Web Service Recommendation With Reconstructed Profile From Mashup Descriptions

Yang Zhong, Yushun Fan, Wei Tan, *Senior Member, IEEE*, and Jia Zhang, *Senior Member, IEEE*

***Abstract*— Web services are self-contained software components that support business process automation over the Internet, and mashup is a popular technique that creates value-added service compositions to fulfill complicated business requirements. For mashup developers, looking for desired component services from a sea of service candidates is often challenging. Therefore, web service recommendation has become a highly demanding technique. Traditional approaches, however, mostly rely on static and potentially subjectively described texts offered by service providers. In this paper, we propose a novel way of dynamically reconstructing objective service profiles based on mashup descriptions, which carry historical information of how services are used in mashups. Our key idea is to leverage mashup descriptions and structures to discover important word features of services and bridge the vocabulary gap between mashup developers and service providers. Specifically, we jointly model mashup descriptions and component service using author topic model in order to reconstruct service profiles. Exploiting word features derived from the reconstructed service profiles, a new service recommendation algorithm is developed. Experiments over a real-world data set from ProgrammableWeb.com demonstrate that our proposed service recommendation algorithm is effective and outperforms the state-of-the-art methods.**

***Note to Practitioners*—Service recommendation accuracy for mashup creation is often limited due to poor quality of service descriptions. Mashup descriptions contain valuable information about functions and features of its component services, which can be leveraged to enhance descriptive quality of original service profiles. Based on the assumption, this paper proposes a novel two-phase service recommendation framework to facilitate mashup creation. Specifically, our approach reconstructs service profiles by extracting appropriate words from historical mashup descriptions. Then, a novel service recommendation algorithm is developed by exploiting popularity and relevance measures hidden in the reconstructed profiles. Moreover, we propose the rules of dominant words discovery and employ it to further refine our algorithm.**

Manuscript received March 3, 2016; revised July 17, 2016 and October 20, 2016; accepted October 29, 2016. This paper was recommended for publication by Associate Editor J. Civera and Editor M. P. Fanti upon evaluation of the reviewers' comments. This work was supported in part by the National Natural Science Foundation of China under Grant 61673230, in part by the National Science and Technology Support Program of China under Grant 2012BAF15G01, and in part by the National High Tech Research and Development Program under Grant 2012AA02A613. *(Corresponding author: Yushun Fan.)*

Y. Zhong and Y. Fan are with the Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: zhongy12@mails.tsinghua.edu.cn; fanyus@mail.tsinghua.edu.cn).

W. Tan is with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (e-mail: wtan@us.ibm.com).

J. Zhang is with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: jia.zhang@sv.cmu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASE.2016.2624310

***Index Terms*— Author topic model (ATM), mashup creation, mashup descriptions, service recommendation.**

I. Introduction

SERVICES computing enables a computing paradigm to allow service providers to publish various resources as web services (nowadays usually in the form of Web APIs) over the Internet [1]. With the popularity of Web 2.0 and related technologies, the number of web services has been growing rapidly [2], [3]. To fulfill comprehensive business goals, mashup has emerged as a popular technique to create value-added composite services by combining multiple individual services as components [4]. The popularity of mashup has driven the development of several service repositories (i.e., mashup platform) [5] in recent years, which continuously collect web services and mashups. Representative examples include ProgrammableWeb.com[1] from Bell Lab (oriented to generic services) and myexperiment.org[2] from the universities of Southampton, Manchester, and Oxford in the U.K. (oriented to scientific services). The rapid growth in the number of available services, however, has laid an obstacle in front of an inexperienced mashup developer, who has to spend a great amount of time to search for desired candidate service components [33]. Therefore, service recommendation has become critical to facilitate developers in finding suitable services. Without causing confusion, we will use (mashup) developer and (service) user interchangeably in this paper.

Service recommendation has become a demanding research topic in recent years. Specifically, based on a mashup description given by a user, a desired recommendation algorithm would come up with a ranked list of services, where higher ranked services are more likely to meet the user's need for mashup creation. Existing approaches mainly depend on static service descriptions offered by service providers, usually in the format of Web service description language (WSDL) documents, tags, and content descriptions. However, if a service description is poorly described or subjective to service provider's understanding, these approaches may fail to rank the service properly even though the service is relevant to the query. Furthermore, service descriptions are offered by service providers, while queries are made by mashup developers. Thus, there may exist a vocabulary gap between them.

To overcome these limitations, we propose to reconstruct service profiles from existing mashup descriptions, in other words, from how services have been consumed by users.

[1] http://www.programmableweb.com

[2] http://www.myExperiment.org

Our rationale is that service usage history carries meaningful information about services in mashup developers' vocabulary. Such constructed service profiles will be more objective based on user experiences. In addition, the occurrences of services as components in multiple mashups reflect their popularities, which provide valuable extra sources to enhance descriptive quality of service profiles. Furthermore, as time goes by with more mashups become available, service profiles can be enriched and become more accurate and objective.

Based on the three intuitions, we propose a novel way of applying the author topic model (ATM) [18] to generate new representation for each service from mashup descriptions and structures. Our key idea is to model component services as authors and mashup descriptions as documents. By learning the model, we can extract the co-occurrence of services and words in mashups, and thus gradually build comprehensive service profiles.

In addition, we believe the co-occurrence of services and words in mashups reveals both of their relevance and popularity from a user's perspective. First, more popular services get higher exposure to their descriptive words and vice versa. In addition, the more relevant a word is with respect to a service, the word is more likely to co-occur with the service in the same mashup. Based on such two assumptions, we further develop a novel service recommendation algorithm by identifying and emphasizing word features in the reconstructed service profiles.

The main contributions of this paper are summarized in three-fold as follows.

1) We propose a novel way of generating a new representation for services by leveraging mashup descriptions and structures. To the best of our knowledge, this is the first effort in the services computing area to automatically and systematically refine and enrich service profiles from mashup descriptions to facilitate service recommendation. We employ ATM to jointly model mashup descriptions and component services. The model captures the co-occurrence of service and word in mashups so that the reconstructed service profiles bridge the vocabulary gap between service providers and mashup developers.
2) We develop a service recommendation algorithm based on the reconstructed profiles. A novel term weighting method is further incorporated to enhance the algorithm by deriving and exploiting word features contained in the reconstructed profiles.
3) Comprehensive experiments on a real-world data set from ProgrammableWeb.com demonstrate that our proposed algorithm significantly outperforms the state-of-the-art methods by various evaluation metrics.

The remainder of this paper is organized as follows. Section II summarizes the related work. Section III introduces necessary definitions to describe an evolving service repository and then formulates the service recommendation problem. Section IV presents the probabilistic generative model for mashup creation to reconstruct service profiles. Section V proposes the novel service recommendation algorithm. Section VI reports the experimental results, and Section VII concludes this paper.

II. Related Work

Service recommendation is one of the key problems in the field of services computing. In this section, we discuss some prior works related to this paper divided into three categories: semantic-aware recommendation, QoS-aware recommendation, and network-aware recommendation.

