Tsinghua Science and Technology

Volume 22 | Issue 3 Article 6

2017

Extracting Relevant Terms from Mashup Descriptions for Service Recommendation

Yang Zhong
the Department of Automation, Tsinghua University, Beijing 100084, China.

Yushun Fan
the Department of Automation, Tsinghua University, Beijing 100084, China.

TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 06/09 pp293–302
Volume 22, Number 3, June 2017

Extracting Relevant Terms from Mashup Descriptions for Service Recommendation

Yang Zhong and Yushun Fan*

Abstract: Due to the exploding growth in the number of web services, mashup has emerged as a service composition technique to reuse existing services and create new applications with the least amount of effort. Service recommendation is essential to facilitate mashup developers locating desired component services among a large collection of candidates. However, the majority of existing methods utilize service profiles for content matching, not mashup descriptions. This makes them suffer from vocabulary gap and cold-start problem when recommending components for new mashups. In this paper, we propose a two-step approach to generate high-quality service representation from mashup descriptions. The first step employs a linear discriminant function to assign each term with a component service such that a coarse-grained service representation can be derived. In the second step, a novel probabilistic topic model is proposed to extract relevant terms from coarse-grained service representation. Finally, a score function is designed based on the final high-quality representation to determine recommendations. Experiments on a data set from ProgrammableWeb.com show that the proposed model significantly outperforms state-of-the-art methods.

Key words: service recommendation; topic model; mashup descriptions; linear discriminant function

1 Introduction

With the prevalence of web services and related technologies, service computing has become an essential part in the information era. Due to the increasing number of web services, mashup has emerged as a popular technique to facilitate developers creating new web applications through service compositions[1]. Several mashup platforms[2] have drawn great attention in recent years such as IBM's ProgrammableWeb.com and myExperiment.org created by the universities of Southampton, Manchester, and Oxford in the UK. These websites serve as a platform for users to publish and consume web services and their compositions. However, mashup developers suffer from the growing number of available services since it is a challenging and time-consuming task to search for desired component services in a vast repository. Service recommendation and discovery is a popular solution to information overload faced by mashup developers. To avoid confusion, we will use (mashup) developer and (service) user interchangeably in this paper.

Service recommendation and discovery is a hot research theme in the service computing society. By analyzing developer's query intentions, service recommendation aims to generate a list of component services in response to facilitate mashup creation. Existing methods are mainly based on content matching between user queries and service profiles from service providers[3,4]. However, there are two inherent drawbacks with these approaches. Firstly, if a service profile is poorly described by its provider, these methods can hardly rank the service high although it is relevant to the query. Moreover, service profiles

• Yang Zhong and Yushun Fan are with the Department of Automation, Tsinghua University, Beijing 100084, China. E-mail: zhongy12@mails.tsinghua.edu.cn; fanyus@tsinghua.edu.cn.
∗ To whom correspondence should be addressed.
Manuscript received: 2016-05-07; revised: 2016-09-28; accepted: 2016-10-20

are written by service providers while queries are made by mashup developers, leading to a vocabulary gap between the two. Mashup developers may not exactly know what terms they should input when searching a service. Recently, several latent factors models are proposed for service recommendation leveraging mashup-service usage matrix[5, 6]. However, both methods ignore mashup descriptions and therefore suffer from cold-start problem for new mashups.

To overcome these drawbacks, we propose to derive high-quality service representations by extracting relevant terms from mashup descriptions, which usually consists of feature words of component services and mashup-specific terms. Since mashup descriptions are written in the vocabulary of developers, the new representation naturally fills the vocabulary gap. Also, the more historical mashups a service is co-invoked by, the higher the descriptive quality of its new representation will be. For example, a service may have terms like "local restaurant reviews" in its profile. If a mashup developer searches for "find nearby food", profile-based methods fail to rank the service high since its profile has no common terms with the query. However, if there is a historical mashup employing the particular service and containing words like "searching food nearby" in its description text, the service may be assigned with terms such as "food" and "nearby" in its new representation. Thus the service may be discovered with the help of the new representation.

We propose a two-step approach to accomplish this goal. As a preliminary treatment, we define a linear discriminant function to assign one component service to each word in mashup descriptions. Terms assigned to the same service in a mashup description can be viewed as a separate comment for the service in developer's vocabulary. Obviously, not all terms in the developer's comments are relevant to its responsible service. Therefore, we further propose a novel probabilistic topic model to identify relevant terms and filter noisy words. High-quality service representations can be obtained by aggregating relevant terms in all corresponding developer comments.

Based on the new representation, we design a score function to measure the matching degree between services and user queries. The key idea is to employ a distribution of term counts over services, which integrates relevance and popularity.

The main contributions of this paper are summarized as follows:

(1) We propose a two-step approach to Extract Relevant Terms (ERT) for component services from mashup descriptions. The relevant terms of a service constitute a high-quality representation with clear advantages over traditional service profile.

