SR-LDA: Mining Effective Representations for Generating Service Ecosystem Knowledge Maps

Bing Bai
Tsinghua National Laboratory for Information Science and Technology Department of Automation Tsinghua University Beijing 100084, China bb13@mails.tsinghua.edu.cn

Yushun Fan*
Tsinghua National Laboratory for Information Science and Technology Department of Automation Tsinghua University Beijing 100084, China fanyus@tsinghua.edu.cn

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

Jia Zhang
Carnegie Mellon University Silicon Valley Moffett Field CA 94035, USA jia.zhang@sv.cmu.edu

***Abstract*—As the quantity of Web services grow continuously, it becomes more challenging for developers to navigate through and make use of them. Thus, a knowledge map consisting of a summary of individual services and their relations has become increasingly useful. There are two challenges in building such a knowledge graph for Web service ecosystems. First, services keep evolving in terms of function and usage pattern, while their descriptions typically remain static and obsolete. Second, service profiles usually comprise some common background terms which do not differentiate services. To address the two challenges, we developed a novel tailored topic model, named Service Representation-LDA (SR-LDA), to mine effective representations beyond service profiles to build a knowledge map. The key idea is to incorporate mashup descriptions as an indication of service evolution, and introduce a global filter to identify and filter out background terms. Extensive experiments show that the tailored model is more effective than baselines. The methodology of building knowledge maps for the real-world ProgrammableWeb service ecosystem based on the learned representations is also presented, together with the analyses of representative functionality patterns.**

***Keywords*-service ecosystem; topic model; knowledge map**

I. Introduction

With the advancement of Service Oriented Software Engineering (SOSE), developers can compose multiple functionalities offered by different providers to build service compositions, or so-called mashups [1], [2]. As such software reusability and interoperability can make it faster to market and bring cost-reduction for software development, it becomes a matter of course that we witness a boom in the number of services.

However, the rapidly increasing number of Web services also brings challenges for mashup developers, as well as service ecosystem managers. Despite many service recommendation/discovery approaches, which can help when developers are ready to build mashups [3], [4], [5], people still get stuck when wondering fuzzy problems like: "What are the functional relationships between the various services?" "Are there any interesting services that can bring appealing functionalities to build mashups?" In these scenarios, knowledge maps of service ecosystems can provide a practical overview of all the services on the web. Mining these knowledge maps may also identify services that are interesting to investigate [6].

In order to generate knowledge maps, building service representations is the first and foundational step. A direct method is to apply vanilla Latent Dirichlet Allocation (LDA) [7] to build latent topic models for service profiles. However, considering the following characteristics of service ecosystems, vanilla LDA cannot build very effective service representations.

- **The profiles are static while the services are evolving.** Once published, the profiles generally remain static, but later the services might evolve in terms of function and usage pattern [8]. In addition, software developers may find ways to use the services that are different from the service publishers' initial thoughts.
- **Service profiles usually comprise some service ecosystem-specific background terms.** For example, the description of *Google Maps* is "*The Google Maps API allows for the embedding of Google Maps onto web pages of outside developers, using a simple JavaScript interface or a Flash interface. It is designed to work on both mobile devices as well as traditional desktop browser applications…*" Background terms such as "developers," "interface" and "applications" cannot differentiate services and will bring adverse impact on the representations.

To the best of our knowledge, no existing algorithm is designed to solve the aforementioned issues. In this paper, we propose a novel model that is tailored to mine effective representations of services aiming for generating knowledge maps. The basic idea of the proposed SR-LDA

* Corresponding Author

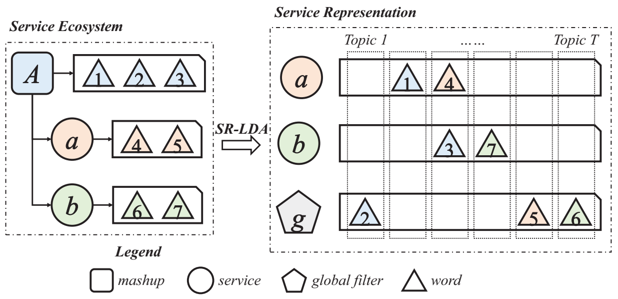


Figure 1: The basic idea of the proposed SR-LDA. We build service representations from both mashup descriptions and service profiles. Additionally, we introduce a global filter to identify and down-weight background terms.

is illustrated in Fig. 1. A service ecosystem consists of all kinds of services, and some of them may be composed as mashups. For example, mashup A involves two services, i.e., service a and service b, each of them is described by a few words (terms), i.e., word 1 to word 7.