A. Semantic-Aware Recommendation

Early works on semantic-aware service recommendation focused on the application of techniques from information retrieval, such as TF/IDF and vector space model (VSM), on the WSDL documents of services [6], [34]. They represented both user query and services as vector of words and calculated relevance scores using Cosine similarity between corresponding vectors. However, these methods require exact matching between queries and services, which is responsible for their unsatisfactory performance in practice. Reference [23] represented user's preferences by keywords and proposed a user-based collaborative filtering (CF) algorithm for service recommendation. However, its performance depends on the quality of domain thesaurus, which usually calls for great human effort.

To address the challenge, [7] proposed a probabilistic approach for service discovery based on latent dirichlet allocation (LDA), which introduced latent topics as a bridge between services and words extracted from corresponding WSDL documents. Relevance scores of services against user query are calculated based on query likelihood. Recently, [8] further proposed a user tagging augmented LDA (TA-LDA) model for service clustering and discovery, which integrated both the WSDL documents of services and tagging data. However, the prevalence of RESTful services makes it difficult to obtain WSDL documents of services. Moreover, due to business interest and other issues, service providers are unwilling to offer WSDL documents [9]. So, in this paper, we employ unstructured description data of services and mashups as sources of content rather than structured WSDL documents.

Another group of researchers focused on logic-based approaches. Reference [10] employed OWL-S to describe services and proposed an ontology-based method for service discovery. A hybrid discovery approach, which combines logic and content similarity, is also proposed in [26]. However, construction of ontology often involves a huge amount of human effort and is trapped in high computational complexity, which makes it intractable over large service candidates. Reference [36] presented a unified framework to facilitate and accelerate mashup development. Our recommendation algorithm can be integrated into the framework to help developers discover interesting component services for further development and deployment.

B. QoS-Aware Recommendation

A number of research work centered on nonfunctional properties of services (QoS) [11]. QoS-aware service recommendation aims to help users find services that can meet their QoS requirements among a list of functionally equivalent service candidates.

Collaborative filtering, which is based on the assumption that similar users tend to consume similar items, has been introduced for QoS prediction recently [12], [35]. Reference [27] employed temporal information to refine the similarity measurement in neighborhood-based CF. Location information is also incorporated to cluster services and users, and personalized recommendation is conducted based on the clustering results [25]. Reference [24] proposed a hidden Markov model-based approach to help users locate services with the optimal response time for their requests.

The aforementioned methods focused on individual services, while other methods concentrated at composition level. Reference [28] employed Monto Carlo simulations to predict the probability distributions of a WS-BPEL service orchestration. Reference [29] used partial selection technique to find the optimal service composition defined by an abstract workflow.

However, QoS information is not always available. Therefore, instead of using QoS attributes, we leverage objective description data about services and mashups to support service recommendation.

C. Network-Aware Recommendation

Another group of researchers proposed to exploit network analysis for service recommendation. In [13], service usage patterns of an evolving service repository, myExperiment, are studied and a recommendation algorithm is proposed based on service correlations in the network [14]. Wang *et al.* [32] investigated user behavior patterns in a service repository by analyzing user-API and user-tag network. The static structure and dynamic evolution of another popular service repository, ProgrammableWeb, is studied in [30] by means of network analysis. Furthermore, [15] presented a service recommendation approach based on link prediction in a dynamic service co-occurrence network. A recent work [16] employed association mining techniques over service network to understand positive and negative collaboration patterns among services.

However, these network-based methods do not take into account users' functional requirements, and therefore fail to exploit content information for recommendation.

In this paper, we have proposed a novel way of reconstructing service profiles from mashup descriptions and structures, which differentiates this paper from traditional methods based on original provider-offered service profiles. The reconstructed profiles from mashup descriptions have the following benefits to improve the accuracy of service recommendation.

1) When searching a service, mashup developers may not use the exact terms used by service providers. The reconstructed profiles describe the functions and features of services in mashup developers' vocabulary, and thus bridge the vocabulary gap between service providers and mashup developers.
2) The original provider-offered service profiles are sometimes poorly described, and even may be unavailable under certain circumstances. However, the reconstructed profile of a service is collectively described by mashups calling it, which guarantees its descriptive quality.

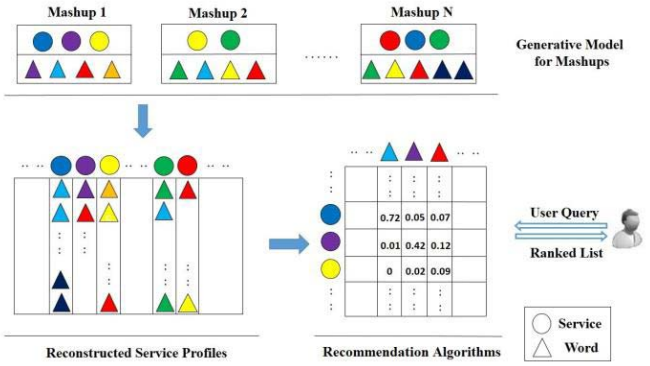

Fig. 1. Overview of the proposed service recommendation framework. The generative model for mashups is trained on all historical mashups to generate reconstructed service profiles, based on which recommendation algorithms are developed to recommend services in response to user queries.

III. Problem Definition

In this section, we first present several definitions concerning data schema of mashup and service in an evolving service repository, and then formulate the problem considered in this paper.

Definition 1 (Service): We denote S as the set of all services in a service repository. Each service comprises a collection of words offered by its provider as its original profile, which typically describes the functionality of the corresponding service.

Examples of original service profile in practice include WSDL documents, tags, and content descriptions.

Definition 2 (Mashup): We denote M as the set of all mashups in a service repository. Similarly, we associate a collection of words for each mashup as its mashup descriptions. $component\ service_m$ represents the collection of component services employed by mashup m.

Based on the above-mentioned definitions, we formulate the problem of service recommendation for mashup creation as follows.

Problem (Service Recommendation for Mashup Creation): Based on the text descriptions of services and mashups, service recommendation algorithms need to receive user query and return a ranked list of services, where higher ranked services are more likely to meet the user's need for mashup creation.

Such service recommendation algorithms for mashup developers can be deployed in an online web service repository, such as ProgrammableWeb.com. Based on the data, especially textual content on mashups and services collected by the service repository, our algorithm can conduct analysis and build objective service profiles. Our recommendation method can be added as a new feature at ProgrammableWeb.com, showing users a ranked list of services when she inputs a query in natural language.

To solve the problem, we propose a recommendation framework as shown in Fig. 1. Mashups are first modeled by ATM [18], where mashup descriptions are viewed as documents and component services as authors. The model is trained with all historical mashups available in the service repository to identify which word corresponds to which service in a mashup. Words assigned with the same service in all mashups constitute a reconstructed service profile. With the

new representation of services, a recommendation algorithm is designed based on co-occurrence count of service–word pairs to calculate relevance scores of services against user queries. Finally, services are ranked in a descending order of relevance scores and returned to the requesting user.

IV. Model Framework

In this section, we will discuss the probabilistic generative model for mashups and subsequent parameters estimation for service profile reconstruction (SPR).

