Topic-aware Web Service Representation Learning

MIN SHI, Florida Atlantic University, USA; Hunan University of Science and Technology, China
YUFEI TANG and XINGQUAN ZHU, Florida Atlantic University, USA
JIANXUN LIU, Hunan University of Science and Technology, China

The advent of Service-Oriented Architecture (SOA) has brought a fundamental shift in the way in which distributed applications are implemented. An overwhelming number of Web-based services (e.g., APIs and Mashups) have leveraged this shift and furthered development. Applications designed with SOA principles are typically characterized by frequent dependencies with one another in the form of heterogeneous networks, i.e., annotation relations between tags and services, and composition relations between Mashups and APIs. Although prior work has shown the utility gained by exploring these networks, their analysis is still in its infancy. This article develops an approach to learning representations of the Web service network, which seeks to embed Web services in low-dimensional continuous vectors with preserved information of the network structure, functional tags, and service descriptions, such that services with similar functional properties and network structures are mapped together in the learned latent space. We first propose a topic generative model for constructing two topic distribution networks (Mashup-Topic and API-Topic) from the service content. Then, we present an efficient optimization process to derive low-dimensional vector representations of Web services from a tri-layer bipartite network with the Mashup-Topic and API-Topic networks on two ends and the Mashup-API composition network in the middle. Experiments on real-word datasets have verified that our approach is effective to learn robust low-rank service representations, i.e., 25% F1-measure gain over the state-of-the-art in Web service recommendation task.

CCS Concepts: • **Software and its engineering → Software creation and management**; • **Applied computing** → *Document management and text processing*; • **Computing methodologies** → *Learning latent representations*;

Additional Key Words and Phrases: Web services, Mashups, service representation, network embedding, probabilistic topic model

ACM Reference format:
Min Shi, Yufei Tang, Xingquan Zhu, and Jianxun Liu. 2020. Topic-aware Web Service Representation Learning. *ACM Trans. Web* 14, 2, Article 9 (April 2020), 23 pages.
https://doi.org/10.1145/3386041

1 INTRODUCTION

Many complex systems in the real world take the form of networks, e.g., social, citation, and biological networks [1]. Similarly, Web services do not exist for independent use alone, but for composition with other modular services [2, 3], naturally forming a network between collaborating

This work is supported in part by the US National Science Foundation (NSF) through Grant Nos. IIS-1763452 and CNS-1828181, and the National Natural Science Foundation of China under Project No. 61872139.
Authors' addresses: M. Shi, Y. Tang, and X. Zhu, the Department of Computer & Electrical Engineering and Computer Science, Florida Atlantic University, FL 33431; emails: {mshi2018, tangy, xzhu3}@fau.edu; J. Liu, the School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan 411201, China; email: ljx529@gmail.com.

1559-1131/2020/04-ART9
https://doi.org/10.1145/3386041

services, e.g., Application Programming Interfaces (APIs) [4]. One such example is combining the addresses and photographs of related locations in Google Maps to create a new Mashup. Networked systems inherently interpret rich structure and semantic relations between entities [5, 6]. For example, in Facebook, users form a homogenous social network where the edges can indicate similar preference shared by corresponding users on each end; in search engines, queries and webpages form a heterogeneous bipartite network, where the edges can indicate users' click behavior, providing a valuable signal for relevance [7]. In the bipartite composition network of Web services, edges usually reveal the functional similarity and information interplay between linked services, i.e., the operation fulfillment of a Mashup service heavily relies on the execution and data exchange of some component API services.

Researchers across many disciplines have become increasingly aware of the importance of analyzing these aforementioned information networks to support many emerging applications [8, 9]. For example, in social networks, classifying users into meaningful interest groups is useful for many downstream tasks such as user search, targeted advertising, and recommendation. Existing work has also confirmed the powerful benefits brought to Web service recommendation [6] and mobile application recommendation [8] by considering the implicit knowledge derived from service linking relations. However, analyzing Web service networks specifically and comprehensively by considering various types of entities (e.g., Mashups, APIs, and functional tags) and relationships (e.g., composition relationships and annotation relationships) is still in its infancy.

Traditionally, to analyze a network, the first step is to obtain the representations of all vertices usually in the form of an adjacency matrix with elements indicating whether pairs of vertices are adjacent or not [10, 11]. However, such bag-of-word representations are subject to the curse of dimensionality and cannot capture rich structural and semantic information [12]. Recently, network embedding, also known as network representation learning, was proposed to learn low-dimensional feature vectors for each vertex while preserving underlying rich network structure and other auxiliary information such as textual content [13, 14]. To date, existing works have primarily focused on learning homogenous networks where vertices are typically of the same type [5, 15]. They cannot be directly generalized to Web service network for the following two reasons:

(1) Web service networks are not directly comparable to homogeneous networks. For example, there are two types of entities (e.g., Mashup and API) in a composition Mashup-API network, where each Mashup has functional dependencies with all the composed APIs [16]. In other words, the functional semantics of each Mashup should be close to the collective functional semantics of its entire composed APIs.
(2) In addition to the heterogeneous dependency between Mashup and API services, there are essentially implicit relationships between entities of the same type, i.e., the intra-layer relationships between Mashups are characterized by their shared functional tags. It is necessary to reveal such subtle relationships to accurately characterize affinities between Web services.

Figure 1 shows a real Web service network from ProgrammableWeb, which has three types of entities organized in three layers of an interdependent bipartite network, with two Mashup-Tag and API-Tag networks on each end and a Mashup-API network in the middle. In this article, we aim to develop an effective approach for learning the low-dimensional semantic or vector representations of Mashup and API services from the tri-layer bipartite network shown in Figure 1. The learning process seeks to reach two objectives: (1) the semantic of each Mashup should be similar to the collective semantics of all its component APIs; and (2) the Mashups (or APIs) with shared functional properties should be similar. In addition, recent studies [13, 17] show that the content information of a network is capable of measuring the affinities between vertices aligned with that

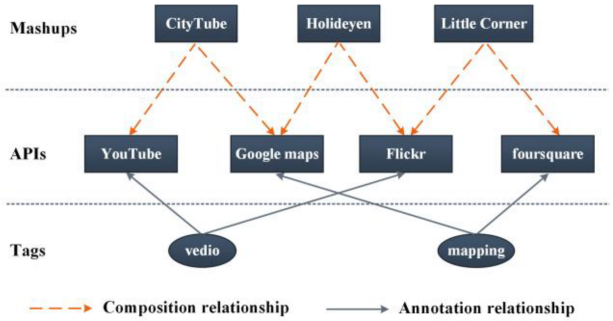


Fig. 1. An example Web service network among Mashups, APIs, and tags. For simplicity, the annotation relationships between Mashups and tags are not shown for the case where Mashups and APIs share the tag space. The composition relationships characterize the functional dependencies between Mashups and APIs, while the annotation relationships characterize the functional similarities between Mashups and APIs.

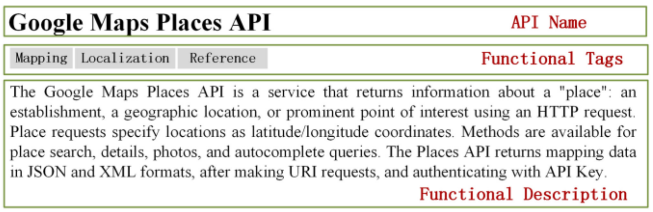


Fig. 2. A real API service from ProgrammableWeb. The description describes complex functional semantics abstracted by multiple tags.

revealed by network structures. For example, Figure 2 shows a real-world example API with rich content such as functional tags and description. Apart from the tags that have been considered in Figure 1, we argue that it is necessary to consider the functional descriptions to enhance the structure-based embedding, which may repair some missing relationships between vertices that have not been captured by the sparse network connectivity.

However, learning a vector representation for each service while simultaneously preserving network structures (composition relations) and the functional contents (tags and descriptions) is nontrivial. In this article, we propose to abstract the description and tags of each service with a functional topic distribution vector to model relations between services from the content perspective. As a result, the Mashup-Tag and API-Tag networks in Figure 1 will be transformed into Mashup-Topic and API-Topic distribution networks that reflect the subtler and richer functional semantics (see Figure 5 in Section 3.2). The topic distribution networks are then used to reconstruct the service representations. Meanwhile, we force the semantic (representation) of each Mashup service to be equivalent to the collective semantics of its composed API services to model relationships from the network structure perspective. The two aspects are optimized in a unified framework to derive the final service representations.

Specifically, our contributions are threefold:

(1) We propose a novel generative model, PV-LDA, to elicit functional topics from descriptions and tags of Web services. To learn accurate latent topics from the sparse and noisy service content, we first obtain the semantic relevance distribution between words and functional tags in each based on the widely used paragraph vector model. Then, the topic assignment for each word is refined by a collective weighted Gaussian influence from the related tags.

(2) We present an effective approach called TWSRL to represent each service as a low-rank semantic vector by encoding the rich structure and content information. The content information is preserved through representation learning from the bipartite Mashup-Topic and API-Topic distribution networks while the structural information is preserved via representation optimization within the bipartite Mashup-API composition network.
(3) Extensive experiments on two real-world datasets demonstrate the effectiveness of our proposed approach to learning Web service representations on two popular tasks, including the Web service classification and recommendation.

The rest of this article is organized as follows: Section 2 defines the problem. Section 3 introduces the proposed topic model for functional topic extraction, followed by the proposed network embedding approach for Web service representation learning. Section 4 describes the extensive experiments used to evaluate the proposed approach. Section 5 reviews the related work. Finally, Section 6 concludes the article.

2 PROBLEM DEFINITION

In this section, we first define the well-studied Web service network with a tri-layer bipartite structure, followed by the formulation of the **T**opic-aware **W**eb **S**ervice **R**epresentation **L**earning (TWSRL) problem. In this article, Web services refer to all web-based Mashup and API services.

