2013 IEEE 10th International Conference on Services Computing

A Probabilistic Approach for Web Service Discovery

Chune Li, Richong Zhang, Jinpeng Huai, Xiaohui Guo, Hailong Sun
School of Computer Science and Engineering
Beihang University
Beijing, China
Email: {lichune, zhangrc, huaijp, guoxh, sunhl}@act.buaa.edu.cn

***Abstract*—Web service discovery is a vital problem in service computing with the increasing number of services. Existing service discovery approaches merely focus on WSDL-based keyword search, semantic matching based on domain knowledge or ontologies, or QoS-based recommendations. The keyword search omits the underlying correlations and semantic knowledge or QoS information is not always available. In this paper, we propose a probabilistic service discovery approach to help web service users to retrieve related services and to improve the search performance. Specifically, we apply a probabilistic model to characterize the latten topics between services and queries, and then propose a matching method based on the topic relevance. Experiments on services from a real service repository confirm the feasibility and efficiency of this proposed method.**

***Keywords*-service discovery; web service; LDA; topic model.**

I. Introduction

The number of web services and public APIs is growing at a terrific pace. This fact provides developers opportunities to enjoy the easiness of invoking existing services and makes possible the collaboration between heterogeneous systems. However, without prior knowledge of the functionalities of these services, users have to spend a great amount of time to search for useful information and the most related services. Therefore, there is a need to develop service discovery approaches for service repositories to assist users locating required services.

Existing service recommender systems solely help users discover good quality services in terms of non-functional properties, such as reliability, availability, response time, throughput and etc. These methods treat the quality of web service as the prediction target and exploit collaborative filtering approach to recommend high-quality services. These recommendation approaches do not take users functional requirements into consideration. Keyword search approaches can somehow assist users to discover potential services. But this approach would fail to return any result if the query keyword input is not exactly the word used to describe the service. To solve this problem, researchers have proposed semantic based approaches. These approaches are highly dependent on the service semantic web and there is no unified standard work on how to build this web. Besides, matching algorithms used by these approaches, mostly based on the logic or graphs, are always time expensive.

To overcome these limitations and improve the performance of the service search engine, in this work, we propose a probabilistic approach for service discovery. In specific, we first identify the features that may reflect the functional attributes of services. Then, we deliver the probabilistic approach to characterize dependencies between the hidden topics of services and the functional representative features of the services. In addition, a function for calculating the probability of a service matching a searcher's intent (the hidden topics of query keywords the user inputs) is proposed. The matching method is based on underlying topic correlations. This approach avoids complex semantic annotation, and catches the relations between words at the same time. At the end, we conduct experiments on a real service repository consisting of more than 10000 services. We present four service retrieval case studies to evaluate the performance of our approach. In comparison with the most commonly used keyword based service discovery approaches, the experimental results confirm the effectiveness of our probabilistic approach. The improved service discovery framework can be applied to service retrieval engines to help users to find their required services quite precisely, it can also serve as tools in service computing platform to help developers to discover useful services.

The remainder of this paper is organized as follows. Section II introduces existing approaches and the related research domains. Section III discusses the representative features of services and delivers an introduction of the problem definition. Section IV introduces a topic model and the probabilistic approach for web service retrieval. Section V presents experimental evaluation of our approach. Finally, section VI concludes the paper.

II. Related Works

In this section, we introduce prior works related to this study, mainly including two aspects of studies: service discovery and applications of topic models.

978-0-7695-5026-8/13 $26.00 © 2013 IEEE
DOI 10.1109/SCC.2013.107

A. Service discovery

Service discovery is one of the key problems that have been widely researched in Service Oriented Architecture (SOA) based systems. The first works were simple keyword-based and category-based search on UDDI [1]. Then Woogle [2] extracted some components from WSDL documents and made use of TF/IDF method to recommend services similar with the existed one. Christian Platzer et.al [3] built a WSDL search engine based on the vector space model (VSM). They modeled both a web service and a query as a vector of words with TF-IDF weights and retrieved results were ranked by the cosine correlation between them. More recently, Ourania Hatzi et.al [4] proposed a specialize framework to collect and retrieve services in both WSDL and OWL-S standards by extending and adapting TF-IDF model. These keyword search methods can achieve low accuracy and recall performance when the description in WSDL documents is insufficient. Advanced data mining techniques like classification and clustering have also been applied to the discovery of services [5] [6]. These methods can improve the time performance of retrieval and recommend relevant services on a user request. However, in the traditional classification or clustering methods, each service can only be grouped to one cluster deterministically, which is not suitable for services with complicated semantics.