A. Generative Model for Mashups

Original service profiles are provided by service providers, and are mainly about their functions and features. However, mashup developers may not exactly know what terms they need to use when searching a service. For example, a service may contain a phrase like "local restaurant recommendation" that describes its function. If a user searches for "find nearby food" that has no common words with the original profile, then traditional algorithms will not rank the aforementioned service high even though the two phrases are about the same function. In such cases, the mashup descriptions may play a role to bridge the vocabulary gap between service providers and mashup developers. Assume that there is an existing mashup that calls the aforementioned service and whose description contains the phrase "locating food nearby," the service may be assigned with words such as "food" and "nearby" in its reconstructed profile. In this way, the service can be discovered by matching query terms with its reconstructed profile. Moreover, original service profiles are sometimes poorly described and even may be unavailable under certain circumstances [9]. Therefore, we resort to mashup descriptions for new service representations.

Mashup description is usually a mixture of descriptions of its component services. Therefore, not all terms in a mashup description are relevant to a specific component service. We take an actual mashup *Lunchbox* from ProgrammableWeb.com for example. The mashup employs Google Maps and Yelp as component services and has description text as follows: "This tool shows you **restaurant ratings** around your *location*. If you want to go beyond your normal **lunch** *places*, this is a great app." As we can see, italic words describe the functionality of Google Maps, while bold words refer to Yelp. Therefore, we need to design a method to automatically identify relevant terms for each component service from mashup descriptions.

We resort to ATM to accomplish this goal. In ATM, each author is represented as a multinomial distribution over topics, and each topic is modeled as a multinomial distribution over words. The model represents each document as a mixture of topics, where the mixture weights are determined by its corresponding authors. We apply a similar idea to ATM, and analyze the mashup creation process in a probabilistic manner. Specifically, we assume that the mashup descriptions are generated by its component services. Each service is modeled by a mixture of topics, and each topic is represented by a probability distribution over words.

Table I summarizes the notations used in this model. Word tokens in descriptions of mashup m can be defined as a word vector $w_m = (w_{\mathrm{mi}})_{i=1}^{N_m}$. For brevity, we concatenate all mashups' word vectors to obtain a vector form of all mashups' descriptions $W = (w_m)_{m=1}^{|M|}$. Vector form of service assignment X and topic assignments Z can be defined in a similar way.

TABLE I
Notions Used in This Model

Symbol	Description
T	Number of topics
$\|M\|$	Number of mashups
$\|S\|$	Number of services
$\|V\|$	Number of distinguish words in all mashup descriptions
V	The set of distinguish words
X	Vector form of service assignments
Z	Vector form of topic assignments
W	Vector form of all mashups' descriptions
$\|W\|$	Number of word tokens in W
CS	Vector form of all mashups' composite services
w_{mi}	The ith word token in descriptions of mashup m
N_m	Number of word tokens in descriptions of mashup m
z_{mi}	Topic assigned with w_{mi}
x_{mi}	Service assigned with w_{mi}
θ_s	The parameter of multinomial distribution over topics specific to service s
ϕ_z	The parameter of multinomial distribution over words specific to topic z
α, β	The parameters of Dirichlet priors to the multinomial distribution θ_s and ϕ_z

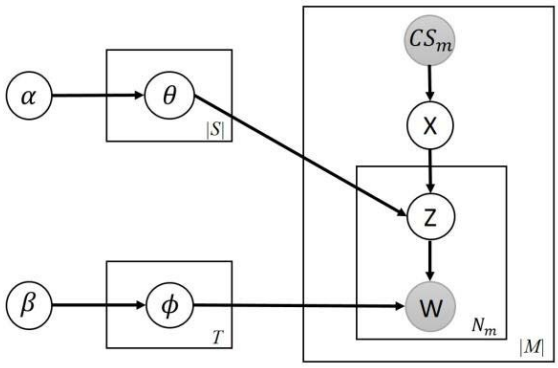

Fig. 2. Graphical model of mashup creation process.

The generative process of mashup creation can be described as follows.

1) For each topic z = 1:T, draw $\phi_z \sim$ Dirichlet (β).
2) For each service s in S, draw $\theta_s \sim$ Dirichlet (α).
3) For each word w_{mi}:
   a) draw a service x_{mi} uniformly from component service$_m$;
   b) draw a topic $z_{\mathrm{mi}} \sim$ Multinomial ($\theta_{x_{\mathrm{mi}}}$);
   c) draw a word $w_{\mathrm{mi}} \sim$ Multinomial ($\phi_{z_{\mathrm{mi}}}$).

The graphical model corresponding to this process is shown in Fig. 2. x indicates the service responsible for a given word, chosen from component service$_m$. Each service is associated with a multinomial distribution over topics θ, chosen from a Dirichlet prior α. The mixture weights of the chosen service is used to select a topic z and a word is finally generated according to the distribution over words specific to the chosen topic, which is drawn from a Dirichlet prior β.

Algorithm 1: Gibbs Sampling

Input:
1) Hyper-parameters α and β
2) The set of mashups M
3) Iteration number N
4) Number of topics T
5) Mashup descriptions W
6) Component services CS

Output
1) Service by word count matrix RSP

Procedure
01. Initialize Z and X randomly
02. **For** $iter = 1{:}N$
03. **For** each word token w_{mi} in W
04. Sample z_{mi} and x_{mi} according to equation (1)
05. **End**
06. **End**
07. Initialize a $|S| \times |V|$ zero matrix RSP
08. **For** $j = 1{:}|W|$
09. $RSP(X(j), W(j)) = RSP(X(j), W(j)) + 1$
10. **End**

B. Model Learning

We complete the service profile reconstruction by solving the generative model for mashups. There are four sets of unknown variables in the model: 1) distribution over topics of services θ_s; 2) distribution over words of topics ϕ_z; 3) topic assignments Z; and 4) service assignments X. A variety of algorithms have been proposed to estimate parameters of topic models. In this paper, we employ an approach analogous to [19] to derive a Gibbs sampler for our model.

We sample the topic z_{mi} and service x_{mi} for word token w_{mi} according to the following equation:

$$P(z_{\mathrm{mi}} = j, x_{\mathrm{mi}} = k | w_{\mathrm{mi}} = l, Z^{\neg i}, X^{\neg i}, \mathrm{W}^{\neg i}, \mathrm{CS}_m) \propto \frac{c_{kj}^{\neg i} + \alpha}{\sum_z \left(c_{kz}^{\neg i} + \alpha\right)} \times \frac{n_{jl}^{\neg i} + \beta}{\sum_v \left(n_{jv}^{\neg i} + \beta\right)} \quad (1)$$

where c_{kz} is the number of times that topic z has been associated with service k, and n_{jv} is the number of times that word v has been assigned to topic j. The superscript $\neg$ denotes a quantity excluding the current instance.

After sampling a sufficient number of iterations, we can get an estimation of the four sets of parameters. Traditional topic models employ topic distributions for query likelihood calculation. On the contrary, we omit the hidden topics and directly reconstruct service profile by converting all service–word pairs, i.e., W and X into a service by word count matrix denoted by RSP. Entries in RSP, which is a $|S| \times |V|$ matrix, represent the number of times some word (indicated by column index) have been employed by users to describe a particular service (indicated by row index) in all historical mashup descriptions.

