Domain-aware Service Recommendation for Service Composition

Bofei Xia, Yushun Fan*, ChengWu, Keman Huang
Tsinghua National Laboratory for Information Science and Technology, Department of Automation, Tsinghua University
Beijing 100084, China

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

Jia Zhang
Carnegie Mellon University
Silicon Valley
jia.zhang@sv.cmu.edu

Bing Bai
Tsinghua National Laboratory for Information Science and Technology, Department of Automation, Tsinghua University
Beijing 100084, China
bj13@mails.tinghua.edu.cn

***Abstract*—Service compositions inherently require multiple services each with its domain-specific functionality. Therefore, how to mine matching patterns between services in relevant domains and compositions becomes crucial to service recommendation for composition. Existing methods usually overlook domain relevance and domain-specific matching patterns, which restrict the quality of recommendations. In this paper, a novel approach is proposed to offer domain-aware service recommendation. First, a K Nearest Neighbor variant (vKNN) based on topic model Latent Dirichlet Allocation (LDA) is introduced to cluster services into semantically coherent domains. On top of service domain clustering results by vKNN, a probabilistic matching model Domain Router (DR) based on Extreme Learning Machine (ELM) is developed for decomposing a requirement to relevant domains. Finally, a comprehensive Domain Topic Matching (DTM) model is built to mine relevant domain-specific matching patterns to facilitate service recommendation. Experiments on a large-scale real-world dataset show that DTM not only gains significant improvement at precision rate but also enhances the diversity of results.**

Keywords—Service recommendation, LDA topic model, Domain-aware Service Clustering, Extreme Learning Machine, Domain-specific matching pattern

I. INTRODUCTION

Service composition plays a key role in services computing and how to facilitate the construction of service composition have attracted significant attentions [1, 13]. In recent years, service recommendation has been proved to be a promising solution to support effective and efficient service composition [2, 16]. With the prevailing of RESTful services, many online repositories allow service providers to describe services functionality in plain text (e.g., descriptive text or tag). When a developer gives a composition requirement also in text, many existing methods directly search the entire set of services in a repository and recommend services that may satisfy the functionality needs of the given requirement [3]. Various non-functional QoS features are also taken into account in service recommendation [9].

However, these methods oftentimes overlook the fact that some services in a repository have similar functionalities and form an implicit but inherent service domain [15]. For example, Map services such as *Google Map* and *Bing Map* form a "map" domain, and *Facebook*, *Twitter*, etc. form a "social network" domain. Such domains delimitate semantically coherent service groups. Moreover, popular services from different domains become popular for different and domain-specific reasons. For example, *Google Map* becomes popular in domain 'Map' for its rich and accurate map information, while *Twitter* is popular in domain 'Social' because of its attractive user interface. These domain-specific, non-functional criteria determine the popularity of services, and are embedded in historical usage information. We name them domain-specific matching patterns. Therefore, services in each individual domain have similar functionalities and enjoy their own matching patterns regarding composition. A previous study has suggested that such domain-specific hidden information may further enhance the performance of service recommendation [11].

In many repositories, e.g., ProgrammableWeb[1] or PW for short, services are organized in *categories* such as advertising, enterprise and file sharing. This categorization is helpful in searching for services; however, it is currently realized through a manual process and no domain-specific matching pattern is derived to help service recommendation. Furthermore, developers are usually not able to explicitly express the relevant domains in the given requirements. Therefore, without any knowledge about these domains, most existing works had to mine the matching patterns between the entire set of services and compositions, taking into no consideration of the domain-specific matching patterns. This largely restricts the quality of recommendation results.

In this paper, we propose a three-step approach to overcome the aforementioned restrictions and offer domain-aware service recommendation for solution construction.

First, a domain-aware service clustering method based on a *K Nearest Neighbor variant* (*vKNN*) is introduced to extract features of services and further cluster them into different domains. vKNN leverages both content (the topic model Latent Dirichlet Allocation (LDA) [4]) and popularity (services' historical usage information), which differentiate it from traditional clustering methods. This step provides a basis for mining domain relevance and domain-specific matching patterns in subsequent steps.

Second, on top of vKNN, a probabilistic matching model - *Domain Router* (*DR*) model - is designed to decompose a textual requirement to relevant domains. Based on LDA and Extreme Learning Machine (ELM) [5], DR first transforms a

[1] http://www.programmableweb.com/
*Yushun Fan is the Communication Author : fanyus@tsinghua.edu.cn

textual requirement description to requirement topic features. Afterward it analyzes the functional requirements involved in the previously derived topic features, and predicts which domains are relevant to the requirement.

Finally, based on the proposed vKNN and DR, a comprehensive *Domain Topic Matching* (*DTM*) model is presented to enable and facilitate domain-aware service recommendation. The recommendation results are represented in the form of *domain relevance ranking order* together with *per domain recommendation list,* which considers both candidate domains' relevance and favorable services in each domain.