Definition 1 (Mashup-API Composition Network). Let $\mathbf{G}_{m,a} = (\mathbf{V}_m, \mathbf{V}_a, \mathbf{E}_{m,a})$ be the bipartite Mashup-API composition network, where $\mathbf{V}_m = \{v_i\}_{i=1,\ldots,|\mathbf{V}_m|}$ and $\mathbf{V}_a = \{v_j\}_{j=1,\ldots,|\mathbf{V}_a|}$ are the Mashup service and API service sets, respectively. $\mathbf{E}_{m,a} \in \mathbf{V}_m \times \mathbf{V}_a$ defines the inter-set edges of the composition relationship between Mashups and APIs, i.e., $\mathbf{E}_{m,a}(i,j) = 1$ indicates the jth API is a member of the ith Mashup.

Definition 2 (Mashup-Topic and API-Topic Distribution Networks). Let $\mathbf{G}_{m,t} = (\mathbf{V}_m, \mathbf{V}_t, \mathbf{E}_{m,t})$ be the bipartite Mashup-Topic distribution network. $\mathbf{V}_t = \{v_k\}_{k=1,\ldots,|\mathbf{V}_t|}$ denotes the set of latent topics used to abstract the diverse functional properties of Web service contents (tags and textual descriptions). $\mathbf{E}_{m,t} \in \mathbf{V}_m \times \mathbf{V}_t$ defines a set of edges representing the topic distributions of Mashups, i.e., $\mathbf{E}_{m,t}(i,k) = w_{i,k}$ indicates that the ith Mashup is relevant to the corresponding functional semantic revealed by the kth topic, with the relevance intensity quantified by $w_{i,k}$. Similarly, let $\mathbf{G}_{a,t} = (\mathbf{V}_a, \mathbf{V}_t, \mathbf{E}_{a,t})$ be the bipartite API-Topic distribution network, where $\mathbf{E}_{a,t} \in \mathbf{V}_a \times \mathbf{V}_t$ defines a set of edges representing the topic distributions of APIs, with each edge carrying a weight indicating the relevance intensity between the corresponding API and functional topic. It is worth noting that the latent topic space $\mathbf{V}_t$ is shared by all Mashup and API services.

Problem Definition. The tri-layer bipartite Web service network is represented as $\mathbf{G} = (\mathbf{G}_{m,t}, \mathbf{G}_{m,a}, \mathbf{G}_{a,t})$ or $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, which is a tri-layer bipartite network (see Figure 5 in Section 4.2), with the Mashup-Topic and API-Topic networks on each end and the Mashup-API network in the middle, where $\mathbf{V} = \mathbf{V}_m \cup \mathbf{V}_a \cup \mathbf{V}_t$ and $\mathbf{E} = \mathbf{E}_{m,t} \cup \mathbf{E}_{m,a} \cup \mathbf{E}_{a,t}$. Web service representation learning aims to represent each node $v_i \in \mathbf{V}$ with a low-dimensional semantic vector $\mathbf{h}_{v_i} \in \Re^{n_d}$, i.e., learning a mapping $f : \mathbf{G} \rightarrow \{\mathbf{h}_{v_i}\}_{i=1,\ldots,|\mathbf{V}|}$ so both the Web service functional content and network structures can be well preserved, where n_d is the dimension of learned node vectors. In this article, the expression "semantic vector" and "semantic representation" are used interchangeably.

3 METHODOLOGY OVERVIEW

As previously discussed, it is beneficial to incorporate content information together with network structures for semantically incremental representation learning [13, 17]. However, it is prohibitive to directly model the various forms of content (e.g., tags and descriptions) and the composition

Table 1. Notations Used in the Proposed Models

Notations	Description
$\mathbf{V}_m$	The set of Mashup services
$\mathbf{V}_a$	The set of API services
$\mathbf{V}$	The set of Mashup and API services, $\mathbf{V} = \mathbf{V}_m \cup \mathbf{V}_a$
$\mathbf{V}_t$	The set of latent topics
$\mathbf{E}_{m,t}$	The set of edges representing the topic distributions of Mashups
$\mathbf{E}_{a,t}$	The set of edges representing the topic distributions of APIs
$\mathbf{D}$	The set of description texts of Mashup and API services
$\mathbf{A}$	The set of tags of a specific service $s \in \mathbf{D}$
$\mathbf{W}$	The set of words of the service $s \in \mathbf{D}$
N	The total number of words in the service repository
T	The number of functional topics trained in PV-LDA
$\boldsymbol{\theta}$	The functional topics distribution of all services, $\boldsymbol{\theta} \in \mathbb{R}^{\lvert\mathbf{D}\rvert \times T}$
w_i	The ith word $w_i \in \mathbf{W}$
a_j	The jth tag $a_j \in \mathbf{A}$
Z_i	The topic assignment for word $w_i \in \mathbf{W}$
Z_j	The topic assignment for tag $a_j \in \mathbf{A}$
$\boldsymbol{\varphi}_w$	The words distribution for topic Z_i, $\boldsymbol{\varphi}_w \in \mathbb{R}^{1 \times N}$
$\boldsymbol{\varphi}_a$	The tags distribution for topic Z_j, $\boldsymbol{\varphi}_a \in \mathbb{R}^{1 \times N}$
$\boldsymbol{\pi}$	The relevance distribution of word w_i with tags in $\mathbf{A}$, where $\boldsymbol{\pi} = \{\sigma_{i,k}\}_{k=1,\ldots,\lvert\mathbf{A}\rvert}$
$p_{i,t}$	The probability that the ith word is assigned with topic t, where $t = Z_i$
$p_{k,t}$	The probability that the kth tag is assigned with topic t, where $t = Z_i$
$\boldsymbol{\gamma}$	The probability distribution of tags in $\mathbf{A}$ on topic t, where $\boldsymbol{\gamma} = \{p_{k,t}\}_{k=1,\ldots,\lvert\mathbf{A}\rvert}$
$\sigma_{i,k}$	The semantic relevance value between ith word $w_i \in \mathbf{W}$ and the kth tag $a_k \in \mathbf{A}$
$\delta_{i,t}$	The collective weighted influence by all tags in $\mathbf{A}$ exerted on word w_i being assigned topic t, where $t = Z_i$
$\alpha,\ \tau$	The prior parameters for the model
β_w, β_a	The prior parameters for the model

relations in a unified framework. Although there are some existing embedding methods such as PLANE [18] and TriDNR [19] that can collectively incorporate contents and structures, unfortunately almost all of them treat the contents of each vertex as a set of flat word features to characterize a simplex semantic of the corresponding vertex. This assumption is inappropriate for Web service contents with complex semantics paired with multiple functional tags (refer to the API service in Figure 2 as an example). Therefore, in this section, we first propose reformulating the content of each service as a collection of topics following some distribution shown in Figure 5, with each topic carrying a different part of the functional semantics. We then couple service-topic distribution networks with the Mashup-API composition network for unified Web service representation learning. For easy reference, Table 1 has defined the notations used in our method.

3.1 Topic Distribution Learning of Web Services

In the past, probabilistic topic models have been widely used to extract the latent semantics of natural language documents [18, 20] based on the Gibbs sampling process [21] in an unsupervised manner. Such a paradigm works well when substantial observed word samples (e.g., large corpus and long documents) are available for ample feature statistics and topic inference [22, 23]. However, Web service descriptions are usually short and presented with various writing styles,

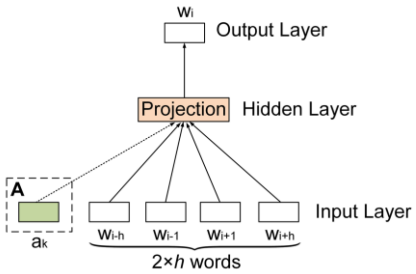


Fig. 3. The paragraph vector model.

i.e., developers may use different phrases to express the same concepts [24], making it difficult to discover the rich feature co-occurrence patterns heavily relied on by most existing topic models.

To address the aforementioned problem, we propose a novel probabilistic generative model trained in a semi-supervised way by taking the functional tags as prior knowledge to guide the Gibbs sampling process. More specifically, we argue that the functional tags have exhibited a prior high-level abstraction of the corresponding functional description (refer to Figure 2 as an example). For a specific service, each tag reveals a functional semantic contributed by all words following a relevance distribution, i.e., different words in the Web service description could carry different functional semantics. This observation has motivated us towards the semi-supervised topic modeling; that is, topic assignments for tags together with the semantic relevance distribution between words and functional tags in each service can help guide and refine the topic sampling for common words. In addition, functionally equivalent Web services may use different tags or descriptions that express similar semantics [6, 16]. Linking tags with their corresponding words together would help to bridge the vocabulary gap between similar services caused by either different tags or different description words expressing the same concepts.

With this in mind, we first obtain the relevance distribution between words and functional tags for each service based on the widely used Paragraph Vector Model (PVM) [25]. PVM is a powerful neural network model to derive the semantic vectors of terms that allows for the similarity calculation between words and tags. PVM takes the service descriptions and tags (e.g., each word w_i or tag a_k first initialized with a random vector representation $\mathbf{h}_{w_i}$ or $\mathbf{h}_{a_k}$) as inputs and then outputs the optimized word and tag vector representations. The vector optimization framework of PVM is shown in Figure 3, which seeks to use word context together with related functional tags to predict the corresponding target word within the same word sequence moving window. More specifically, given the word sequence $\mathbf{W} = w_1, w_2, \ldots, w_{|\mathbf{W}|}$ of a service description $s \in \mathbf{D}$, the learning objective of PVM is to maximize the probability over all service descriptions and tags:

$$\mathcal{L} = \sum_{s}^{\mathbf{D}} \sum_{i=1}^{|\mathbf{W}|} \sum_{k=1}^{|\mathbf{A}|} \log p(w_i | w_{i-h}, \ldots, w_{i+h}, a_k), \tag{1}$$

where $\mathbf{W}$ and $\mathbf{A}$ are the word set and tag set of service s. The moving window size is noted as h and $w_{i-h}, \ldots, w_{i+h}$ are words surrounding the target word w_i in the sequence. The prediction of word w_i can be estimated by a multiclass classifier defined by the softmax function [25]:

$$p(w_i | w_{i-h}, \ldots, w_{i+h}, a_k) = \frac{\exp\left(\bar{\mathbf{h}}_{w_i}^T \mathbf{h}_{w_i}\right)}{\sum_{j=1}^{N} \exp\left(\bar{\mathbf{h}}_{w_i}^T \mathbf{h}_{w_j}\right)}, \tag{2}$$

where N is the total number of words in the vocabulary. $\bar{\mathbf{h}}_{w_i}$ and $\mathbf{h}_{w_i}$ are the input and output (e.g., optimized representation) vectors of word w_i, respectively. $\bar{\mathbf{h}}_{w_i}$ is computed by concatenating ($\oplus$)

