Leveraging Incrementally Enriched Domain Knowledge to Enhance Service Categorization

Jia Zhang, Carnegie Mellon University, Silicon Valley, USA

Jian Wang, State Key Lab of Software Engineering, Computer School, Wuhan University, China

Patrick C.K. Hung, University of Ontario Institute of Technology, Canada

Zheng Li, State Key Lab of Software Engineering, Computer School, Wuhan University, China

Neng Zhang, State Key Lab of Software Engineering, Computer School, Wuhan University, China

Keqing He, State Key Lab of Software Engineering, Computer School, Wuhan University, China

ABSTRACT

This paper reports the authors' study over an open service and mashup repository, ProgrammableWeb, which groups stored services into predefined categories. Leveraging such a unique structural feature and hidden domain knowledge of the service repository, they extend the Support Vector Machine (SVM)-based text classification technique to enhance service-oriented categorization. An iterative approach is presented to automatically verify and adjust service categorization, which will incrementally enrich domain ontology and in turn enhance the accuracy of service categorization.

Keywords: *Enriched Domain Knowledge, Mashup Repository, Open Service, ProgrammableWeb, Service Categorization, Support Vector Machine*

INTRODUCTION

The ultimate goal of cloud computing is to enable everything as a service (XaaS) (NIST, 2011), where Software as a Service (SaaS) is one core objective. While software being published as universally accessible Web services, users can leverage existing services and quickly compose new value-added business processes and services. However, as cloud has become an unprecedented driving factor to encourage people to publish and share software as services, how to effectively and efficiently discover interested services from a "cloud" of resources remains a big challenge.

DOI: 10.4018/jwsr.2012070103

Copyright © 2012, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

One major technique is to establish service registries (Zhang et al., 2007) as centralized "service yellow pages" to help users find interested services. Earlier Universal Description, Discovery, and Integration (UDDI) registries are going out of date, however. Two major reasons are their tight binding to SOAP/WSDL services and their over standardization. In recent years, REpresentational State Transfer (REST) service, a light-weight HTTP Request/Response-based service style, has rapidly emerged and caught significant momentum (Pautasso et al., 2008). Thus, many non-UDDI service registries have been developed. Among them, The ProgrammableWeb (PW, http://www.programmableweb.com, acquired by Alcatel-Lucent in 2010) has become a popular one.

Without adopting the heavy UDDI standard, ProgrammableWeb provides a repository that allows people to publish reusable Web services in various formats (protocols including REST and SOAP), called Web APIs. Meanwhile, PW allows people to publish API-based applications, called mashups. A mashup represents a value-added business process leveraging one or more existing APIs published in PW. Such a light-weight service repository has attracted extensive attention. Since its inception in late 2005, the number of services published at PW has increased rapidly. Up to September 7, 2012, 7190 services and 6,763 mashups have been published at PW. Among the published services, 70% are REST services, 21% are SOAP services, 5% are JavaScript services, and 2% are XML-RPC services. Since APIs at PW represent reusable service components, throughout this paper, we will use the terms *API* and *service* interchangeably.

As the number of services accumulates at PW, it is important to facilitate users in querying and finding interested services (Gomadam et al., 2008). However, the current querying power at PW is limited. At publishing time, service providers are allowed to attach some user-defined name tags. Unlike UDDI that intends to regulate a comprehensive ontology system, ProgrammableWeb adopts a straightforward strategy. Every service is manually categorized into one of a preset list of domains (68 domains up to September 7, 2012) (Arabshian et al., 2012). The assigned domain name and provider-defined tags associated with the service are combined to support keyword-based search function.

Such an API search mechanism may cause confusion and decrease search accuracy. First, the manual process of service categorization may not be accurate. As a matter of fact, API "ShowMyIP" was originally classified in domain "Mapping"; and was moved to domain "Internet" later. In the metadata of the API, its description, summary and tags contain some representative keywords of domain "Internet" such as "IP" and "Internet." Second, it may be difficult to decide one single domain for some APIs, because some predefined domains overlap with each other conceptually. For example, domains Travel, Transportation, and Weather share many common concepts. For another example, the aforementioned API "ShowMyIP" does relate to the category "Mapping" in addition to the category of "Internet." Third, PW presets a special domain named "Other" and a significant number of services are found left in the category. Currently, 199 services are listed in the category of "Other," which is the top 14th category with the most number of services (over the entire 68 preset domains). Fourth, user-defined tags may be *ad hoc* and inconsistent, and sometimes lack of tag (Gomadam et al., 2008), cannot effectively help users find their interested services.