Our experiments on PW data set show that DTM gains a 30% improvement at overall precision rate, compared to state-of-the-art functionality or popularity based approaches that ignore domain knowledge. Moreover, DTM also prevails compared to the recommendation based on the original service categorization offered by PW, mainly due to the advantage of vKNN over PW's (manual) categorization. DTM also achieves a 20% improvement for long tail recommendation, i.e., recommending the not-so-popular services, which enhances the diversity of recommendation results [8]. The superiority of DTM is a result of our mining and exploiting of domain-aware knowledge, and our consideration of historical usage. The remainder of the paper is organized as follows. Section 2 illustrates the overall methodology; Section 3 presents the recommendation model and algorithms; Section 4 presents the experiments and results on a real-world dataset; Section 5 discusses the related work and Section 6 draws a conclusion.

II. Overview of Methodology

The overview of our proposed methodology is illustrated in Figure 1. Aiming to mine domain-specific service matching patterns to support service composition, our methodology is divided into an offline service clustering phase and an online service recommendation phase.

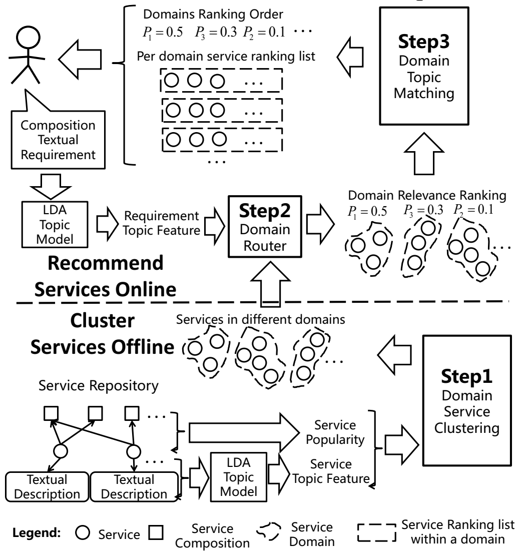

Fig. 1. Methodology of domain-aware service recommendation for composition. P_d denotes the relevance probability of domain *d*.

The offline phase provides a basis for domain-aware recommendation. A Domain Service Clustering method takes as input both service topic features (*captured from services' textual description by the LDA topic model*) and service popularity (*extracted from service historical usage records by composition*), and outputs services clustering results. Within each resulting domain, services have similar functionality and share the same matching pattern.

In the online recommendation phase, when received a developer's textual requirement (in form of description text, tag, etc.), the first task is to decompose the requirement into relevance domains. The Domain Router takes the textual requirement as input, and predicts the relevance probabilities of each domain, leading to a domain relevance ranking order. Afterwards, a Domain Topic Matching method considers both the domain relevance order and domain-specific matching patterns, and returns "Per domain service ranking list." In this way, the developer is informed "which domains are potential relevant to your requirement" and "which services may satisfy your needs in each domain." Such a user-friendly form of recommendation makes it easier for developers to select proper services to satisfy a certain aspect of need.

III. Modeling and Algorithms

As shown in Figure 1, our methodology introduces a three-step approach to realize domain-aware service recommendation for composition: a *K Nearest Neighbor variant* (*vKNN*) method for Domain-aware Service Clustering (section A), a *Domain Router* (*DR*) model for domain relevance ranking (section B), and a *Domain Topic Matching* (*DTM*) model for "per domain service ranking list" (section C).

A. Domain-aware Service Clustering

1) Clustering motivation

The first step is to automatically cluster services into semantically coherent domains. According to our empirical study [1, 6] of service usage patterns, from each functionality aspect, there may exist one core service that gains the most popularity. Most services added later into the repository usually follow one of the core services and its functionality. Therefore, the traditional KNN that only considers the functionality similarity is insufficient for domain service clustering. Here we introduce a two-phase clustering strategy, named K-Nearest-Neighbor variant (vKNN) to group services. In vKNN, both *functionality* and *popularity* of services are taken into account.

2) Information gathering for clustering

A service's functionality can be extracted from its textual material (description text, service tag, etc.). Similar to our previous work [15], we use a probabilistic topic model LDA [4] to map the service's functionality to a fixed-length vector named 'topic feature' vector. Every element in the topic feature vector ranges from 0 to 1, which represents the probability of whether a service is relevant to a certain functionality (or topic). The summation of all the elements of a topic feature vector equals 1. We define a ***STF*** matrix (defined in Table I) to formally represent the topic features of all services.

A service's popularity is measured by the number of compositions that invoke the service. By analyzing compositions in a repository, we can obtain all the services' popularity information from their historical usage. Similar to [14], we formalize the historical usage information in a ***CS*** matrix (defined in Table I). The summation of the *i*th column of ***CS*** is assigned as the popularity of service *i* (**Algorithm 1**, line 1-3).

3) *Two-phase clustering strategy*

The proposed vKNN clustering method contains two phases. In the first phase, we extract *core services* in the repository considering both its topic feature similarity and popularity. In the second phase, we firstly rank all the non-core services by their popularity. Similar to the KNN method, we cluster each service to one domain considering its topic distribution's Kullback Leibler (KL) distance [7]. With all involved notions listed in Table I, we describe the pseudo code of our vKNN clustering algorithm in **Algorithm 1**.