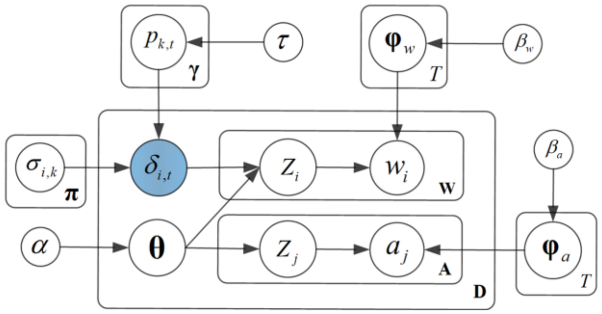


Fig. 4. The notation representation of PV-LDA model.

the vector of tag a_k and the averaged vector of all words within the moving window by:

$$\bar{\mathbf{h}}_{w_i} = \mathbf{h}_{a_k} \oplus \left(\frac{1}{2h} \sum_{-h \le j \le h, j \ne 0} \mathbf{h}_{w_{i+j}} \right). \tag{3}$$

The PVM can be trained via the gradient descent [25] to iteratively optimize word and tag semantic representations. After training the PVM, a semantic relevance value is computed between each word $w_i \in \mathbf{W}$ and each tag $a_k \in \mathbf{A}$ by the cosine method:

$$\sigma_{i,k} = \cos(\mathbf{h}_{w_i}, \mathbf{h}_{a_k}). \tag{4}$$

With the relevance distributions obtained in Equation (4), we then propose a generative model for functional topic extraction as shown in Figure 4, called Paragraph Vector augmented LDA, or PV-LDA for simplicity. Please refer to Table 1 for the relevant notations used in PV-LDA.

PV-LDA assumes that each service s is composed of a set of words ($\mathbf{W}$) together with a set of tags ($\mathbf{A}$) generated by a mixture of latent variables or topics. Each service follows a distribution over all topics $\boldsymbol{\theta}$, with each topic (Z_i or Z_j) being responsible for generating a set of words and tags. The Gibbs training process of PV-LDA is used to infer the posterior functional topic distributions $\boldsymbol{\theta}$ of services from all observed words and tags based on the prior Dirichlet allocation parameters β_w and β_a. PV-LDA is summarized in Algorithm 1. First, the model samples a mixture of words and tags for each functional topic (Lines 1–4). It then samples topic distribution for each service s (Line 6), which is subsequently used to literately generate all included tags (Lines 7–10) and words (Lines 11–16). When sampling the topic for word w_i, the collective Gaussian influence enforced by all tags have been drawn previously (Lines 12–13) to supervise the final topic assignment (Line 14) of word w_i, with $p_{i,t}$ and τ being the mean and standard deviation, respectively.

The inference of all latent variables in PV-LDA can be performed through the Gibbs sampling process, where a Markov Chain Monte Carlo chain is created to update the topic assignment of all words and tags. Similar to the basic LDA model [26], the probability to assign topic t to tag a_j in PV-LDA is given by:

$$p(Z_j = t) \propto \frac{n(a_j, t)_{\neg j} + \beta_a}{\sum_{q \in \mathbf{W} \cup \mathbf{A}} n(q, t) + N\beta_a} \times \frac{n(s, t)_{\neg j} + \alpha}{\sum_{f \in [1,T]} n(s, f) + T\alpha}, \tag{5}$$

where $n(a_j, t)$ is the number of times topic t is allocated for tag a_j, and $n(s, t)$ is the number of times topic t appears in tags and words of service s. The symbol of $\neg j$ means the jth tag is not counted in the statistic. Then, by taking the semantic relevance with tags as prior auxiliary information, the

ALGORITHM 1: The Generative Procedure of PV-LDA

Required: words **W** of a service s
Required: tags **A** of a service s
Required: semantic relevance distribution $\boldsymbol{\pi}$ of each word $w_i \in \mathbf{W}$ with all tags in **A**
procedure begin:
1. **for each** topic $t \in [1, T]$ **do**
2. sample the mixture of words $\boldsymbol{\varphi}_w \sim Dirichlet(\beta_w)$
3. sample the mixture of tags $\boldsymbol{\varphi}_a \sim Dirichlet(\beta_a)$
4. **end for**
5. **for each** service $s \in \mathbf{D}$ **do**
6. sample the mixture of topics $\boldsymbol{\theta} \sim Dirichlet(\alpha)$
7. **for each** tag $a_j \in \mathbf{A}$ **do**
8. sample a topic $z_j \sim Multinomial(\boldsymbol{\theta})$
9. sample the tag $a_j \sim Multinomial(\boldsymbol{\varphi}_a)$
10. **end for**
11. **for each** word $w_j \in \mathbf{W}$ **do**
12. for each tag $a_k \in \mathbf{A}$, sample $p_{k,t} \sim \mathcal{N}(p_{i,t}, \tau)$
13. sample $\delta_{i,t} \sim Multinomial(\sigma_{i,k}, p_{k,t})$
14. sample a topic $z_i \sim Multinomial(\boldsymbol{\theta}, \delta_{i,t})$
15. sample the word $w_i \sim Multinomial(\boldsymbol{\varphi}_w)$
16. **end for**
17. **end for**
end procedure

probability to assign topic t to word w_i is given by:

$$p^*(Z_i = t) \propto p_{i,t} \times \prod_{k \in [1, |\mathbf{A}|]}^{p_{k,t} \in \boldsymbol{\gamma}, \sigma_{i,k} \in \boldsymbol{\pi}} \frac{\sigma_{i,k}}{\sqrt{2\pi}\tau} \exp\left(-\frac{(p_{k,t} - p_{i,t})^2}{2\tau^2}\right), \tag{6}$$

where $p_{i,t}$ is calculated by:

$$p(Z_i = t) \propto \frac{n(w_i, t)_{\neg i} + \beta_w}{\sum_{q \in \mathbf{W} \cup \mathbf{A}} n(q, t) + N\beta_w} \times \frac{n(s, t)_{\neg j} + \alpha}{\sum_{f \in [1,T]} n(s, f) + T\alpha}, \tag{7}$$

where $n(w_i, t)$ is the number of times topic t is allocated for word w_i. Equation (7) calculates the sampling probability $p_{i,t}$ of word w_i on topic t similar to the basic LDA model without considering the topic enhancement of tags. However, in PV-LDA the topic sampling of a common word is refined by the collective weighted Gaussian influence of all tags (the second term in Equation (6)). The reason for choosing the normal distribution to characterize the influence of tag a_k on word w_i is twofold: (1) the influence increases dramatically when a_k and w_i have very close probability on topic t; (2) the influence decreases gradually otherwise. Moreover, the prior semantic relevance of $\sigma_{i,k}$ between a_k and w_i exists to control the final influence intensity exerted by each individual tag, i.e., if a_k and w_i are not semantically relevant (e.g., $\sigma_{i,k}$ is very small), then the corresponding Gaussian influence would degrade to be invisible. With above favorable properties, the proposed PV-LDA model is able to learn functional topics preserving both the tags and descriptions in an enhanced manner.

After above topic sampling process, the functional topic distribution of a service s can be represented as $\boldsymbol{\theta} = \{\boldsymbol{\theta}_t\}_{t=1,\ldots,T}$, with $\boldsymbol{\theta}_t$ being computed by [16]:

$$\boldsymbol{\theta}_t = \frac{n(w_i, t) + \beta_w}{\sum_{q \in \mathbf{W} \cup \mathbf{A}} n(q, t) + N\beta_w}. \tag{8}$$

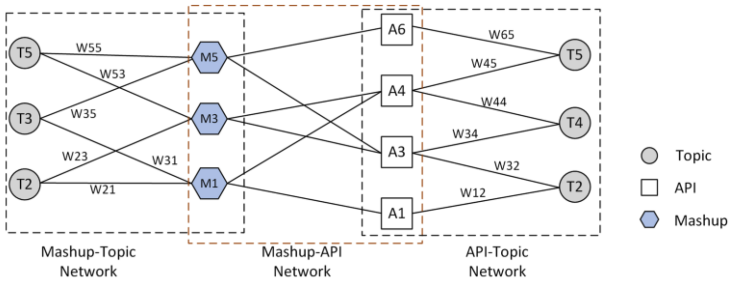


Fig. 5. The hierarchical bipartite Web service network structure among topics, Mashups, and APIs.

3.2 Representation Learning of Tri-layer Bipartite Web Service Network Structure

To this end, we have derived the API-Topic and Mashup-Topic distribution networks based on Equation (8) with all learned topics being topic nodes in the network. Each Mashup or API service has edges with its relevant topics, where edges are weighted indicating the probabilities of services belonging to the respective topics. Combined with the Mashup-API composition network, the target network studied in this article exhibits a tri-layer bipartite structure shown in Figure 5. It visualizes with the topic distribution networks on two ends revealing the functional semantic distributions of services from the content perspective (e.g., M5 connects to topics T3 and T5, which are learned from the text content associated with M5), and the composition network in the middle capturing the semantic interplay between Mashups and APIs. In the following, we elaborate on the unified optimization process to learn representations of services simultaneously from the network contents and structures.