Table 1 shows some motivating examples. The second column shows the category name assigned to the Web API (whose name is in the first column) by ProgrammableWeb. However, as shown in the third column, our study indicates that these Web APIs should belong to several categories (domains). The names of the Web APIs even imply such cross-relationships. For example, users should be able to find the API "BestParking" from the Travel, Transportation, or Mapping categories. (The numbers represent the similarity between a Web API to a corresponding category. For another example, the API "StrikeIron Address Distance" is listed in

Copyright © 2012, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

Table 1. Motivating examples

API Name	Original Category	Categories Related
BestParking	Travel	Travel (0.9261) Transportation (0.2732) Mapping (0.4139)
RunwayFinder	Travel	Travel (0.4864) Transportation(0.1715) Mapping (0.8629)
Trazzler	Travel	Travel (0.9319) Transportation (0.2644) Mapping (0.6502)
TripTracker	Travel	Travel (0.9327) Transportation (0.2703) Mapping (0.8807)
Urban Mapping Mass Transit Proximity	Travel	Travel (0.5158) Transportation (0.8632) Mapping (0.4099)
Mapnificent	Transportation	Travel (0.9337) Transportation (0.6907) Mapping (0.7003)
NAC Real-time Routing	Mapping	Travel (0.9289) Transportation (0.2755) Mapping (0.7446)
StrikeIron Address Distance	Other	Travel (0.5586) Transportation (0.2564) Mapping (0.5805)

the category of "Other." However, our study of the descriptions of the API revealed that it should be listed in three categories: Travel, Transportation, and Mapping. The details of how to obtain such similarity will be discussed.)

Therefore, there exists a need for existing non-UDDI service repositories to enhance their service categorization accuracy, and in turn enhance their query and search ability. While such an ability will attract more service users to visit the repository, it will also attract more service providers to publish their services at the site.

This paper reports our approach that adapts the Support Vector Machine (SVM) technique to enhance service categorization on service repositories. While an SVM engine being constructed as a black-box service, we propose a pair of customizable input and output functions to increase its categorization accuracy based on incrementally enriched and refined domain knowledge. Our extended SVM technique has three major contributions: one is to verify and refine existing author/user-centered service categorization; second is to enrich domain ontology; third is to support in automatically annotating (tagging) services. Our experimental results show that our approach surpasses the tag-based search approach used by ProgrammableWeb, as well as directly applying text document classification approaches to service categorization.

The remainder of the paper is organized as follows. First, we discuss related work. Then, we introduce our extensions to SVM. Afterwards, we present our ontology-empowered SVM technique, and how it can be used to support service categorization, respectively. Followed

Copyright © 2012, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

by a presentation experimental settings and preliminary results and analysis. Finally, we then draw conclusions.

RELATED WORK

The work presented in this paper will further advance the research in semantics-empowered Web services discovery. Although the literature has witnessed a rich set of research results in the area, most of them are oriented to SOAP services supported by a formal service description model embedded in WSDL. OWL-S (Coalition, 2004) describes services in terms of profile, process, and grounding. SAWSDL (Kopecky et al., 2007) proposes to annotate WSDL components such as inputs/outputs with references to ontologies. WSMO (Vitvar et al., 2007) advocates to model Web services using four major elements of ontologies, Web services, goals, and mediators. Paolucci et al. (2002) map DAML-S service profiles to UDDI records; Sivashanmugam et al. (2004) map semantic Web service descriptions in SAWSDL to UDDI. Dong et al. (2004) present a clustering method to cluster parameters present in inputs and outputs of WSDL operations. Hess and Kushmerick (2003) apply Naive Bayes and SVM machine learning methods to categorize WSDL files in manually defined hierarchies. In contrast, this research accumulates semantic data from fragmental descriptive elements from various types of services not limited to SOAP services.

Most of semantic services discovery research performs profile-based service signature (I/O) matching (Klusch et al., 2006). OWLS-MX (Klusch et al., 2006) and WSMO-MX (Klusch & Kaufer 2009) propose to combine logic-based reasoning and syntactic concept similarity computations in OWL-S. Sbodio et al. (2010) propose to use SPARQL as a formal language to describe the pre- and post-conditions of services. Junghans et al. (2010) propose a practical formalism to describe functionalities and service requests. In contrast, we focus on enriching domain ontology and leveraging it to classify services.

Since the *ad hoc* standard Web Service Descriptions Language (WSDL) does not carry semantic information, automatic service discovery remains a challenge. As one important aspect, service categorization research, i.e., service classification, aims to explore a mechanism to automatically verify the domain(s) to which a service belongs. Such research efforts generally follow two directions. One direction is to extend WSDL with abilities to carry semantic information of services. Leveraging semantic web technologies, a number of languages have been developed, including OWL-S (Martin et al., 2007), WSDL-S (Akkiraju et al., 2005), and WSMO (Klusch & Kaufer, 2009).

The other direction focuses on eliciting semantic knowledge from textual description documents associated with Web services, leveraging the Information Retrieval (IR) techniques. Typical strategy is to extract features from Web service description documents using IR methods such as text retrieval, vector model and clustering. However, few of the existing efforts address the issue of high dimensionality of feature space. If Web service description documents carry a large number of different terms, which are treated as features by IR classifiers, a large feature space may lead to low performance and low accuracy.