TABLE I. NOTIONS IN DOMAIN SERVICE CLUSTERING

N	The total # of services
K_s	The total # of topics on service side
S_i	Service *i* in repository
STF	$N \times K_s$ matrix, *STF*(*i*,*k*) represents the probability of topic *k* given S_i
C_i	Composition *i*
M	The total # of composition
CS	$M \times N$ matrix, *CS*(*i*,*j*) =1 if S_j is used by C_i ,otherwise *CS*(*i*,*j*) =0
D	The total # of service domains
D_d	Servic domain *d*
S_{di}	The *i*th service clustered in domain *d*
N_d	The # of services clustered in domain *d*
C_S	$1 \times D$ vector, $C_S(d)$ represents the core service in each domain *d*

Algorithm 1: variant K Nearest Neighbor clustering

```
Input: STF matrix, CS matrix
Output: $\{S_{di} : 1 \le d \le D, 1 \le i \le N_d\}$
1. For $i = 1$ to $N$
2.     $S_i$.populariry = sum(CS(:, i))
3. End For
4. $d = 1$
5. For $r = 1$ to $N$  //KLD(a,b) calculates KL Distance of a,b
6.     For $j = 1$ to $d$
7.        If KLD(the rth popular service, $C_S(j)$ ) $\ge$ threshold
8.              distribute rth popular service as $C_S$ (d++)
9.              Break
10.       End If
11.    End For
12.    Break if $d \ge D$
13. End For
14. For $i = 1$ to $N$
15.    If $S_i$ not in $C_S$
16.    For $d = 1$ to $D$
17.        calculate $KLD(S_i, \frac{1}{N_d}\sum_{Si \in D_d} STF(i,:))$
18.    End for
19.    Distribute $S_i$ to $D_d$ with minima KLD
20.    $N_d$ ++
21. End For
```

Lines 1-3 obtain the service popularity based on historical usage information. Lines 4-13 identify the core services in each domain according to their popularity and topic features. Afterwards, for each unclustered service, lines 16-18 caculate its average KL distance to the clustered services topic distribution in each domain. Finally line 19 assigns it to the domain to which it achieves a minima KL distance. After the iteration, all the services in the repository are clustered into a certain domain.

The main difference between vKNN and existing clustering methods is that vKNN considers service usage patterns extracted from historical records. vKNN first identifies some popular services with different topic features and assigns them as core services in different domains, before clustering the non-core services. In this way, services within each domain not only have relevant functionality but also share the same matching patterns regarding compositions. Thus, mining the *domain relevance with composition requirement* and *domain-specific matching pattern regarding composition* becomes possible.

B. Domain Router Model

1) *Model description*

After clustering all the services in domains, when receiving a composition textual requirement, the first task is to decompose the requirement and predict its relevance with each domain. This task is conducted by a *Domain Router* (*DR*) model.

The workflow in a DR is shown in Figure 2. The input of *DR* is the topic features of requirement captured by LDA; the output of DR is the relevance probabilities of each domain. The role of DR is to imitate the mapping function from requirement to domain relevance. Here, we adopt a machine learning method to learn the mapping function.

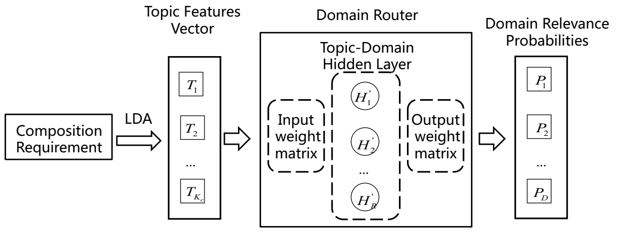

Fig. 2. Working process of Domain Router.

Combine historical usage information and results of domain-aware service clustering, we obtain historical dataset of which compositions use which domain's services. Formally, we define these information as a D_I matrix (illustrated in Table II). Similar to the ***STF*** matrix, we use

LDA to capture topic features of composition requirement from their textual descriptions (requirement text, composition tag, etc.) and formalize them as a ***CTF*** matrix (illustrated in Table II). Afterwards, the core task is turned to learn the implicit mapping relations between matrix ***CTF*** and D_I. We adopt the Extreme Learning Machine (ELM) [5], which provides fast learning speed and powerful learning scalability, to establish a 'Topic-Domain' hidden layer between them. The mapping relation is embedded in the input weight matrix I_R, the bias vector B_R, and the output weight matrix O_R. To avoid ambiguity, we list the notions used in DR in Table II.