3.2.1 Network Content Preservation. The content preservation is achieved by representation learning based on the Mashup-Topic and API-Topic distribution networks. To learn meaningful and discriminative representations, we argue that the learned service representations should be capable of reconstructing the original networks, i.e., services with similar topic distributions should also have similar semantic representations and vice versa. Similar to the first-order relations modeling mechanism adopted in LINE [27], we focus on the observed local proximities (e.g., each service exhibits top K most relevant topics out of the T topics in our setting) over the whole topic distribution networks, where the joint probability between the ith Mashup (or the jth API) and the kth topic is defined as:

$$p_{m,t}(i,k) = \frac{w_{ik}}{\sum_{e_{ik} \in \mathbf{E}_{m,t}} w_{ik}} \tag{9}$$

or

$$p_{a,t}(j,k) = \frac{w_{jk}}{\sum_{e_{jk} \in \mathbf{E}_{a,t}} w_{jk}}, \tag{10}$$

where w_{ik} (or w_{jk}.) is the probability/weight that the ith Mashup (or the jth API) emphasizes the kth topic. Equations (9) and (10) reflect the empirical closeness of each pair of observed nodes, which can then be used to reconstruct the affinities between nodes in the embedding space. We represent the joint probability between the ith Mashup (or the $j$thAPI) and the kth topic in the embedding space by inner product through a sigmoid transformation. They are represented as:

$$\hat{p}_{m,t}(i,k) = \frac{1}{1 + \exp\left(-\mathbf{h}_{v_i}{}^T \mathbf{h}_{v_k}\right)}, \tag{11}$$

$$\hat{p}_{a,t}(j,k) = \frac{1}{1 + \exp\left(-\mathbf{h}_{v_j}{}^T \mathbf{h}_{v_k}\right)}, \tag{12}$$

where the $\mathbf{h}_{v_i}$, $\mathbf{h}_{v_j}$, and $\mathbf{h}_{v_k}$ are the low-rank representations of the corresponding Mashup, API, and topic, respectively. The optimization goal is to minimize the difference between the empirical distributions under the functional topic spacand the semantic distributions under the embedding space. We choose the Kullback-Leibler (KL) divergence as difference measure and minimize the following objectives corresponding to the Mashup-Topic and API-Topic distribution networks, respectively:

$$\mathcal{L}_1 = KL(p_{m,t}|\hat{p}_{m,t}) = \sum_{e_{ik}\in\mathbf{E}_{m,t}} p_{m,t}(i,k)\log\left(\frac{p_{m,t}(i,k)}{\hat{p}_{m,t}(i,k)}\right) \propto -\sum_{e_{ik}\in\mathbf{E}_{m,t}} w_{ik}\log\hat{p}_{m,t}(i,k), \tag{13}$$

$$\mathcal{L}_2 = KL(p_{a,t}|\hat{p}_{a,t}) = \sum_{e_{jk}\in\mathbf{E}_{a,t}} p_{a,t}(j,k)\log\left(\frac{p_{a,t}(j,k)}{\hat{p}_{a,t}(j,k)}\right) \propto -\sum_{e_{jk}\in\mathbf{E}_{a,t}} w_{jk}\log\hat{p}_{a,t}(j,k). \tag{14}$$

The optimization process of Equations (13) and (14) guarantees that the functional semantics derived from the service content can be described by the learned low-dimensional representations of services.

3.2.2 Network Structure Preservation. The structure preservation is achieved by implicit semantic interactions between Mashups and APIs over the Mashup-API composition network. We develop a translation-based mechanism to characterize the composition relationships between Mashups and all member APIs under the principle that each Mashup is semantically equal to its member APIs, i.e., $\mathbf{h}_{M1} \approx \mathbf{h}_{A1} + \mathbf{h}_{A4}$ in Figure 5, where $\mathbf{h}_{M1}$, $\mathbf{h}_{A1}$, and $\mathbf{h}_{A4}$ are vector representations of services *M1, A1*, and *A4*, respectively. The basic idea is that semantic of each Mashup is collectively composed by corresponding APIs, which respects the fact that a Mashup service is normally developed by integrating multiple member APIs. Such a property would greatly facilitate the service composition recommendation. For example, if we have selected a base API for a candidate Mashup of known functionalities, then it would be highly likely to discover other appropriate component APIs based on the above conservation principle of functional semantics. Therefore, the optimization goal attempts to minimize the difference of semantic representations between each Mashup and its entire member API consort. It can be performed by minimizing a margin-based ranking creation [28] over the Mashup-API composition network:

$$\mathcal{L}_3 = \sum_{i=[1,|\mathbf{V}_m|]}^{\{e_{i,j}\}_{j=[1,|\mathbf{C}_i|]}\subseteq\mathbf{E}_{m,a}} \left[\mathcal{M} + \Gamma\left(\bar{\mathbf{h}}_{\mathbf{C}_i}, \mathbf{h}_{\sqsubseteq_i}\right) - \Gamma\left(\bar{\mathbf{h}}_{\mathbf{C}_i}, \mathbf{h}_{\sqsubseteq_{\neg i}}\right)\right], \tag{15}$$

where $\mathcal{M} > 0$ is a margin hyper parameter, $\mathbf{C}_i$ is the set of member/linked APIs of Mashup v_i, and $v\neg_i$ is a sampled negative Mashup that is different from the positive Mashup v_i, i.e., there are no links between $v\neg_i$ and APIs in $\mathbf{C}_i$ observed from the Mashup-API composition network. $\bar{\mathbf{h}}_{\mathbf{C}_i}$ is the averaged pair-wise sum over semantic representations of all APIs in $\mathbf{C}_i$, which is computed by Equation (16):

$$\bar{\mathbf{h}}_{\mathbf{C}_i} = \frac{1}{|\mathbf{C}_i|}\sum_{j=1}^{|\mathbf{C}_i|}\mathbf{h}_{v_j}. \tag{16}$$

d is a function for measuring the closeness between two vector representations, i.e., the squared Euclidean distance as adopted in this article. The loss function in Equation (15) favors slower values of energy for all observed positive links than for the negative ones, which converges to give Mashups similar semantic representations with their composed APIs.

3.2.3 Joint Optimization. The joint learning process from the content and structure perspectives is summarized in Algorithm 2. The low-dimensional representation of each node is first randomly initialized with each vector value generated following a normal distribution (Lines 1–3). Then, all node representations are trained and optimized in mini-batches (Lines 4–13). To learn discriminative representations that capture the composition relations between Mashups and APIs, we randomly sample negative examples of Mashups (Lines 7–10) that have no links with APIs in C_i. Finally, the training process performed in Algorithm 2 minimizes the following collective objective:

$$\mathcal{L} = (1-\lambda)\mathcal{L}_1 + (1-\lambda)\mathcal{L}_2 + \lambda\mathcal{L}_3, \tag{17}$$

where λ is a hyper parameter set to balance the representation learning from the Web service content and structure aspects.

ALGORITHM 2: The Joint Representation Learning Process of Web Services

Input: the hierarchical Web service network **G** of topics, Mashups, and APIs
Output: the low-dimensional vector representations $\{\mathbf{h}_{v_i}\}_{i=1,\ldots,|\mathbf{V}|}$ of all nodes in network **G**
process begin:
1. **for each** vertex $v_i \in \mathbf{V}$ **do**
2. Initialize $\mathbf{h}_{v_i}$ by the truncated normal distribution, with standard deviation equals to $\frac{1}{\sqrt{n_d}}$
3. **end for**
4. **loop**
5. sample mini-batches $mt_{batch} = \{e_{ik}\}$, $at_{batch} = \{e_{jk}\}$, $ma_{batch} = \{e_{i,j}\}_{j=[1,|\mathrm{C}_i|]}$ of size b from the Mashup-Topic, API-Topic and Mashup-API networks, respectively
6. ${ma_{batch}}^* = \{\}$
7. **for each** positive Mashup $v_i \in ma_{batch}$ **do**
8. sample a negative Mashup $v_{\neg i}$
9. ${ma_{batch}}^* \leftarrow \{e_{\neg i,j}\}_{j=[1,|\mathrm{C}_i|]}$
10. **end for**
11. optimize w.r.t. Equation (14) over mt_{batch}
12. optimize w.r.t. Equation (15) over at_{batch}
13. optimize w.r.t. Equation (16) over ma_{batch} and ${ma_{batch}}^*$
14. **until** the iteration times end
15. **end procedure**

Optimization of Equation (17) is intractable, since learning has to emphasize both the content and structure information to search for an optimal solution in the unconstrained low-dimensional semantic space. To solve this problem, we adopt Stochastic Gradient Decent (SGD) algorithm as the optimization strategy using mini-batch training, which, respectively, minimizes the three components in Equation (17) as below.

First, we update the embedding vectors $\mathbf{h}_{v_i}$ and $\mathbf{h}_{v_k}$ involved in $(1-\lambda)\mathcal{L}_1$ by:

$$\mathbf{h}_{v_i} = \mathbf{h}_{v_i} + \eta\left\{(1-\lambda)w_{ik}\left[1-\sigma\left({\mathbf{h}_{v_i}}^T\mathbf{h}_{v_k}\right)\right]\cdot\mathbf{h}_{v_k}\right\}, \tag{18}$$

$$\mathbf{h}_{v_k} = \mathbf{h}_{v_k} + \eta\left\{(1-\lambda)w_{ik}\left[1-\sigma\left({\mathbf{h}_{v_i}}^T\mathbf{h}_{v_k}\right)\right]\cdot\mathbf{h}_{v_i}\right\}, \tag{19}$$

where σ is the sigmoid function and η is the learning rate of the SGD algorithm.