Some researchers exploit the structure of WSDL documents to categorize Web services. For example, our previous work (Zhang et al., 2011) and Liu et al. (2011) leverage bipartite graphs to calculate similarity between Web services based on WSDL documents; Patil et al. (2004) adopt a schemamatching similarity matching method. In contrast, our work presented in this paper focuses on leveraging domain knowledge to highlight significant dimensions of feature space.

The IR community has created a wealth of clustering algorithms and techniques (Yang & Liu, 1999). In contrast to these general-purpose

Copyright © 2012, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

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.

As REST services become popular, many researchers have started to explore how to add semantics to REST services. SA-REST (Sheth et al., 2007) semantically annotates REST services by adding annotations to Web pages that describe the services. Semantic Bridge for Web Services (SBWS) (Battle and Benson 2008) provides custom annotations to WADL (Web Application Description Language) (Hadley, 2009) documents. However, these annotate processes are difficult and costly (Lee & Kim, 2011).

Generic search engines such as Google can help discover Web services (APIs) and mashups through keywords. The ProgrammableWeb site leverages Google search facility to provide tag-based search function. Studies have indicated the limitation of keyword-based search method without considering semantic data (Lee & Kim, 2011). In contrast, we propose an approach to verify and refine user tags to enhance service and mashup discovery.

Gomadam et al. (2008) adopted the traditional term vector similarity approach and faceted classification (Ranganathan, 1962) to categorize the Web APIs at ProgrammableWeb. A term vector and a tag vector are built for each domain. Each API is compared against the term vector and the tag vector of each domain using the cosine similarity approach. They also present a PageRank-inspired service utilization (serviut) approach to rank services in a category. In contrast, our approach proposes to leverage domain knowledge to enhance traditional SVM (a popular text categorization method) in the context of service categorization. In addition, we leverage their serviut rank method as part of our service recommendation criteria.

Domain-specific ontologies are normally developed manually through the collaboration of highly skilled domain experts and ontology engineers (Lee & Kim, 2011). A number of works have been reported on automatic acquisition of semantic information to build ontology. Sabou et al. (2005b) present an approach that extracts domain ontologies from textual documents associated with Web services. Segev and Zheng (2010) propose an ontology bootstrapping method that automatically generates concepts and their relations in a domain from WSDL files. Our earlier research presents an approach that automatically extract semantic information from WSDL files, based on the hypothesis that service developers usually follow naming convention (Zhang et al., 2011). Lee and Kim (2011) use Sun's WADL (Hadley, 2009) (SUN) to describe syntactical information of REST services, and study how to enable similarity search over RESTful services based on their syntactic and semantic descriptions. A learning ontology method is proposed to semantically describe REST services, which groups parameter names of services into semantically meaningful concepts and capture relationships between words contained in a parameter name. In contrast, we focus on incrementally build domain concepts and their relationships from fragmental semantic data.

Liu et al. (2011) 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 semantic information including descriptions, tags, and categorization information.

Zhang and Li introduce the concept of service cluster (Zhang & Li, 2004) 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.

EXTENDED SVM TECHNIQUE

Existing service categorization usually adopts Information Retrieval (IR) similarity models such as vector space models, probabilistic models, and information theory-based mod-

Copyright © 2012, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

els (Dasgupta et al., 2011). Their underlying technique is semantic similarity measurement between services, either based on keywords (Corella & Castells, 2006) or on ontology (Bianchini et al., 2006). The former method uses the Term Frequency – Inverse Document Frequency (TF-IDF) (Jones, 1972) technique to build a vector space; the latter leverages taxonomy, information content (IC), or concept property to calculate similarity between services. However, two reasons make service categorization challenging if we directly apply these methods. First, services are usually published without comprehensive descriptions. For example, we can only find on ProgrammableWeb service information from its name, domain name, user-specified tags, summary, and short description. Second, it is known that acquisition of domain ontologies is difficult and costly (Sabou et al., 2005a; Lee & Kim, 2011). Therefore, this research aims to address these challenges by exploring an approach that incrementally establishes domain knowledge, and leverages such knowledge to automatically verify and enhance service categorization.

Applying SVM Technique

It is known that the Support Vector Machine (SVM) method outperforms (accuracy of classification) other text categorization methods (Yang & Liu, 1999), especially when the number of dimensions of the documents under consideration is significant. Verifying and justifying the categorization of services may not be a trivial task, since service descriptions are free text documents that may comprise various terms. Therefore, it is suitable to apply the SVM method for service categorization.