Note that services adopted by more mashups have a greater chance to be exposed to words and vice versa. In addition, the more relevant a word is with respect to a service, the word is more likely to co-occur with the service in the same mashup. Therefore, the matrix captures both the popularity and relevance through the count number. The details of Gibbs sampling is summarized in Algorithm 1.

V. Recommendation Algorithm

In this section, we introduce our algorithms based on *service profile reconstruction*.

A. Basic Version

Specifically, for each word in the vocabulary V, we define a conditional probability distribution on the set of services S, which can be instantiated as follows:

$$p(s|w) = \frac{\mathrm{RSP}(s, w)}{\sum_k \mathrm{RSP}(k, w)}. \quad (2)$$

We can interpret $p(s|w)$ as the probability that service s will satisfy user's need if she inputs a word w in her query. A basic version of our algorithm calculates the relevance score of service s with respect to query Q by the following equation:

$$r(s, Q) = \sum_{w \in Q} p(s|w). \quad (3)$$

Note that we accumulate the contribution of every word in the query, rather than getting the product. Summing over the words contributions is equivalent to calculating arithmetic mean, which is more sensitive to extreme values compared with geometric mean strategies (multiply all words' contributions).

B. Dominant Words

Now, we present how to refine the basic algorithm by incorporating dominant words, which will be introduced in this section.

In the basic algorithm, we calculate the relevance score through (3), where words occurred in the query are equally weighted. However, we discover from RSP that a particular service may account for a large portion of co-occurrences with a given word. In other words, $p(s|w)$ is close to 1 for some service–word pair (s,w) in the reconstructed profile, which means that the occurrence of word w in user query nearly indicates the adoption of service s by the user. Naturally, words with this property should have more weight in relevance calculation. We name such words as dominant words to emphasize their important role in identifying desirable component services from user query.

In this paper, we propose two simple but effective rules to discover such dominant words based on (2).

1) $p(s|w) \geq g$, where g is a constant between 0 and 1.
2) $RSP(s, w) \geq h$, where h is a constant positive integer.

The first rule comes from the essential nature of dominant words. Since the numerator of $p(s|w)$ is $RSP(s,w)$, the second rule requires that service-word pair should have co-occurred for at least a certain number of times to support the high confidence of $p(s|w)$.

To exploit dominant words for better recommendation, we propose to assign them overwhelming weight in calculating relevance scores such that services with dominant word can be

placed ahead of others without dominant word. Our algorithm accomplish this goal simply by adding the length of user query to $p(s|w)$ when calculating relevance scores by (3) if w is a dominant word and s is the identified service. We prove the following theorem to demonstrate why this adjustment is effective.

Theorem: For any two services s_1 and s_2, where s_1 has at least one dominant word w in user query Q of length L, while s_2 has none, if a constant C is added to the relevance scores of s_1 calculated by (3) to guarantee $r(s_1, Q) + C \geq r(s_2, Q)$, then C should satisfy the following condition: $C \geq L - g$.

Proof: It is obvious that $r(s_2, Q) \leq L$, since $p(s|q) \leq 1$ for every word q in Q. Moreover, $r(s_2, Q)$ can reach L when $p(s_2|q) = 1$ and $RSP(s_2,q) < h$ for every word q in Q. Therefore, C should guarantee $r(s_1, Q) + C \geq L$, leading to $C \geq L - r(s_1, Q)$. For any service s_1, $r(s_1, Q) \geq p(s_1, w) \geq g$, which can be simply derived from (3) and the definition of dominant word. Since C is shared by all services with dominant word, C should satisfy the following condition: $C \geq \max_{s1}(L - r(s_1, Q)) = L - g$.

The theorem ensures that any constant greater than $L - g$ can satisfy our need. Since g is greater than 0, we simply set C to be L. Finally, the proposed algorithm is summarized in Algorithm 2.

The proposed algorithm can be divided into two stages: offline (Lines 01–09) and online (Lines 10–22). The offline part only needs to be conducted when more mashups are collected by the service repository. On the other hand, the online part performs every time when receiving a user query for mashup creation. Line 01 is the implementation of SPR based on ATM; Lines 02–09 describe the generation of dominant words from *RSP*; Lines 10–18 evaluate the relevance of services against online user query; Line 19 generates a ranked list of services for the user by sorting all services in descending order of the calculated relevance scores.

Algorithm 2: SPR

Input:
1) S: The set of all services
2) M: The set of all mashups
3) V: The set of unique words
4) *component service*: Vector form of all mashups' component services
5) W: Vector form of all mashups' mashup descriptions
6) T: Number of topics in ATM
7) N: Number of iterations in Gibbs sampling
8) α and β: Hyper-parameters in ATM
9) g and h: Thresholds for dominant words generation
10) Q: User query

Output:
1) LS: Ranked list of services

Procedure:
01. $RSP = GibbsSampling(\alpha, \beta, N, T, M, W, CS)$
02. **For** each word w in V
03. **For** each service s in S
04. Calculate $p(s|w)$ by equation (2)
05. **If** $p(s|w) \geq g$ and $RSP(s, w) \geq h$
06. Add (s,w) to DW
07. **End**
08. **End**
09. **End**
10. $L = \text{length}(Q)$
11. **For** each service s in S
12. Calculate relevance score r (s, Q) by equation (3)
13. **For** each word w in Q
14. **If** $(s,w) \in DW$
15. $r(s, Q) = r(s, Q) + L$
16. **End**
17. **End**
18. **End**
19. Return $LS = \text{sort}$ $(S, r(s,Q), \text{'descend'})$

C. Computational Complexity

This section discusses the upper bound on the computational complexity of *SPR*. In the following discussion, we assume that the online query has a number of P word tokens on average.

The complexity of Gibbs sampling (Line 01) to estimate the generative model for mashups is bounded by $O(NT|W|)$. From (2), we know the complexity of dominant words discovery (Lines 02–09) is $O(|S||V|)$. Therefore, the overall complexity of the offline part is $O\{(|S| + NT)|W|\}$, since $|V|$ is always less or equal to $|W|$. Similarly, the complexity of calculating relevance scores of services (Lines 10–18) is bounded by $O(P|S|)$, and the ranking of services (Line 19) is $O(|S| \log |S|)$. Therefore, the overall complexity of the online part is $O\{(P + |S|) \log |S|\}$.

From the previous discussion, we have demonstrated that our recommendation algorithms have polynomial complexity, and thus are computationally feasible in practice (both offline and online) to support real-time query.

VI. Experiments

In this section, we evaluate the proposed methods on a real-world data set crawled from the ProgrammableWeb.com. A collection of experiments were conducted to compare the state-of-the-art methods with our approaches for service recommendation.

A. Experimental Setup

1) Data Set Construction: We extracted the metadata of services and mashups from ProgrammableWeb.com, the largest online repository of web services and mashups [30]. The data used in our experiments spans from September 2005 to July 2013. Each service contains metadata such as name, tags, and descriptions. Every mashup contains the information, such as name, tags, creation date, descriptions, and a list of component services.