(2) We further design a score function based on the new representation to recommend component services. The score function captures both service-word relevance and their respective popularity in a unified way.

(3) Comprehensive experiments on a real-world data set from ProgrammableWeb.com illustrate that the proposed method significantly outperforms state-of-the-art approaches in recommendation accuracy.

The remainder of this paper is organized as follows. Section 2 introduces necessary definitions to describe a mashup platform and then formulates the service recommendation problem. Section 3 presents the relevant terms extraction framework and service recommendation algorithm. Section 4 reports the experimental results and Section 5 summarizes the related work. Section 6 concludes the paper.

2 Problem Definition

In this section, we introduce necessary definitions related to mashup platform and then formulate the service recommendation problem.

We denote V as the set of unique words that appear in all service profiles and mashup descriptions. S represents the set of web services collected by a mashup platform. Every service $s \in S$ contains a bag of words SP_s offered by its provider as profile. The service profile usually accounts for its core functions and features. Typical examples include Web Service Description Language (WSDL) documents[7] , tags[8], and unstructured text descriptions[9]. We denote M as the set of mashups created in a mashup platform. Similarly, a bag of words MD_m is associated with every mashup $m \in M$ to give a necessary description. CS_m denotes the set of component services invoked by m. Based on these definitions, we now give a formal statement of the problem considered in this paper.

Problem: Component Service Recommendation for Mashup Developer. Given a text query from mashup developer, component service recommendation algorithm aims to generate a ranked list of services in response, where higher ranked services should have a greater chance to be invoked by the query user.

The component service recommendation algorithm can be deployed in a mashup platform such as ProgrammableWeb.com. Given the text query, the algorithm can facilitate developers finding desirable component services and shorten development cycle.

The overview of the proposed model is illustrated in Fig. 1. Firstly, each word in mashup descriptions is associated with one component service by a linear discriminant function. Words with the same component service in a mashup constitute a separate developer comment. The second step uses a novel probabilistic topic model to extract relevant terms from developer comments that are related to each service. Finally, relevant terms are preserved and aggregated to form a new service representation, upon which a score function is further designed to measure the matching degree between services and user queries. A ranked list of services is returned to the requesting developer in a descending order of scores.

3 Model Framework

In this section, we will present our two-step approach to extract relevant terms from mashup descriptions, upon which a score function is further proposed to generate recommendation lists.

3.1 Service assignment for terms

Generally, feature words of component services and mashup-specific terms constitute a mashup description. For example, a mashup that invokes Google Maps and Yelp may have the following description: "The application allow users to visualize **local restaurants** on the *map* along with **customer reviews** according to your real-time *GPS position*." Terms in italics refer to Google Maps while bold terms are relevant to Yelp, with others being mashup-specific. To extract

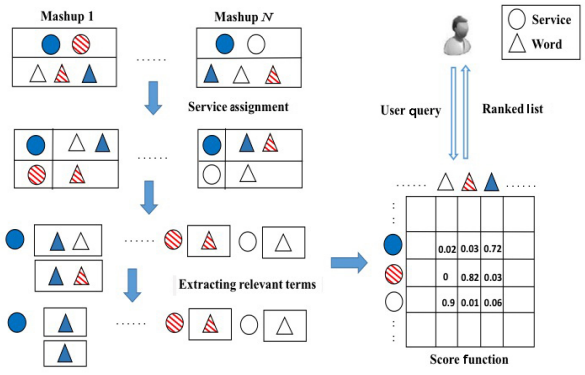

Fig. 1 Overview of the proposed service recommendation framework.

relevant terms from mashup descriptions for services, we propose to generate a coarse-grained representation first and elaborate on it in the second step.

Specifically, we conduct service assignment term by term through a linear discriminant function, which is designed based on the following three assumptions:

(1) Service assignment for terms is restricted to its corresponding component services.

(2) A word is more likely to be assigned to services containing it in the profiles.

(3) The more times a word co-occurs with a service in historical mashups, the more likely it is to be assigned with that service.

Next, we give several mathematical notations to make our statement concise. $I(s,m)$ is 1 if service s belongs to CS_m and 0 otherwise. Similarly, $I(w,s)$ is 1 if word w belongs to SP_s and 0 otherwise. We define $\mathrm{tf}(w,m)$ as term frequency of word w in MD_m. Therefore the co-occurrence times of service s with word w in all historical mashups $\mathrm{co}(s,w)$ can be calculated as follows:

$$\mathrm{co}(s,w) = \sum_{m \in M} I(s,m) \cdot \mathrm{tf}(w,m) \tag{1}$$

Based on our assumptions, service assignment $x(w)$ for word w in mashup m can be determined by the following equation:

$$x(w) = \arg\max_{s \in \mathrm{CS}_m} \lambda I(w,s) + (1-\lambda)\frac{\mathrm{co}(s,w)}{\sum_{s \in S} \mathrm{co}(s,w)} \tag{2}$$

where $0 \leqslant \lambda \leqslant 1$ is introduced to control the dependence on service profiles.

