprehensive service representations on the fly. SR-LDA is further equipped with a global filter that helps to filter out background terms automatically. Based on the generated high-quality service representations, we present a knowledge map for a service ecosystem, which helps to visualize services and their functionality patterns in an intuitive manner.

In our future work, we plan to incorporate additional information to further enhance service representations, such as the comments on the Web services and temporal information explicitly. In addition, we are interested in investigating the evolution of a service ecosystem by rebuilding the representations and visualization regularly. We also plan

to explore the possible applications of the representations learned by SR-LDA, such as service recommendation and discovery.

ACKNOWLEDGMENTS

This research has been partially supported by the National Key Research and Development Program of China (No.2018YFB1402500), the National Natural Science Foundation of China (No.61673230), and the High-Tech Ship Research Project of China (No.17GC26102.01). Yushun Fan is the corresponding author.

REFERENCES

[1] L.-J. Zhang, J. Zhang, and H. Cai, *Services Computing*. Springer, 2007.

[2] 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.

[3] D. Ardagna and B. Pernici, "Adaptive service composition in flexible processes," *IEEE Transactions on Software Engineering*, vol. 33, no. 6, pp. 369–384, 2007.

[4] A. P. Barros and M. Dumas, "The rise of web service ecosystems," *IT professional*, vol. 8, no. 5, pp. 31–37, 2006.

[5] G. Kang, M. Tang, J. Liu, X. F. Liu, and B. Cao, "Diversifying web service recommendation results via exploring service usage history," *IEEE Transactions on Services Computing*, vol. 9, no. 4, pp. 566–579, 2016.

[6] B. Cao, X. Liu, M. M. Rahman, B. Li, J. Liu, and M. Tang, "Integrated content and network-based service clustering and web apis recommendation for mashup development," *IEEE Transactions on Services Computing*, 2017.

[7] X. Zhu, X.-Y. Jing, D. Wu, Z. He, J. Cao, D. Yue, and L. Wang, "Similarity-maintaining privacy preservation and location-aware low-rank matrix factorization for qos prediction based web service recommendation," *IEEE Transactions on Services Computing*, 2018.

[8] L. Ren and W. Wang, "A granular svm-based method for top-n web services recommendation," *IEEE Transactions on Services Computing*, 2019.

[9] C. Zins, "Knowledge map of information science," *Journal of the American Society for Information Science and Technology*, vol. 58, no. 4, pp. 526–535, 2007.

[10] B. Bai, Y. Fan, W. Tan, and J. Zhang, "SR-LDA: mining effective representations for generating service ecosystem knowledge maps," in *Proc. IEEE Int. Conf. Serv. Comput.* IEEE, 2017, pp. 124–131.

[11] M. Rosen-Zvi, C. Chemudugunta, T. Griffiths, P. Smyth, and M. Steyvers, "Learning author-topic models from text corpora," *ACM Transactions on Information Systems*, vol. 28, no. 1, p. 4, 2010.

[12] M. Rosen-Zvi, T. Griffiths, M. Steyvers, and P. Smyth, "The author-topic model for authors and documents," *arXiv preprint arXiv:1207.4169*, 2012.

[13] X. Liu and I. Fulia, "Incorporating user, topic, and service related latent factors into web service recommendation," in *Proc. IEEE Int. Conf. Web Serv.* IEEE, 2015, pp. 185–192.

[14] S. Lyu, J. Liu, M. Tang, G. Kang, B. Cao, and Y. Duan, "Three-level views of the web service network: an empirical study based on programmableweb," in *Proc. IEEE International Congress on Big Data*. IEEE, 2014, pp. 374–381.

[15] Y. Zhong, Y. Fan, K. Huang, W. Tan, and J. Zhang, "Time-aware service recommendation for mashup creation," *IEEE Transactions on Services Computing*, vol. 8, no. 3, pp. 356–368, 2015.

[16] 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 composition recommendation," *IEEE Transactions on Services Computing*, 2018.

[17] W. Li, Y. Feng, D. Li, and Z. Yu, "Micro-blog topic detection method based on BTM topic model and K-means clustering algorithm," *Automatic Control and Computer Sciences*, vol. 50, no. 4, pp. 271–277, 2016.

[18] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 2, pp. 224–227, 1979.

[19] T. Caliński and J. Harabasz, "A dendrite method for cluster analysis," *Communications in Statistics*, vol. 3, no. 1, pp. 1–27, 1974.

[20] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[21] T. L. Griffiths and M. Steyvers, "Finding scientific topics," *Proceedings of the National academy of Sciences*, vol. 101, no. suppl 1, pp. 5228–5235, 2004.

[22] M. Jacomy, S. Heymann, T. Venturini, and M. Bastian, "Forceatlas2, a continuous graph layout algorithm for handy network visualization," *Medialab Center of Research*, vol. 560, 2011.

[23] M. Bastian, S. Heymann, and M. Jacomy, "Gephi: an open source software for exploring and manipulating networks," in *Proc. International AAAI Conference on Weblogs and Social Media*, 2009.

[24] M. Aznag, M. Quafafou, and Z. Jarir, "Correlated topic model for web services ranking," *Int. J. Adv. Comput. Sci. Appl*, vol. 4, 2013.

[25] G. Xun, Y. Li, W. X. Zhao, J. Gao, and A. Zhang, "A correlated topic model using word embeddings." in *IJCAI*, 2017, pp. 4207–4213.