TABLE II. ADDITIONAL NOTIONS USED IN DOMAIN ROUTER

D_I	$M \times D$ matrix indicating whether C_i used service in a certain domain: $D_I(i, d)=1$ if C_i uses at least a service in domain *d*; otherwise, $D_I(i, d)=0$
CTF	$M \times K_c$ matrix, *CTF*(*i*,*k*) represents the probability of topic *k* given composition C_i
T_i	Topic *i* on composition side
K_c	The total # of topics on composition side
R	The total # of hidden units in domain router
$H_i^{'}$	The *i*th unit of topic-domain hidden layer
I_R	$R \times K_c$ input weight matrix of *DR*
B_R	$R \times 1$ vector, bias of hidden layer units of *DR*
O_R	$R \times D$ output weight matrix of *DR*
C_Q	$1 \times K_c$ topic feature vector of a new requriement
P	$1 \times D$ domain relevance probability vector regarding new composition requirement
P_i	The relevance probability of domain *i*

2) *Model Training*

As aforementioned, the kernel task here is to learn the model parameters which include the input weight matrix I_R between composition topics feature vector and hidden layer, the bias vector B_R of hidden layer, and the output weight matrix O_R between hidden layer and the domain relevance probabilities result. After obtaining the ***CTF*** matrix and D_I matrix as training dataset, we follow the three training steps in [5]. The first step is to randomly assign I_R and B_R. Then calculate the hidden layer output ***HLOutput***. Finally, calculate the Moore-Penrose inverse matrix of ***HLOutput*** and obtain O_R.

Especially, we resort to the SVD approach for calculating *Moore-penrose* inverse matrix of ***HLOutput*** (**Algorithm 2**. line 15). The SVD result of ***HLOutput*** is illustrated as follows,

$$\boldsymbol{HLOutput} = U \Lambda V^H$$

where

$$\Lambda = \begin{pmatrix} \Lambda_1 & 0 \\ 0 & 0 \end{pmatrix}, \Lambda_1 = diag(\sigma_1, \sigma_2 ... \sigma_r) \qquad (1)$$

σ_r is the single value of matrix ***HLOutput***, *r*= rank(***HLOutput***) satisfying $\sigma_1 \geq \sigma_2 \geq ... \sigma_{r-1} \geq \sigma_r > 0$. Then the Moore-penrose inverse matrix of ***HLOutput*** can be obtained as follows,

$$H^+ = V \Lambda^+ U^H$$

where

$$\Lambda^+ = \begin{pmatrix} \Lambda_1^+ & 0 \\ 0 & 0 \end{pmatrix}, \Lambda_1^+ = diag(1/\sigma_1, 1/\sigma_2 ... 1/\sigma_r) \qquad (2)$$

We summarize the details of the model training in **Algorithm 2**.

Algorithm 2: DR paremeters inference

```
Input: CTF matrix, CS matrix,
       { $S_{di}: 1 \leq d \leq D, 1 \leq i \leq N_d$ }
Output: { $I_R, B_R, O_R$ }
1.  For i = 1 to M
2.      tempdI =zeros(1,D)  // initialize tempdI as 1×D
                            // zero matrix
3.      For d = 1 to D
4.          If $C_i$ used service in $D_d$
5.              tempdI(1,d) = 1
6.          End If
7.      End For
8.      $D_I$ (i,:) = tempdI
9.  End For
10. Initialize $I_R$ by random value ∈[-1, 1]
11. Initialize $B_R$ by random value∈[0, 1]
12. BExtend = $B_R$ (:,ones(M,1)) // extend $B_R$ to a
                              // $R \times M$ matrix by column
13. tempHLOutput = $I_R$ • CTF'+ BExtend
14. HLOutput = $e^{-tempHLOutput^2}$
15. $H^+$ = Moore-penrose inverse matrix of HLOutput
16. $O_R$ = $(H^+)'$ • $D_I$
17. Return $I_R$ matrix, $B_R$ vector, and $O_R$ matrix
```

Lines 1-9 combine historical usage information and results of domain-aware service clustering, find which domains are involved in compositions and organize the information into matrix D_I. Lines 10-17 follow the ELM training process and use matrix ***CTF*** and D_I to learn the parameter of DR.

3) *Domain Relevance Ranking*

When receiving a new composition requirement, we first use LDA to extract topic feature C_Q of the requirement. After inputting C_Q into trained DR, we obtain the relevance probabilities vector *P* at its output side. **Algorithm 3** shows the pseudo code of the Domain Relevance Ranking process of *DR*.

Algorithm 3: Domain Relevance Ranking

```
Input: New Composition requirement
       $I_R$ matrix, $B_R$ vector, $O_R$ matrix
Output: Relevance ranking order of domains
```

```
1. $C_Q \leftarrow$ new composition requirement
2. tempHLOutput = $I_R \bullet C_Q + B_R$
3. HLOutput = $e^{-tempHLOutput^2}$
4. $P$ = HLOutput' $\bullet$ $O_R$
5. For $d$ = 1 to $D$
6.     Ranking domain $d$ according to $P_i$
7. End For
```

Line 1 uses LDA to infer new composition requirement's topic features. Lines 2-4 use the trained DR model to predict the each domain's relevance probability. Lines 5-7 rank all the domains according to their relevance probability.

In contrast to the traditional keyword-based methods, DR uses LDA to transfer textual requirement to topic features. Furthermore, DR learns the historical 'requirement-domain' mapping relation at the topic level. Even though developers may not express the domain explicitly with keywords, DR can still predict relevant domains, which will facilitate service recommendation process.

C. Domain Topic Matching Model

1) Model description

Through DR, we can obtain the domain relevance ranking order. We have further designed a *Domain Topic Matching* (*DTM*) model, aiming to inform developers "which services may better satisfy your needs in each domain."

