2011 IEEE International Conference on High Performance Computing and Communications

Leveraging Fragmental Semantic Data to Enhance Services Discovery

Jian Wang*, Jia Zhang**, Patrick C.K. Hung***, Zheng Li*, Jianxiao Liu*, Keqing He*
*State Key Lab of Software Engineering, Computer School, Wuhan University, China
**Department of Computer Science, Northern Illinois University, USA
***University of Ontario Institute of Technology, Canada
*jianwang@whu.edu.cn, **jiazhang@cs.niu.edu, ***Patrick.Hung@uoit.ca, *hekeqing@whu.edu.cn

***Abstract*—As one foundational technology of cloud computing, services computing is playing a critical role to enable provisioning of software as a service (SaaS). However, how to effectively and efficiently discover proper available services from the cloud of resources remains a big challenge. This paper reports our continuous efforts on semantic services discovery. We extend the Support Vector Machine (SVM)-based text clustering technique in the context of service-oriented categorization in a service repository, and propose an iterative process to incrementally enrich domain ontology. A popular Web 2.0 mashup platform is used as a testbed; and preliminary evaluation results are reported.**

I. INTRODUCTION

One key advantage and goal of cloud computing is resource pooling, meaning that various types of resources can be shared on the cloud [1]. Leveraging existing services available in the cloud, users can compose new value-added processes and further publish them as reusable services. However, as cloud has become an unprecedented driving factor to encourage people to publish and share resources as services, how to effectively and efficiently discover interested services from the cloud of resources remains a big challenge.

One major technique is to establish service registries [2] as centralized yellow pages to help users find related services. Earlier Universal Description, Discovery, and Integration (UDDI) registries are going out of date – one major reason is that it is ambitious to manage all kinds of Web services. Thus, in recent years, various less formal and more domain/usage-specific service registries have emerged. For example, the BioCatalogue (http://www.biocatalogue.org) site manages over 1,600 life science-specific services; the programmable web (http://www.programmableweb.com, PWeb) site manages over 3,500 services for users to design mashups.

Naturally, such service registries all provide a layer of technique to facilitate users in querying and finding interested services. However, their current querying power is usually preliminary. Taking the PWeb as an example, it requires that users select one specific domain from a preset list. This requirement may cause confusion. First, some user-categorized domains at service registration time may not be accurate. Furthermore, some services naturally belong to multiple domains, because some predefined domains overlap with each other. For example, domains "travel," "shipping," and "weather" share many common concepts. Moreover, the PWeb presets a domain named "others" and a significant number of services are left in the category. Currently, 158 services are listed in the category of "others," which is the top 5 category with the most number of services (Other top 4 categories are: Internet (268), Social (245), Mapping (206), Reference (170)).

As a result, there exist needs for existing service repositories to enhance their query and search ability. While such ability will attract more service users to visit the repository, it will also attract more service providers to publish services at the site.

This paper reports our continuous efforts on semantic services discovery, extracting fragmental semantic data to support services discovery. Specially, we aim to explore an international standard-compatible approach to annotate and classify services on the service registry side. Our basic hypothesis is that, much knowledge is hidden in the service repositories and can be leveraged to enable and facilitate services discovery.

As integral parts of an ISO project, Metamodel Framework for Interoperability (MFI, http://metadata-stds.org/19763/index.html), Role-Goal-Process-Service (RGPS) framework [3] aims at providing a language for users to describe personalized requirements involving domain-related services, toward an ultimate goal of enabling on-demand service provisioning. This paper reports our first attempt to extend the RGPS concepts onto service registry side and describe registered services, so that semantic match making can be conducted between user requirements and service descriptions through a meet-in-the-middle strategy. As a starting point, here we focus on describing services to facilitate their categorization.

We extend the Support Vector Machine (SVM) technique in this project in the context of service categorization on service repositories. While the SVM engine being like a black-box, we propose an input function and an output function for it to increase its categorization accuracy based on incrementally constructed domain knowledge. Our extended SVM technique has two goals: one is to verify and adjust existing author/user-centered service categorization; the other one is to enrich domain ontology.

The remainder of the paper is organized as follows. In

978-0-7695-4538-7/11 $26.00 © 2011 IEEE
DOI 10.1109/HPCC.2011.149

Section II, we discuss related work. In Section III, we introduce our service semantic model. In Sections IV and V, we present our ontology-empowered SVM technique and the consequent methodology, respectively. In Section VI, we present experimental settings and preliminary results and analysis. In Section VII, we draw conclusions.

II. Related Work

Segev and Zheng [4] propose an ontology bootstrapping method that automatically generates concepts and their relations in a domain from WSDL files. Lee and Kim [5] study how to enable similarity search over RESTful services based on syntactic and semantic descriptions. In contrast to their work, we focus on incrementally build domain concepts and their relationships from fragmental semantic data.

Liu et al. [6] derive semantic relations between services based on their associated tags, and consequently build a directed service graph to guide potential service composition. In contrast, we categorize services based on their associated sematic information including descriptions, tags, and categorization information.