We construct a SVM model by formalizing service classification as an optimization problem (Boser et al., 1992). 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 an n-dimension vector, and $y_i \in \{1, -1\}$ indicating whether a service belongs to the domain or not. The SVM model aims to find a solution to:

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

subject to:

$$y_i\left(w^{\mathrm{T}}\varnothing\left(x_i\right)+\mathrm{b}\right)\geq 1-\xi_i,\ \xi_i \geq 0\ .$$

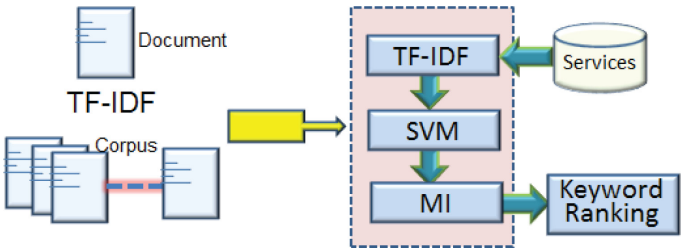
Training vectors x_i are mapped into a higher dimensional space by the function $\varnothing$. SVM will find a linear separating hyperplane with the maximal margin in the higher dimensional space. $C > 0$ is a penalty parameter of the error service.

We construct a single-class SVM aiming to find a line to split the domain relevance and domain irrelevance in high-dimensional vectors (Yang & Liu, 1999). As we discussed earlier, a service may fit in one domain the best while naturally relating to multiple domains. Our goal is to find out all such relative domains, as well as the similarity between one service and each related domain (details will be discussed later). That is why we decided to use the single-class SVM.

Figure 1 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, a vector space is constructed from all input services based on the TF-IDF formula. Second, a SVM classification model is built based on a selected training service set, and then runs over the entire service repository (i.e., testing service 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

Copyright © 2012, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

Figure 1. Directly applying SVM technique



to represent the ranking of the keyword in the domain.

Issues and Strategy

Applying the above approach to the service repository ProgrammableWeb, we found that the categorization results are not satisfactory in a number of domains. Analyzing the reasons, we identified several significant issues. Successful 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. For example, the Internet domain contains 472 Web APIs, and the Social domain contains 407 APIs. However, some domains contain a smaller number of services. For example, the *Dictionary* category contains 14 Web APIs, and the Politics domain contains 5 APIs. Such small training sets cannot train a good SVM model. Meanwhile, *ad hoc* service categorization decided by ProgrammableWeb may not always be accurate (Arabshian et al., 2012).

After carefully examining ProgrammableWeb, we noticed one unique feature. The services registered at the repository are organized into 68 separate domains. In other words, services categorized within the same domain should share the same domain ontology. Thus, we made an important hypothesis: *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. The supporting assumption is that, descriptive information associated with published services in a domain represents ontology and domain knowledge of the domain. Such information includes service name, domain name, summary, tags, and short description. For SOAP services, descriptive information also includes operation names in WSDL files, usually following naming conventions and implying semantic intension (Zhang et al., 2011).

Our idea is to incrementally build knowledge for each domain based on information derived from its comprised registered services. We thus enhanced the algorithm used to measure keyword-based service similarity, i.e., TF-IDF, which does not take into account features in corresponding 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.

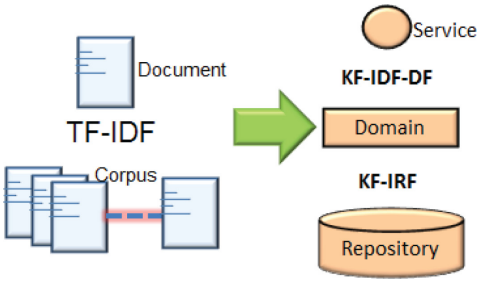
Enhancement to TF-IDF

Figure 2 illustrates our extensions to the traditional TF-IDF (Jones, 1972). 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.

$$tf - idf(w_{j,i}) = tf_{j,i} \times idf_j = \frac{c_{j,i}}{\Sigma_k c_{k,i}} \times \log \frac{|\{d\}|}{1 + |\{d : t_{j,i} \in d\}|} \quad (1)$$

Copyright © 2012, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

Figure 2. Extensions to TF-IDF



In contrast to TF-IDF that measures between one document and corpus (the entire set of documents), we break TF-IDF into two parts. As shown in Figure 2, two new concepts are introduced: (1) keyword frequency – inverse document frequency – domain frequency (KF-IDF-DF) and (2) 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. In the context of ProgrammableWeb, as well as similar tag-empowered service repositories, all tags are considered as highlighted keywords – important terms represent a domain. As the first step, we use ranked domain-specific keywords to represent the relationships among domain ontology.

Equation (2) shows how we calculate the significance of keyword (k) in service (s) regarding 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.

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

As new services are added into a domain, its domain ontology will increase. 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, considering 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 ($\beta \in [0,1]$) is set to be 1).