In DTM, each domain is assigned a uniform probabilistic model called *Topic Matching* (*TM*), which mines the implicit matching patterns of a certain domain regarding composition. Figure 3 illustrates how TM of domain *d* works. The input of TM is the topic features of requirement captured by LDA; the output of TM is the predicted service topic feature S_T^d (defined in Table III). Finally, all the services in domain *d* are ranked according to their topic feature's KL distance with S_T^d. In this process, TM of domain *d* imitates the matching patterns between topic features of requirement and topic features of services in domain *d*. We have developed a machine learning method to learn the matching patterns.

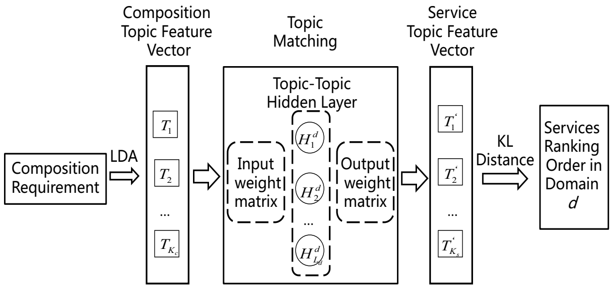

Fig. 3. Procedure for topic matching.

The core component of TM is an ELM-based "topic-topic" hidden layer. After learning from the historical usage information, the matching patterns of domain *d* are embedded in the input weight matrix I_d, the bias vector B_d, and the output weight matrix O_d. The notions used in DTM are listed in Table III.

TABLE III. ADDITIONAL NOTIONS IN DOMAIN TOPIC MATCHING

T_i'	Topic *i* on service side
H_i^d	The *i*th unit in topic-topic hidden layer of domain *d*
L_d	The # of units in hidden layer of domain *d*
I_d	$L_d \times K_c$ input weight matrix of domain *d*
B_d	$L_d \times 1$ vector, bias of hidden layer units of domain *d*
O_d	$L_d \times K_s$ output weight matrix of domain *d*
M_d	The # of composition using service of domain *d*
C_{TF}^d	$M_d \times K_c$ matrix of composition topic feature for training TM of domain *d*
M_{STF}^d	$M_d \times K_s$ matrix of service topic feature for traning TM of domain *d*
S_T^d	$1 \times K_s$ infered service topic feature vector of domain *d*

2) Model training

Using clustering results and the historical usage information, we can obtain the training data for TM of domain *d*: C_{TF}^d matrix at input side and M_{STF}^d matrix at output side. The parameters are learned following the ELM training process whose pseudo code is shown in **Algorithm 4**.

```
Algorithm 4: TM parameter inference
Input: STF matrix, CTF matrix, $D_I$ matrix
       $\{S_{di} : 1 \le d \le D, 1 \le i \le N_d\}$
Output: $\{I_d, B_d, O_d, 1 \le d \le D\}$
1. For $d$= 1 to $D$
2.     $M_d$ = sum($D_I$ (:, $d$))
3.     $M_{STF}^d = zeros(M_d, K_s)$
4.     $C_{TF}^d = zeros(M_d, K_c)$
5. End For
6. For $i$ = 1 to $M$
7.     For $d$ = 1 to $D$
8.         If $D_I$ ($i$, $d$) == 1
9.             $C_{TF}^d(end+1,:) \leftarrow$ CTF($i$,:)
10.            Randomly choose $S_{dj}$ used in $C_i$
11.            $M_{STF}^d(end+1,:) \leftarrow$ STF($j$,:)
12.        End If
13.    End For
14. End For
15. For $d$ = 1 to $D$
16.    Infer $\{I_d, B_d, O_d\}$ by $\{C_{TF}^d, M_{STF}^d\}$
17. End For
```

Lines 1-14 acquire the training data for TM of each domain. Lines 15-17 learn the parameters similar to **Algorithm 2**.

3) Domain Topic Matching based Recommendation

Via DR, the new requirement is transferred to C_Q. In this phase, inputting C_Q to TM of each domain and following the TM working process, we can acquire the service ranking order of each domain. Combining with domain relevance ranking order, the domain-aware recommendation result is obtained in **Algorithm 5**.

```
Algorithm 5: DTM service recommendation
Input: $C_Q, \{I_d, B_d, O_d, 1 \le d \le D\}$
Output: domain-aware recommendation result
1. For $d = 1$ to $D$
2.     $C_Q \rightarrow TM(I_d, B_d, O_d) \rightarrow S_T^d$
3.     rank all the services in domain $d$
4. End For
5. Return domain relevance ranking order & service
          ranking order within domain
```

Lines 1-4 obtain per domain per service ranking list: line 2 inputs C_Q to trained TM of domain *d* and gains S_T^d similar to the method in **Algorithm 3**. Line 3 ranks all the services in domain *d* according to their topic feature's KL distance with S_T^d. Line 5 recommends services combining domain relevance probability and services ranking order inside of each domain. For example, if there are 10 domains, DTM first places the first service in each domain at top 10 corresponding to the relevance order of their domain. Afterwards, DTM places the second service in each domain at 11~20 also corresponding to the relevance order of their domain, and so on.