Then, in a similar way, we update the embedding vectors $\mathbf{h}_{v_j}$ and $\mathbf{h}_{v_k}$ involved in $(1-\lambda)\mathcal{L}_2$ by:

$$\mathbf{h}_{v_j} = \mathbf{h}_{v_j} + \eta\left\{(1-\lambda)w_{jk}\left[1-\sigma\left({\mathbf{h}_{v_j}}^T\mathbf{h}_{v_k}\right)\right]\cdot\mathbf{h}_{v_k}\right\}, \tag{20}$$

$$\mathbf{h}_{v_k} = \mathbf{h}_{v_k} + \eta\left\{(1-\lambda)w_{jk}\left[1-\sigma\left({\mathbf{h}_{v_j}}^T\mathbf{h}_{v_k}\right)\right]\cdot\mathbf{h}_{v_j}\right\}. \tag{21}$$

Finally, we update the embedding vectors $\mathbf{h}_{v_i}$ and all $\{\mathbf{h}_{v_j}\}_{v_j \in \mathbf{C}_i}$ involved in the last term $\lambda \mathcal{L}_3$ by:

$$\mathbf{h}_{v_i} = \mathbf{h}_{v_i} + \eta\{\lambda(\mathbf{h}_{v_i} - \bar{\mathbf{h}}_{\mathbf{C}_i})\}, \tag{22}$$

$$\mathbf{h}_{v_j} = |\mathbf{C}_i| \cdot \{\mathbf{h}_{v_j} + \eta\left[\lambda\left(\mathbf{h}_{v_i} - \mathbf{h}_{v_{\neg_i}}\right)\right]\}. \tag{23}$$

From above updating rules, we can observe that Mashups, APIs, and topics are being projected into the same latent space, where the representations of Mashups and APIs learned both from the content (by Equations (18)–(21)) and the structure (by Equations (22)–(23)). The two aspects influence each other based on the shared latent parameters.

3.3 Applications of Web service Representation Learning

The Web service representations learned by the proposed approach in this article can have multiple downstream applications in real-world scenarios. We discuss three promising application domains as follows:

Web Service Functionality Classification. Service functionality classification is the process of identifying functional categories of unlabeled Web services, which can classify new services or find the potential application domains for existing services. The task is normally performed by considering the content and structure similarities between labeled and unlabeled services. Since the embedding space in this article has preserved both service description and structure information, we can simply apply off-the-shelf machine learning algorithms such as Support Vector Machine [29] by taking representations of Web services as input features.

Web Service Functionality Clustering. Service functionality clustering is a popular task that aims group Mashup or API services into different clusters, where each cluster represents an application domain. The clustering results can improve service searching and management processes. Based on the learned service representations with well-encoded contextual and structural features with respect to closeness between services, one can use algorithms such as K-means to fulfill the task.

Web Service Functionality Visualization. Service functionality visualization projects Web services into a two-dimensional Euclidean space, which allows users to have a straightforward observation of the functionality distribution and the affinity between each pair of service. Since the learned service representations are continuous, we can easily map them into a two-dimensional visualization space using a dimensionality reduction/visualization algorithm such as t-SNE [1].

4 EXPERIMENTS

To evaluate the proposed models for Web service representation learning, we design two types of popular tasks: Web service recommendation [24, 30] and classification [24, 31]. We mainly investigate two questions: (1) Can the proposed PV-LDA model elicit more accurate functional semantics when compared with the state-of-the-art topic models?; (2) Can the proposed embedding approach learn robust representations that are helpful in other downstream applications?

4.1 Experimental Settings

4.1.1 Datasets. We evaluate the proposed models on two real-world datasets named Webservice and Wiki, respectively. The Wiki is considered a supplemental set that has richer text content information and presents analogous data structure as the Webservice set, i.e., each Web page document links to a collection of others, therefore the proposed topic-aware embedding approach still applies. The two datasets are described as follows:

Webservice is a Web service dataset recently crawled from ProgrammableWeb during August 2018. In total, we collected 19,718 Web services, where the numbers of API and Mashup services are 13,460 and 6,258, respectively. There are 26,246 composition links between Mashups and APIs

Table 2. Statistics Information of the Network Datasets

Items	Webservice	Wiki
# Nodes	19,718	2,405
# Edges	26,246	17,981
# Labels	478	17
# Vocabulary	25,028	4,973
# Words per node	36	647
# Tags per node	4	–

in the bipartite Mashup-API network. Each service node is associated with a functional description together with some functional tags. The average number of words and tags for every node are 36 and 4, respectively. Each service corresponds to a major functional category out of the total 487.

Wiki is a Web page linking network that contains 2,405 Web pages from 17 categories with each Web page classified as a relevant category. Each page node links to a collection of others and there are 17,981 hyperlinks in total between these Web pages. Every page node is described by a text with 647 average words.

The statistic information regarding these two datasets is shown in Table 2. It is necessary to mention that when evaluating the proposed method on the Wiki dataset, we first construct the Web page-Topic distribution network from Wiki contents based on the basic LDA model [21]. Then, coupled with the Web page linking network, the topic distribution network is used for representation learning based on the process described in Section 3.2, i.e., each Web page node is semantically close to the collective semantic of all linked Web pages.

4.1.2 Evaluation Protocols. First, we perform Web service recommendation (or link prediction) [32] based on the similarities between Mashups and Services in learned representations. We randomly remove 20% of links from the training set and take them as a test set to evaluate the service recommendation performance. We choose Recall, Precision, and F-measure as evaluation metrics defined as follows:

$$Recall@N_R = \frac{|CR(m) \cap Rec(m)|}{|CR(m)|}, \tag{24}$$

$$Precision@N_R = \frac{|CR(m) \cap Rec(m)|}{|Rec(m)|}, \tag{25}$$

$$F - measure@N_R = \frac{2 \times Recall \times Precision}{Recall + Precision}, \tag{26}$$

where CR(m) denotes all member APIs of Mashup m, and Rec(m) represents the recommended API services. |CR(m)| and |Rec(m)| are the number of real member APIs and the recommended number of APIs, respectively. For each Mashup service m, we recommend N_R API services, which ranges from 1 to 6.

We then benchmark node classification performance based on both the Webservice and Wiki datasets. Using similar settings to that of previous work [27, 19], we build a classifier based on the linear kernel Support Vector Machine (SVM) [29] on the training set (e.g., all involved nodes with categories known for training), where the low-dimensional node representations learned by the proposed embedding approach are taken as input features for SVM. Then, based on the trained SVM classifier, we predict a category for each node in the test set and compare with their respective

ground-truth categories to calculate the accuracy. The classification performance is evaluated by the following two metrics:

$$Micro - F1 = \frac{\sum_{i=1}^{\Lambda} 2TP^i}{\sum_{i=1}^{\Lambda} (2TP^i + FP^i + FN^i)}, \tag{27}$$

$$Macro - F1 = \frac{1}{\Lambda} \sum_{i=1}^{\Lambda} \frac{2TP^i}{(2TP^i + FP^i + FN^i)}, \tag{28}$$

where Λ indicates the total number of categories involved in the node classification task, TP^i, FN^i and FP^i denote the number of true positive, false negatives and false positives w.r.t. the ith resulting category, respectively.

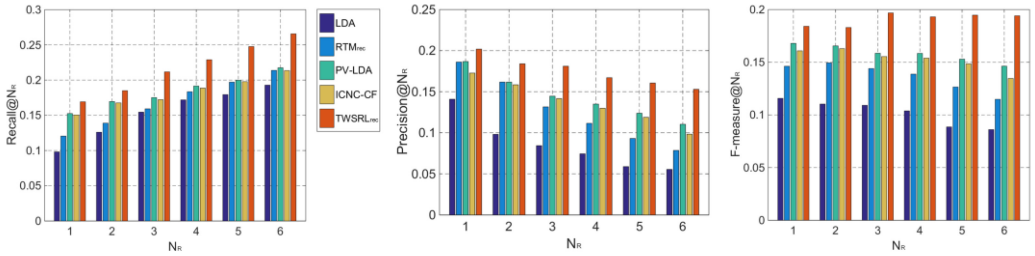
4.1.3 Baselines.

Web service recommendation:

- **LDA**: It recommends API services whose descriptions are similar (e.g., cosine method) to that of the target Mashup based on the topic distribution vectors learned by the standard LDA model.
- **RTM$_{rec}$** [33]: It recommends Web APIs whose descriptions are similar to that of the target Mashup in semantics derived by the Relational Topic Model (RTM). RTM extends the basic LDA model by additionally considering the composition relationships between APIs and Mashups during the training process, where Mashups and APIs with links are more likely to follow similar topic distributions.
- **PV-LDA**: The proposed Paragraph Vector augmented LDA model in this article, which simultaneously learns from the functional descriptions and tags in a semi-supervised manner.
- **ICNC-CF** [24]: It first clusters Mashups based on the latent topics learned by a two-level topic model considering the relationships between Mashups. Then, with the aid of historical invocation history between Mashup clusters and API services, it explores using item-based collaborative filtering to rank and recommend diverse API services.
- **TWSRL$_{rec}$**: It recommends API services whose representations are similar (e.g., cosine method) to that of the target Mashup based on the proposed topic-aware embedding approach proposed in this article. In addition, the translation-based mechanism (introduced in Section 3.2.2) is adopted while ranking the candidate APIs for recommendation.

Multi-category Node classification:

- **DeepWalk** [34]: This method preserves only the network structure information by the truncated random walk over the whole network. It then learns the node representations by using the Skip-Gram model.
- **Node2vec** [35]: Compared with DeepWalk, this method adopts a more flexible random walk process by simultaneously capturing the local and global structure of the network; however, both Node2vec and DeepWalk preserve only the network structure information.
- **LINE** [27]: It is a structure-preserving embedding method that encodes both the local and global network structures. This method adopts an edge-sampling algorithm to address the limitation of classical stochastic gradient descent while improving both the effectiveness and the efficiency of inference.
- **PLANE** [18]: It is an extended version of RTM that specifically trains a latent low-dimensional vector for representing each document node. The node structure, content, and latent topics are preserved in the final network representations.



(a) Recall (b) Precision (c) F-measure

Fig. 6. Web service recommendation performance.

- **TriDNR** [19]: A state-of-the-art method that exploits network structure, node content, and label information for node representation learning. The content and structure learning enhance each other through shared latent parameters.
- **TWSRL**: It is the proposed approach for topic-aware Web service representation learning in this article, which learns semantics from both the network contents and structures.