Once a service is published, its profile remains static, but the service may evolve. Such information is hidden in the descriptions of mashups that involve the service. That is to say, instead of modeling services solely based on their profiles, we incorporate mashups' descriptions to build more comprehensive and up-to-date service representations. However, a mashup might use multiple services, and the words used to describe the mashup can correspond to anyone of them. Thus we resort to the idea of author topic model [9], [10] and develop a probabilistic method that can automatically assign words to corresponding services based on maximizing the posterior probability. By incorporating mashup descriptions and service profiles, the first issue can be settled.

To tackle the second issue, we introduce a concept of "global filter." If a word is not quite related to some specific services and appears roughly uniformly in many profiles, it is more likely to be a background term and thus assigned to the global filter. Through this method, we can automatically identify and down-weight the adverse impact of the background terms.

By incorporating the aforementioned ideas, we can obtain significantly more effective service representations for knowledge map generation. The main contributions of this paper are summarized as follows:

- We designed a tailored model that adapts to the evolving service ecosystem and can automatically identify background terms, thus the model can learn more effective service representations.
- We tested our model in the real-world ProgrammableWeb dataset, and both quantitative and qualitative analyses show that our model is more effective than baselines.
- We introduced the methodology of building knowledge maps for service ecosystems based on the learned representations, and found three representative functionality patterns.

The rest of this paper is organized as follows. Section II describes the framework of our model. Section III shows how the optimal parameters are obtained. Section IV reports the experimental results. Section V reports how knowledge map is generated and our analyses. Section VI lists the related work and Section VII draws conclusions.

II. Model Framework

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

A. 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 this notation is related to service j, and $j = 1 : J$. $SD_j = \{w_{j1}, w_{j2}, \ldots, w_{jn_j}\}$ is the set of n_j words (terms) used in the profile of service j.

For mashups, we use subscript "i" to denote that this notation is related to mashup i, and $i = 1 : I$. $CS_i = \{cs_{i1}, cs_{i2}, \ldots, cs_{ih_i}\}$ is the set of h_i component services involved in mashup i, and $MD_i = \{w_{i1}, w_{i2}, \ldots, w_{in_i}\}$ is the set of n_i words used to describe mashup i.

As illustrated above, in this model we mainly use the involvement relationship between mashups and services, the service profiles and the mashup descriptions. Note that the service profiles include, but are not limited to content descriptions, category information, tags and WSDL documents.

B. Generative model for SR-LDA

The original service profiles are static, and contain too many service ecosystem-specific background terms, thus the representations built directly from the profiles cannot be very effective. In this paper, we incorporate the descriptions of mashups as a source of information for service evolution, and introduce a global filter to automatically filter the background words. In order to achieve the above objectives, we define a tailored generative process. For conciseness, we use the subscript "$J+1$" to represent the global filter. For example, notation $\boldsymbol{\theta}_{J+1}$ represents the topic proportions of the global filter.

The generative process is defined as follows. Assuming that there are T topics,

1) For each topic $z = 1 : T$, draw word proportions $\boldsymbol{\phi}_z \sim$ Dirichlet(β).
2) For the global filter, draw topic proportions $\boldsymbol{\theta}_{J+1} \sim$ Dirichlet(α).
3) For each service $j = 1 : J$,

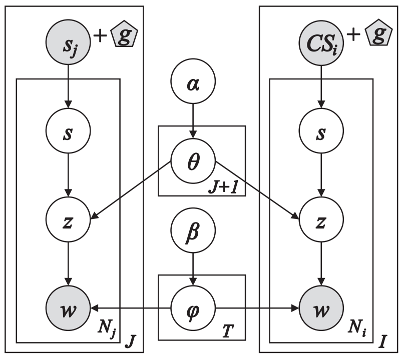


Figure 2: The graphical model for the proposed SR-LDA. Words in the service profiles and mashup descriptions are assigned to services, and a global filter is introduced for better performance.