Equation (3) shows how we 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 maximum keyword frequency 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 (we will explain using examples below):

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

Consider two distribution scenarios as shown below, where the frequencies of a keyword in two repositories are the same (118). In scenario (a), the frequency of the keyword in domain d1 is 100, and its frequency in domain d2 is 18. In scenario (b), the keyword counts

Copyright © 2012, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

100 times in domain d1, and counts twice in nine other domains (d2~d10).

	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10
(a)	100	18								
(b)	100	2	2	2	2	**2**	2	2	2	2

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 ProgrammableWeb is a web service registry, many services in its comprising domains possess the same keyword "service" with high frequency. Such a keyword is insignificant in representing any domain. Therefore, the fraction $(1-\frac{\left|\{d:k\in d\}\right|}{|D|})$ in Equation (3) intends to lower the ranking of such keywords.

Meanwhile, in scenario (a), the keyword is likely to represent domain d1 (frequency 100) more than domain d2 (frequency 18). In scenario (b), the keyword is likely to represent domain d1 (frequency 100) more than domain d2 (frequency 2). The fraction $(\frac{num(k,d)}{\sum_{d_i\in D} num(k,d_i)})$ in Equation (3) reflects such a consideration.

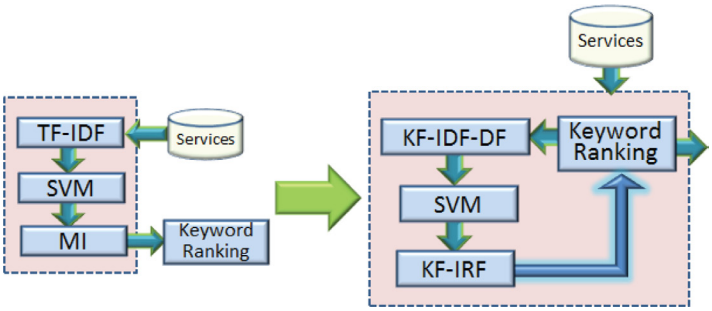
After calculating the $kf-irf_{k,d}$ value of each term in a domain, all terms in the domain can be sorted. Its order in the ranked list will be used in Equation (2).

SVM-EXTENDED METHODOLOGY

Based on our extensions to TF-IDF, we propose an ontology-empowered SVM methodology for service categorization. Figure 3 outlines our overall idea; and Figure 4 lists the 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 re-rank domain-related keywords (steps 11-14).

Different from the traditional waterfall-like SVM methodology, as shown in Figure 3, 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-

Figure 3. SVM-extended methodology for service categorization



Copyright © 2012, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

Figure 4. Ontology-empowered SVM approach

Algorithm: Extended SVM
Input: a collection of tagged 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**

IDF-DF and reconstruct the vector space, and rerun the entire process for another round (steps 6-16).

Figure 4 illustrates the pseudo code of the extended SVM methodology. The input is a collection of services in one specific domain (together with their related documents); 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 (Porter, 1980) 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 traditional TF-IDF algorithm to remove insignificant terms (steps 2-5).

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 Figure 3, we iteratively refine and enrich 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 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).

SERVICE CATEGORIZATION

Recall that our main goal is to facilitate service categorization and discovery. Our SVM-extended approach helps to verify whether a service belongs to a domain, i.e., whether an API falls into a category. As discussed earlier, an API may naturally show features of multiple categories. Therefore, we propose a two-phase

Copyright © 2012, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

method to verify service categorization, as shown in Figure 5.

Two-Phase Service Categorization

In phase 1, we intend to identify the candidate domains where a service shows corresponding features. In phase 2, we calculate the similarity between the service and the candidate domains. Using the extended-SVM approach discussed in the last section, an API is examined against each category and classified as either domain-relevant or domain-irrelevant. At the end of phase 1, a set of candidate categories are obtained for each service.

Although a service may belong to multiple domains, it may show different levels of relevance to different domains. Phase 2 thus intends to quantify the relevance of services to domains.

Numerous research work has been conducted to calculate the similarity between services. For example, Jiang et al. (2011) applied the collaborative filtering method to calculate similarity between services. In contrast, we calculate similarity between a service and a domain to evaluate the relevance of the service to the domain. Our key idea is to model a domain as a representative vector space, and then transform the service-domain relevance problem into a similarity computation problem between two vector spaces. If the similarity between a service and a domain exceeds a predefined threshold, the service is considered relate to the domain.

The rationale is that a domain can be represented by a keyword-oriented vector space comprising a collection of significant terms. Thus, a domain is modeled as a ranked vector of keywords derived from our SVM-extended algorithm discussed: $\overrightarrow{DKV}$ (Domain Keyword Vector). Our introduced KF-IRF (Equation 3) is used to calculate the ranking of a keyword in a domain. Note that the model can be refined through iterations of verification processes driven by comprising services. Without losing generality, only top N (e.g., 150) keywords with high significance will be considered. As discussed, each service can be modeled as a term vector: $\overrightarrow{SKV}$ (Service Keyword Vector). Currently we compute cosine similarity between $\overrightarrow{DKV}$ and $\overrightarrow{SKV}$ as below. Note that the

Figure 5. Two-phase service categorization

Algorithm: Two-phase service categorization.
Input: a collection of services (**S**) and a set of domains (**D**).
Output: categorized services with similarity.