Zhou et al. [7] presents an algorithm that automatically generates hierarchical concept relationships from social annotations. In our research, we leverage the algorithm to generate the initial domain ontology hierarchy.

Semantic Automated Discovery and Integration (SADI) [8] framework is able to retrieve SADI services. While SADI search only compares the input/output OWL class URLs in SADI services, our work considers more semantic conditions (e.g., functional profile and tags).

Zhang and Li introduce the concept of service cluster [9] to represent a collection of available services provided by multiple service providers to perform a specific common function. Here we borrow the concept and extend the SVM technique to help verify and justify service clusters.

There have been a lot of efforts on semantic services discovery, most of which performing profile-based service signature (I/O) matching [10]. OWLS-MX [10] and WSMO-MX [11] propose to combine logic-based reasoning and syntactic concept similarity computations in OWL-S. Sbodio et al. [12] propose to use SPARQL as a formal language to describe the pre- and post-conditions of services. Junghans et al. [13] propose a practical formalism to describe functionalities and service requests. In contrast, we focus on enriching domain ontology and leveraging it to classify services.

The Information Retrieval (IR) community has created a wealth of clustering algorithms and techniques [14]. In contrast to these general-purpose text categorization technologies, our work aims at services discovery and targeting on service repositories with embedded ontological information, which can be exploited to facilitate service categorization when the scale of the training data set is not large enough.

III. Service Semantics Model

Extending our previous work on service semantic modeling [15], we propose an ISO-standard RGPS-compatible service semantic model. Such a model will serve as a contractual template between service providers and service consumers, so that semantic match making can be conducted between user requirements and service descriptions.

As shown in Fig. 1 illustrating in a UML class diagram, we propose a service semantics model comprising static semantics and behavioral semantics, in a composition relationship. Static semantics represents published information of a service together with its associated descriptions contributed by either creators or users. It consists of functional profile (describe what a service can do), goal (objectives of the service provided by service creators) and comment (user comments and tags), also in a composition relationship. Behavioral semantics describes how a service should be used in the best way, comprising a set of aspects including I/O parameters, constraint, condition, usage pattern and QoS property.

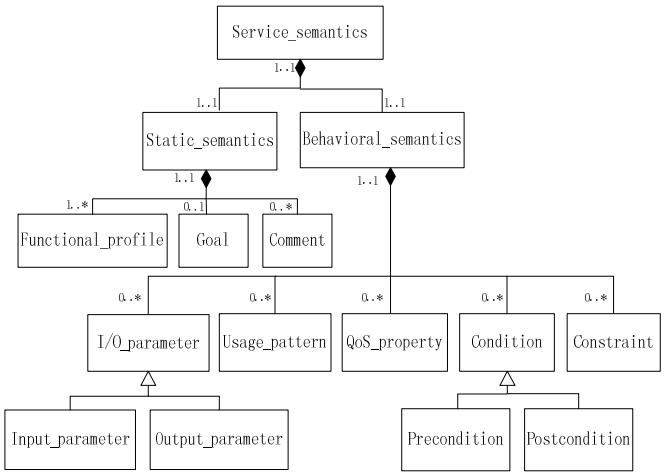

Fig 1. Service semantics model.

We will use an example of payment service to illustrate how our service semantics model can be used to describe a service. The semantic specifications of the example service are summarized in Table I, some being extracted from its published description file (WSDL file) and some from related documents such as registration documents. One integral element of our service semantics model is the functional profile of a service. Existing studies focus on either service interfaces (i.e., input/output signatures) or comprehensive documents (e.g., user manual). In contrast, we define the technical functional profile of a service as a combination of information that can be extracted from the standard descriptions of the service (e.g., in WSDL): service name, portType name, operation names and data variable names defined in service operations. The goal indicates that the service aims at providing a payment service. Creator input such

as keywords are inserted into the goal category to complement service descriptions from published code (e.g., free of charge). The comments can be *ad hoc* fragmental documents such as social tags and user comments (e.g., good user experience).

Table I. An example of service semantics model.

Static semantics	
functional profile	service_name：PaymentService portType_name: PaymentServicePortType operation_name: make_payment input_message_name: paymentRequest output_message_name: paymentResponse
Goal	provide payment service for users
comment	Payment, Free
Behavioral semantics	
input parameter	card_ID, card_PIN, transaction_Amount
output parameter	transaction result(success/failure)
precondition	validated PIN and enough balance
postcondition	the balance of the card is decreased
Constraint	An operation must be received from users in every 15 minutes
usage pattern	Usually used with a shipping service
QoS property	High security

The input parameters of the service are card_ID, card_PIN, and transaction_Amount, while the output parameter is a Boolean variable transaction result. The precondition of the service is that the input PIN has to be valid and the balance of the card is greater than the value of the input transaction_Amount. The postcondition is that the payment is charged and the balance of the card is decreased after the transaction is completed. The constraint of the service is that if no operation is received from users for 15 minutes during the execution process of the service, the connection will be expired for security reasons. The usage pattern indicates that the service is usually used followed by a shipping service if a merchandize is involved. The QoS property denotes that the security of the service is guaranteed.