Besides WSDL-based service search, many studies have focused on automated service discovery with semantic languages. OWLS-MX [7] described services in semantic OWL-S, and exploited logic-based reasoning for parameters matching and service retrieval. In [8], semantic annotations SAWSDL were used to automatically find service compositions. Xiao et.al. [9] proposed a behavior-based model making use of web service business process execution language WS-BPEL to model the service interaction process and automatically recommend similar services. These automatic formalized semantics approaches can achieve high precision. However, the semantic information is not always available and the annotating and automatic reasoning process faces the challenge of low time performance and high complexity in practice.

Another group of researchers were considering on the non-functional aspects such as web service selection and recommendation based on quality of service (QoS) [10], customer ratings [11] or service popularity [12]. However, the non-functional information is also always not available and these methods can only serve as a complement of service discovery.

B. Application of Topic Model

Since the introduction of LDA [13], the topic model and its various extensions have been widely used. Y. Lu et.al. [14] investigated probabilistic topic models for different text mining tasks including document clustering, text categorization and ad-hoc retrieval. Rosen-Zvi et.al. extended LDA to author topic model (AT) to model the relationship of both documents and authors [15]. In [16], an LDA extension relational topic model (RTM) was proposed to model the links between documents.

Besides text documents, topic based approach has been used to recommend tags of resources [17], and to reveal community structure [18]. In the software area, LDA has also become popular recently. In [19], topic models were learnt over software artifacts to navigate the software architecture. In [20], topics were modeled over software commit-log comments in source control systems to support analysis of software maintenance activities.

III. Web Service and Service Discovery

Web services are "self-contained, self-describing and modular applications that can be published, located, and invoked across the web" [21]. Technically, the web services description language (WSDL) provides a formal, computer-readable description of web services. We analyze the information provided by the WSDL document and retrieve important features to enable the process of service discovery.

A. Features from WSDL

A WSDL file is an XML-formatted document which provides information about what functions the service provides, where it locates and how to invoke it. A WSDL version 1.1[1] document describes a web service using the following seven major components [22]:

- *types*: XML schema types;
- *message*: definitions of communication messages;
- *operation*: descriptions of service actions;
- *portType*: a set of operations;
- *binding*: concrete communication protocols;
- *port*: endpoints with bindings and network addresses
- *service*: a list of endpoints.

Since we focus on the functional attributes of services, the information like communication protocols and network address become less important. The following segments and properties in WSDL document can be used as the features for the service discovery:

- Message Name
- Operation Name
- Service Name
- Text descriptions defined in the *documentation* segment for all the components in the WSDL document.

B. Service Discovery Problem Definition

We use S and WO to represent services and the observed features of services. More specifically, $WO = \{w_1, w_2, .., w_W\}$ is a dictionary, where $w_i (1 \leq i \leq W)$ denotes the i^{th} feature word, and W is the total number of words.

[1]The current version of WSDL specification is 2.0. Version 1.1 is used since almost all of our data are following this version.

$S = \{s_1, s_2, .., s_D\}$ is a set of services, where $s_d (1 \leq d \leq D)$ represents the d^{th} web service, and D is the total number of services.

We denote a service $s \in S$ as a set of words: $s = \{w_{s,1}, w_{s,2}, .., w_{s,n(s)}\}$, where each component $w_{s,i} \in WO (1 \leq i \leq n(s))$ is a word extracted from the service's WSDL features described before, $n(s)$ is the length of the service features.

In the service discovery process, a user specifies a query q as a set of words, $q = \{w_{i_1}, .., w_{i_n}\}$, describing the task he/she wants to implement or information he/she wants to acquire. The retrieval system searches services in S and return top k web services whose functions are most matched to the query.

IV. Topic Model based Service Discovery

In this section, we first introduce LDA, one of the most popular topic models, including the representation and inference. Then we propose a topic-based probabilistic service discovery approach.

A. LDA Model

The basic probabilistic topic model LDA (Latent Dirichlet Allocation) is a "generative probabilistic model for collections of discrete data such as text corpora" [13]. It views documents in the text corpora as bags of words and assumes that each document exhibits multiple topics, while each topic is modeled as a probabilistic distribution over all words. The generative process for all documents in a corpus is as follows:

1) For each topic j in 1:T, draw a topic distribution over words $\phi_j \sim Dirichlet(\beta)$
2) For each document d in the corpus:
   (a) Draw topic proportions $\theta^{(d)}|\alpha \sim Dirichlet(\alpha)$
   (b) For each of the N_d words in document d:
      i. Draw a topic assignment $z|\theta^{(d)} \sim Multinomial(\theta^{(d)})$
      ii. Draw a word $w|z, \phi_{1:T} \sim Multinomial(\phi_z)$

where T is the number of topics and W is the number of words in the corpus dictionary; the proportions parameter α is a length T vector, and the topic parameter β is a length W vector.