1: *verify_service*(S, D)
1.1: **For** each domain $d_j \in D$
1.2: $C_{d_j} \leftarrow \emptyset$
1.3: **End Loop**
1.4: **For** each service $s_i \in S$
1.5: **For** each domain $d_j \in D$
1.6: flag←*extended_SVM*(s_i, d_j)
1.7: **If** flag is *true*
1.8: $C_{d_j} \leftarrow s_i$
1.9: **End If**
1.10: **End Loop**
1.11: **End Loop**
2: *calculate_similarity(C)*
2.1: **For** each domain $d_j \in D$
2.2: $\overrightarrow{DKV_j} \leftarrow extended\_SVM(D_j)$
2.3: **For** each service $s_i \in C_j$
2.4: $\overrightarrow{SKV_i} \leftarrow extended\_SVM(s_i, D)$
2.5: $WT(s_i) \leftarrow extended\_SVM(\overrightarrow{SKV_i})$
2.6: return $sim(WT(s_i), \overrightarrow{DKV})$
2.7: **End For**
2.8: **End For**

Copyright © 2012, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

relevance values shown in Table 1 are calculated through this approach.

$$\mathrm{sim}\left(\overrightarrow{SKV}_i, \overrightarrow{DKV}_j\right) = \frac{\overrightarrow{SKV}_i \cdot \overrightarrow{DKV}_j}{|\overrightarrow{SKV}_i| \cdot |\overrightarrow{DKV}_j|} \quad (4)$$

The formula intends to calculate the relevance of a given service *i* (API) and every domain *j* (category). A threshold can be preset, e.g., 0.6. If the resulted relevance exceeds the preset threshold, the API is considered relevant to the category.

Further Refinements

We further exploit obtained domain knowledge to verify and refine service categorization. Specially, we leverage service tags and keyword rankings. In the context of a service repository, tags act important roles to represent the semantics of a service. Therefore, we grant more weight to keywords highlighted by service providers as tags. For example, if a keyword appears in tags, its $kf - idf - df_{k,s,d}$ value can be amplified by a predefined scale (e.g., doubled).

Iterative SVM-extended approach leads to a ranked keyword list for a domain. Such obtained domain knowledge can be used to verify the results of our service categorization, from another direction. If a service carries representative keywords (i.e., ranked high) in a domain, and if such keywords appear multiple times (term frequency or TF), it is likely that the service has higher relevance to the domain. For example, *finance* and *stock* are the top two keywords in the resulted ranked keyword list from the *Financial* category. Consider two services which are considered relevant to domain *Financial* API_1 comprises terms *finance* (TF: 1) and *stock* (TF: 1), and API_2 comprises terms *finance* (TF: 3) and *stock* (TF: 3). Obviously API_2 has a closer relation to the *Financial* domain.

We defined the following method to verify our service categorization results. The weight w_{i,t_k} of term t_k in service *i*'s term vector $\overrightarrow{SKV}_i$ and the weight w_{j,t_k} of the t_k in domain *j*'s term vector $\overrightarrow{DKV}_j$ are defined as follows:

$$w_{i,t_k} = kf - irf_{j,t_k} \times log(1 + F_{i,t_k})$$

$$w_{j,t_k} = \begin{cases} kf - irf_{j,t_k} \times log(1 + F_{i,t_k}) & if\ F_{i,t_k} \geq 1 \\ kf - irf_{j,t_k} & Otherwise \end{cases}$$

where $kf - irf_{j,t_k}$ denotes the *kf-irf* value of term $t_k (k \leq N)$ in the keyword ranking list of domain j; F_{i,t_k} denotes the term frequency of term t_k in service *i*; and the purpose of logarithm operation is to eliminate the effects of term frequency. In other words, the top *N* keywords in domain *j* will be used to examine the term list in service *i*. If t_k does not exist in service *i*, then $w_{i,t_k} = 0$, and $w_{j,t_k} = kf - irf_{j,t_k}$, otherwise, both of them equal to $kf - irf_{j,t_k} \times log(1 + F_{i,t_k})$.