a) Draw topic proportions $\boldsymbol{\theta}_j \sim$ Dirichlet(α)

b) For each word $w_{jn} \in SD_j$,

i) Draw a service assignment $s_{jn} \sim$ Uniform($\{j, J+1\}$)

ii) Conditioned on s_{jn}, draw a topic assignment $z_{jn} \sim$ Mult($\boldsymbol{\theta}_{s_{jn}}$).

iii) Conditioned on z_{jn}, draw the word $w_{jn} \sim$ Mult($\boldsymbol{\phi}_{z_{jn}}$).

4) For each mashup $i = 1 : I$,

a) For each word $w_{in} \in MD_i$,

i) Draw a service assignment $s_{in} \sim$ Uniform($CS_i \bigcup \{J+1\}$)

ii) Conditioned on s_{in}, draw a topic assignment $z_{in} \sim$ Mult($\boldsymbol{\theta}_{s_{in}}$).

iii) Conditioned on z_{in}, draw the word $w_{in} \sim$ Mult($\boldsymbol{\phi}_{z_{in}}$).

As illustrated at 3)b) and 4)a), we assume that the words in a service's profile can only be assigned to the service itself or the global filter, while the words in the descriptions of mashups can be assigned to any one of the component services, or the global filter. Based on such a framework, words that are used to describe mashups can also fertilize the representations of services (i.e., the topic proportions $\boldsymbol{\theta}_{1:J}$). What's more, as the global filter corresponds to all the profiles and descriptions, 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 the impact on the representations. Through these targeted designs, SR-LDA can build effective representations for Web services.

The graphical model for this process is shown in Fig. 2.

III. Parameter Learning from Data

Effective service representations can be built by solving the generative model of SR-LDA. If we assume that there are T topics, the variables that we are interested in include: (1) service-topic proportions $\boldsymbol{\Theta} = \boldsymbol{\theta}_{1:J+1}$, (2) topic-word proportions $\boldsymbol{\Phi} = \boldsymbol{\phi}_{1:T}$. Our aim is to maximize the posterior distribution on $\boldsymbol{\Theta}$ and $\boldsymbol{\Phi}$.

For conciseness, we merge the data from the service part and the mashup part together when explaining the model training procedure. We define

$$D = \{SD_1, \dots, SD_J, MD_1, \dots, MD_I\},$$

in other words, D contains all the description content in the service ecosystem, and $|D| = I + J$. Similarly, we define

$$S = \Big\{\{1, J+1\}, \dots, \{J, J+1\}, \\ \{CS_1 \bigcup \{J+1\}\}, \dots, \{CS_I \bigcup \{J+1\}\}\Big\},$$

which means that S contains the corresponding service associated to D. We use the subscript "k" to denote that the notation is related to the merged variables. By merging the service part and mashup part together, we can get more concise expressions for model training.

Similar with [9], our inference scheme is based upon maximizing the observation that

$$p(\boldsymbol{\Theta}, \boldsymbol{\Phi} | D, S, \alpha, \beta) \\ = \sum_{\mathbf{z},\mathbf{s}} p(\boldsymbol{\Theta}, \boldsymbol{\Phi} | \mathbf{z}, \mathbf{s}, D, S, \alpha, \beta) P(\mathbf{z}, \mathbf{s} | D, S, \alpha, \beta),$$

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 $\boldsymbol{\Theta}$ and $\boldsymbol{\Phi}$ can be obtained by using Gibbs sampling algorithm. First an empirical sample-based estimate of $P(\mathbf{z}, \mathbf{s} | D, S, \alpha, \beta)$ can be obtained. Afterwards for any specific sampling instance, we can get the expectation of $\boldsymbol{\Theta}$ and $\boldsymbol{\Phi}$ directly by exploiting the conjugation of Dirichlet distribution and multinomial distribution.