Fig. 1 represents the relationships between documents, observed words and latten topics clearly. The global corpus probability described by the model is

$$p(\mathbf{w}) = \prod_{j=1}^{T} p(\phi_j|\beta) \prod_{d} p(\theta^{(d)}|\alpha)(\prod_{w \in d} p(z|\theta^{(d)}) p(w|\phi_z)) \tag{1}$$

What we are interested in are the hidden variables: per-word topic assignment z, per-document topic proportions $\theta^{(d)}$ and per topic distribution over words ϕ_j. A well-principled Markov chain Monte Carlo (MCMC) method, the Gibbs Sampling can be used to inference the hidden variables. In MCMC [23], a Markov chain is constructed and topic assignment samples are taken from the chain which in turn change the state of the chain. Repeat these procedure until the chain converges to the target distribution. The final topic assignments in the chain are the estimations for each z. After all word topic assignments are got, the other two hidden variable θ and ϕ can be estimated:

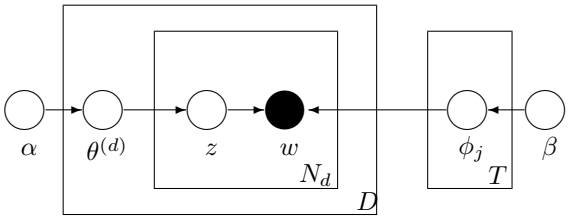


Figure 1. Graphical model representation of LDA. The shaded node represents observed words. The plates indicate replications. There are T topics represented in the right plate, D documents represented in the left outer plate. While the inner plate represents each topics and words in a document.

$$\hat{\phi}_j^{(w)} = \frac{n_j^{(w)} + \beta}{n_j^{(\cdot)} + W\beta} \tag{2}$$

$$\hat{\theta}_j^{(d)} = \frac{n_j^{(d)} + \alpha}{n_\cdot^{(d)} + T\alpha}. \tag{3}$$

where $n_j^{(\cdot)}$ denotes the number of times words are assigned to topic j, $n_j^{(w)}$ denotes the number of times word w is assigned to topic j; $n_j^{(d)}$ denotes the number of times words from document d is assigned to topic j, and $n_\cdot^{(d)}$ is the length of document d.

B. LDA-based Service Discovery

When both web services and users' queries are represented as sets of words, we can use LDA to model the topics for words and topic mixtures for services, and then calculate the relevance between services and the query based on the topic distributions.

In our application, the "observed words" are feature words extracted from service WSDL documents. The "documents" are web services. Hidden topics of documents can be explained as domains of web services. After the Gibbs sampling on the features for every service, the distribution over service feature words for topics $\phi_j^{(w)}$ and topic proportions for services $\theta_j^{(s)}$ can be calculated by Eq. 2 and Eq. 3 separately.

After modeling the relationships of the observed web service features, the services, and hidden topics, the probability of a service matching the query can be measured by

the conditional probability of the query given the candidate service:

$$\begin{aligned} p(q|s) &= \prod_{w \in q} p(w|s) \\ &= \prod_{w \in q} \sum_{j=1}^{T} p(w|j)p(j|s) \\ &= \prod_{w \in q} \sum_{j=1}^{T} \phi_j^{(w)} \theta_j^{(s)} \end{aligned} \tag{4}$$

Note that the equation associates words of query and service via topics. Services with feature words whose topics are similar to topics of query words are more likely to match the query. In other words, these services are relevant to the query on topics.

In this probabilistic approach for service discovery, topics of services and words are modeled once for all. Then when a user performs a service search with a query, the system calculates the probability of each service matching the query based on acquired topics, and ranks all the services based on the matching probability.

V. Experiments and Results

In this section, we apply our probabilistic topic-based service discovery method to a real service repository. The process of our method includes: a). extracting features from WSDL documents and preprocessing to get "documents" in LDA model; b). determining the input parameters of the model especially the topic number and modeling on service documents; c). calculating matching probability and ranking to find top relevant results.

A. Data Set and Preprocessing

Our experimental data is from serviceXchange platform[2], a web service repository developed by us. The platform has a web service crawler to collect web services from the web, including popular web service web sites, like webservices.seekda.com, www.xmethods.com, etc. .