IV. Experiments

To the best of our knowledge, ProgrammableWeb.com is by far the largest online repository of Web APIs (i.e., services) and their mashups (i.e., compositions). PW organizes Web APIs into different categories according to their functionalities. Therefore, we adopt it as our testbed. Note that we use the terms 'service' and 'composition' to denote Web APIs and mashups, respectively.

A. Data Set

We crawled the service and composition data from the ProgrammableWeb.com over the last 8-year time period (June 2005 to June 2013). In the data set, descriptive text of services and compositions consists of their *Description File*, *Service Tags* and *Summary*. After removing meaningless or vacant compositions and services, the data set can be profiled in Table IV.

TABLE IV. Data set on ProgrammableWeb.com

Total # of services	7,186
Total # of original service categories from PW	62
Total # of compositions	6,813
# of services used in at least one composition	1,155
Average # of services in one composition	2.075
Total # of terms in compostions corpus	205,494
Total # of terms in services corpus	350,101

B. Evaluation Metrics

1) MAP@N

Mean Average Precision @ top N services in ranking list is defined as follows:

$$MAP@N = \frac{\sum_{r=1}^{N}\left(\frac{N_r}{r} \bullet I(r)\right)}{N_{used}} \qquad (3)$$

Where N_r denotes the number of actually used services in the top *r* services of the ranking list, *I(r)* indicates whether the service at ranking position *r* is actually used and N_{used} represents the total number of actually used services in composition.

2) NDCG@K

Normalized Discounted Cumulative Gain @ top K services in ranking list is defined as follows:

$$NDCG@K = \frac{1}{N_K}\sum_{j=1}^{K}\frac{(2^{r(j)}-1)}{\log_2(1+j)} \qquad (4)$$

Where *r(j)* represents the relevant score {0,1} of the *j*th recommended service on the ranking list and N_K represents the ideal maximum score that the cumulative component can reach.

C. Baseline Methods

We chose four types of common recommendation methods as baseline methods: TF-IDF and LDA are functionality-based methods; PopK is popularity-based method; STM considers both functionality and popularity, without domain knowledge; OTM recommends services based on original service clutsering resultes offered by PW.

1) TF-IDF

For the TF-IDF method, we first obtain the Term Frequency-document Matrix (TDM) from the corpus, and transfer TDM to Tf-idf Weighted-Document Matrix (TWDM). Afterwards, services are recommended according to their tf-idf weight similarity with composition requirements measured by KL divergence as defined below:

$$D_{KL}(query \| service) = \sum_{i=1}^{N} TWDM(i,q)\log\frac{TWDM(i,q)}{TWDM(i,s)} \qquad (5)$$

Where *N* denotes the total number of terms, *TWDM(i,q)* denotes the tf-idf weight of the *i*th term given the *query* and *TWDM(i,s)* denotes the tf-idf weight of the *i*th term given a *service*.

2) LDA

For the LDA method, after training LDA model with the corpus, we infer the topic distribution of the composition query. Subsequently, we directly calculate the KL divergence between topic distribution of query and each service as follows:

$$D_{KL}(query \| service) = \sum_{i=1}^{K} P(i \mid query)\log\frac{P(i \mid query)}{P(i \mid service)} \qquad (6)$$

Where *K* denotes the total number of topics, *P(i|query)* denotes the probability of topic *i* given the query, *P(i|service)* denotes the probability of topic *i* given a service. Similarly, all the services are ranked by their topic distribution KL divergence with the query.

3) PopK

For the PopK method, similar to the approach in [18], we recommend the most popular services in each relevant service domain.

4) Single Topic Matching (STM)

STM is a special case of the proposed DTM. STM considers the entire set of services as a single domain, without considering domain-aware influence.

5) ***Original Topic Matching (OTM)***

The difference between *OTM* and *DTM* is that *OTM* recommends services based on the service categorization offered by PW rather than vKNN.

D. Experiment Results

1) Overall performance comparison

We consider the recommended services to be positive only if the services are actually used in the composition; otherwise, the recommended services are considered to be negative. After 10-fold cross validation, the MAP and NDCG of the six methods on varying sizes of recommendation list are shown in Figure 4. As illustrated, the DTM (fix number of domains at 10) method outperforms on both MAP and NDCG. The TF-IDF and LDA only calculate the functionality similarity between service and composition requirements, thus show a relatively poor performance. PopK performs relatively well because the services on this data set present a strong power-law distribution, which means that the popular services gain more preference from developers [15]. STM takes both functionality and popularity into account; however, without considering domain-specific matching patterns restricts its performance. The service categorization offered by PW takes no consideration of service popularity, which impacts on OTM. Taking advantage of domain-aware service clustering by vKNN, our DTM method considers both domain relevance and domain-specific matching pattern, thus results in a superior recommendation ratio.