Specifically, in this research project, such a model will help us compare similarity between services. As the first step, we will use the combination of {functional_profile} ∪ {goal} ∪ {comment} to guide service categorization. Service similarity calculation leveraging the entire structure of our service semantics model will be our future work.

IV. Extended SVM Technique

Existing service categorization usually adopts Information Retrieval (IR) similarity models such as vector space models, probabilistic models, and information theory-based models [16]. Their underlying technique is semantic similarity measurement between services, either based on keywords [17] or on ontology [18]. The former method uses the Term Frequency – Inverse Document Frequency (TF-IDF) [19] technique to build a vector space; the latter leverages taxonomy, information content (IC), or concept property to calculate similarity between services.

A. Applying SVM

It is known that the Support Vector Machine (SVM) method outperforms (accuracy of clustering) other text categorization methods [14], especially when the number of dimensions of the documents to be considered is significant. Verifying and justifying the categorization of services may not be a trivial task, since services may have many semantic aspects to be considered as shown in our service semantics model (Section II). Therefore, it is suitable to apply the SVM method for service categorization.

Fig. 2 shows the high-level workflow of how we directly apply the SVM approach to conduct service categorization. The input of the process is a repository of services; the output is the classified services and the ranking of the domain keywords (domain ontology). The workflow comprises three phases. First, all input services are transformed into a vector space based on the TF-IDF formula. Second, a SVM classification model is built based on a selected training set, and then runs over the entire repository (i.e., testing set) to classify each comprising service as either domain-relevant or domain-irrelevant. Third, the mutual information (MI) value of each keyword in the testing set is calculated to represent the ranking of the keyword in the domain.

The construction of the SVM classification model can be formalized as an optimization problem [20]. Given a training set of pairs (x_i, y_i), $i \in S_T$ where $x_i \in S^n$ representing each service in the form of a n-dimension vector, and $y_i \in \{1, -1\}$ indicating whether a service belongs to the domain or not. A SVM model aims to find a solution:

$$\min_{w,b,\xi} \left(\frac{1}{2} w^T w + C \sum_{i=1}^{|S_T|} \xi_i\right)$$

$$\text{subject to: } y_i(w^{\mathrm{T}} \emptyset(x_i) + \mathrm{b}) \geq 1 - \xi_i,\ \xi_i \geq 0$$

Training vectors x_i are mapped into a higher dimensional space by the function $\emptyset$. SVM finds a linear separating hyperplane with the maximal margin in the higher dimensional space. $C > 0$ is a penalty parameter of the error service.

B. Issues and Considerations

Applying the above approach to service repository (e.g., PWeb), we found that the categorization results are not satisfactory in a number of domains. Analyzing the reasons, we identify several significant issues. Successful

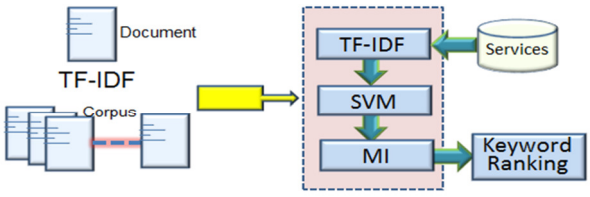

Fig. 2. Directly applying SVM technique.

training of the SVM classification model heavily depends on the scale and the precision of the training set. A service repository, however, usually has an unbalanced distribution of services in different domains. Some domains contain a smaller number of services. Meanwhile, *ad hoc* service categorization inputted by service providers may not always be accurate.

After carefully examining PWeb, we noticed one unique feature. The services registered at the repository are organized into 55 separate domains. In other words, services registered within the same domain should share the same domain ontology. Our hypothesis is that, such domain ontology may help build a more instructive vector space as the input to the SVM, so as to enhance the quality of the training set and in turn enhance categorization accuracy.

In detail, we extended the algorithm used to measure keyword-based service similarity, i.e., TF-IDF, which does not take into account domain knowledge. When TF-IDF calculates the significance of a term in a document, it does not consider the significance of the term in the corresponding domain.

C. Extensions to TF-IDF

Fig. 3 illustrates our extensions to the traditional TF-IDF [19]. TF-IDF aims to calculate the weight ($w_{j,i}$) of every term ($t_{j,i}$) inside of document (d_i). It indicates the importance of the term, against the entire document repository. Equation (1) shows that, the more documents in which a term appears, the less important the term is.

$$tf\text{–}idf(w_{j,i}) = tf_{j,i} \times idf_j = \frac{c_{j,i}}{\sum_k c_{k,i}} \times log \frac{|\{d\}|}{1+|\{d:t_{j,i} \epsilon d\}|} \quad (1)$$

In contrast to TF-IDF measuring between documents and corpus (entire set of documents), we propose to break TF-IDF into two parts. As shown in Fig. 3, two concepts are introduced: keyword frequency – inverse document frequency – domain frequency (KF-IDF-DF) and keyword frequency – inverse repository frequency (KF-IRF). KF-IDF-DF intends to measure between a service and its corresponding domain; while KF-IRF intends to measure between a domain and the entire service repository. Note that keywords here represent terms that are significant enough to facilitate service categorization. This also explains our rationale of using keywords instead of all terms: keywords will decide service categorization.