According to section III-A, we extracted names and documentation features from these WSDL documents. To generate bags of words as input of LDA model, several natural language preprocessing tasks were done on features:

1) Filtering HTML tags. In the extracted $documentation$ segments, there are many HTML format substrings which have little contribution to the function description of web services. So a HTML tag filter was built based on regular expression. It filtered all HTML escape sequence like "&lt;" and all contents inside the angles brackets like the "string" in "<string>" and "<string/>". Besides, all white space characters and all non-alphabetic characters were also removed.

[2]www.servicexchange.cn

Table I
THE SIZE OF FINAL DATA SET

Item	Service	Dictionary	Word Occurrence
Size	12167	17856	357989

2) Separating long combined word strings. After filtering HTML tags in the feature strings, we have got a set of word strings separated by blank space. However, there are many long word strings which combined by several words, especially strings in the $name$ segments, such as *ValidateEmailWebService* and *getStockInfoByCode*. Note that most combined strings follow Pascal or Camel casing rules, that is the first letter in each word (including/expect the first word) is capitalized and other letters are in lower case. Based on this rules, we did a separation and got a set of words for every service WSDL document.
3) Filtering language stop words. Like text documents in any language, there are some common short function words which have little lexical meaning but occur frequently in the corpus. We chose about 600 English words as stop words and skipped every occurrence of these words.
4) Filtering frequent words and rare words. For each word, we counted the number of services in which the word occurs. Then top 20 frequent words and words which occur only once were removed. Since the former are most technically words used in service description such as *service*, *web*, *type*, *xml*, etc. And the latter are often not correct spelling words, like *abc* and *branchen*.

B. Training the Model

1) Choosing input parameters: We used Matlab Topic Modeling Toolbox [23] to train the LDA topic model on services features. The main inputs of the model are two vectors which contain the word and document indices for each word occurrence in the documents. After preprocessing, we got a bag of words for each service. Then a dictionary was created and each word occurrence in the services features were indexed. The size of the final data set is shown in Table I.

Another important input of the model is the topic number T which can affect the model results greatly. If T is small, the topics would be very broad. Otherwise, too many topics may lead to the overleap in meaning. We applied a standard Bayesian statistics method to determine the topic number [23]: the parameter T is chosen by maximizing the posterior probability of the model given the observed data $p(T|\mathbf{w})$. Further, assuming that each weak prior constraints on the number of topics, T can be chosen by maximizing $p(\mathbf{w}|T)$, the main component of $p(T|\mathbf{w})$. And $p(\mathbf{w}|T)$ can be approximated by taking the harmonic mean of a set of

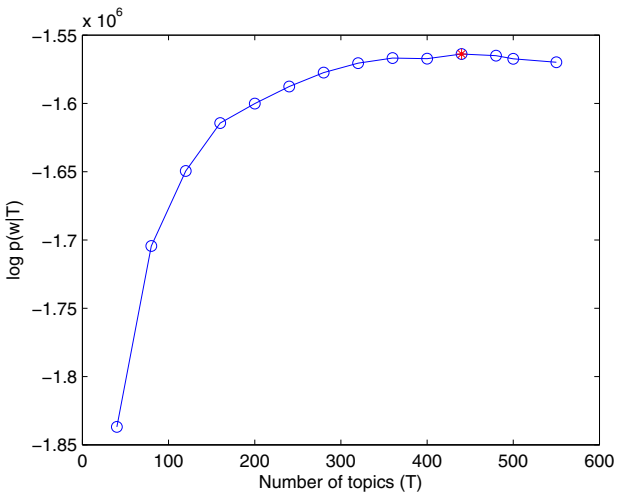


Figure 2. Topic number selection results, with topic numbers on the horizontal axis and the log-likelihood probability on the vertical one, peak value marked by a star.

values of $p(\mathbf{w}|\mathbf{z}, T)$, while $p(\mathbf{w}|\mathbf{z}, T)$ can be computed by

$$p(\mathbf{w}|\mathbf{z}) = \prod_{j=1}^{K} \frac{\Gamma(W\beta)}{\Gamma(\beta)^W} \frac{\prod_w \Gamma(\beta + n_j^{(w)})}{\Gamma(W\beta + n_j^{(\cdot)})} \tag{5}$$

Here $\Gamma(\cdot)$ is the standard gamma function.