When service assignment for all terms is completed, we combine terms assigned to the same service in a single mashup description to form a separate developer comment on the service. Coarse-grained representation for a service can be derived by aggregation of corresponding developer comments.

3.2 Relevant terms extraction

Obviously, not all terms in the coarse-grained representation is relevant to the corresponding service. Therefore we further propose a novel probabilistic topic model to identify relevant terms for services from the developer's comments.

Specifically, we assume terms in each developer comment on a particular service are generated from two topics: (1) a common topic reflecting features of the service and (2) a comment-specific topic reflecting features of the originated mashup. The first topic is responsible for relevant terms while the second topic

would likely generate noisy words. Table 1 summarizes the notations used in this model.

The generative process of developer comments is formally described as follows:

For each service $s \in S$
  Draw $\phi_s \sim$ Dirichlet(β)
  For each developer comment d on s
    Draw $p_d \sim \beta(a, b)$
    Draw $\phi_d \sim$ Dirichlet(β)
    For each word w_{di}
      Draw a switch $y_{di} \sim$ Binomial(p_d)
      If y_{di}=1
        Draw a word $w_{di} \sim$ Multinomial(ϕ_s)
      Else
        Draw a word $w_{di} \sim$ Multinomial(ϕ_d)

The graphical model corresponding to this process is shown in Fig. 2. As we can see, w_{di} are observable variables and denoted by shaded circles, while a, b, and β are pre-defined hyper-parameters in the model. Others are latent variables and we seek to estimate them using Gibbs sampling[10].

Specifically, we sample the switch variable y_{di} for word token w_{di} in developer comment on service s according to the following equation:

$$p(y_{di} = 1 | w_{di} = j, s, \cdot) \propto \frac{n_{d1}^{\neg i} + b}{N_d - 1 + a + b} \cdot \frac{c_{sj}^{\neg i} + \beta}{\sum_v (c_{sv}^{\neg i} + \beta)} \tag{3}$$

$$p(y_{di} = 0 | w_{di} = j, s, \cdot) \propto \frac{n_{d0}^{\neg i} + a}{N_d - 1 + a + b} \cdot \frac{c_{dj}^{\neg i} + \beta}{\sum_v (c_{dv}^{\neg i} + \beta)} \tag{4}$$

Table 1 Notations used in the topic model.

Symbol	Description
d	Developer comment on a service
N_d	Number of word tokens in d
w_{di}	The i-th word token in d
y_{di}	Switch variable assigned with w_{di}
p_d	The parameter of Binomial distribution specific to d
ϕ_s	The parameter of multinomial distribution over words specific to service s
ϕ_d	The parameter of multinomial distribution over words specific to comment d
β	The parameters of Dirichlet priors to the multinomial distribution ϕ_s and ϕ_d
a, b	The parameters of Beta priors to the binomial distribution p_d

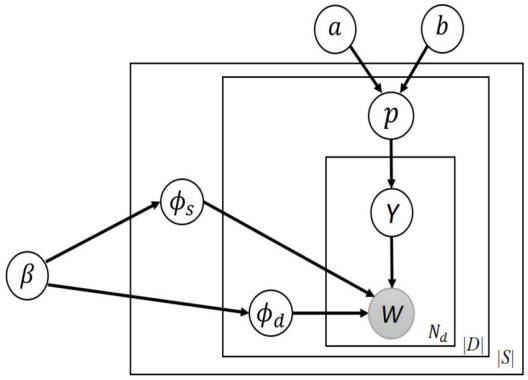

Fig. 2 Graphical model of relevant terms extraction.

where n_{d1} is the number of tokens whose switch variable has been set to 1 in comment d and n_{d0} is the number of tokens whose switch variable has been set to 0 in comment d. Also, c_{sv} is the number of times that word v has been assigned to feature topic of service s and c_{dv} is the number of times that word v has been assigned to specific topic of comment d. The superscript "¬" denotes a quantity excluding the current instance.

After sampling sufficient iterations, we can get a reliable estimation of the latent variables. To extract relevant terms for services, we preserve word tokens with their corresponding switch variables equal to one and drop the others. Finally, we generate high-quality service representations by aggregation of the relevant terms in all developer comments.

3.3 Recommendation algorithm

Based on the high-quality service representation, we now present our service recommendation algorithm for mashup creation.