Equation (2) shows how to calculate the significance

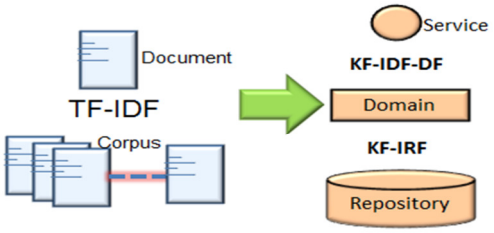

Fig. 3. Extensions to TF-IDF.

of keyword (k) in service (s) for domain (d). rank(k,d) represents the rank of the keyword (k) in the domain ontology (d). If a keyword highly represents a domain (it ranks in the top Ω keywords, e.g., the top 100 keywords), its tf-idf value will be amplified.

$$kf\text{–}idf\text{–}df_{k,s,d} = \begin{cases} tf\text{–}idf_{k,s} \cdot \left(1 + \left(1 - \left\lfloor \frac{rank(k,d)}{\sqrt{\Omega}} \right\rfloor / \sqrt{\Omega}\right) \cdot \beta\right) & rank(k,d) \le \Omega \\ tf\text{–}idf_{k,s} & otherwise \end{cases} \quad (2)$$

Since a domain ontology will keep on evolving, its top ranked keywords are divided into sections (i.e., square root of Ω) to decide its amplifier scale. For example, if we consider the top 100 keywords for a domain, its top 10 ($\sqrt{100}$) keywords will be put into one section and will amplify their tf-idf value by 1.1 (if the coefficient β (β ∈ [0.1]) is set to be 1).

Equation (3) shows how to calculate the ranking of keyword (k) in domain (d). num(k,d) represents the frequency of the keyword (k) in the domain ontology (d). It is divided by the number of services in the domain for normalization purpose. The frequency of the keyword is further adjusted by the distribution of the keyword over the corresponding domains in a service repository. α is a coefficient that can be adjusted in specific domains.

$$kf\text{–}irf_{k,d} = \frac{num(k,d)}{|\{s: s \epsilon d\}|} \cdot \left(\alpha \cdot \left(1 - \frac{1}{\log \frac{|D|+1}{|\{d:k \epsilon d\}|}}\right) + (1-\alpha) \cdot \frac{num(k,d)}{\sum_{d_i \epsilon D} num(k,d_i)}\right) \quad (3)$$

Consider two distribution scenarios as shown below, where the frequencies of a keyword in two repositories are the same (109). In scenario (a), the frequency of the keyword in domain d1 is 100, and its frequency in domain d2 is 9. In scenario (b), the keyword counts 100 times in domain d1, and counts one time in nine other domains (d2~d10).

	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10…
(a)	100	9								
(b)	100	1	1	1	1	1	1	1	1	1

If a keyword appears in many domains (e.g., scenario (b)), it is more likely that it is less important in representing any domain. For example, since the PWeb is a web service registry, many services in its comprising domains possess the same keyword "service" with high frequency, which is insignificant in representing any domain. Therefore, the fraction $\left(1 - \frac{1}{\log \frac{|D|+1}{|\{d:k \epsilon d\}|}}\right)$ in Equation (3) will lower the ranking of such keywords.

Meanwhile, in scenario (a), the keyword is likely to represent domain d1 (frequency 100) more than to domain d2 (frequency 9). In scenario (b), the keyword is likely to represent domain d1 (frequency 100) more than to domain

d2 (frequency 1). The fraction $\left(\frac{num(k,d)}{\sum_{d_i \in D} num(k,d_i)}\right)$ in Equation (3) reflects such a consideration.

V. SVM-Extended Methodology

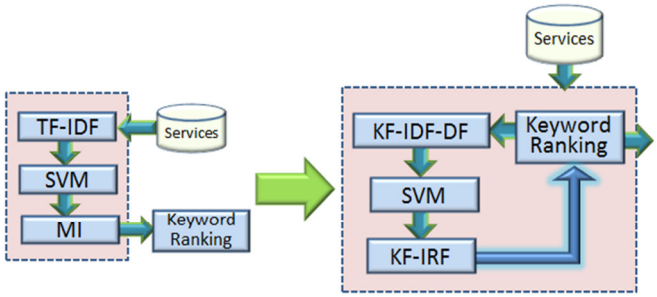

Fig. 4. SVM-extended methodology.

Based on our extensions to TF-IDF, we propose an ontology-empowered SVM methodology. Fig. 4 illustrates our overall idea; and Table II lists the detailed pseudo-code algorithm. Domain ontology, keyword ranking as explained in Equation (2), is used in our KF-IDF-DF formula to assist in creating the vector space (step 8). Generated vector space with all normalized service vectors are sent to the SVM machine for classification (step 10). Afterwards, our KF-IRF formula is used to rank domain-related keywords (steps 11~14).

Table II. Ontology-empowered SVM approach.