We estimate $p(\mathbf{w}|T)$ for T from 40 to 550. To get a set of $p(\mathbf{w}|\mathbf{z}, T)$ for each value of T, we run 3 Markov chains and take 4 samplers with a lag of 200 iterations for each chain. In these sampling, other input parameters of the model are set as the suggested default values, i.e. $\beta = 200/W$; $\alpha = 50/T$. We calculate $p(\mathbf{w}|\mathbf{z}, T)$ for each sample and the harmonic mean on the 12 samples for each T are achieved as the value of $p(\mathbf{w}|T)$.

Topic selection results are shown in Fig. 2. From the results we can see with an increasing number of topics, the log-likelihood probability first increases and then decreases and the peak value is got when the number of topics is 440. Therefore by maximizing posterior probability of the model, 440 is chosen as the topics number value in our experiment.

2) Training and Results: We set input parameters values as described before(T=440, α=50/440, β=200/17856, iterations=1000) and train LDA model on the service data set.

Table II shows three discovered topics and corresponding words as an example. (These topics are chosen since they will be used in the following parts of this article.) Top 10 probable words of each topic are listed. The first line is the probability of each topic in the corpus. Other lines are words and their probabilistic likelihood in the corresponding topic. By analyzing the words in each topic, we can see these three topics might be *weather*, *parcel post* and *travel booking* separately.

Table II
TOPIC EXAMPLES GOT BY LDA

TOPIC 108	0.00238	TOPIC 382	0.0025	TOPIC 105	0.00457
weather	0.04907	weight	0.04392	hotel	0.03089
temperature	0.04152	length	0.02958	booking	0.02648
longitude	0.02737	package	0.02689	departure	0.02648
latitude	0.02548	carrier	0.02331	price	0.02501
station	0.02548	shipment	0.02331	availability	0.02452
direction	0.02454	width	0.02331	flight	0.02255
speed	0.02454	contact	0.02062	arrival	0.02206
wind	0.02454	pickup	0.01972	room	0.02108
forecast	0.0236	instructions	0.01883	destination	0.02059
average	0.01982	customer	0.01614	class	0.01765

Table III
FOUR OF OUR QUERIES

No.	Query	Intention
Q1	weather forecast	Get weather information for next few days.
Q2	parcel shipment	Find delivery services to mail parcels.
Q3	book airline ticket	Find ticket booking services to book a flight.
Q4	book flight	Find ticket booking services to book a flight.

C. Service Discovery Experiments

In this subsection, we construct user queries and evaluate the service retrieval results of different methods.

1) Metrics: We evaluate service retrieval results by Normalized Discounted Cumulative Gain(NDCG) [24], which can be calculated by the following equation:

$$NDCG@K = z_k \sum_{i=1}^{k} \frac{2^{r(i)} - 1}{log_2(1+i)} \tag{6}$$

where $r(i)$ is the relevance of the i-th result, z_k is a normalized coefficient which makes the NDCG value of the perfect ranking is 1 and other rankings less than 1. The ratio $\frac{2^{r(i)}-1}{log(1+i)}$ places stronger emphasis on relevant documents and penalizes the importance proportional to the position of the result.

A four-level scoring {3,2,1,0} is used for the relevance value r in our experiment. Each of these four levels respectively represents strong relevant, quite relevant, not so relevant and irrelevant.

2) Query: A user's query is a set of words describing his/her expectation for web services. The choices of words are very subjective and random. In Table III we describe four of our queries we have tested.

We describe these queries since they are quite easy to determine whether a service is relevant to the query, which is critical for the evaluation of the results. The functions asked by these queries are in common use. And there are lots of services on meteorology, geography and travel, all of which may provide these functions. Note that the intentions of Q3 and Q4 are the same.

3) Topic Proportions of Retrieved Services: We performed a service retrieval for the most

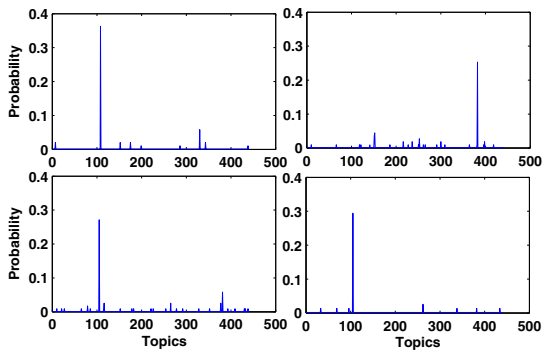


Figure 3. Topic proportions of top-1 service for four queries. (Q1: top-left; Q2:top-right; Q3:bottom-left; Q4:bottom-right)

relevant one service for each query. The services we have got are *DASWorldWeather* (http://almiyah.gov.ae/webservices/WorldWeather.asmx), *walker* (http://sws-challenge.org/shipper/v2/walker), *Service* (http://webconnect.akbartravelsonline.com/service.asmx), *travelSearch* (http://ts.afnt.co.uk/travelSearch.asmx). All these four service are satisfactory results.