For each mashup in the data set, there is an unstructured description consisting of a bag of words that describe its functionality and features. Before the descriptions can be used in our experiments, we have to apply several natural

TABLE II
BASIC PROPERTIES OF PROGRAMMABLE WEB DATA SET

Number of services	7,077
Number of mashups	6,594
Number of unique words	13,648
Average number of component services per mashup	2.1

language preprocessing techniques to the data. In this paper, we extended the preprocessing method similar in [31] to the raw texts to extract meaningful words as follows.

1) *Words Extraction:* First of all, we tokenize the text data according to the separator.
2) *Pruning:* The second task is to discard words that are not meaningful for recognizing the service. Examples include some articles, such as *a, an, the*; some prepositions, such as *in, on, with, by, for, at, about, from,* and so on; some adverbs, such as *where, when, quite,* and so on.
3) *Suffix Striping:* Third, suffix stripping is performed to get stem words. For example, *map*, *mapping*, *maps*, and *mappings* will be replaced with the same stem *map*.
4) *Error Correction:* Special characters not identified by natural language processing tools are recovered. Meaningful words, such as EC2 and S3, are added back, which are usually filtered by natural language processing tools, since they are not standard words.
5) *Text Augmentation:* Since the name and tags of a mashup may contain meaningful words to complement its description data, we enhance the semantics of mashup description by including its name and tags. For example, a mashup named "Tweets and Diggs" has word "tweet," which can substantially improve the descriptive quality of its description that originally does not contain it.

Similar actions are performed on the descriptions of services, and the processed description of a service can be viewed as original profile offered by its provider. Table II summarizes the basic properties of our data set.

To examine the performance of the proposed approach, we divided the whole data set into two parts at four different time points, and thus get four test cases denoted from *D1* to *D4*. As shown in Fig. 3, we use the historic data (data before the dividing point in the timeline) for training and the rest part for validation. For each mashup appeared in the testing data set, we use its descriptions as user query and its component services as the ground truth.

2) Evaluation Metric: Following previous works [20], [22], we evaluate the recommendation performance of different methods with two metrics, namely mean average precision (MAP), and recall (Rec@K).

MAP is a widely used evaluation metric in recommender systems. In this paper, it measures how well an approach can rank adopted services higher in the recommendation list. The equation for calculating average MAP is presented as follows:

$$\text{MAP} = \frac{1}{|\text{CS}_m|} \sum_{s \in \text{CS}_m} \frac{\text{top}(s, m)}{\text{rank}(s, m)} \tag{4}$$

rank(s,m) is the ranking position of service s in the recommended list of services for a testing mashup m, and top(s,m) is the number of component services of mashup m, which appear in the Top-K result of the ranked list, where K is equal to rank(s,m).

Fig. 3. Generation of training and testing data sets.

MAP is a real number between 0 and 1. Methods with higher MAP present a better recommendation performance than those with lower MAP.

Another commonly used metric Rec@K is the fraction of adopted services that are contained in the Top-K recommendations. Equations for calculating Rec@K are presented as follows:

$$\text{Rec@}K = \frac{|\text{Top} - K(m) \cap \text{CS}_m|}{|\text{CS}_m|} \tag{5}$$

where Top-$K(m)$ is the *Top-K* recommended services for mashup m. Since the average number of component services per mashup is close to two according to Table II, and K is fixed at five in our experiments.

3) Comparison Methods: We compare the following methods with our approach (*SPR* and its basic version *Service Profile Reconstruction-BAsic version*) for service recommendation.

a) Vector space model: It represents each service as a feature vector of words w_s based on corresponding original profile offered by its provider [6]. User query Q is also modeled as a vector of words w_q and Cosine similarity is adopted to score the relevance between service and user query

$$Sim\_vsm(s, Q) = \frac{w_s \cdot w_q}{\|w_s\| \|w_q\|}. \tag{6}$$

b) Latent Dirichlet allocation: It employs LDA to model the generation of service descriptions with latent topics as a bridge between services and words [7]. With a model learned, we can leverage the estimated parameters, topic distribution of services $p(k|s)$, and word distribution of topics $p(w|k)$, to calculate the relevance score of services against user query Q as follows:

$$Sim\_\text{LDA}(s, Q) = \prod_{w \in Q} \sum_{k} p(w|k)\, p(k|s). \tag{7}$$

c) Tagging augmented LDA: It incorporates tags into the LDA framework introduced previously. The relevance calculation is almost identical to LDA with a minor difference that the topic distribution of a service is obtained by averaging over topic distribution of its included tags [8].

d) Service usage frequency (SUF): It recommends services in descending order of service usage frequency regardless of mashup descriptions.

e) Collaborative filtering: It leverages service compositions of past mashups to make recommendation. The basic idea is that if the query mashup m_q has the same or similar functionality as some mashup m, then m_q is likely to employ the same component services as m. CF recommends services for user query Q using the following equation:

$$Sim\_cf(s, Q) = \sum_{m \in M} r(m, Q) I(s, m) \tag{8}$$

where $r(m,Q)$ is the similarity between historical mashup m and the user query Q. Similar to [21], we model the mashup descriptions by the LDA and calculate the similarity as the likelihood of generating the current user query according to the estimated parameters specific to mashup m. The indicator function $I(s,m)$ is 1 if the service s belongs to CS_m and 0 otherwise.

All baselines are evaluated under the optimal settings. Next, we set up the parameters of SPR and SPR-BA. For hyperparameters in the ATM, we empirically set $\alpha = 50/T$ and $\beta = 0.01$. The number of topics is set to 50 and the number of iterations in Gibbs sampling is set to 100. For threshold parameters in dominant words of SPR, we set $g = 0.5$ and $h = 1$ empirically by experiments, which will be explained in Section VI-B. All experiments were conducted on a Core 2 Duo 3.00-GHz machine with 4-GB RAM.

B. Quantitative Analysis

1) Recommendation Performance: Table III lists the comparison of performance of service recommendation for mashup creation using various methods on four test cases.

The methods leveraging mashup descriptions (CF and our proposed methods) perform much better than those depending on original service profiles (VSM, LDA, and TA-LDA) and SUF. The reason for the performance gap is that mashup descriptions can bridge the vocabulary gap between mashup developers and service providers. Moreover, our proposed algorithm SPR and its basic version clearly outperform the CF. The reason is twofold. On one hand, our proposed algorithms derive word features based on co-occurrence of service and word in mashup descriptions, which integrates both popularity and relevance measures. On the other hand, SPR and SPR-BA employ the weighted average mean strategy to aggregate the contribution of individual words in user query, which is more sensitive to extreme feature values. Finally, the proposed algorithm SPR obtains significant improvement over its basic version SPR-BA by discovering dominant words, and emphasizing them in relevance evaluation, which demonstrates the effectiveness of dominant words.

2) Impact of T: In this section, we examine how parameters influence the recommendation performance of the proposed methods. With prior knowledge that there are about 40 service domains in ProgrammableWeb.com, we display in Fig. 4 the MAP of SPR and SPR-BA on *D1* with T varied from 10 to 70 with a step length of 10. All the other parameters are fixed as before.