Mathematically, we denote all service representations by a matrix $\boldsymbol{R}$ where each entry $R(s, w)$ is equal to the term frequency of word w in representation of service s. We observe that the distribution of term frequency over services can be leveraged as a good measure of both relevance and popularity. Firstly, the more relevant a term is to a service, it is more likely to co-occur with the service in developer comments and corresponding term frequency is higher. In addition, the more popular a service is (invoked by more historical mashups), it has a greater chance to be assigned with terms and thus corresponding term frequency is higher. Similar conclusion can be drawn for popular words. Therefore we propose to employ a distribution of the term frequency over services to design a score function

for service ranking.

Specifically, the distribution of term frequency over services for each word w is calculated as follows:

$$p(s|w) = \frac{R(s,w)}{\sum_{k \in S} R(k,w)} \tag{5}$$

Given a collection of words as a user query Q, we calculate the matching score between service s with respect to Q by accumulating the contribution of every word in the query:

$$f(s,Q) = \sum_{w \in Q} p(s|w) \tag{6}$$

Services with higher scores are more likely to satisfy the needs of query user. Therefore, we generate a ranked list of services in descending order of the matching scores in response to the requesting user.

4 Experiments

4.1 Dataset construction

Evaluation of the proposed model ERT is conducted on a real-world dataset from ProgrammableWeb.com, which is the largest mashup platform so far[11].

We crawled from the website the metadata of services and mashups ranging from September 2005 to July 2013. Every mashup contains a list of component services and is associated with natural language text to describe its details. Mashup name and tags are also combined into corresponding description text to enhance its semantics. Similarly, each service has a bag of words to describe its functionality and features (i.e., service profiles). Before the raw texts can be used in our experiments, several natural language preprocessing tasks need to be completed. In this paper, we employ a similar approach proposed in Ref. [12] as follows:

(1) **Tokenization**. As an initial treatment, we tokenize the unstructured description text by means of a separator.

(2) **Filtering**. In the second step, we remove stop words that are not helpful for description.

(3) **Suffix Striping**. Thirdly, we get stem words by stripping suffixes from words. For example, *notify*, *notification*, and *notifications* will be replaced with the same stem *notify*.

(4) **Error Correction**. Meaningful special characters filtered by standard parsing tools are recovered. Examples include *EC2* and *S3* in Amazon APIs.

Table 2 summarizes the basic properties of the final data set. To evaluate the performance of our model, the whole data set is divided into training set and testing set at four points along the timeline. As shown in Fig. 3, data before the dividing point is used for training and the remaining for performance testing. Therefore we obtained four test cases denoted from D1 to D4. Mashup description in the testing set is regarded as developer query and the corresponding component services as the ground truth.

Table 2 Basic properties of ProgrammableWeb data set.

Number of services	7077
Number of mashups	6594
Average number of component services per mashup	2.1
Number of unique words	13 648
Number of word tokens in all service profiles	33 531
Number of word tokens in all mashup descriptions	80 286

4.2 Evaluation metrics

The evaluation of the component service recommendation is based on four metrics, namely Mean Average Precision (MAP), Recall (Rec@K), Precision (Pre@K), and F1 measure (F1).

MAP measures ranking accuracy of component services in the recommendation list presented to the requesting user. The equation for calculating MAP is described as follows:

$$\text{MAP} = \frac{1}{|\text{CS}_m|} \sum_{s \in \text{CS}_m} \frac{\text{top}(s,m)}{\text{rank}(s,m)} \tag{7}$$

where $\text{rank}(s,m)$ is the ranking of service s in the recommendation list for a testing mashup m and $\text{top}(s,m)$ is the number of hits in top $\text{rank}(s,m)$ of the recommendation list. Higher MAP values indicate better performance.

Another metric Rec@K is the proportion of component services that are hit in the Top-K recommendations. Pre@K is the proportion of Top-K recommendations that are hit. Calculation of Rec@K

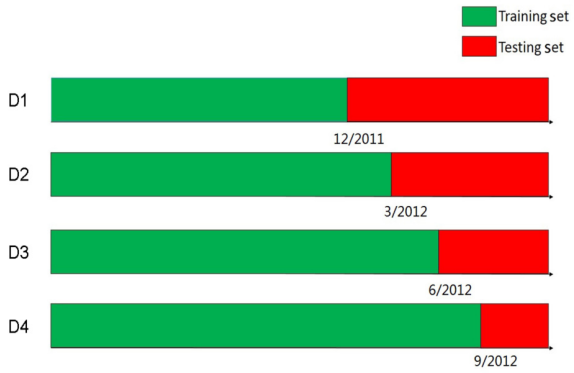

Fig. 3 Illustration of training and testing data sets.

and Pre@K is presented in the following equations, respectively:

$$\text{Rec@K} = \frac{|\text{TK}(m) \bigcap \text{CS}_m|}{|\text{CS}_m|} \tag{8}$$

$$\text{Pre@K} = \frac{|\text{TK}(m) \bigcap \text{CS}_m|}{|\text{TK}(m)|} \tag{9}$$

where TK(m) represents Top-K services recommended for mashup m. We fix K at a reasonable value 5 in this paper, since the average number of component services per mashup is around 2. With recall and precision, the final metric F1 can be calculated as follows:

$$\text{F1} = \frac{2\text{Pre@5} \cdot \text{Rec@5}}{\text{Pre@5} + \text{Rec@5}} \tag{10}$$

4.3 Comparison methods

We compare the following methods with the proposed model ERT for service recommendation.