In Fig. 3, we give the topic proportions of these four services. The main topic for each service is corresponded to topics listed in Table II. Besides the main topic, there are several secondary topics in each service. Note that services acquired for Q3 and Q4 have the same main topic. However, they still have difference in other topics proportions.

4) Comparative Experiments and Results : We compare our topic-based service discovery methods with two keyword based methods: TF/IDF and vector space model (VSM). The TF/IDF value between a word and a document is defined as below:

$$tf(w,d) = \frac{n_{w,d}}{n_{.,d}} \tag{7}$$

$$idf(w) = log(\frac{D}{N_w}) \tag{8}$$

$$tfidf(w,d) = tf(w,d) * idf(w) \tag{9}$$

where $n_{w,d}$ is the times word w occurs in document d, $n_{.,d}$ is the length of document d; D is the number of documents in corpus, N_w is the number of documents which include word w. We calculated the TF/IDF based relevance between a query q and a service s using the following equation:

$$tfidf(q,s) = \sum_{w \in q} tfidf(w,s) \tag{10}$$

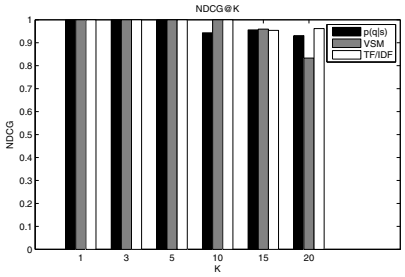
Vector space model represents both documents and queries as vectors of words with TF/IDF weight:

$$\mathbf{v}_d = [v_{d1}, v_{d2}, .., v_{dW}] \tag{11}$$

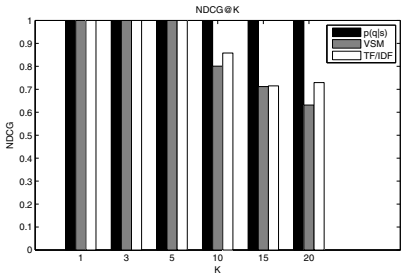
Each component v_{di} in this vector is the TF/IDF value between the i-the word of dictionary and document d, i.e. $tfidf(w_i, d)$. Then cosine similarity between each service vector and query vector are calculated.

$$cos(\mathbf{v}_q, \mathbf{v}_s) = \frac{\mathbf{v}_q \cdot \mathbf{v}_s}{\| \mathbf{v}_q \| \| \mathbf{v}_s \|} \tag{12}$$

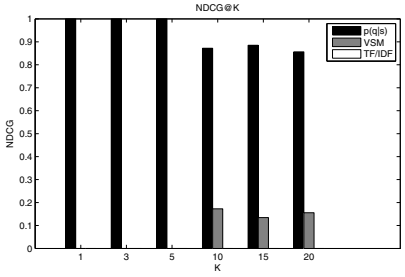
Then the services with high cosine similarity are VSM retrieved results.



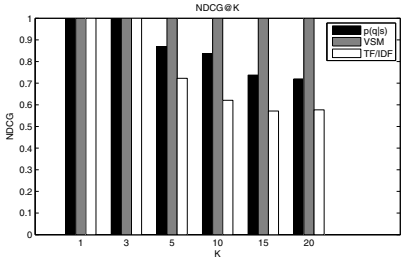
(a) weather forecast



(b) parcel shipment



(c) book airline ticket



(d) book flight

Figure 4. Comparison of NDCG@K

The NDCG@K results for K equal to 1, 3, 5, 10, 15, 20

Table IV
RETRIEVED WEB SERVICES NOT DIRECTLY RELEVANT TO QUERIES

Query	$p(q \mid s)$	VSM	TF/IDF
Q1	MeteoService(get air station)	Service(adserver forecast) Varsel(pollen forecast)	DigitalForecastServiceImplService
Q2	-	PropertyInformation(land parcel) Webservices.nl(property service)	IGeoKeysFacadeService(land parcel) PropertyTaxes
Q3	HotelsService AmendmentService(hotel booking amend) Export(cruises price)	AddressSearch(address search, validation) DataService(CSI club data) Books(author, pages, etc.)	TicketCentral(get applications' accredit) ISBN(books detail) stWSDL(ticket image,encoding)
Q4	HotelsService AmendmentService(hotel booking amend) Export(cruises price)	-	ISBN(books detail) autocomplete(GetBookList) Books(author, pages, etc.)