TABLE III
RECOMMENDATION PERFORMANCE BY DIFFERENT METHODS

Dataset	Method	MAP	Rec@5
D1	VSM	5.93%	8.42%
	LDA	17.09%	30.01%
	TA-LDA	19.86%	38.11%
	SUF	25.82%	30.02%
	CF	36.84%	42.23%
	SPR-BA	43.51%	57.44%
	SPR	**54.07%**	**59.09%**
D2	VSM	5.70%	8.71%
	LDA	12.44%	26.21%
	TA-LDA	14.61%	33.95%
	SUF	26.74%	31.71%
	CF	36.75%	41.78%
	SPR-BA	42.98%	55.84%
	SPR	**52.98%**	**57.43%**
D3	VSM	5%	6.38%
	LDA	13.19%	28.98%
	TA-LDA	15.86%	36.39%
	SUF	27.88%	32.98%
	CF	38.55%	43.59%
	SPR-BA	45.10%	56.86%
	SPR	**53.55%**	**57.65%**
D4	VSM	6.26%	6.92%
	LDA	13.99%	25.84%
	TA-LDA	19.63%	38.27%
	SUF	26.60%	32.07%
	CF	41.94%	45.99%
	SPR-BA	48.01%	60.52%
	SPR	**54.42%**	**61.77%**

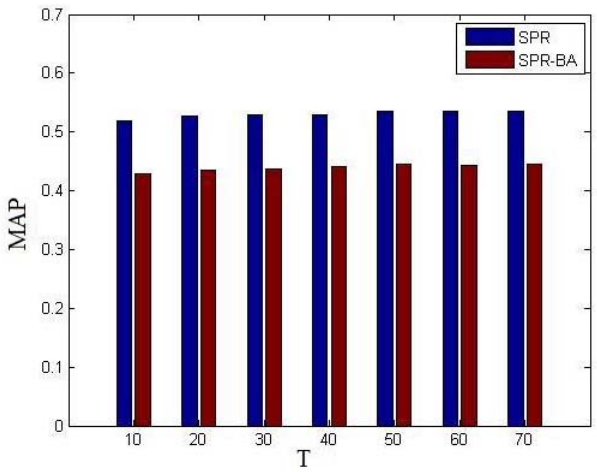

Fig. 4. MAP of the proposed methods with the number of topics varied.

Taking SPR for example, we can see that although the performance changes with T varied, the largest difference is less than 0.02. This demonstrates that the SPR is not sensitive to the number of topics. Similar conclusion can be drawn for the SPR-BA.

Since the computational complexity of both the SPR and the SPR-BA are linear with T, T should be as small as possible

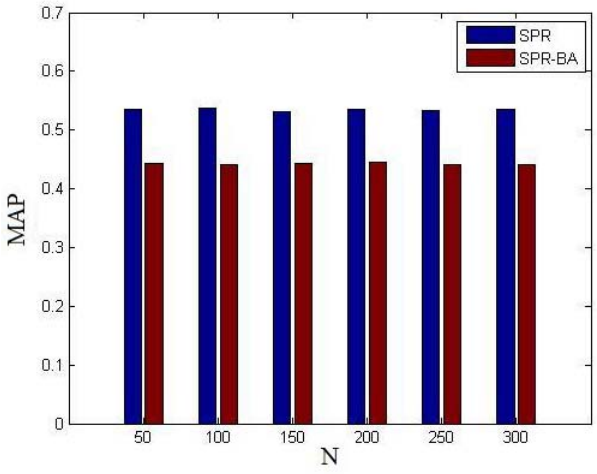

Fig. 5. MAP of the proposed methods with a number of iterations in Gibbs sampling varied.

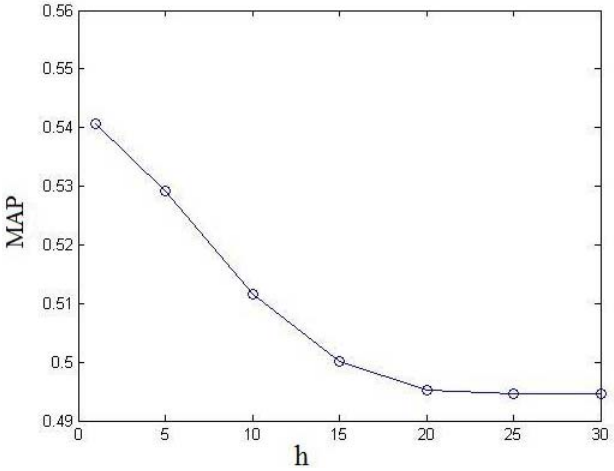

Fig. 6. MAP of SPR on *D1* with g = 0.5 and h varied.

to save computation cost. We can observe that the MAP is the highest when T is equal to 50 and changes a little with T increased. So, we choose T to be 50 in our experimental settings.

3) Impact of N: In this section, we examine the effect of the number of Gibbs sampling iterations N on recommendation performance of our proposed methods.

Fig. 5 shows the MAP of the SPR and the SPR-BA on *D1* with N varied. We can see that both the algorithms can get good performance in just 100 iterations and their MAP become stable with N increased. This suggests that our algorithms are efficient, and have a good convergence property.

4) Impact of g and h: In this section, we examine the effect of the constants g and h on recommendation performance of the SPR. According to the definition of dominant word, g should be large enough to ensure that the word is dominant. So, we first fix g at a typical level 0.5, and study the effect of h on the performance of the SPR.

Fig. 6 shows the MAP of the SPR on *D1* with $g = 0.5$ and h varied from 1 to 30 with a step value of 5. The MAP of the SPR decreases as h increases, and becomes stable when h is up to 20. The explanation is that increase in h narrows the range of dominant words. As h becomes sufficiently large,

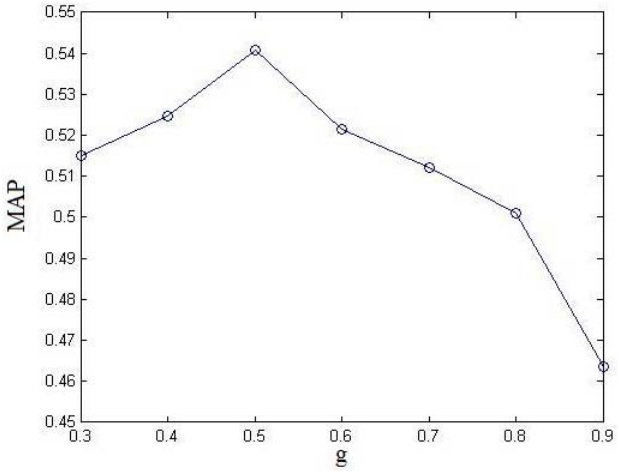

Fig. 7. MAP of SPR on *D1* with $h = 1$ and g varied.

there will be no dominant words and the SPR is reduced to SPR-BA. Therefore, h should be as small as possible, and we fix h at 1 to further investigate the effect of g on the performance of SPR.