We will thus obtain a weighted term vector $\mathrm{WT}(\overrightarrow{SKV}_i)$ of service *i* defined as:

$$\mathrm{WT}(\overrightarrow{SKV}_i) = \left\{\left(t_k,\ w_{i,t_k}\right)\right\}$$

and a weighted term vector $\mathrm{WT}(\overrightarrow{DKV}_j)$ of domain *j* defined as:

$$\mathrm{WT}(\overrightarrow{DKV}_j) = \left\{\left(t_k,\ w_{j,t_k}\right)\right\}$$

The following formula verifies whether a service is relevant to a domain, by calculating the cosine similarity between the weighted term vectors $\overrightarrow{SKV}_i$ and $\overrightarrow{DKV}_j$. Compared to the equation (4), here we leverage the representative keywords in the domain to examine the results.

Copyright © 2012, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

$$\begin{aligned}
&\mathrm{sim}\left(\overrightarrow{SKV_i}, \overrightarrow{DKV_j}\right) \\
&= \frac{\mathrm{WT}\left(\overrightarrow{SKV_i}\right).\mathrm{WT}(\overrightarrow{DKV_j})}{\left|\mathrm{WT}\left(\overrightarrow{SKV_i}\right)\right| \mid \mathrm{WT}(\overrightarrow{DKV_j}) \mid} \\
&= \frac{\sum_{k=1}^{N}(w_{i,t_k} \times w_{j,t_k})}{\sqrt{\sum_{k=1}^{N} w_{i,t_k}{}^2} \times \sqrt{\sum_{k=1}^{N} w_{j,t_k}{}^2}}
\end{aligned} \quad (5)$$

EXPERIMENTS AND DISCUSSIONS

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

Testbed Establishment and Experimental Preparation

We use the publically available ProgrammableWeb as our testbed. The first step is to grab a complete image of the PW repository, meaning that we fetch the available metadata of each of its comprising Web APIs and mashups. PW 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[1], until the end of 2011, one Atom feed document can carry information for up to 20 services. Therefore, to retrieve information for the entire set of over thousands of services, we had to send many requests. We found that not all services can be extracted in this approach. The process was stopped at some service pages, whose XML documents are broken or because of network overtime.

An alternative is the more labor-intensive crawler approach, which goes to every corresponding Web page and extracts relative data based on embedded HTML tags. The crawler that we adopted is the open-source Heritrix (http://crawler.archive.org/). However, crawling all comprising hierarchical pages from such a large-scale website is not a trivial task. Its efficiency is much lower than the first PW API-based approach. As a result, we decided to adopt a hybrid approach. PW API method is adopted to retrieve as much as possible service metadata. When exceptions occur, the crawler method is used to fetch the corresponding metadata. Figure 6 lists the pseudo code of constructing an "image" of the PW site.

The key concern is the efficiency, since crawler takes much longer time to retrieve the metadata of a service than using PW APIs. We leveraged PW API to retrieve API metadata in pages with a capacity of 20 in default (Step 2). If an exception occurs when retrieving a page, we tuned the page capacity to 1 to fetch the rest services in the original page (Steps 8-13). A stack structure was used to store services that cannot be retrieved using PW API approach (Steps 7, 12). Then for each block in the stack comprising continuous service ids, the crawler was invoked to retrieve the corresponding service metadata, which will be integrated with the data retrieved through the PW API approach (Steps 14-18).

We implemented an SVM Web service leveraging the LIBSVM (Chang and Lin 2011), 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.

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 each of the 68 preset PW 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

Copyright © 2012, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

Figure 6. Pseudo code of constructing PW testbed

```
Algorithm: Construction of PW Image
Input: PW site
Output: Organized PW service/mashup metadata.
1: Initialization()
   lastNormal←-1; pageSize←20;
2: Loop (catchPage(pageSize)) do
3:    sid←generateServiceId()
4:    generateServiceMetaFile(sid)
5:     lastNormal←sid
6:    If (exception)
7:       stack←sid //exception point
8:       Loop (remainedService(page)) do
9:          sid←generateServiceId()
10:        catchPage(1)
11:        If (exception)
12:         Then stack←sid
13:       End Loop
14:       group_sids_in_stack()
15:       For each sid_block_i ∈ Stack
16:          lastNormal←fsid-1; nextNormal←lsid+1
17:          crawl(lastNormal, nextNormal)
18:       End For
19:       integrate()
20:    End If
21:End Loop
```

relevance to the domain name, according to their understanding and common sense.

Based on the constructed baseline of keyword ranking, we ran three techniques (TF-IDF, MI, and our KF-IRF) over the entire ProgrammableWeb testbed. According to Church and Hanks (1990), the Mutual Information (MI) between a term t and category c is defined as:

$$I\left(t,c\right)=\sum_{e_t\in\{0,1\}}\sum_{e_c\in\{0,1\}}p(e_t,e_c)\log\frac{p(e_t,e_c)}{p(e_t)p(e_c)}$$

Each method will result in a ranked keyword list. Table 2 lists the top 15 ranked keywords in the domain *Travel*, using three methods TF-IDF, MI, and KF-IRF, respectively. It is shown that our method (KF-IRF) performs slightly better than the other two methods. Some top keywords resulted from MI and TF-IDF are not relative to the domain, such as the keyword "availability" listed by MI, and the keywords "search" and "property" listed by TF-IDF.