are showed in Figure 4. From the results we can see, in all instances calculating the likelihood probability of queries conditioned on services gives quite good results in this service discovery application while VSM and TF/IDF are sometimes seriously inefficient. More specifically, all the three methods give good results for the first query *weather forecast*, the NDCG value are more than 0.8. For Q2, our method gives a perfect result, while the NDCG value of VSM and TF/IDF method can drop to 0.6 and 0.7. Our method is relatively better. The results given by keyword-based methods for the third query *book airline ticket* are awfully bad: VSM method gives no right recommendations until top 10 services while TF/IDF method never retrieves a related service. For the last query, VSM method gives a perfect result, while our method is relatively a little worse, however still better than TF/IDF method.

5) Discussion: To discuss the different approaches, we show some services which are retrieved but are not directly relevant to the queries in Table IV. (The two empty cells in the table mean that top 20 results retrieved by our probabilistic method for Q2 and retrieved by VSM method for Q4 are all services relevant to queries.) The difference between probabilistic topic model based method and keyword based method is quite clear:

The results given by probabilistic methods are either relevant to queries, or with related topics. In our experiment cases, hotel service is retrieved when asking for flight, airport information are given when asking for weather. However, both booking flights and booking hotels are travel related, while wether information is essential when choosing airports. These results although cannot fulfill the initial intention of the queries, are good suggestions in practice.

VSM and TF/IDF method can give totally irrelevant results if some words in the query are polysemous. For example, both the methods give land property service for query *parcel shipment* since parcel has the meaning of *an extended area of land* besides the meaning of *a collection of things wrapped or boxed together*. The similar occasion happens for queries including word *book*. Note that for the similar query *book airline ticket* and *book flight*, the keyword based method give good results for the latter while extremely bad results for the former. This occurs when services are described using *flight* more than *airline*.

The disadvantages of the keyword based methods are obvious. Their retrieval performance heavily depends on the user query. While the topic based method is quite robust and nearly always gives good results. Besides, topic based method gives related suggestions.

Based on this discussion, we can conclude that when users are not so decided what functions or information exactly they want, when the queries for web service are quite fuzzy or with a broad topic, topic based methods may be better since these methods can suggest related web services. While the need for services is very exact and the way of describing it is very single, keyword based methods can give quite accurate results. A good combination of keyword based methods and probabilistic topic based methods may make use of their advantages to improve the performance of service discovery.

VI. CONCLUSION

Today's developers want to receive satisfactory APIs or web services, but they are not willing to spend much time and effort for their surfing for required services. Therefore, we believe that our proposed approach, aiming to automate the process of analyzing and ranking services to retrieve the most related services they queried, will be useful.

In this study, we first analyze features that may describe the functional attributes of web services. Then we propose a probabilistic framework and a retrieval model for web services. In addition, we evaluate the performance of the proposed approach on real web service data set and the results confirms the effectiveness and the efficiency of our system. We note that this work can serve as a service discovering engine for any service repositories and also the tools that assist developers to discover useful services.

In the future work, we plan to incorporate more web service features, such as providers of services which may be found by analyzing *name space* and *soap address* fragments

of WSDL, and the adequacy of descriptions about service functions, to improve the performance of our approach. Also, in this current study, we do not consider the non-functional attributes of services. We are going to take into account these attributes to provide both high quality and high related services for users.

ACKNOWLEDGMENT

This work was supported partly by National Natural Science Foundation of China (No. 61103031), partly by China 863 program (No. 2012AA011203), partly by the Fundamental Research Funds for the Central Universities (No. YWF-12-RHRS-016), partly by A Foundation for the Author of National Excellent Doctoral Dissertation of PR China, partly by Beijing Nova Program and partly by Program for New Century Excellent Talents in University.

REFERENCES

[1] T. Bellwood, L. Clément, D. Ehnebuske, A. Hately, M. Hondo, Y. L. Husband, K. Januszewski, S. Lee, B. McKee, J. Munter *et al.*, "Uddi version 3.0," *Published specification, Oasis*, vol. 5, pp. 16–18, 2002.

[2] X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang, "Similarity search for web services," in *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*. VLDB Endowment, 2004, pp. 372–383.