Term Frequency-Inverse Document Frequency (TF/IDF). Based on service profiles, a vector of term frequency weighted by inverse document frequency w_s is employed to represent a service[3]. The user query Q is represented as a vector of term frequency w_q and finally cosine similarity is used to score the matching degree between service and user query:

$$\text{tf\_idf}(s, Q) = \frac{w_s \cdot w_q}{\|w_s\| \|w_q\|} \tag{11}$$

Unigram Language Model (ULM). The baseline models each service profile by a multinomial distribution over terms $p(w|s)$ and maximum likelihood is used to obtain the parameter estimation[13].

$$p(w|s) = \frac{\text{tf}(w, s) + \alpha}{\sum\limits_{v} (\text{tf}(v, s) + \alpha)} \tag{12}$$

where $\text{tf}(w, s)$ is the term frequency of word w in service s and α is introduced to smooth zero probability values. Finally the query likelihood under profile of service s is calculated as a ranking score:

$$\text{ulm}(s, Q) = \prod_{w \in Q} p(w|s) \tag{13}$$

Latent Dirichlet Allocation (LDA). The baseline models service profiles by LDA[14] as stated in Ref. [4]. Latent semantic structures (i.e., topic distribution of services $p(k|s)$ and word distribution of topics $p(w|k)$) are leveraged to calculate the matching degree between services and user query Q:

$$\text{lda}(s, Q) = \prod_{w \in Q} \sum_{k} p(w|k) \cdot p(k|s) \tag{14}$$

Mashup Description based Collaborative Filtering (MDCF). Based on the assumption that similar mashups invoke similar component services, MDCF generates the recommendation list by the following equation:

$$\text{mdcf}(s, Q) = \sum_{m \in M} r(m, Q) \cdot I(s, m) \tag{15}$$

where $r(m, Q)$ represents similarity between historical mashup m and the user query Q. As stated in Ref. [15], we first employ LDA to model mashup descriptions and the query likelihood under mashup m is calculated based on latent topics as similarity measure.

Coarse-grained Service Representation (CSR). The baseline is a simplified version of the proposed model ERT where relevant terms extraction is eliminated and Eq. (5) is calculated based on coarse-grained service representation.

All baselines generate recommendation list in descending order of calculated scores and are evaluated under the optimal settings. Finally, we set up parameters of ERT. The parameter in service assignment for terms λ is tuned to be 0.2. For hyper-parameters in the topic model for relevant terms extraction, we empirically set $a = b = 1$ and $\beta = 0.01$. The number of iterations in Gibbs sampling N is set to 50. All experiments were conducted on a Core 2 Duo 3.00 GHz machine with 4 GB RAM.

4.4 Quantitative analysis

4.4.1 Recommendation performance

In this section, we compare the state-of-the-art methods with our approach on four test cases. MAP, Rec@5, Pre@5, and F1 of different methods are reported in Table 3.

As we can see, methods leveraging mashup descriptions (MDCF, CSR, and ERT) clearly outperform those profile-based approaches (TF/IDF, ULM, and LDA) in all evaluation metrics. The vocabulary gap between mashup developers and service providers is responsible for the large performance gap. Furthermore, the proposed model ERT and its simplified version CSR perform much better than MDCF. The reason is that the proposed methods can directly evaluate matching degrees at service level with the help of component service assignment while MDCF scores services using composition as a bridge. Last but not the least, ERT obtains significant improvement over its simplified version CSR in MAP by preserving relevant terms and discarding noisy words, which makes evaluation in Eq. (5) more accurate. The reason for the marginal improvement of ERT over CSR on

Table 3 Recommendation performance by different methods.

Dataset	Method	MAP (%)	Rec@5 (%)	Pre@5 (%)	F1 (%)
D1	TF/IDF	5	5	3	4
	ULM	13	18	13	15
	LDA	18	30	15	20
	MDCF	36	42	16	23
	CSR	43	58	22	32
	ERT	50	60	23	33
D2	TF/IDF	5	4	3	3
	ULM	14	17	11	13
	LDA	12	27	14	18
	MDCF	36	42	17	24
	CSR	42	56	21	31
	ERT	50	58	22	32
D3	TF/IDF	4	3	2	2
	ULM	12	16	11	13
	LDA	12	30	16	21
	MDCF	38	44	18	26
	CSR	42	56	20	29
	ERT	49	57	21	31
D4	TF/IDF	6	5	3	4
	ULM	14	28	10	15
	LDA	13	17	16	16
	MDCF	40	46	21	29
	CSR	45	58	26	36
	ERT	51	59	27	37

Rec@5, Pre@5, and F1 is two-fold. On one hand, CSR already achieves very good performance; on the other hand, Rec@5, Pre@5, and F1 do not differentiate rankings of true items in the recommendation lists, which is contrary to MAP.