Algorithm: Extended SVM
Input: a collection of annotated services (**S**) in a domain (d).
Output: ranked domain keywords.
1: *build_corpus*(S)
1.1: **For** each service $s_i \in S$
1.2: $\{k_i, f_i\} \leftarrow extract\_terms(s_i)$
1.3: $\{k_i, f_i\} \leftarrow normalize\_terms(\{k_i, f_i\})$
1.4: $\{k_{j,d}, f_{j,d}\} \leftarrow \{k_i, f_i\}$
1.5: $\{K_l, F_l\} \leftarrow \{k_i, f_i\}$
1.6: **End For**
2: **For** each service $s_i \in S$
3 tf-idf(s_i, $\{K, F\}$)
4: $revise(s_i); revise(\{k_{j,d}, f_{j,d}\})$
5: **End For**
6: **Loop** $(remains(\{k_{j,d}, f_{j,d}\}))$ **do**
7: **For** each service $s_i \in S$
8: $build\_vector\_space(s_i)$ ←tf-idf-df(s_i)
9: **End For**
10: SVM(S)
11: **For** each keyword $k_i \in K_d$
12: tf-irf(k_i)
13: $\{k_{j,d}, f_{j,d}\}$←rerank(K_d)
14: **End For**
15: **goto** Step 2
16:**End Loop**

Different from the traditional waterfall-like SVM methodology, as shown in Fig. 4, our extended SVM methodology does not stop at one single round. Instead, we adopt an iterative approach to incrementally enhance categorization quality. As explained in Equation (2), domain-specific keyword ranking can be used to highlight the importance of specific keywords (value of the attribute in the vector) when building the vector space. After one iteration, the ranking of the keywords in a domain may be changed. In other words, the domain knowledge may be enriched. Therefore, such enhanced domain knowledge can be reapplied to the KF-IDF-DF and reconstruct the vector space, and rerun the entire process for another round (steps 6~16).

As shown in Table II, the input of the extended SVM methodology is a collection of services in one specific domain; the output is enriched domain knowledge in the form of a list of ranked keywords. As the initialization phase, step 1 reads in all services, extracts all terms, normalizes them (e.g., applying Porter stemming algorithm [21] for prefix and affix removal and Wordnet [http://www.wordnet.princeton.edu] for solving the synonym issue), and adds them to the domain ontology (step 1.4) and repository ontology (step 1.5). The repository ontology will be used for the TF-IDF algorithm to remove insignificant terms (steps 2~5).

It is known that acquisition of domain ontologies is difficult and costly [22]. In the context of a service repository, meanwhile, services are continuously registered into domains. Therefore, the corresponding domain ontologies have to be incrementally built and enriched.

As shown in Fig. 4, we iteratively extract domain ontology (to revise keyword ranking) based on SVM-based service categorization process. The initial keyword ranking is obtained by counting word frequency (step 1.4) and then removing insignificant terms through the TF-IDF algorithm (step 4). Afterwards, each round of SVM-based categorization process (steps 6~16) will revise the keyword ranking; and the resulting list will serve as an input for the next iteration of categorization process. The termination criteria can be set when the resulting keyword ranking remains unchanged (for example, when the top 50 keywords ranking remains unchanged in new iterations).

Extracted domain ontology can be further organized into an ontology graph. Fig. 5 shows a segment of the

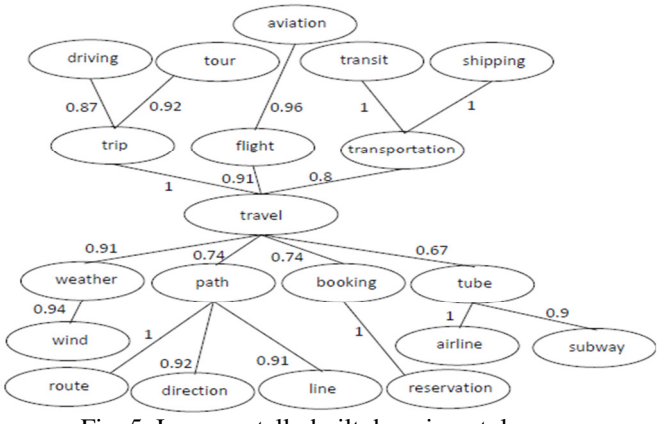

Fig. 5. Incrementally built domain ontology.

ontology graph in our study. To build a useful domain ontology, we will try to keep only high-rank keywords (for example, the top 200 keywords in the domain) in the domain ontology. The method introduced in [7] can be used to automatically build a hierarchical ontology graph. An edge between two nodes represents the similarity between the two keywords, whose original value can be retrieved from the Wordnet. Popularity of a keyword can be attached to the corresponding node as an annotation (attribute), which can be used for future ontology-based services discovery.

VI. Experiments and Discussions

We have conducted a series of experiments to evaluate our proposed technique and methodology.