[3] C. Platzer and S. Dustdar, "A vector space search engine for web services," in *Web Services, 2005. ECOWS 2005. Third IEEE European Conference on*. IEEE, 2005, pp. 9–pp.

[4] O. Hatzi, G. Batistatos, M. Nikolaidou, and D. Anagnostopoulos, "A specialized search engine for web service discovery," in *Web Services (ICWS), 2012 IEEE 19th International Conference on*. IEEE, 2012, pp. 448–455.

[5] Y. Wu, C. Yan, Z. Ding, P. Wang, C. Jiang, and M. Zhou, "A relational taxonomy of services for large scale service repositories," in *Web Services (ICWS), 2012 IEEE 19th International Conference on*. IEEE, 2012, pp. 644–645.

[6] K. Elgazzar, A. E. Hassan, and P. Martin, "Clustering wsdl documents to bootstrap the discovery of web services," in *Web Services (ICWS), 2010 IEEE International Conference on*. IEEE, 2010, pp. 147–154.

[7] M. Klusch, B. Fries, and K. Sycara, "Automated semantic web service discovery with owls-mx," in *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*. ACM, 2006, pp. 915–922.

[8] G. C. Hobold and F. Siqueira, "Discovery of semantic web services compositions based on sawsdl annotations," in *Web Services (ICWS), 2012 IEEE 19th International Conference on*. IEEE, 2012, pp. 280–287.

[9] Z. Xiao, D. Cao, C. You, and H. Mei, "Extracting behavioral models from ws-bpel processes for service discovery," in *Services Computing, 2009. SCC'09. IEEE International Conference on*. IEEE, 2009, pp. 300–307.

[10] M. Zhang, X. Liu, R. Zhang, and H. Sun, "A web service recommendation approach based on qos prediction using fuzzy clustering," in *Services Computing (SCC), 2012 IEEE Ninth International Conference on*. IEEE, 2012, pp. 138–145.

[11] A. Srivastava and P. G. Sorenson, "Service selection based on customer rating of quality of service attributes," in *Web Services (ICWS), 2010 IEEE International Conference on*. IEEE, 2010, pp. 1–8.

[12] S. Elfirdoussi, Z. Jarir, and M. Quafafou, "Popularity based web service search," in *Web Services (ICWS), 2012 IEEE 19th International Conference on*. IEEE, 2012, pp. 683–689.

[13] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *the Journal of machine Learning research*, vol. 3, pp. 993–1022, 2003.

[14] Y. Lu, Q. Mei, and C. Zhai, "Investigating task performance of probabilistic topic models: an empirical study of plsa and lda," *Information Retrieval*, vol. 14, no. 2, pp. 178–203, 2011.

[15] M. Rosen-Zvi, T. Griffiths, M. Steyvers, and P. Smyth, "The author-topic model for authors and documents," in *Proceedings of the 20th conference on Uncertainty in artificial intelligence*. AUAI Press, 2004, pp. 487–494.

[16] J. Chang and D. M. Blei, "Hierarchical relational models for document networks," *The Annals of Applied Statistics*, vol. 4, no. 1, pp. 124–150, 2010.

[17] R. Krestel, P. Fankhauser, and W. Nejdl, "Latent dirichlet allocation for tag recommendation," in *Proceedings of the third ACM conference on Recommender systems*. ACM, 2009, pp. 61–68.

[18] D. Li, Y. Ding, C. Sugimoto, B. He, J. Tang, E. Yan, N. Lin, Z. Qin, and T. Dong, "Modeling topic and community structure in social tagging: The ttr-lda-community model," *Journal of the American Society for Information Science and Technology*, vol. 62, no. 9, pp. 1849–1866, 2011.

[19] H. U. Asuncion, A. U. Asuncion, and R. N. Taylor, "Software traceability with topic modeling," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1*. ACM, 2010, pp. 95–104.

[20] A. Hindle, N. A. Ernst, M. W. Godfrey, R. C. Holt, and J. Mylopoulos, "Automated topic naming to support analysis of software maintenance activities," in *The 33rd International Conference on Software Engineering, ICSE*, 2011.

[21] D. Tidwell, "Web services-the webs next revolution," *IBM developerWorks*, 2000.

[22] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana *et al.*, "Web services description language (wsdl) 1.1," 2001.

[23] T. L. Griffiths and M. Steyvers, "Finding scientific topics," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 101, no. Suppl 1, pp. 5228–5235, 2004.

[24] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of ir techniques," *ACM Transactions on Information Systems (TOIS)*, vol. 20, no. 4, pp. 422–446, 2002.