4.4.2 Impact of λ

Now we further study how parameter λ in service assignment for terms influences the recommendation performance of our model. Figure 4 shows the MAP of CSR and ERT on D1 with λ increased from 0 to 1 with a step length of 0.1. All other parameters are fixed as described in Section 4.3.

It is obvious from Eq. (2) that higher λ means more trust on service profiles and less belief on co-occurrence in mashup descriptions. From Fig. 4, MAP is modest when we depend solely on co-occurrence in mashup descriptions (i.e., λ=0). However, MAPs of both ERT and CSR rise steadily with λ increased from 0 to 0.2 and reach maximum at 0.2, which explains our choice for λ in experimental settings. Beyond that, MAP of ERT declines as λ proceeds to 0.9 while CSR remains unchanged. Finally, MAPs of the two

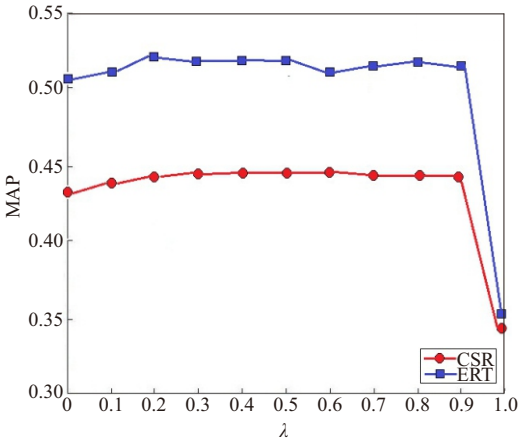

Fig. 4 MAP of ERT and CSR with λ varied.

algorithms decrease significantly when $\lambda=1$, which means that service assignment based entirely on service profiles is insufficient. For example, *tweet* is a feature word of Twitter API while it is not contained in the corresponding service profile. Experiments show that terms of *tweet* are heavily misclassified when $\lambda=1$. However, with the help of co-occurrence information in mashup descriptions by multiple developers, the problem can be greatly alleviated.

4.4.3 Impact of N

In this section, we evaluate how the number of Gibbs sampling iterations, N, influences the recommendation performance of ERT. To study the effect, MAP of ERT on D1 is presented in Fig. 5 with N varied from 10 to 50 with a step value of 10. All other parameters are fixed as described in Section 4.3.

From Fig. 5, MAP of ERT can achieve modest performance in just 10 iterations and the largest difference is less than 1.5%. The experiments prove that our model displays good convergence. The value of MAP is highest when N equals 50, which explains

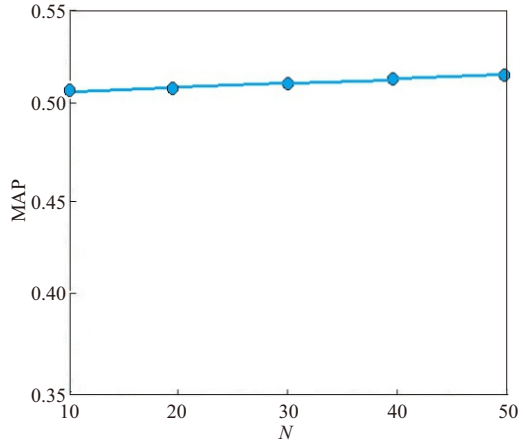

Fig. 5 MAP of ERT with N varied.

our choice for N in the experimental settings.

4.5 Qualitative analysis

In this section, we take an additional step to examine the effect of relevant terms extraction proposed in Section 3.2. The word *translate* is used as an example. We applied our method on D1 with the parameters fixed as previously defined. Table 4 lists the Top-5 services in term frequency distribution based on coarse-grained and high-quality service representation, respectively.

From Table 4, the term counts of services relevant to *translate* (Google Translate and Google Ajax Language) are almost unchanged after relevant terms extraction. In contrast, the counts of irrelevant services (Twilio, Google Maps, and Gravatar) are cleared or halved by the same process. All these efforts contribute to a more accurate estimation of $p(s|\text{translate})$ in Eq. (5) and therefore a higher MAP. From the discussion, we can conclude that the topic model proposed in Section 3.2 for extracting relevant terms is effective.

5 Related Work

In this section, we discuss several representative works related to our study. These works can be divided into three categories based on their methods: functionality-based, Quality of Service (QoS)-based, and network-based service recommendation.