4.1.4 Parameter Settings. There are many parameters that should be defined beforehand. In this article, some hyper parameters such as α are set either by previous experiences [6, 20] or several trials. To train the proposed PV-LDA model, hyper parameters α, β_w, β_a, and τ are set as 0.1, 0.1, 0.05, and 1.0, respectively. Other parameters such as λ and T are selected based on sensitivity testing results. In this article, r percent of network nodes with labels known are used for training the SVM classifier. The rest are split into two parts, where 20% are used for validation to select the parameters and 80% are used to evaluate the trained model. For comparison, the rest parameters are set as follows: The number of topic (T) is set as 20 for PV-LDA, and the number (K) of relevant topics chosen for each vertex is 4. The dimension (d_n) for the learned representations is set as 100. The balance parameter λ is set as 0.2. The ranking margin $\mathcal{M}$ is set as 1.0, and the learning rate adopted in the SGD algorithm is set as 0.05. In the node classification task, r is set as 70%. For each r value, the experiment repeats 10 times, and the average results as well as their standard deviations are reported.

4.2 Experimental Results

4.2.1 Web Service Recommendation. Figure 6 shows the service recommendation results achieved by five topic model-based methods. The recommendation process for each candidate Mashup repeats five times, and the average results are reported. From the Recall, Precision, and F-measure results, we have the following three significant observations:

(1) When increasing the number of Web services, Recall performance increases, while the opposite is observed in Precision. This is due to the fact that more services recommended increases the hit rate, but meanwhile introduces more noise. F-measure is a trade-off between the Recall and Precision results. We can see from Figure 6(c) that LDA performs the worst, which demonstrates the fact that Web service content lacks quality features [24, 33]. As we know, the basic LDA model needs a substantial enough training corpus to perform well [40], which is difficult to satisfy in experiments. In comparison, all other baselines achieve better results, which are based on the topic models (e.g., RTM), assisted by auxiliary information during the unsupervised training process. This demonstrates that it is necessary to incorporate some valuable side information such as service structures to help derive more accurate functional semantics.

Table 3. The distribution of Web services in top 20 categories

Category	Number	Category	Number
Tools	*841*	*Video*	*412*
eCommerce	*740*	*Telephony*	*395*
Social	*739*	*Reference*	*392*
Financial	*632*	*Payments*	*385*
Search	*570*	*Government*	*385*
Messaging	*528*	*Science*	*304*
Enterprise	*511*	*Sports*	*265*
Photos	*459*	*Advertising*	*265*
Music	*440*	*Education*	*264*
Travel	*419*	*Email*	*262*

(2) Although RTM outperforms LDA, it is not competitive with the proposed model PV-LDA. RTM considers the linking relations between Mashups and APIs to influence the topic sampling process, enforcing linked services into similar topic distributions. However, when compared with PV-LDA, the inferiority of RTM is most likely caused by the content not fully being learned due to the sparsity of features. PV-LDA addresses this problem by including tags as valuable prior functional semantic information to enhance the training data as well as refine word topic assignment, which is more effective than simply expanding from the structure level as in RTM. Moreover, we can observe from Figure 6 that PV-LDA is superior to the two-level topic model ICNC-CF. The primary reason is that the descriptions are normally sparse and littered with noisy or irrelevant word features that could confuse the regular topic sampling process adopted in ICNC-CF. In comparison, tags associated with services are able to reveal the functionalities described in the descriptions, therefore it is helpful to leverage the tag information to highlight the relevant functional features in descriptions for improved topic assignment. The observation again demonstrates the effectiveness of the proposed mechanism of word embedding augmented LDA model for functional topics extraction of Web services.

(3) From observations of all metrics, $\text{TSWRL}_{\text{rec}}$ performs significantly better compared with other topic model baselines, with the average F-measure increasing 25.06% over the ICNC-CF, 39.76% over RTM, and 86.73% over the LDA. The reason is that the proposed embedding approach has seamlessly combined functional content and network structures to learn rich semantic feature representations of services, where the learned vector representation is capable of reflecting the affinities between Mashups and APIs. In addition, we adopted the translation-based mechanism to model the relationships between Mashups and their member APIs, such that the conservation of functional semantics would help rank and recommend appropriate services.

4.2.2 Multi-category Node Classification. Classifying Web services is important to improve the efficiency of distributed service computing [31, 36], such as convenient service management and fast service retrieval. Typically, the feature representation of services is of paramount importance for classification performance. We compare the proposed approach with several state-of-the-art baselines of learning network representations. The experimental results on Webservice and Wiki datasets are, respectively, summarized in Tables 4 and 5, where 20 categories of Web services shown in Table 3 are examined in the experiment. The first and second best performances

Table 4. Node Classification Results on Webservice dataset

Metrics	%r	10	30	50	70
Macro-F1	DeepWalk	0.078±0.005	0.110±0.004	0.113±0.006	0.116±0.006
	LINE	0.201±0.012	0.256±0.014	0.273±0.010	0.282±0.011
	Node2vec	**0.233±0.012**	**0.296±0.011***	**0.324±0.011***	***0.335±0.013***
	PLANE	0.146±0.011	0.173±0.011	0.178±0.012	0.192±0.017
	TriDNR	0.218±0.019	0.266±0.012	0.284±0.009	0.297±0.011
	TWSRL	***0.232±0.013***	***0.286±0.009***	***0.312±0.008***	**0.336±0.007**
Micro-F1	DeepWalk	0.148±0.007	0.185±0.007	0.210±0.009	0.226±0.009
	LINE	0.293±0.010	0.348±0.008	0.374±0.011	0.390±0.013
	Node2vec	0.325±0.008	0.393±0.007	0.428±0.008	0.448±0.011
	PLANE	0.203±0.007	0.238±0.008	0.254±0.008	0.263±0.011
	TriDNR	***0.391±0.009***	***0.431±0.006***	***0.443±0.006***	***0.449±0.008***
	TWSRL	**0.424±0.008***	**0.460±0.007***	**0.466±0.006***	**0.475±0.005***

Table 5. Node Classification Results on Wiki dataset

Metrics	%p	10	30	50	70
Macro-F1	DeepWalk	0.217±0.014	0.273±0.009	0.296±0.010	0.303±0.017
	LINE	0.345±0.013	0.432±0.013	0.458±0.014	***0.481±0.019***
	Node2vec	0.382±0.011	***0.454±0.014***	**0.493±0.013***	0.513±0.012
	PLANE	0.162±0.011	0.216±0.009	0.234±0.012	0.192±0.017
	TriDNR	***0.385±0.016***	0.441±0.013	0.452±0.012	0.479±0.016
	TWSRL	**0.415±0.013***	**0.463±0.013**	***0.472±0.013***	**0.486±0.016**
Micro-F1	DeepWalk	0.323±0.014	0.398±0.010	0.426±0.011	0.442±0.013
	LINE	0.445±0.012	0.538±0.007	0.567±0.013	0.581±0.012
	Node2vec	0.451±0.012	0.569±0.009	0.609±0.006	***0.629±0.008***
	PLANE	0.244±0.008	0.311±0.009	0.340±0.010	0.365±0.012
	TriDNR	***0.564±0.011***	***0.606±0.010***	***0.616±0.012***	0.628±0.012
	TWSRL	**0.578±0.012***	**0.631±0.010***	**0.643±0.011***	**0.656±0.013***

have been highlighted with bold-faced and italic bold-faced fonts, respectively. In addition, we perform a student t-test with $p < 0.05$ on the comparative results; for each training ratio, the best result is marked with * if it significantly outperforms others. We draw the following three major observations from the results:

(1) Among all baselines, DeepWalk, LINE, and Node2vec are structure-preserving methods, while others aim to preserve both network content and structure information. We can observe that in most cases incorporating the content information could generate better embedding performance on both datasets, i.e., TriDNR and TWSRL are significantly better than DeepWalk and LINE with different training set ratios. A convincing explanation is that network contents are reliable information to characterize affinities between vertices in a similar way aligned with the network structure. For example, services connecting each other tend to share identical or similar functional properties revealed by their respective functional descriptions and tags. In addition, we argue that there are many missing composition links between Mashups and services, since an appropriate API is not used because of many other functionally equivalent alternatives. For instance, "Google

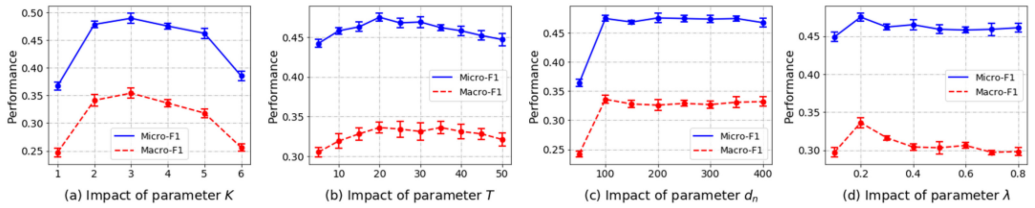


Fig. 7. Parameter influence on Webservice dataset.

Maps" is functionally equivalent to "Bing Maps" in the Webservice dataset, where "Google Maps" was used 2,576 times compared with the latter being used only 32 times. We believe that considering service contents other than composition structures would repair some missing and potential links throughout the sparsely connected network.

(2) The proposed approach TWSRL performs better than the state-of-the-art model TriDNR, i.e., the average Macro-F1 and Micro-F1 for service representation learning improved by 2.5% and 2.8%, respectively. TriDNR effectively models the content and structure in a unified framework, where the learned low-dimensional vectors can well preserve information in a reciprocally enhanced manner. However, in TriDNR the service content is considered as only a set of plain word attributes to reveal a simplex semantic supporting a corresponding structure link. This paradigm is not appropriate for Web service contents or long texts that usually exhibit rich and complex semantics. For example, a Web service may demonstrate multiple functionalities and interact with others for different aspects. A Web page in Wikipedia may describe many aspects of topics and cite many others of different subject matters. In this paper, we propose abstracting network content as a set of topics revealing different aspects of functional semantics. Through learning from the Mashup-Topic and API-Topic distribution networks, we are able to gather more subtle and discriminative vector representations than TriDNR. The comparison results presented in Tables 4 and 5 demonstrate the effectiveness of our topic-aware network embedding approach.