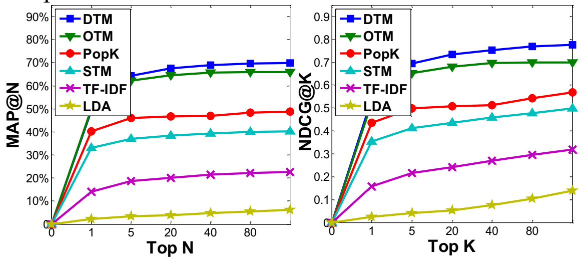

Fig. 4. MAP and NDCG of six methods on different sizes of recommendation list.

2) Long tail performance comparison

'Long tail' recommendation gains a lot of momentum in recent years [8]. According to our earlier empirical study [1, 6], the service usage pattern on the ProgrammableWeb.com shows a significant power-law distribution. By our study, nearly 5% popular services ever occurred in 78% compositions, and the other 95% services are in long tail. Therefore, we cannot ignore the large amount of services in long tail for their potential values. We report the recommendation performance on long tail services in Figure 5. In the context of service recommendation, 20 is a reasonable length of candidate service list. Thus, we compare the performance of MAP@20 and NDCG@20 for each method and report the results in Table V.

From Table V, it can be seen that PopK sharply decreases in the two metrics as it only recommends popular services but ignores services in long tail. OTM and STM are also subject to the power-law influence and shows poor performance in recommending services in long tail. On the contrary, TF-IDF does not consider service popularity, which in turn gets rids of the power-law influence and becomes better in long tail recommendation. LDA also decrease in both MAP and NDCG. Though DTM suffers a performance degradation, it still performs the best for the long tail recommendation. We conclude that domain-aware method can remedy the influence of power-law and contributes to a higher performance in long tail recommendation.

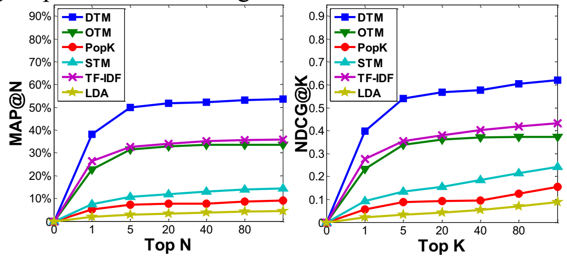

Fig. 5. MAP and NDCG of six methods on long tail recommendation.

TABLE V. THE COMPARISON BETWEEN **OVERALL** AND **LONG TAIL**

	MAP@20			***NDCG@20***		
	Overall	LT	variation	Overall	LT	variation
DTM	69.73%	53.21%	-16.52%	0.7681	0.6042	-0.1639
OTM	64.52%	32.94%	-31.58%	0.6807	0.3618	-0.3189
PopK	48.23%	8.59%	-39.64%	0.5422	0.1242	-0.418
STM	39.88%	13.89%	-25.99%	0.4769	0.215	-0.2619
TFIDF	22.23%	35.67%	13.44%	0.2954	0.4159	0.1205
LDA	5.4%	4.22%	-1.18%	0.1034	0.0702	-0.0332

3) Amount of service domains

The proposed DTM method provides a flexible mechanism adjustable to the amount of the service domains. In this section, we discuss the impact of the amount of domains. We examine the MAP@20 and NDCG@20 with varying amount of service domains shown in Table VI.

TABLE VI. IMPACT OF AMOUNT OF SERVICE DOMAINS

Amount Of Domains	***MAP@20***		***NDCG@20***	
	Overall	LT	Overall	LT
1	39.88%	13.89%	0.4769	0.215
5	52.4%	36.42%	0.5933	0.448
10	69.73%	53.21%	0.7681	0.6042
15	65.93%	57.46%	0.7167	0.632
20	63.41%	57.26%	0.6925	0.6212

In some range, DTM shows a general trend of better performance with more domains. Especially, DTM can significantly improve the precision rate in long tail recommendation. When there is only one group, the DTM is equivalent to STM and shows a relative low MAP (13.89%) and NDCG (0.215). When the amount of domains is 10 or larger, DTM mines the implicit usage patterns in long tail better. Thus, the MAP and NDCG of DTM are over 50% and 0.6, respectively. However, too many presetting domains may weaken too much the status of small number of popular services in domains. As a result, it leads to slight decreasing in MAP and NDCG when it comes to overall metric.

V. Related work

Service recommendation has gained significant momentum in recent years. Among them, trace norm regularized matrix factorization technique is adopted in [9] to predict QoS features to support service recommendation. A social-aware service recommendation approach for Mashup composition is proposed in [3], comprising a coupled matrix model and Latent Dirichlet Allocation (LDA) model. However, most existing works overlook domain knowledge at service recommendation [15], especially when domain information is not labeled explicitly by service providers.

Service clustering has been studied in the context of web service repository or ecosystem. An ontology-based service clustering method is adopted in [10] to improve service discovery. In [11], service-oriented categorization is addressed with an extended SVM-based text clustering technique. Zhou et al. [12] present an approach of leveraging and integrating both clustering and ranking methods toward higher performance.

We go a step further in this paper. Our proposed approach first clusters services into different domains, then automatically decomposes requirements into domains and goes deep into each domain for matching proper services.