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

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

where g_{jt} is the number of word tokens assigned to topic t and service j at the same time, and c_{tw} is the frequency at which word w is assigned to topic t. The superscript $\neg$ denotes a quantity excluding the current instance.

After sampling a sufficient number of burn-in iterations, the sampler will converge and we can accumulate the results for several iterations, average them and compute the expectation of the true posterior with equations. For any specific sampling instance $\mathbf{z}$ and $\mathbf{s}$,

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

Table I: Notations Used in SR-LDA

Symbol	Description
i	Subscript for mashups
j	Subscript for services
k	Subscript for merged variables
MD_i	Description content for mashup i
SD_j	Profile content for service j
$D = \{D_k\}$	D_k is the kth merged description content
$\mathbf{w} = \{w_{kn}\}$	w_{kn} is the word for nth word token in D_k
W	Vocabulary size
CS_i	Component services of mashup i
$S = \{S_k\}$	S_k is the corresponding service set for D_k
T	Number of topics
$\mathbf{\Phi} = \boldsymbol{\phi}_{1:T}$	Topic-word proportions. ϕ_{tw} indicates the probability of word w given topic t
$\mathbf{\Theta} = \boldsymbol{\theta}_{1:J+1}$	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	Dirichlet prior for service-topic proportions
β	Dirichlet prior for topic-word proportions
$\mathbf{z} = \{z_{kn}\}$	z_{kn} is the topic assignment for w_{kn}
$\mathbf{s} = \{s_{kn}\}$	s_{kn} is the service assignment for w_{kn}
g_{jt}	Number of word tokens assigned to topic t and service j at the same time
c_{tw}	Frequency at which word w is assigned to topic t
N_{burn}	Number of burn-in iterations
N_{acc}	Number of accumulation iterations

Algorithm 1: Parameter learning for SR-LDA

```
Input: $\alpha$, $\beta$, $D$, $S$, $T$, $N_{burn}$ and $N_{acc}$
Output: Optimal $\mathbf{\Theta}$ and $\mathbf{\Phi}$
Procedure:
01  Initialize z and s randomly
02  For $iter = 1 : N_{burn} + N_{acc}$
03    For each word token $w_{kn}$
04      Update $g_{jt}$ and $c_{t_w}$ for all $j$, $t$ and $w$
05      Sample $z_{kn}$ and $s_{kn}$ according to Eq. (1)
06    End
07    If $iter \geq N_{burn} + 1$
08      Record the sampling results of $z_{kn}$ and $s_{kn}$
09    End
10  End
11  Calculate the average of recorded z and s
12  Calculate expectational $\mathbf{\Phi}$ according to Eq. (2)
13  Calculate expectational $\mathbf{\Theta}$ according to Eq. (3)
```

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

The notations used here are summarized in Table I. Assuming that the number of burn-in iterations is N_{burn} and the number of accumulation iterations is N_{acc}, the procedure of parameter learning can be described in Algorithm 1.

IV. Experiments

In this section, we first introduce the data set we used for evaluation, the evaluation metric and the baseline methods. We then discuss the quantitative and qualitative results compared with baselines[1].

Table II: Information about ProgrammableWeb Data Set

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

A. Data set

ProgrammableWeb.com is the largest online repository of public Web services and their mashups [4], and it has been used in a number of experiments of Web service research [3], [4]. We crawled the data of ProgrammableWeb.com from June 2005 to August 2016. The profile of services include their textual description, tags and category information. After processing the profiles by stemming and stop words removing, we got a vocabulary size of 9,555. Detailed information about the data set we used is listed in Table II.

B. Evaluation scheme

The target of our method is to learn effective representations from service ecosystem to fertilize the generation of knowledge maps. However, the quality of representations is hard to measure directly. Intuitively, better representations can (1) detect the similarities of services in the same domain; and (2) detect the differences of services in different domains. Based on the intuition, we adopted an indirect method to evaluate the quality of service representations.