Early works on functionality-based service recommendation mainly employed techniques from information retrieval to calculate the matching degree of functionality between services and user queries[3]. In Ref. [3], service and query are represented by vector of words extracted from WSDL documents and cosine similarity is employed to measure their relevance. However, the method relies on exact term matching between query and services, leading to its limited performance. Meng et al.[16] proposed to characterize a service user by keywords and presented an algorithm based on collaborative filtering for service recommendation. The limitation of this approach is that it requires great human effort to offer high-quality domain knowledge. To overcome the drawbacks of keyword-based methods, Li et al.[4] introduced LDA to model service descriptions extracted from WSDL documents and topic-level semantics were explored to enhance the evaluation of their relevance. Another group of researchers focused on ontology-based approaches. Hobold and Siqueira[17] extended WSDL documents to annotate services by SAWSDL and proposed a graph-based method for service composition. However, ontology construction usually calls for human effort and is time-consuming due to its high complexity. Moreover, as the popularity of REST-ful APIs and unstructured service description, it is difficult to obtain WSDL documents[9]. Recently, several approaches have been proposed to incorporate service usage history into profile-based recommendation[5, 6]. Liu and Fulia[5] followed the idea of collaborative topic regression and combined user-service matrix and service descriptions into a unified latent factor model. In Ref. [6], service correlation is discovered from co-occurrence of services in mashups and incorporated as a regularization term into the traditional matrix factorization framework for mashup-service usage. Both methods utilize service usage history but ignore mashup descriptions. Therefore these methods are limited to in-matrix prediction and suffer from the cold-start problem for new mashup. In contrast, our model leverages both mashup-service usage history and mashup descriptions for a new service representation, which alleviates the cold-start problem and fills the vocabulary gap.

Much research is related to the QoS-based recommendation. Collaborative filtering is introduced in Ref. [18] for missing QoS prediction. Top similar services are identified and their QoS values are aggregated as prediction. Service compositions with optimal QoS is also widely studied. Recently, Zhang et al.[19] proposed an improved Fruit Fly Optimization Algorithm to solve the problem with enhanced global searching ability. Reputation is an important QoS factor and emerged as a hot topic in recent years. Wu et al.[20] proposed a dynamic weight formula to calculate service reputation from historical ratings and unfair ratings were removed leveraging the idea of olfactory fatigue phenomenon. Fan et al.[21] employed LDA to cluster services into different domains and recommended services with the highest reputation in each domain. QoS-based recommendation centers on non-functional

Table 4 Case study on relevant terms extraction.

	Term count of services	
	Coarse-grained	High-quality
Google Translate	36	36
Google Ajax Language	15	14
Twilio	8	0
Google Maps	2	0
Gravatar	2	1

properties of web services and is different from our focus.

There are several methods that exploit social networks for service recommendation. Zhang et al.[22] studied people-service-workflow social network from myExperiment.org and proposed a recommendation algorithm for workflow composition by employing service correlations in the network. The method does not take content information into consideration and therefore fails to respond to the user query. Maaradji et al.[23] proposed to derive implicit social networks from user's composition and consumption activities, based on which user's social proximity and relevance of services are combined to determine the recommendations. However, it focused on recommendation for service users, not for mashups.

6 Conclusion

In this paper, we proposed a two-step approach to derive high-quality service representations from mashup descriptions. Based on the coarse-grained representation from component service assignment in the first step, a novel probabilistic topic model is further applied to extract relevant terms and filter out irrelevant ones. Finally, we design a score function based on the derived high-quality representation to evaluate the matching degree between services and queries to generate the recommendation list. Experiments on the dataset from ProgrammableWeb.com demonstrate the effectiveness of the proposed model.

For future work, we plan to integrate component service assignment into the proposed topic model framework for a better representation.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China (No. 61673230).

References

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

[2] M. Weiss and G. R. Gangadharan, Modeling the mashup ecosystem: Structure and growth, *R& D Management,* vol. 40, no. 1, pp. 40–49, 2010.

[3] C. Platzer and S. Dustdar, A vector space search engine for web services, in *third European Conference on Web Services (ECOWS'05)*, Vaxjo, Sweden, 2005, pp. 62–71.

[4] C. Li, R. Zhang, J. Huai, X. Guo, and H. Sun, A probabilistic approach for web service discovery, in *2013 IEEE International Conference on Services Computing,* Santa Clara, CA, USA, 2013, pp. 49–56.

[5] X. Liu and I. Fulia, Incorporating user, topic, and service related latent factors into web service recommendation, in *2015 IEEE International Conference on Web Services*, New York, NY, USA, 2015, pp. 185–192.

[6] L. Yao, X. Wang, Q. Sheng, W. Ruan, and W. Zhang, Service recommendation for mashup composition with implicit correlation regularization, in *2015 IEEE International Conference on Web Services*, New York, NY, USA, 2015, pp. 217–224.