(3) From the results, PLANE performs the worst among all methods even though both network content and structure information have been modeled. The reason is most likely due to the fact that sparse content information has impeded accurate semantic extraction, since PLANE is an LDA-based model that requires large data to perform well, and there is no additional semi-supervised information used in PLANE to mitigate this problem. It is interesting to observe that Node2vec performs slightly better than the proposed TWSRL model w.r.t. Macro-F1 performance on Webservice dataset, while TWSRL significantly outperforms Node2vec on Wiki dataset. There are two possible reasons behind this phenomenon. First, Node2vec adopts a more flexible way for capturing the network structure relationships [12] that is helpful specifically when the connectivity density (e.g., 1.33 for Webservice set compared with 7.48 for Wiki set) is sparse. Second, Wiki set contains richer text content than Webservice, with the average numbers of word features per node 647 and 36, respectively. The richer text information can help learn more accurate node relations that could subsequently boost the structure-based node relation modeling.

4.2.3 Parameter Sensitivity. We designed a series of experiments on both datasets to test the sensitivity of parameters with regards to K, T, d_n, and λ. Figure 7(a) and Figure 8(a) show the impacts of parameter K on the representation learning performance by running the multi-category classification task. The larger value of K means that each vertex exhibits more semantic topics (i.e., assuming the vertex content contains more fragmented semantic information), while the smaller

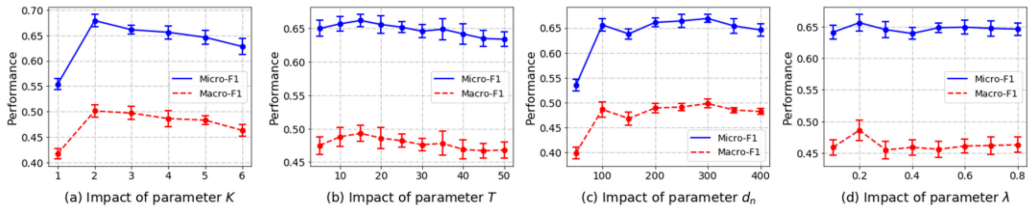


Fig. 8. Parameter influence on Wiki dataset.

value of K means content is centralized to reveal simplex semantics. The results show an upward tendency followed by a gradual decline after peaking when K equals to 3 and 2 on Webservice and Wiki datasets, respectively. Similarly, parameter T is also a parameter to indicate the semantic compactness of network content. We can see from Figure 7(b) and Figure 8(b) that the performance change with different values of T, where the best settings are 20 and 15, respectively. The influences of embedding sizes are shown in Figure 7(c) and Figure 8(c); they increase sharply from 50 to 100 and then fluctuate slightly afterwards. The best embedding sizes for Webservice and Wiki are 100 and 300, respectively. Last, λ. is set to balance the learning between content aspect and structure aspect. Larger value will give bias to emphasize the network structure (e.g., the optimization will focus on minimizing the structure part in Equation (17)) while preserving both the content and structure information. From Figure 7(d) and Figure 8(d), we can see that 0.2 is the best setting for both two datasets.

5 RELATED WORK

A large spectrum of effort has been devoted to the area of learning representations from social networks [14], citation networks [18], biological networks [37], and so on. We first summarize the general network embedding techniques from structure-based and content-based perspectives. Then, we survey existing works related to Web service representation learning.

5.1 Structure-preserving Network Embedding

Early work mainly focuses on structure-preserving network embedding, which seeks to find an optimal scheme for precisely characterizing affinities between vertices based on the structural properties of a given network, such as neighborhood relations [34], high-order relations [38], and communities [39].

DeepWalk [18] is a pioneering work intentionally encoding neighborhood relationships. Affinities between vertices are determined by the truncated random walk over the whole network, where vertices within a walk are forced to have similar semantic representations based on the Skip-Gram model [40]. However, DeepWalk is not expressive enough to capture the diversity of connectivity patterns in a network. Node2vec [38] thus adopts a more flexible way of capturing neighborhood relations by an additional second-order random walk strategy, which can smoothly interpolate between the breadth-first and depth-first samplings. Similarly, Tang et al. proposed a model LINE to preserve simultaneously the first- and second-order proximities simultaneously [27]. The first-order proximity is defined by the observed edges between vertices, while the second-order proximity is derived from the transitivity of neighbor relations (e.g., neighborhoods' neighbors). In addition to the microscopic structure information preserved, some methods further extend to explore the macroscopic properties such as communities and groups [39, 41]. For example, Wang et al. [39] proposed a Modularized Nonnegative Matrix Factorization model for network embedding, which enables the preservation of both local node proximities and global community structures.

The above methods are mainly based on shallow models (e.g, Skip-Gram) in which the learning ability might be limited. Deep neural network models have therefore been proposed to break this constraint. For example, Wang et al. [42] proposed a deep autoencoder model with multiple non-linear layers to preserve the neighbor structure of nodes. Cao et al. [43] proposed a deep network embedding method to capture the weighted graph structure and represent nodes of non-linear structures. However, preserving only structural information may be insufficient for expressing the rich affinities between vertices in a network, i.e., a qualified API is not composed (it will cause a missing link in the Web service network) by a Mashup because there are many functionally equivalent APIs. To address this issue, content information (e.g., tags and textual descriptions) has been widely utilized to enhance the structure-based methods, especially when the network connectivity is sparse.

5.2 Content-preserving Network Embedding

In addition to the structure properties that explicitly define the closeness of vertices, content information associated with specific networks naturally reveal the similarities between nodes, i.e., services having some identical tags should share some common functional semantics [44]. A vast number of content-preserving approaches have been proposed so far [17, 18].

Tu et al. [45] proposed a label information augmented network embedding algorithm based on matrix factorization. It adopts support vector machines (SVM) and incorporates label information to learn representations that generate the optimal classification boundary. Yang et al. [13] proved that matrix factorization is equivalent to the DeepWalk learning process and proposed a text-associated DeepWalk model. Their work incorporates text features of vertices into network representation learning under the framework of matrix factorization. However, computational cost of the matrix factorization–based methods is subject to the network scale. Sun et al. [46] treats content as a special kind of node and integrates text modeling and structure modeling in a unified framework. Pan et al. also proposed a unified learning framework, TriDNR, to embed both the structure and content information [19]. TriDNR forces it to learn representations simultaneously from linking relations, labels, and textual content, where learning from the structural and content aspects influence each other under shared parameters.

Nearly all existing methods simply model content information as a set of word features to help learn a simple semantic representation that tries to explain corresponding network structures. However, such an assumption is not reasonable when content features of vertices exhibit multiple types of semantics, i.e., a Web service may have multiple tags with each indicating a different aspect of functional property. To address this problem, in this article, we propose to abstract content (e.g., tags and descriptions) of every Web service as a collection of functional topics, where each topic encodes a part of semantics of that service content.

5.3 Web Service Representation Learning

Web service representation has not been specifically studied. Prior work generally utilizes the feature extraction step to support the completion of advanced tasks such as Web service clustering [16, 36] and recommendation [30, 24, 47].

Platzer et al. [48] represented Web services based on the Vector Space Model (VSM), with each word feature constituting a relevant dimension. However, one obvious drawback of VSM is that the vector dimensionality is subject to the vocabulary size, which may suffer from the curse of dimensionality and feature sparsity. Ma et al. [49] used the Singular Value Decomposition (SVD) technique to map services into a latent semantic space, which can significantly reduce the vector dimensionality and meanwhile encode the interactions between services efficiently. However, SVD usually involves undesirable computation complexity especially when the input feature

matrix is too large. To address this problem, many researchers have appealed to probabilistic models that derive latent topic distributions of Web services in an unsupervised manner. For example, the Probabilistic Latent Semantic Analysis (PLSA) [31] and Latent Dirichlet Allocation (LDA) [16] have been used to elicit semantics from service descriptions and then represent each service with a low-rank topic distribution vector. However, the standard PLSA and LDA models rely on a substantial enough number of observed examples to perform well, which may not be feasible to Web services in which content information is often sparse and noisy [24, 20]. In the past, a large amount of effort has been devoted to mitigating these problems. Cao et al. [50] proposed to reduce the influence of irrelevant and noisy words in service descriptions through a statistic-based feature reduction process (e.g., based on the notion of TF-IDF). Analogously, to reduce the impact of sparse word features in service descriptions, Shi et al. [20] proposed a word-augmented LDA training process that uses words within the same cluster to help choose and refine the optimal topic assignment for each other. In addition to adaptation from the text content perspective, some works also seek to explore the rich service structure relations. For example, Cao et al. [24] and Li et al. [33] proposed to consider the composition links between services for improved service semantic extraction and representation. Although these methods could achieve improved results in representation learning, they consistently suffer from at least one of the following three limitations: First, existing methods are often application-oriented, which means they target a specific application such as Mashup or API service clustering, which may result in suboptimal generalization ability for learned representations when applied to other tasks such as service functionality classification. Second, many existing works either only consider text information [50] or simply model service structures as plain information where linked services are forced to have similar semantics. Such a paradigm fails to fully capture the semantic interactions between services reflected by the network structure, i.e., a Mashup service is often reflected by its composed API services as a whole. Finally, the service representations learned based on topic models are discrete, which means different dimensions in the topic distribution vectors of services are not continuous and therefore cannot measure the closeness between service nodes in Euclidean space.

In comparison, we specifically focus on Web service representation learning in this article, which simultaneously models service content and structure information in a unified optimization framework. The learned vector representations are continuous to measure the closeness or distance in a Euclidean space, where services with similar functional features and topology structures are mapped together. In addition, we propose taking tags as prior information to highlight the important word features for topic sampling, which is also in stark contrast with previous methods.

6 CONCLUSIONS

We studied the problem of Web service representation learning by exploring a tri-layer bipartite service network. The contributions of this article are threefold: (1) To effectively preserve the content/text information, we proposed a generative model PV-LDA to incorporate words and tags for semi-supervised topics training; (2) We proposed a unified optimization framework that first reconstructs the service representations from the Mashup-Topic and API-Topic distribution networks. The learned service representations from topic distribution networks are then enhanced by the learning from Mashup-API composition network based on a translation principle; (3) We presented how to optimize the proposed embedding approach in a unified way. The extensive experiments on two real-world datasets demonstrated the learned low-rank vector representations are effective and robust in service recommendation and functionality classification.