The evaluation works in the following way: similar with [11], first we perform K-means clustering upon the representations, then we calculate the Davies-Bouldin index (DBI) [12] to measure the results. DBI is defined as follows

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

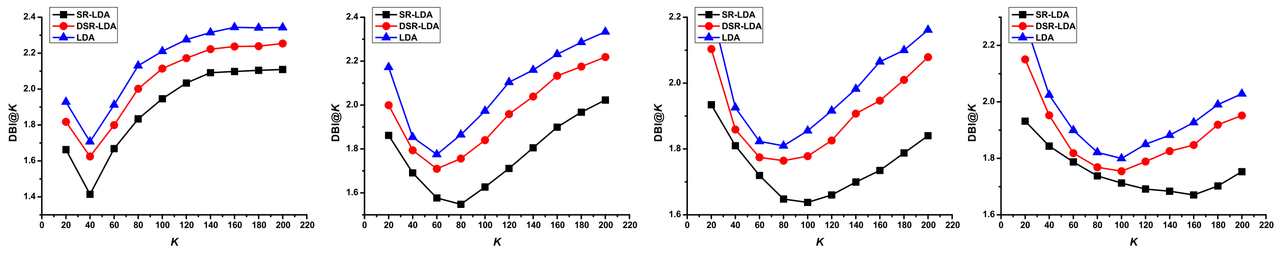
where

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

is the average distance within cluster C, K is the number of clusters and $d_{cen}(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j)$ indicates the distance between centers of two clusters, i.e., the distance between $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}_j$.

Obviously, a lower value of DBI indicates more effective representations, i.e., items are more similar within a cluster, and the clusters are separated better, which corresponds to the aforementioned intuition.

[1]The data set and results are published at http://www.simflow.net/team/baibing/sr-lda.zip



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

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

C. Baselines and hyperparameter settings

The baselines used for comparison include:

- **LDA**. For this baseline, we used the profiles of services and applied LDA [7] to extract topics of each service. This baseline can provide evidence for how well the vanilla method works in service ecosystems.
- **DSR-LDA (Degenerated SR-LDA)**. For this method, we abandoned the global filter in the proposed SR-LDA. This baseline can provide evidence for how background terms bring adverse impact on building effective representations.

For all runs of our model and baselines, we used $\alpha = 50/T$ and $\beta = 0.01$. We computed the results of T values of 40, 80, 120 and 160 topics. For all values of T, we ran five different Gibbs sampling chains on 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. The DBI results reported are the average of these five Gibbs sampling chains. For qualitative evaluation and knowledge map generation, we set T to 120 based on the inflection point of perplexity curve [7].

D. Quantitative comparisons

The DBI results of SR-LDA and baselines with different topic numbers T are shown in Fig. 3. We can find that at all tested topic numbers T and cluster numbers K, vanilla LDA gets the largest DBI and SR-LDA performs the best. Comparing the results of baseline LDA and DSR-LDA, we can find that taking the description of mashups into consideration can promote the quality of service representations. While comparing the results of the proposed SR-LDA and the baselines, we can draw more interesting conclusions.

Firstly, the gaps between SR-LDA and DSR-LDA are generally larger than those between DSR-LDA and LDA, indicating that background terms are bringing more adverse impact on building high-quality representations.

Secondly, we can find that the DBI results of SR-LDA show a different trend compared with the results of the two baselines, especially when T is large. For example, when T is set to 120, 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 is more significant. This 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 can give better DBI results at a variety of topic number T and cluster number K, indicating that representations given by SR-LDA are more effective.

E. Qualitative analyses

We also conducted qualitative analyses on the topics given by different methods. Table III 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. Due to space constraints, we only report the results of SR-LDA and DSR-LDA. This can provide information about the role of the global filter.

As illustrated in the Table, 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 these words. As for DSR-LDA, Topic #23 and #29 are the most similar topics to the corresponding topics. As DSR-LDA cannot filter these words, the quality of service representations suffers from the adverse impact of these background terms.

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 actually also related to this topic. Both methods can discover this phenomenon from the descriptions 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

Table III: 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%

in this topic, and some environmentally-friendly energy-related services like *Clean Power SolarAnywhere* and *AMEE* also show up in this topic. This shows that without filtering background terms, the quality of function-related topics will also suffer.