A. Experimental setup

We use Web 2.0 service/mashup registry (http://www.programmableweb.com, PWeb) as our testbed, mainly because of its popularity. Currently, PWeb has accumulated over 3,500 APIs (data gathered on Jul. 25th, 2011). PWeb provides a set of programmable APIs to allow users to fetch some descriptive data about their registered services: summary, tag, description and category information. An APIkey has to be applied and granted before using these APIs. Upon request, an XML file. called Atom feed document, will be responded carrying requested information. However, one Atom feed document can carry information for up to 20 services. Therefore, to retrieve information for the entire set of over 3,500 services, we had to send many requests. We found that all services can be extracted in this approach. Our code continuously stopped at some service pages without meaningful error messages.

Therefore, we tried a more labor-intensive approach, using a crawler to go to every service page and extract relative data based on embedded tags. The crawler we adopted is Heritrix (http://crawler.archive.org/). However, crawling all comprising hierarchical pages from such a big website is not a trivial task. The crawler kept on stopping in the process, even after our many times of efforts. As a result, we had to leverage both methods and combine the information from both of them to receive a more complete picture of the service repository.

By examining the various services retrieved from the PWeb, we selected its three related categories: travel, shipping, and weather. The rationale is as follows. First, the three categories are closely related to the concept of "travel." Second, combing the services from the three categories will provide a relatively larger training set: travel (87), shipping (11), and weather (15).

We implemented an SVM engine leveraging the LIBSVM [23], a library with Java APIs for supporting SVM-based classification and regression analysis.

All of our algorithms and experiments are developed in Java, and conducted on PCs with Intel Core 2 CPU T7300, @2 GHz and 2 GB main memory, running the Windows XP operating system.

B. Domain Ontology Construction

We designed experiments to evaluate the effectiveness of building domain ontology using our method. The first step is to build a base line. From the three selected categories, we wrote code to identify all terms in the forms of verb and noun, and then rank all of them by their frequency in the domain. Then focusing on the top 100 terms, three graduate/undergraduate students were asked to manually adjust the rankings of the terms and sort them by their relevance to the concept of *travel*.

Based on the constructed baseline of keyword ranking, we ran three techniques (TF-IDF|D, MI, and our approach) over the entire PWeb testbed. Particularly, applying TF-IDF alone requires that we adjust its formula, so that a term frequency is counted in the entire domain (including all services in the domain) instead of in one document (service). Therefore, we use TF-IDF|D to refer to Term Frequency – Inverse Domain Frequency.

$$tf\text{-}idf(k,d)|D = \frac{num(k,d)}{\sum_{k_i \epsilon D} num(k_i, d_j)} \times log \frac{|D|}{1 + |\{d: k \in d\}|}$$

Each method will result in a ranked keyword list. Table III lists the top 10 ranked keywords identified by the three approaches, respectively.

Table III. The top 10 ranked keywords.

MI	TF-IDF|D	KF-IRF
travel	Travel	travel
booking	Flight	weather
hotel	Weather	booking
weather	Transit	flight
transit	Taxi	transit
shipping	Booking	hotel
social	Airport	shipping
information	Traveler	Airport
flight	Hotel	Forecast
mapping	Trip	Trip

To precisely compare the effectiveness of generating domain ontology using the three methods, we calculate the standard deviation for each method as follows:

$$d(test, base) = \sqrt{\frac{\sum_{i=1}^{n} \lfloor (rank_{test}(test_i) - rank_{base}(test_i))/\gamma \rfloor^2}{n}} \quad (4)$$

where *base* denotes the normalized ranked keyword list generated by domain experts; *test* denotes the ranked keyword list generated by a categorization method; γ is a normalization factor.

The basic idea is to measure diversity of how much variation between the ranking of each keyword in one approach from that in the baseline. The standard deviation of one method checks the dispersion of all comprising keywords. A lower standard deviation indicates that the ranking method tends to be more accurate. Because domain ontology has to be incrementally built, the ranking of a specific keyword in a domain is not absolute.

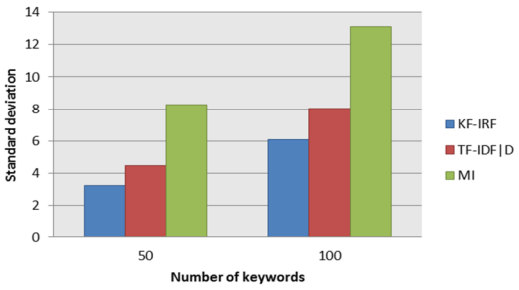

Fig. 6. Comparison of keyword ranking.

Meanwhile, our baseline ranking is also depended on human decisions. Therefore, we use a normalization factor to eliminate such random factors. For example, a keyword ranked as second or third is considered no difference. When setting $\gamma = 5$, we obtained the effectiveness comparison among the three approaches, by considering the top 50 keywords and top 100 keywords, respectively (on the left and right). As shown in Fig. 6, our method shows the lowest standard deviation.

As discussed in Section V, our ontology-empowered SVM methodology adopts iterations to incrementally reach better service categorization accuracy. We thus studied the convergence rate of our approach. We set the termination criterion as the standard deviation remains unchanged in two iterations (without losing much accuracy, we consider the top 100 keywords). As shown in Fig. 7, using different scales of testing sets including 80, 200, 500, 1000, 3000 services, the convergence rate remains at 3. In other words, at most three iterations are needed to get the best categorization results.