[7] X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang, Similarity search for web services, in *Proceedings of the* 13*th International Conference on Very Large Data Bases,* Toronto, Canada, 2004, pp. 372–383.

[8] B. Xia, Y. Fan, C. Wu, B. Bai, and J. Zhang, A method for predicting service deprecation in service systems, *Tsinghua Science and Technology*, vol. 22, no. 1, pp. 52–61, 2017

[9] A. Jain, X. Liu, and Q. Yu, Aggregating functionality, use history, and popularity of APIs to recommend mashup creation, in *International Conference on Service-Oriented Computing*, 2015, pp. 188–202.

[10] I. Porteous, D. Newman, A. Ihler, A. Asuncion, P. Smyth, and M. Welling, Fast collapsed gibbs sampling for latent dirichlet allocation, in *Proceedings of the* 14*th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* Las Vegas, NV, USA, 2008, pp. 569–577.

[11] K. Huang, Y. Fan, and W. Tan, An empirical study of programmable web: a network analysis on a service-mashup system, in *2012 IEEE 19th International Conference on Web Services,* Hawaii, HI, USA, 2012, pp. 552–559.

[12] B. Xia, Y. Fan, W. Tan, K. Huang, J. Zhang, and C. Wu, Category-aware API clustering and distributed recommendation for automatic mashup creation, *IEEE Transactions on Services Computing,* vol. 8, no. 5, pp. 674–687, 2015.

[13] C. Zhai and J. Lafferty, A study of smoothing methods for language models applied to ad hoc information retrieval, in *Proceedings of the* 24*th ACM SIGIR International Conference on Research and Development in Information Retrieval,* New Orleans, IL, USA, 2001, pp. 334–342.

[14] D. M. Blei, A. Y. Ng, and M. I. Jordan, Latent dirichlet allocation, *Journal of Machine Learning Research,* vol. 3, pp. 993–1022, 2003.

[15] B. Bai, Y. Fan, K. Huang, W. Tan, B. Xia, and S. Chen, Service recommendation for mashup creation based on time-aware collaborative domain regression, in *2015 IEEE International Conference on Web Services,* New York, NY, USA, 2015, pp. 209–216.

[16] S. Meng, W. Dou, X. Zhang, and J. Chen, KASR: A keyword-aware service recommendation method on MapReduce for big data application, *IEEE Transactions on Parallel and Distributed Systems,* vol. 25, no. 12, pp. 3221–3231, 2014.

[17] G. C. Hobold and F. Siqueira, Discovery of semantic web services compositions based on sawsdl annotations, in *2012 19th IEEE International Conference on Web Services,* Hawaii, HI, USA, 2012, pp. 280–287.

[18] Z. Zheng, H. Ma, M. R. Lyu, and I. King, QoS-aware web service recommendation by collaborative filtering, *IEEE Transactions on Services Computing,* vol. 4, no. 2, pp. 140–152, 2011.

[19] Y. Zhang, G. Cui, Y. Wang, X. Guo, and S. Zhao, An optimization algorithm for service composition based on a improved FOA, *Tsinghua Science and Technology,* vol. 20, no. 1, pp. 90–99, 2015.

[20] Y. Wu, C. Yan, Z. Ding, G. Liu, P. Wang, C. Jiang, and M. Zhou, A novel method for calculating service reputation, *IEEE Transactions on Automation Science and Engineering,* vol. 10, no.3, pp. 634–642, 2013.

[21] Y. Fan, K. Huang, W. Tan, Y. Zhong, J. Yao, S. Nepal, and S. Chen, Domain-aware reputable service recommendation in heterogeneous manufacturing service ecosystem, *International Journal of Computer Integrated Manufacturing,* vol. 28, no. 11, pp. 1178–1195, 2015.

[22] 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 *2011 IEEE International Conference on Services Computing,* Washington, DC, USA, 2011, pp. 48–55.

[23] A. Maaradji, H. Hacid, R. Skraba, A. Lateef, J. Daigremont, and N. Crespi, Social-based web services discovery and composition for step-by-step mashup completion, in *2011 IEEE International Conference on Web Services,* Washington, DC, USA, 2011, pp. 700–701.

Yushun Fan received the PhD degree in control theory and application from Tsinghua University, China, in 1990. He is currently a professor with the Department of Automation, Director of the System Integration Institute, and Director of the Networking Manufacturing Laboratory, Tsinghua University. From 1993 to 1995, he was a visiting scientist, supported by Alexander von Humboldt Stiftung, with the Fraunhofer Institute for Production System and Design Technology (FHG/IPK), Germany. He has authored 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.

Yang Zhong received the BS degree in 2012 from Tsinghua University, China. He is currently a PhD candidate at the Department of Automation, Tsinghua University. His research interests include services computing, service recommendation, and big data.