Based on the results, we can conclude that the topics learned by SR-LDA are more effective. This can be helpful for the generation of service ecosystem knowledge maps.

V. Generation of Service Ecosystem Knowledge Maps

In this section, we show how to use the representations to generate knowledge maps for service ecosystem. Firstly, we show the settings of knowledge map generation, then we demonstrate the resulting map and take four example Web services for further analysis.

A. Settings of knowledge map generation

Based on the representations learned by SR-LDA, we can finally build knowledge maps for Web service ecosystems. We take nodes in the map for services, and undirected edges between nodes for the relationships between services.

The diameter of a node is proportional to the logarithm of how many times the service is involved by mashups, i.e.,

$$diameter_j \propto \log(pop_j).$$

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

$$weight_{j,j'} = \frac{\boldsymbol{\theta}_j^{\mathrm{T}} \boldsymbol{\theta}_{j'}}{\|\boldsymbol{\theta}_j\| \|\boldsymbol{\theta}_{j'}\|}.$$

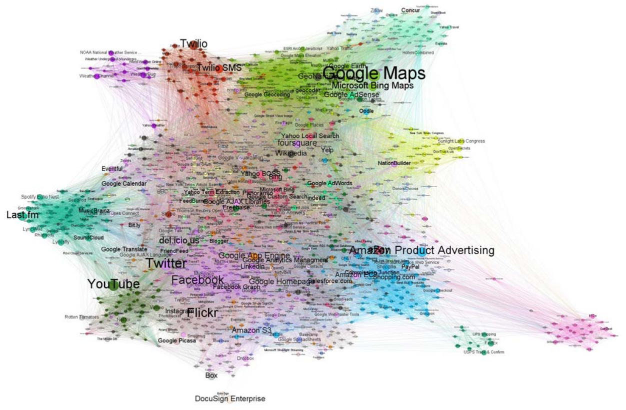


Figure 4: Overview of the knowledge map generated based on representations learned by SR-LDA.

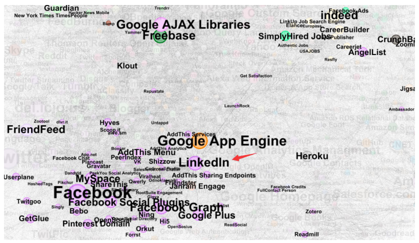
The color of a node is related to the primary category given by the service's provider.

With these settings, we use the ForceAtlas2 [13] algorithm and Gephi [14] to generate service ecosystem knowledge map. The basic idea of ForceAtlas2 is to simulate the gravity and repulsion and find a stationary state of the graph.

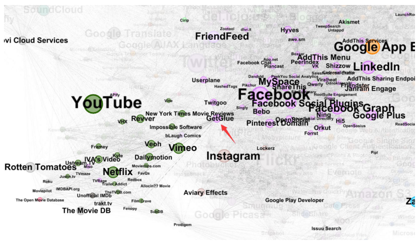
B. Analyses of the overview

Based on the aforementioned settings, we generate a knowledge map for ProgrammableWeb service ecosystem[2].

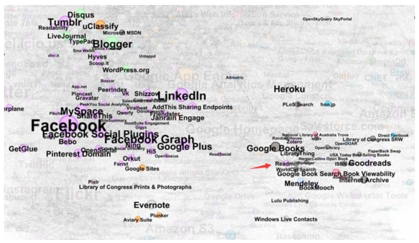
[2]In this paper, we only report the knowledge map of ever-used services for demonstration. The full version is published on the web.



(a) Inclining: *LinkedIn*



(b) Balancing: *GetGlue*



(c) Realigning: *Readmill*

Figure 5: Three example Web services and their neighbors. All these services belong to the "Social" primary category, however, we can find that they are of different functionality patterns.

The overview of the resulting map is shown in Fig. 4.

From the overview, we can find that there are many service clusters, and each cluster is dominated by a few giant services. Such as *Google Maps* and *Microsoft Bing Maps* of "Mapping" category, *YouTube* and *Netflix* of "Video" category and so on. Besides, there are also clusters that are relatively independent of others, like the cluster in the lower right corner, which is mainly Bitcoin-related services. Generally, the services within the same category are gathered together.