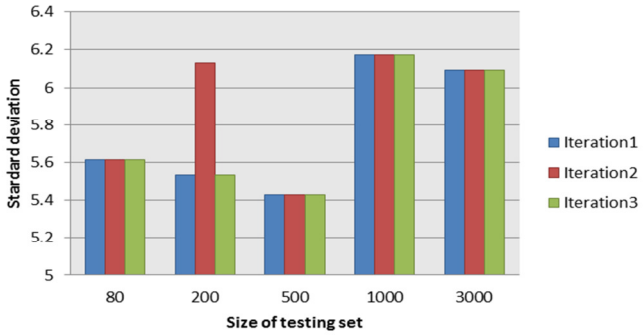

Fig. 7. Comparison between different iterations.

C. Accuracy Analysis

We designed a set of experiments to compare the service categorization accuracy between using our ontology-empowered SVM methodology and directly applying the SVM methodology. We adopted two indexes, *Precision* and *Recall*, to evaluate the performance of service categorization. *Precision* is the fraction of the services that are correctly considered as relevant to the target domain; *Recall* is the fraction of the relevant services that has been correctly categorized into the target domain. *Precision* and *Recall* are formally defined as follows:

$$Precision = \frac{|C_R|}{|C|}; Recall = \frac{|C_R|}{|R|}$$

where,

- C represents the set of services that are categorized as relevant to a domain; $|C|$ denotes the number of services in C.
- R represents the set services that should be categorized as relevant; $|R|$ indicates the number of services in R.
- C_R represents the set of services as the intersection of the sets R and C; $|C_R|$ indicates the number of services in C_R.

We evaluated the Precision/Recall values of using our methodology and using the traditional SVM methodology, for categorizing travel-related services. The same experiments were conducted over difference scales of testing sets containing different number of services: 80, 200, 500, 1000, 3000. As explained in Section V, the experiments were repeated until the termination criterion is met (the precision rate remains). Table IV summarizes our findings.

Table IV. Categorization accuracy comparisons.

Test# \ #test		80	200	500	1000	3000
TF-IDF	precision	95.65%	85.19%	74.19%	64.86%	40%
	recall	91.67%	95.83%	95.83%	100%	100%
Iteration 1	precision	100%	92.31%	82.76%	68.57%	45.28%
	recall	95.83%	100%	100%	100%	100%
Iteration 2	precision	100%	88.89%	82.76%	64.86%	44.44%
	recall	95.83%	100%	100%	100%	100%
Iteration 3	precision	100%	92.31%	82.76%	66.67%	44.44%
	recall	95.83%	100%	100%	100%	100%

As shown in Table IV, our methodology outperforms the traditional SVM in both precision and recall indexes. For the Recall index, as long as the scale of the testing set becomes large enough (more than 80 services), our methodology always reaches 100%, meaning that our method is good at identifying all services containing significant domain-related keywords. For the Recall index, our method outperforms the traditional SVM even from the first iteration, and terminates quickly (no more than 3 times for the test set containing 3,000 services). In general, using our method, values of both precision and recall indexes become better in newer iterations, until remaining unchanged when iterations stop.

In our experiments, we used the set of services originally categorized by the service creators as the reference points: only the 87+11+15=113 services are considered travel-related. When examining the other services that are categorized as travel-related by our methodology, we found that their descriptions do contain high-rank keywords according to our constructed domain ontology. For example, the service named "Yahoo Traffic" is categorized as "others" on the PWeb. However, its descriptions contain several high-rank travel-related

keywords including: traffic (7:15), transit (1:5), route (1:21). A number pair (X:Y) represents the appearance frequency of the keyword in the service descriptions and the ranking of the keyword in the domain, respectively.

Such findings indicate that, our methodology has the ability to: 1) justify existing categorization of services; 2) identify services that belong to multiple domains; and 3) find service that should be categorized into different domains. For the second and third types of services, we chose to add tags to them, so that such information will support further services discovery.

VII. Conclusions

The unique structural feature of service repositories and their hidden domain knowledge inspire us in extending the traditional SVM methodology. Our proposed technique is particularly valuable in building service search engines oriented to small- to middle-scale service repositories.

We plan to continue our research in the following directions. First, we will study the effectiveness and efficiency of our approach on every category in the PWeb. Second, we will explore SVM techniques that can categorize multiple domains. Third, we will use our technique to enhance existing service repository-oriented service search engine.

Acknowledgement

The authors thank Yijie Shen, Yueting Yang, and Ran Chen for their coding help in the project. The work described in this paper is partially supported by the National Basic Research Program of China (973) under Grant No. 2007CB310801, the National Natural Science Foundation of China under Grant No. 60970017, the Fundamental Research Funds for the Central Universities No. 3101034, and the National Science Foundation of USA under grant IIS-0959215.

References

[1] NIST, "Cloud Computing Definition", 2011, Available from: csrc.nist.gov/publications/drafts/800-145/Draft-SP-800-145_cloud-definition.pdf.