VI. Conclusions

Service compositions inherently contain various domain-specific functionalities to fulfill complex requirements. However, due to the lack of explicit service domain categorization, most state-of-the-art service recommendation methods only mine the matching pattern between services and composition requirements without a finer grained classification of domains. This significantly restricts the accuracy and relevance of recommendation.

In this paper, a novel three-step approach is presented to provide domain-aware service recommendation. To overcome the absence of explicit service domain categorization, we first cluster services into different domains using topic features. Then, we mine domain relevance and domain-aware matching patterns regarding composition. Finally, we give recommendation in the form of a list of domains, each with a ranked list of services. The experimental results show that our approach gains a 30% improvement in MAP@20 and achieves a 20% improvement for long tail recommendation, which proves the advantages of domain-aware recommendation over state-of-art methods.

Our current clustering algorithm classifies services into individual domains. Future work will extend the approach and consider services associated with multi-domains. We also plan to incorporate users' profile and other information for personalized and context-aware recommendation.

Acknowledgment

This work is partially supported by the National Natural Science Foundation of China (No.61033005, No.61174169) and the Independent Research Fund of Tsinghua University (No. 20111080998) and the Specialized Research Fund for the Doctoral Program of Higher Education (No. 20120002110034).

References

[1] K. Huang, Y. Fan, W. Tan, and X. Li, "Service Recommendation in an Evolving Ecosystem: A Link Prediction Approach", In *Proceedings of IEEE International Conference on Web Services (ICWS)*, pp. 507-514, 2013.

[2] X. Chen, Z. Zheng, X. Liu, Z Huang, and H. Sun, "Personalized QoS-Aware Web Service Recommendation and Visualization", *IEEE Transactions on Services Computing,* 1(6): 35-47, 2013.

[3] W. Xu, J. Cao, L. Hu, J. Wang, and M. Li, "A Social-Aware Service Recommendation Approach for Mashup Creation" In *Proceedings of IEEE International Conference on Web Services (ICWS)*, pp. 107-114, 2013.

[4] D. M. Blei, A. Y. Ng and M. I. Jordan, "Latent Dirichlet Allocation", *The Journal of Machine Learning Research,* 3: 993-1022, 2003.

[5] G. Huang, Q. Zhu and C. Siew, "Extreme Learning Machine: Theory and Applications", *Neurocomputing,* 1(70): 489-501, 2006.

[6] K. Huang, Y. Fan and W. Tan, "An Empirical Study of Programmable Web: A Network Analysis on a Service-Mashup System", In *Proceedings of IEEE International Conference on Web Services (ICWS)*, pp. 552-559, 2012.

[7] M. N. Do and M. Vetterli, "Wavelet-based Texture Retrieval using Generalized Gaussian Density and Kullback-Leibler Distance", *IEEE Transactions on Image Processing,* 1(11):146-158, 2002.

[8] H. Yin, B. Cui, J. Li, J. Yao, and C. Chen, "Challenging the Long Tail Recommendation", In *VLDB*, pp. 896-907, 2012.

[9] Q. Yu, Z. Zheng and H. Wang, "Trace Norm Regularized Matrix Factorization for Service Recommendation", In *Proceedings of IEEE International Conference on Web Services (ICWS)*, pp. 34-41, 2013.

[10] D. Bianchini, V. De Antonellis, B. Pernici, and P. Plebani. "Ontology-based Methodology for e-Service Discovery", *Information Systems,* 4(31): 361-380, 2006.

[11] J. Zhang, J. Wang, P.C.K. Hung, Z. Li, J. Liu, and K. He, "Leveraging Incrementally Enriched Domain Knowledge to Enhance Service Categorization", *International Journal of Web Services Research (JWSR)*, 9(3): pp. 43-66, 2012.

[12] Y. Zhou, L. Liu, C. Perng, A. Sailer, I. Silva-Lepe, and Z. Su, "Ranking Services by Service Network Structure and Service", In *Proceedings of IEEE International Conference on Web Services (ICWS)*, pp. 26-33, 2013.

[13] W. Tan, M. Zhou, *Business and Scientific Workflows: A Web Service-Oriented Approach*, *Wiley-IEEE Press,* 2013.

[14] W. Tan, J. Zhang, and I. Foster, "Network Analysis of Scientific Workflows: a Gateway to Reuse", *IEEE Computer*, 43(9): 54-61, 2010.

[15] K. Huang, J. Yao, Y. Fan, W. Tan, S. Nepal, Y. Ni, and S. Chen, "Mirror, Mirror, on the Web, Which is the Most Reputable Service of them All? - A Domain-Aware and Reputation-Aware Method for Service Recommendation", In *Proceedings of International Conference on Service Oriented Computing (ICSOC)*, pp. 343-357, 2013.

[16] 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 *Proceedings of IEEE International Conference on Services Computing (SCC)*, 2011, pp. 48-55.

[17] K. Huang, Y. Fan, and W. Tan, "Recommendation in an Evolving Service Ecosystem Based on Network Prediction", *IEEE Transactions on Automation Science and Engineering*. Accepted in 2014 and available online. http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6717050.