C. Analyses of representative functionality patterns

To dive into the details of the map, we take three services as examples and analyze their neighbors in Fig. 5. According to the profiles given by the providers, all these services belong to the "Social" primary category, but we can find that they are of quite different functionality patterns.

(1) Inclining. Fig. 5a demonstrates the neighbors of *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*, conforming that *LinkedIn* is "*a business social networking hub*." This shows the phenomenon that a service is majoring in one domain, but also providing auxiliary functionalities.

(2) Balancing. *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 "Social" category and "Video" category in Fig. 5b. If developers want to make compositions relating to both functionalities, *GetGlue* would be a potential choice.

(3) Realigning. *Readmill* is an interesting example for investigation. The profile of *Readmill* 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 involve *Readmill* generally do not focus on "Social" properties. For example, mashup *ReadTracker* is "*an app for keeping track of the books you read*" and *Readmap* "*lets users plot a map of all the places they have been reading*." As a result, the social feature of *Readmill* degenerates and it becomes more related to "Books" category than "Social" as shown in Fig. 5c.

As we can see, such a knowledge map constructed for ProgrammableWeb service ecosystem is able to provide insights for mashup developers and service ecosystem managers, benefiting from the effective representations learned by SR-LDA.

VI. Related Work

In this section, we describe several representative related works and differentiate them with our approach.

A. Topic modeling

In order to generate service ecosystem knowledge maps, we first build representations for services. Topic modeling becomes popular for mining topic proportions from text since Latent Dirichlet Allocation (LDA) was proposed in [7]. Based on LDA, a variety of topic models have been proposed. Correlated Topic Model [15] extends LDA and models the relationship between topics. Author-Topic model [9], [10] models authors and documents at the same time. Dynamic Topic Model [16] captures the topic changing over time. Beyond the framework of LDA, there are also non-linear methods like Stacked Denoising Autoencoders (SDAE) [17]. However, non-linear methods lack interpretability and often require much more training data, thus cannot give as many intuitions as generative topic models [9].

For Web service ecosystems, the relationship of mashup-service-word becomes the primary issue that we need to consider. Thus we have designed a tailored generative process for mashups and services, and introduced a global filter for background terms.

B. Visualization for Web service ecosystem

Among the existing work for visualization in Web service field, [18] propose a QoS-aware service recommendation algorithm and use a recommendation visualization technique to show the similarity of RTT variance of Web services. Beyond the Web service field, knowledge visualization remains an active topic for research. [19] proposes a knowledge map-based method for domain knowledge browsing. [20] presents an automated approach for constructing topic knowledge maps with knowledge structures. [21] proposes t-Distributed Stochastic Neighbor Embedding (t-SNE) for visualization of high-dimension data.

In this paper, we build a knowledge map of ProgrammableWeb service ecosystem based on the representations by SR-LDA. Experiments show that building service ecosystem knowledge maps is able to provide insights for mashup developers and service ecosystem managers.

VII. Conclusions

As the number of reusable Web services keeps growing, making a value-added software has become amazingly convenient. However, the overloaded information of Web services poses a great challenge for mashup developers and service ecosystem managers to gain an overview cognition of all the services in the repository. Building knowledge maps becomes a solution to this problem.

However, considering the unique characteristics of Web service ecosystem, vanilla topic models cannot build effective service representations. In this paper, for the evolution property of Web services, we incorporated mashup descriptions to enrich the static profiles of services. Besides, we introduced a global filter to down-weight the adverse impact of background terms and the quality of representations is further improved. Both quantitative and qualitative experiments show that our proposed SR-LDA is more effective than baselines. Upon the representations learned by SR-LDA, we built a knowledge map of ProgrammableWeb service ecosystem, which yielding some interesting results.

In the future, we plan to incorporate more information, such as the comments on the Web services and temporal information explicitly, for building more effective representations. We also plan to develop better visualization tools for service ecosystems.