[2] L.-J. Zhang, J. Zhang, and H. Cai, *Services Computing*, 2007: Springer.

[3] K. He, J. Wang, and P. Liang, "Towards Semantic Interoperability Aggregation in Service Requirements Refinement", *Journal of Computer Science and Technology*, 2010, 25(6): pp. 1103-1117.

[4] A. Segev and Q.Z. Sheng, "Bootstrapping Ontologies for Web Services", *IEEE Transactions on Services Computing (TSC)*, Dec. 2010, 2010: pp. Preprints.

[5] Y.-J. Lee and C.-S. Kim, "A Learning Ontology Method for RESTful Semantic Web Services", in Proceedings of *IEEE International Conference on Web Services (ICWS)*, 2011, Washington, DC, USA, pp. 251-258.

[6] X. Liu, Q. Zhao, G. Huang, H. Mei, and T. Teng, "Composing Data-Driven Service Mashups with Tag-based Semantic Annotations", in Proceedings of *IEEE International Conference on Web Services (ICWS)*, 2011, Washington, DC, USA, pp. 243-250.

[7] M. Zhou, S. Bao, X. Wu, and Y. Yu, "An Unsupervised Model for Exploring Hierarchical Semantics from Social Annotations", in Proceedings of *6th International Semantic Web and 2nd Asian Semantic Web Conference (ICSW/ASWC)*, 2007, pp. 680-693.

[8] M.D. Wilkinson, L. McCarthy, B. Vandervalk, D. Withers, E. Kawas, and S. Samadian, "SADI, SHARE, and the in silico Scientific Method". 2010, BMC Bioinformatics 11(suppl 12):S7.

[9] L. Zhang and B. Li, "Requirements Driven Dynamic Business Process Composition for Web Services Solutions", *Journal of Grid Computing*, 2004, 2: pp. 121-140.

[10] M. Klusch, B. Fries, and K. Sycara, "Automated Semantic Web Service Discovery with OWLS-MX", in Proceedings of *ACM International Conference on Autonomous Agents*, 2006, Hakodate, Japan, May 8-12, pp. 915-922.

[11] M. Klusch and F. Kaufer, "WSMO-MX: A Hybrid Semantic Web Service Matchmaker", *Web Intelligence and Agent Systems*, Jan., 2009, 7(1): pp. 23-42.

[12] M.L. Sbodio, D. Martin, and C. Moulin, "Discovering Semantic Web Services using SPARQL and Intelligent Agents", *Web Semantics: Science, Services and Agents on the World Wide Web*, Nov., 2010, 8(4): pp. 310-328.

[13] M. Junghans, S. Agarwal, and R. Studer, "Towards Practical Semantic Web Service Discovery", *Lecture Notes in Computer Science (The Semantic Web: Research and Applications)*, 2010, 6089/2010: pp. 15-29.

[14] Y. Yang and X. Liu, "A Re-Examination of Text Categorization Methods", in Proceedings of *The 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1999, pp. 42-49.

[15] J. Zhang, R. Madduri, W. Tan, K. Deichl, J. Alexander, and I. Foster, "Toward Semantics Empowered Biomedical Web Services", in Proceedings of *IEEE International Conference on Web Services (ICWS)*, 2011, Washington DC, USA, Jul. 4-9, pp. 371-378.

[16] S. Dasgupta, S. Bhat, and Y. Lee, "Taxonomic Clustering and Query Matching for Efficient Service Discovery", in Proceedings of *IEEE International Conference on Web Services (ICWS)*, 2011, Washington, DC, USA, pp. 363-370.

[17] M.Á. Corella and P. Castells, "A Heuristic Approach to Semantic Web Services Classification", in Proceedings of *10th International Conference on Knowledge-Based & Intelligent Information & Engineering Systems (KES)*, 2006, pp. 598-605.

[18] D. Bianchini, V. Antonellis, B. Pernici, and P. Plebani, "Ontology-Based Methodology for e-Service Discovery", *Information Systems*, Jun.-Jul., 2006, 31(4-5): pp. 361-380.

[19] K.S. Jones, "A Statistical Interpretation of Term Specificity and Its Application in Retrieval", *Journal of Documentation*, 1972, 28(1): pp. 11-21.

[20] B.E. Boser, I. Guyon, and V. Vapnik, "A Training Algorithm for Optimal Margin Classiers", in Proceedings of *the 5th ACM Annual Workshop on Computational Learning Theory*, 1992, pp. 144-152.

[21] M. Porter, "An Algorithm for Suffix Stripping Program", *Automated Library and Information Systems*, 1980, 14(3): pp. 130-137.

[22] M. Sabou, C. Wroe, C. Goble, and G. Mishne, "Learning Domain Ontologies for Web Service Descriptions: An Experiment in Bioinformatics", in Proceedings of *The 14th ACM International Conference on World Wide Web (WWW)*, 2005, pp. 190-198.

[23] C.-C. Chang and C.-J. Lin, "LIBSVM: a Library for Support Vector Machines", 2011, Available from: http://www.csie.ntu.edu.tw/~cjlin/libsvm/.