Fig. 7 shows the MAP of SPR on D1 with g varied from 0.3 to 0.9 with a step value of 0.1 and h fixed at 1. The MAP of SPR improves as g increases from 0.3 and reach a peak at $g = 0.5$, then declines as g goes beyond 0.5. The reason for the trend is twofolds. As g increases from 0.3 to 0.5, some false dominant words are filtered out, while true dominant words are still preserved. When g further increases beyond 0.5, the SPR drops more and more true dominant words, which leads to its decline in the MAP. Therefore, we set g at 0.5 in this experiment.

C. Qualitative Analysis

1) Service Profile Comparison: To further explore the reason why the SPR and the SPR-BA outperform those baselines, we compare the original service profile offered by service providers with reconstructed service profiles from mashup descriptions.

Take *Google Maps* for example, we list the *Top-10* frequent words of both original and reconstructed profile along with their occurrence times in Table IV. The original profile comes from service descriptions, while the reconstructed profile is obtained from service by word matrix the *RSP*, the output of Algorithm 1 applied to data set *D4* with the same parameter settings as before.

As we can see from Table IV, the reconstructed service profile is significantly different from the original one. Only two words, *map* and *google*, of the *Top-10* words in the original profile remain in the list of the reconstructed service profile. Moreover, the occurrence times of words in the reconstructed profile have increased a lot compared with the static descriptions offered by its provider, since *Google Map* is the most popular web service in the Programmable.com. According to the generative model for mashup creation, *Google Maps* is employed by many mashups, so it has a much greater chance to be assigned to different words in mashup descriptions. Therefore, the occurrence times can be used as an integrated measure of popularity and relevance. Furthermore, words, such

TABLE IV
SERVICE PROFILE COMPARISON

Rank	Original Profile		Reconstructed Profile	
	word	count	word	count
1	map	4	map	1434
2	google	3	google	743
3	local	2	location	389
4	interface	2	mashup	224
5	retrieval	2	find	183
6	develop	2	city	169
7	http	2	world	162
8	simple	2	show	161
9	web	1	place	159
10	application	1	search	158

TABLE V
DOMINANT WORDS GENERATION

Service	Dominant words
Google Maps	map, location, google, world, city
Twitter	tweet, twitter, follow
YouTube	youtube, video
Flickr	flickr, photo
Amazon eCommerce	amazon, ecommerce
Facebook	Facebook
Twilio	Twilio, call
eBay	eBay, auction
Last.fm	Last.fm
del.icio.us	del.icio.us, bookmark
411Sync	wap, sync, capable
foursquare	check-in, foursquare
Amazon S3	storage, S3
Digg	digg, bury

as *location* and *show*, presented in the reconstructed profile are not contained in the original one, which reflects the vocabulary gap between mashup developers and service providers. With the same user query, all these features of the reconstructed service profiles contribute to a more comprehensive relevance score, and thus better recommendation performance.

2) Dominant Words Generation: In this section, we investigate dominant words generated by the SPR on *D4*, where the parameters are set as before. We list part of the dominant words with their identified services in Table V.

As we can observe from Table V, the majority of dominant words can be divided into three categories. The first one is the service's name, which is the most representative word of a service. Examples include Flickr, Facebook, Last.fm, and so on. The second category can be named as the featured word of services. Words in this category are usually peculiar to a particular service. Therefore, the occurrence of these words in user query can be a decisive signal to adopt the corresponding services. Taking Digg for example, one of its dominant word, *bury*, is the name of a button by which users of Digg express their voting to pages. The last category contains words used to describe the core function provided by the service. For example, *call* in Twilio, *auction* in eBay, and *storage* in Amazon S3 fall into this category. These words demonstrate the popularity of their corresponding identified services.

By analyzing the results of dominant words generation from the SPR, we can conclude that it is meaningful to emphasize dominant words in relevance evaluation.

VII. CONCLUSION

In this paper, we have presented a novel way of generating comprehensive profiles for services based on service usage history to bridge the vocabulary gap between mashup developers and service providers. Specifically, we leverage the ATM to capture co-occurrence of service and word in mashups, based on which service profiles are reconstructed, refined, and enriched dynamically. Furthermore, we have developed a service recommendation algorithm by deriving and exploiting word features hidden in the reconstructed profiles. The experimental results on a real-world data set demonstrate the effectiveness of the proposed methods.

For the future work, we plan to incorporate users explicitly into our model to generate user profiles to support personalized service recommendation.

REFERENCES

[1] Z. Zhou, M. Sellami, W. Gaaloul, M. Barhamgi, and B. Defude, "Data providing services clustering and management for facilitating service discovery and replacement," *IEEE Trans. Autom. Sci. Eng.*, vol. 10, no. 4, pp. 1131–1146, Oct. 2013.

[2] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service-oriented computing: A research roadmap," *Int. J. Cooperat. Inf. Syst.*, vol. 17, no. 2, pp. 223–255, Jun. 2008.

[3] M. Maleshkova, C. Pedrinaci, and J. Domingue, "Investigating web services on the World Wide web," in *Proc. IEEE 8th Eur. Conf. Services Comput.*, Dec. 2010, pp. 107–114.

[4] D. Benslimane, S. Dustdar, and A. Sheth, "Services mashups: The new generation of web applications," *IEEE Internet Comput.*, vol. 12, no. 5, pp. 13–15, Sep. 2008.

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

[6] C. Platzer and S. Dustdar, "A vector space search engine for web services," in *Proc. 3rd IEEE Eur. Conf. Services Comput.*, Nov. 2005, pp. 62–71.

[7] C. Li, R. Zhang, J. Huai, X. Guo, and H. Sun, "A probabilistic approach for web service discovery," in *Proc. IEEE 20th Int. Conf. Services Comput.*, Jun. 2013, pp. 49–56.

[8] L. Chen, Y. Wang, Q. Yu, Z. Zheng, and J. Wu, "WT-LDA: User tagging augmented LDA for web service clustering," in *Service-Oriented Computing*. Berlin, Germany: Springer, 2013, pp. 162–176.

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

[10] M. Klusch, B. Fries, and K. Sycara, "Automated semantic web service discovery with OWLS-MX," in *Proc. 5th Int. Joint Conf. Auto. Agents Multiagent Syst.*, May 2006, pp. 915–922.

[11] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-aware middleware for web services composition," *IEEE Trans. Softw. Eng.*, vol. 30, no. 5, pp. 311–327, May 2004.

[12] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "QoS-aware web service recommendation by collaborative filtering," *IEEE Trans. Services Comput.*, vol. 4, no. 2, pp. 140–152, Apr./Jun. 2011.

[13] W. Tan, J. Zhang, and I. Foster, "Network analysis of scientific workflows: A gateway to reuse," *IEEE Comput.*, vol. 43, pp. 54–61, Sep. 2010.

[14] J. Zhang, W. Tan, J. Alexander, I. Foster, and R. Madduri, "Recommend-as-you-go: A novel approach supporting services-oriented scientific workflow reuse," in *Proc. IEEE 9th Int. Conf. Services Comput.*, Jul. 2011, pp. 48–55.

[15] K. Huang, Y. Fan, and W. Tan, "Recommendation in an evolving service ecosystem based on network prediction," *IEEE Trans. Autom. Sci. Eng.*, vol. 11, no. 3, pp. 906–920, Jul. 2014.