Since the proposed approach only considers the immediate composition relationships between Mashup and API services, future work can incorporate high-order structures. For instance, two Mashup service nodes can reach each other through their shared API service nodes in the network.

These second-order structure relations between Mashup services can be leveraged to enhance the affinities between the respective representations.

REFERENCES

[1] D. Zhang, J. Yin, X. Zhu, and C. Zhang. 2018. Network representation learning: A survey. *IEEE Trans. Big Data.* 6, 1 (2018), 3–38. DOI: 10.1109/TBDATA.2018.2850013

[2] B. Cheng, S. Zhao, J. Qian, Z. Zhai, and J. Chen. 2018. Lightweight service mashup middleware with REST style architecture for IoT applications. *IEEE Trans. Netw. Serv. Manag.* 15, 3 (2018), 1063–1075.

[3] B. Cheng, Z. Zhai, S. Zhao, and J. Chen. 2017. LSMP: A lightweight service mashup platform for ordinary users. *IEEE Commun. Mag.* 55, 4 (2017), 116–123.

[4] D. Bianchini, V. D. Antonellis, and M. Melchiori. 2017. WISeR: a multi-dimensional framework for searching and ranking web APIs. *ACM Trans.* Web 11, 3 (2017), 19.

[5] Y. Ma, Z. Ren, Z. Jiang, J. Tang, and D. Yin. 2018. Multi-dimensional network embedding with hierarchical structure. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM'18).* 387–395.

[6] M. Shi, J. Liu, D. Zhou, and Y. Tang. 2018. A topic-sensitive method for mashup tag recommendation utilizing multi-relational service data. *IEEE Trans. Serv. Comput.* In press. DOI: 10.1109/TSC.2018.2805826

[7] M. Gao, L. Chen, X. He, and A. Zhou, 2018. Bine: Bipartite network embedding. In *Proceedings of the International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR'18).* 715–724.

[8] F. Xie, L. Chen, Y. Ye, Y. Liu, Z. Zheng, and X. Lin. 2018. A weighted meta-graph based approach for mobile application recommendation on heterogeneous information networks. In *Proceedings of the International Conference on Service-oriented Computing (ICSOC'18).* 404–420.

[9] W. Chen, I. Paik, and P.C. Hung. 2015. Constructing a global social service network for better quality of web service discovery. *IEEE Trans. Serv. Comput.* 8, 2 (2015). 284–298.

[10] P. Minervini, V. Tresp, C. D'amato, and N. Fanizzi. 2018. Adaptive knowledge propagation in web ontologies. *ACM Trans.* Web 12, 1 (2018), 2.

[11] Q. Gong, Y. Chen, J. Hu, Q. Cao, P. Hui, and X. Wang. 2018. Understanding cross-site linking in online social networks. *ACM Trans.* Web 12, 4 (2018), 25.

[12] P. Cui, X. Wang, J. Pei, and W. Zhu. 2018. A survey on network embedding. *IEEE Trans. Knowl. Data Eng.* 31, 5 (2019), 833–852. DOI: 10.1109/TKDE.2018.2849727

[13] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang. 2015. Network representation learning with rich text information. In *Proceedings of the International Joint Conference on Artificial Intelligence.* 2111–2117.

[14] L. Liao, X. He, H. Zhang, T. S. Chua. 2018. Attributed social network embedding. *IEEE Trans. Knowl. Data Eng.* 30, 12 (2018), 2257–2270. DOI: 10.1109/TKDE.2018.2819980

[15] J. Li, C. Chen, H. Tong, and H. Liu. 2018. Multi-layered network embedding. In *Proceedings of the International Conference on Society for Industrial and Applied Mathematics.* 684–692.

[16] M. Shi, J. Liu, B. Cao, Y. Wen, and X. Zhang. 2018. A prior knowledge based approach to improving accuracy of web services clustering. In *Proceedings of the IEEE International Conference on Services Computing (SCC).* 1–8.

[17] Z. Wang, J. Zhang, J. Feng, and Z. Chen. 2014. Knowledge graph and text jointly embedding. In *Proceedings of the International Conference on Empirical Methods in Natural Language Processing (EMNLP'14).* 1591–1601.

[18] X. Li, W. Chen, Y. Chen, X. Zhang, J. Gu, and M. Q. Zhang. 2017. Network embedding-based representation learning for single cell RNA-seq data. *Nucl. Acids Res.* 45, 19 (2017), e166–e166.

[19] S. Pan, J. Wu, X. Zhu, C. Zhang, and Y. Wang. 2016. Tri-party deep network representation. In *Proceedings of the International Joint Conference on Artificial Intelligence.* 11, 9 (2016), 1895–1901.

[20] M. Shi, J. Liu, D. Zhou, M. Tang, and B. Cao. 2017. WE-LDA: A word embeddings augmented LDA model for web services clustering. In *Proceedings of the International Conference on Web Services.* 9–16.

[21] Y. Teh, D. Newman, and M. Welling. 2007. A collapsed variational Bayesian inference algorithm for latent Dirichlet allocation. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems (NIPS'07).* 1353–1360.

[22] X. Cheng, X. Yan, Y. Lan, and J. Guo. 2014. Btm: Topic modeling over short texts. *IEEE Trans. Knowl. Data Eng.* 26, 12 (2014), 2928–2941.

[23] C. Li, H. Wang, Z. Zhang, A. Sun, and Z. Ma. 2016. Topic modeling for short texts with auxiliary word embeddings. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval* 165–174.

[24] B. Cao, J. Liu, M. M. Rahman, B. Li, J. Liu, and M. Tang. 2017. Integrated content and network-based service clustering and Web APIs recommendation for mashup development. *IEEE Trans. Serv. Comput.* 13, 1 (2017), 1–14.

[25] Q. Le and T. Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the International Conference on Machine Learning.* 1188–1196.

[26] D. M. Blei, A. Y. Ng, and M. I. Jordan. 2003. Latent Dirichlet allocation. *J. Mach. Learn. Res.* 3 (Jan. 2003), 993–1022.

[27] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the International Conference on World Wide Web*. 1067–1077.
[28] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems*. 2787–2795.
[29] A. Abraham, F. Pedregosa, M. Eickenberg, P. Gervais, A. Mueller, J. Kossaifi, A. Gramfort, B. Thirion, and G. Varoquaux. 2014. Machine learning for neuroimaging with scikit-learn. *Front. Neuroinf.* 8, 2 (2014), 1–15.
[30] L. Yao, Q. Sheng, A. H. Ngu, J. Yu, and A. Segev. 2015. Unified collaborative and content-based web service recommendation. *IEEE Trans. Serv. Comput.* 8, 3 (2015), 453–466.
[31] G. Cassar, P. Barnaghi, and K. Moessner. 2010. Probabilistic methods for service clustering. In *Proceedings of the International Workshop on Semantic Web Service Matchmaking and Resource (ISWC'10)*. 4–20.
[32] L. Liu, F. Lecue, and N. Mehandjiev. 2013. Semantic content-based recommendation of software services using context. *ACM Trans.* Web 7, 3 (2013), 17.
[33] C. Li, R. Zhang, J. Huai, and H. Sun. 2014. A novel approach for API recommendation in mashup development. In *Proceedings of the IEEE International Conference on Web Services (ICWS'14)*. 289–296.
[34] B. Perozz, R. Al-Rfou, and S. Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD'14)*. 701–710.
[35] A. Grover and J. Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD'16)*. 855–864.
[36] M. Aznag, M. Quafafou, E. M. Rochd, and Z. Jarir. 2013. Probabilistic topic models for web services clustering and discovery. In *Proceedings of the European Conference on Service-oriented and Cloud Computing (ICSOC'13)*. 19–33.
[37] T. Le and H. Lauw. 2014. Probabilistic latent document network embedding. In *Proceedings of the IEEE International Conference on Data Mining*. 270–279.
[38] C. Yang, M. Sun, and Z. Liu. 2017. Fast network embedding enhancement via high order proximity approximation. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'17)*. 19–25.
[39] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang. 2017. Community preserving network embedding. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 203–209.
[40] T. Mikolov, K. Chen, G. Corrado, J. Dean. 2013. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.
[41] J. Chen, Q. Zhang, and X. Huang. 2016. Incorporate group information to enhance network embedding. In *Proceedings of the ACM International Conference on Information and Knowledge Management*. 1901–1904.
[42] D. Wang, P. Cui, and W. Zhu. 2016. Structural deep network embedding. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1225–1234.
[43] S. Cao, W. Lu, and Q. Xu. 2016. Deep neural networks for learning graph representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 1145–1152.
[44] L. Chen, J. Wu, Z. Zheng, M. R. Lyu, and Z. Wu. 2014. Modeling and exploiting tag relevance for web service mining. *Knowl. Inf. Syst.* 39, 1 (2014), 153–173.
[45] C. Tu, W. Zhang, Z. Liu, and M. Sun. 2016. Max-Margin DeepWalk: Discriminative learning of network representation. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'16)*. 3889–3895.
[46] X. Sun, J. Guo, X. Ding, and T. Liu. 2016. A general framework for content-enhanced network representation learning. arXiv preprint arXiv:1610.02906.
[47] B. Cheng, C. Li, S. Zhao, and J. Chen. 2018. Semantics mining & indexing-based rapid web services discovery framework. *IEEE Trans. Serv. Comput.* In press. DOI: 10.1109/TSC.2018.2831678
[48] C. Platzer and S. Dustdar. 2005. A vector space search engine for web services. In *Proceedings of the IEEE European Conference on Web Services (ECOWS)*. 62–71.
[49] J. Ma, Y. Zhang, and J. He. 2008. Web services discovery based on latent semantic approach. In *Proceedings of the IEEE International Conference on Web Services (ICWS)*. 740–747.
[50] B. Cao, X. F. Liu, J. Liu, and M. Tang. 2017. Domain-aware mashup service clustering based on LDA topic model from multiple data sources. *Inf. Softw. Technol.* 90 (2017), 40–54.

Received February 2007; revised March 2009; accepted June 2009