Acknowledgment

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

References

[1] V. Andrikopoulos, S. Benbernou and M. P. Papazoglou, ”On the Evolution of Services,” *IEEE Transactions on Software Engineering*, 2012, 38, pp. 609-628.

[2] J. Zhang, R. Shi, W. Wang, S. Lu, Y. Bai and Q. Bao, “A bloom filter-powered technique supporting scalable semantic service discovery in service networks,” in proceedings of *IEEE International Conference on Web Services*, 2016, pp. 81-90.

[3] Z. Gao, Y. Fan, C. Wu, W. Tan, J. Zhang, Y. Ni, B. Bai and S. Chen, “SeCo-LDA: mining service co-occurrence topics for recommendation,” in proceedings of *IEEE International Conference on Web Services*, 2016, pp. 25-32.

[4] X. Liu and I. Fulia, “Incorporating user, topic, and service related latent factors into web service recommendation,” in proceedings of *IEEE International Conference on Web Services*, 2015, pp. 185-192.

[5] C. Li, R. Zhang, J. Huai, X. Guo and H. Sun, “A Probabilistic Approach for Web Service Discovery,” in proceedings of *IEEE International Conference on Services Computing*, 2013, pp. 49-56.

[6] C. Zins, “Knowledge map of information science,” *Journal of the Association for Information Science and Technology*, 2007, 58(4), pp. 526-535.

[7] D. M. Blei, A. Y. Ng and M. I. Jordan, “Latent dirichlet allocation,” *Journal of Machine Learning Research*, 2003, 3(Jan), 993-1022.

[8] K. Huang, Y. Fan, W. Tan and X. Li, ”Service recommendation in an evolving ecosystem: a link prediction approach,” in proceedings of *IEEE International Conference on Web Services*, 2013, pp. 507-514.

[9] M. Rosen-Zvi, C. Chemudugunta, T. Griffiths, P. Smyth and M. Steyvers, “Learning author-topic models from text corpora,” *ACM Transactions on Information Systems*, 2010, 28(1), pp. 312-324.

[10] M. Steyvers, P. Smyth, M. Rosen-Zvi and T. Griffiths, “Probabilistic author-topic models for information discovery,” in proceedings of *ACM International Conference on Knowledge Discovery and Data Mining*, 2004, pp. 306-315.

[11] 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 & Computer Sciences*, 2016, 50(4), pp.271-277.

[12] D. L. Davies and D. W. Bouldin, “A cluster separation measure” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1979, (2), pp.224-227.

[13] M. Jacomy, T. Venturini, S. Heymann and M. Bastian, “ForceAtlas2, a continuous graph layout algorithm for handy network visualization designed for the gephi software,” *Plos One*, 2014, 9(6), e98679.

[14] M. Bastian, S. Heymann and M. Jacomy, “Gephi: An Open Source Software for Exploring and Manipulating Networks,” in proceedings of *International Conference on Weblogs and Social Media*, 2009, pp.361-362.

[15] D. M. Blei and J. D. Lafferty, “A correlated topic model of science,” *The Annals of Applied Statistics*, 2007, pp. 17-35.

[16] D. M. Blei and J. D. Lafferty, “Dynamic topic models,” in proceedings of *ACM International Conference on Machine Learning*, 2006, pp. 113-120.

[17] 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*, 2010, 11(Dec), pp. 3371-3408.

[18] X. Chen, Z. Zheng, X. Liu, Z. Huang and H. Sun, “Personalized qos-aware web service recommendation and visualization,” *IEEE Transactions on Services Computing*, 2013, 6(1), pp. 35-47.

[19] J. Hao, Y. Yan, L. Gong, G. Wang and J. Lin, “Knowledge map-based method for domain knowledge browsing,” *Decision Support Systems*, 2014, 61(1), pp. 106-114.

[20] D. Y. Chiu and Y. C. Pan, “Topic knowledge map and knowledge structure constructions with genetic algorithm, information retrieval and multi-dimension scaling method,” *Knowledge-Based Systems*, 2014, 67(3), pp. 412-428.

[21] L. v. d. Maaten and G. Hinton, “Visualizing data using t-sne,” *Journal of Machine Learning Research*, 2008, 9, 2579-2605.