[26] D. Blei and J. D. Lafferty, "Dynamic topic models," in *Proc. International Conference on Machine Learning*. ACM, 2006, pp. 113–120.

[27] C. Wang, D. Blei, and D. Heckerman, "Continuous time dynamic topic models," *arXiv preprint arXiv:1206.3298*, 2012.

[28] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei, "Hierarchical dirichlet processes," *Journal of the American Statistical Association*, vol. 101, no. 476, pp. 1566–1581, 2006.

[29] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[30] J. V. Stone, *Independent component analysis: a tutorial introduction*. MIT press, 2004.

[31] M. E. Wall, A. Rechtsteiner, and L. M. Rocha, "Singular value decomposition and principal component analysis," in *A practical approach to microarray data analysis*. Springer, 2003, pp. 91–109.

[32] C. Hou, J. Wang, Y. Wu, and D. Yi, "Local linear transformation embedding," *Neurocomputing*, vol. 72, no. 10-12, pp. 2368–2378, 2009.

[33] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of machine learning research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[34] Y. Zhong, Y. Fan, W. Tan, and J. Zhang, "Web service recommendation with reconstructed profile from mashup descriptions," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 2, pp. 468–478, 2018.

[35] J. Lu, Y. Zhang, J. Xu, G. Xiao, and Q. Liang, "Data visualization of web service with parallel coordinates and nodetrix," in *Proc. IEEE Int. Conf. Serv. Comput.* IEEE, 2014, pp. 766–773.

[36] B. T. Kumara, I. Paik, H. Ohashi, Y. Yaguchi, and W. Chen, "Context-aware filtering and visualization of web service clusters," in *Proc. IEEE Int. Conf. Web Serv.* IEEE, 2014, pp. 89–96.

[37] O. Adeleye, J. Yu, S. Yongchareon, and Y. Han, "Constructing and evaluating an evolving web-api network for service discovery," in *International Conference on Service-Oriented Computing*. Springer, 2018, pp. 603–617.

[38] X. Chen, Z. Zheng, X. Liu, Z. Huang, and H. Sun, "Personalized qos-aware web service recommendation and visualization," *IEEE Transactions on Services Computing*, vol. 6, no. 1, pp. 35–47, 2013.

[39] F. C. Fang, H. Yuan, and S. D. Kim, "A visualization framework for web service discovery and selection based on quality of service," in *Proc. IEEE Asia-Pacific Service Computing Conference*. IEEE, 2007, pp. 312–319.

[40] C. Ma and Y. He, "An approach for visualization and formalization of web service composition," in *Proc. International Conference on Web Information Systems and Mining*. IEEE, 2009, pp. 342–346.

[41] W. Shou, W. Fan, B. Liu, and Y. Lai, "Knowledge map mining of financial data," *Tsinghua Science and Technology*, vol. 18, no. 1, pp. 68–76, 2013.

[42] J. Hao, Y. Yan, L. Gong, G. Wang, and J. Lin, "Knowledge map-based method for domain knowledge browsing," *Decision Support Systems*, vol. 61, pp. 106–114, 2014.

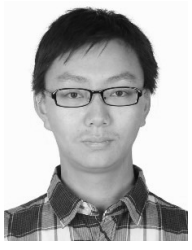
[43] M. Li, X. Lu, L. Chen, and J. Wang, "Knowledge map construction for question and answer archives," *Expert Systems with Applications*, vol. 141, p. 112923, 2020.

[44] Y. Gao, J. Liang, B. Han, M. Yakout, and A. Mohamed, "Building a large-scale, accurate and fresh knowledge graph," *KDD-2018, Tutorial*, vol. 39, 2018.

Junqi Zhang received his B.S. degree in control theory and application from Tsinghua University, China, in 2015. He is currently working towards the Ph.D. degree at the Department of Automation, Tsinghua University. His research interests include services computing, recommender systems and data mining.

Yushun Fan received his Ph.D. degree in control theory and application from Tsinghua University, China, in 1990. He is currently a tenured professor with the Department of Automation, Vice Director of National CIMS Engineering Research Center of China, Director of the System Integration Institute, and Director of the Networking Manufacturing Laboratory, Tsinghua University. He is a member of IFAC TC5.1 and TC 5.2, Vice Director of China Standardization Committee for Automation System and Integration, and an editorial member of the International Journal of Computer Integrated Manufacturing. From September 1993 to 1995, he was a visiting scientist, supported by Alexander von Humboldt Stiftung, with the Fraunhofer Institute for Production System and Design Technology (FHG/IPK), Germany. He has authored 10 books in enterprise modeling, workflow technology, intelligent agent, object-oriented complex system analysis, and computer integrated manufacturing. He has published more than 500 research papers in journals and conferences. His research interests include enterprise modeling methods and optimization analysis, business process re-engineering, workflow management, system integration, modern service science and technology, petri nets modeling and analysis.

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



Bing Bai received his B.S. and Ph.D. degrees in control theory and application from Tsinghua University, China, in 2013 and 2018 respectively, and he is currently a senior researcher with the Cloud and Smart Industries Group, Tencent, Beijing, China. He received the Best Paper Award from the 14th IEEE International Conference on Services Computing (2017). His research interests include data mining and recommender systems.