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

[17] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, Mar. 2003.
[18] M. Rosen-Zvi, T. Griffiths, M. Steyvers, and P. Smyth, "The author-topic model for authors and documents," in *Proc. 20th Conf. Uncertainty Artif. Intell.*, Jul. 2004, pp. 487–494.
[19] I. Porteous, D. Newman, A. Ihler, A. Asuncion, P. Smyth, and M. Welling, "Fast collapsed gibbs sampling for latent Dirichlet allocation," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2008, pp. 569–577.
[20] B. Xia *et al.*, "Domain-aware service recommendation for service composition," in *Proc. IEEE 21st Int. Conf. web Services*, Jun. 2014, pp. 439–446.
[21] Y. Zhong, Y. Fan, K. Huang, W. Tan, and J. Zhang, "Time-aware service recommendation for mashup creation in an evolving service ecosystem," in *Proc. IEEE 21st Int. Conf. web Services*, Jun. 2014, pp. 25–32.
[22] Q. Yuan, G. Cong, and C. Lin, "COM: A generative model for group recommendation," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2014, pp. 163–172.
[23] S. Meng, W. Dou, X. Zhang, and J. Chen, "KASR: A keyword-aware service recommendation method on MapReduce for big data application," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 12, pp. 3221–3231, Dec. 2014.
[24] W. Ahmed, Y. Wu, and W. Zheng, "Response time based optimal web service selection," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 2, pp. 551–561, Feb. 2015.
[25] X. Chen, Z. Zheng, Q. Yu, and M. Lyu, "web service recommendation via exploiting location and QoS information," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 7, pp. 1913–1924, Jul. 2014.
[26] M. Klusch, P. Kapahnke, and I. Zinnikus, "Hybrid adaptive web service selection with SAWSDL-MX and WSDL-analyzer," in *The Semantic web: Research and Applications*. Springer, 2009, pp. 550–564.
[27] Y. Hu, Q. Peng, and X. Hu, "A time-aware and data sparsity tolerant approach for web service recommendation," in *Proc. IEEE 21st Int. Conf. web Services*, Jun. 2014, pp. 33–40.
[28] L. Bartoloni, A. Brogi, and A. Ibrahim, "Probabilistic prediction of the QoS of service orchestrations: A truly compositional approach," in *Service-Oriented Computing*. Springer, Nov. 2014, pp. 378–385.
[29] Y. Chen, J. Huang, and C. Lin, "Partial selection: An efficient approach for QoS-aware web service composition," in *Proc. IEEE 21st Int. Conf. web Services*, Jun. 2014, pp. 1–8.
[30] K. Huang, Y. Fan, and W. Tan, "An empirical study of programmable web: A network analysis on a service-mashup system," in *Proc. IEEE 19th Int. Conf. web Services*, Jun. 2012, pp. 552–559.
[31] K. Huang *et al.*, "Mirror, Mirror, on the web, which is the most reputable service of them all?" in *Service-Oriented Computing*. Springer, Dec. 2013, pp. 343–357.
[32] J. Wang, H. Chen, and Y. Zhang, "Mining user behavior pattern in mashup community," in *Proc. IEEE Int. Conf. Inf. Reuse Integr.*, Aug. 2009, pp. 126–131.
[33] M. Weiss and G. R. Gangadharan, "Modeling the mashup ecosystem: Structure and growth," *R&D Manage.*, vol. 40, no. 1, pp. 40–49, Jan. 2010.
[34] X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang, "Similarity search for web services," in *Proc. 13th Int. Conf. Very Large Data Bases*, vol. 30. Aug. 2004, pp. 372–383.
[35] Q. Zhang, C. Ding, and C. Chi, "Collaborative filtering based service ranking using invocation histories," in *Proc. 18th IEEE Int. Conf.*, Jul. 2011, pp. 195–202.
[36] X. Liu, G. Huang, Q. Zhao, H. Mei, and M. Blake, "iMashup: A mashup-based framework for service composition," *Sci. China Inf. Sci.*, vol. 57, no. 1, pp. 1–20, Jan. 2014.

Yang Zhong received the B.S. degree in control theory and application from Tsinghua University, Beijing, China, in 2012, where he is currently pursuing the Ph.D. degree with the Department of Automation.

His current research interests include services computing and data mining.

Yushun Fan received the Ph.D. degree in control theory and application from Tsinghua University, Beijing, China, in 1990.

From 1993 to 1995, he was a Visiting Scientist, supported by A. von H. Stiftung, with the Fraunhofer Institute for Production System and Design Technology, Berlin, Germany. He is currently a Professor with the Department of Automation, Director of the System Integration Institute, and Director of the Networking Manufacturing Laboratory, Tsinghua University. He has authored ten books in enterprise modeling, workflow technology, intelligent agent, object-oriented complex system analysis, and computer integrated manufacturing. He has published more than 300 research papers in journals and conferences. His research interests include enterprise modeling methods and optimization analysis, business process reengineering, workflow management, system integration, object-oriented technologies and flexible software systems, Petri nets modeling and analysis, and workshop management and control.

Wei Tan (M'12–SM'13) received the B.S. and Ph.D. degrees from the Department of Automation, Tsinghua University, Beijing, China, in 2002 and 2008, respectively.

From 2008 to 2010, he was a Researcher with the Computation Institute, University of Chicago, Chicago, IL, USA, and the Argonne National Laboratory, Lemont, IL, USA. At that time, he was the technical lead of the caBIG workflow system. He is currently a Research Staff Member with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA. He has published more than 70 journal and conference papers, and a monograph entitled *Business and Scientific Workflows: A web Service-Oriented Approach* (272 pages, Wiley-IEEE Press). His research interests include GPU accelerated machine learning, NoSQL, cloud computing, service-oriented architecture, business and scientific workflows, and Petri nets.

Dr. Tan is a member of the Association for Computing Machinery (ACM). He was a recipient of the IEEE Peter Chen Big Data Young Researcher Award in 2016, the Best Paper Award from the ACM/IEEE CCGrid in 2015, the Best Student Paper Award from the IEEE ICWS in 2014, the Best Paper Award from the IEEE SCC in 2011, the Pacesetter Award from the Argonne National Laboratory in 2010, and the caBIG Teamwork Award from the National Institute of Health in 2008. He is an Associate Editor of the IEEE TRANSACTIONS ON AUTOMATION, SCIENCE AND ENGINEERING. He has served in the program committees of many conferences and co-chaired several workshops.

Jia Zhang (M'03–SM'15) received the M.S. and B.S. degrees from Nanjing University, Nanjing, China, and the Ph.D. degree from the University of Illinois at Chicago, Chicago, IL, USA, all in computer science.

She is currently an Associate Professor with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, USA. She has co-authored one textbook titled *Services Computing* and has published over 140 refereed journal papers, book chapters, and conference papers. Her recent research interests include center on service oriented computing, with a focus on scientific workflows, net-centric collaboration, and big data management.

Dr. Zhang is currently an Associate Editor of the IEEE TRANSACTIONS ON SERVICES COMPUTING and the *International Journal of web Services Research*, and the Editor-in-Chief of the *International Journal of Services Computing*.