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

$$d\left(test,base\right)=\sqrt{\frac{\sum_{i=1}^{n}\left|\left(rank_{test}\left(test_i\right)-rank_{base}\left(test_i\right)\right)/\gamma\right|^2}{n}}$$

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 the 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. Meanwhile,

Copyright © 2012, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

Table 2. Top 15 ranked keywords

TF-IDF	MI	KF-IRF
Travel	Travel	Travel
Booking	Booking	Booking
Hotel	Hotel	Hotel
Flight	Flight	Flight
Reservation	Reservation	Trip
Trip	Airline	reservation
Rental	Vacation	Vacation
Vacation	Airport	Rental
Airport	Trip	Airport
Airline	Rental	Airline
Search	Destination	Accommodation
Tours	Accommodation	Tours
Accommodation	Traveler	Destination
Guide	Transportation	Transit
Property	Availability	Traveler

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 Figure 7, our method shows the lowest standard deviation.

As discussed, our domain knowledge-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). Figure 8 summarizes our experimental results over the category *Travel*. Using different scales of testing sets including 1000, 3000, 5000, 7190 services, at most five iterations are needed to get the best categorization results.

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 three indexes, *Precision, Recall*, and *F-measure,* 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; *F-measure* is a weighted average of *Precision* and *Recall*. *Precision, Recall, and F-measure* are formally defined as follows.

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

Copyright © 2012, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

Figure 7. Comparison of keyword ranking

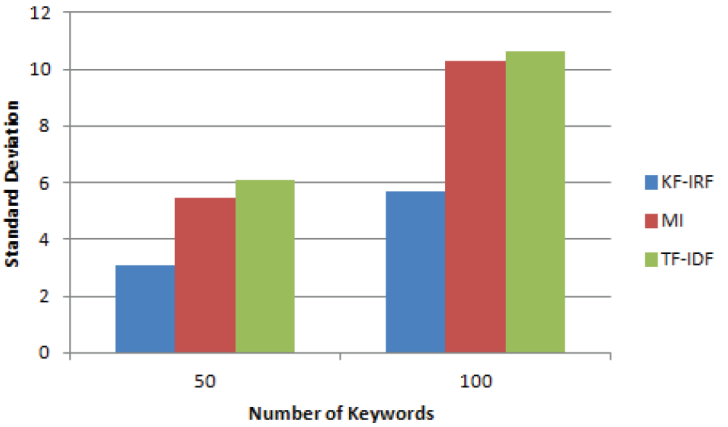
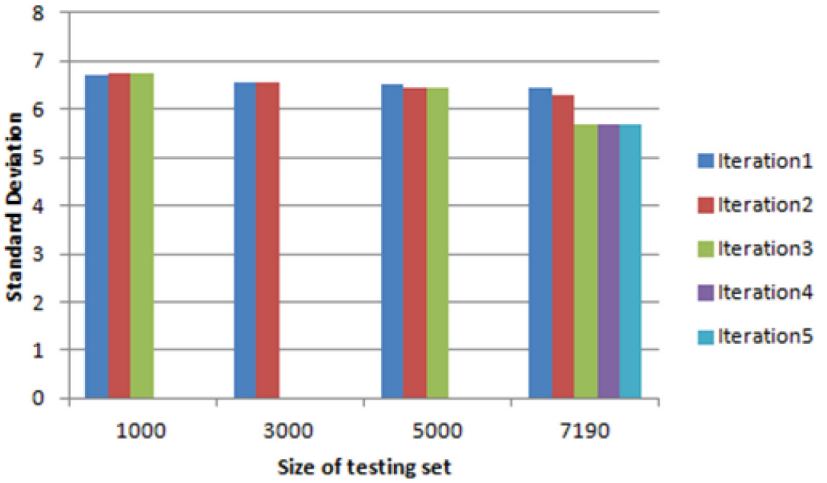


Figure 8. Comparison between different iterations



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

$$F - measure = 2.\frac{Precision \times Recall}{Precision + Recall}$$

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 applied our approach to evaluate all APIs in ProgrammableWeb against the 68 preset categories. The training set for each category comprised two parts: domain-relevant set and domain-irrelevant set. 80% of the APIs listed

Copyright © 2012, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

in the category by ProgrammableWeb formed the domain-relevant set; the same amount of APIs randomly selected from other categories formed the domain-irrelevant set. Table 3 lists the categories that have more than 100 APIs. As we discussed earlier, our approach also relies on the scale of training dataset. Therefore, we only examined the categories that can provide sufficient training sets.

Recall of most categories is 100 and the average recall reaches 99.78. For the precision index, the average reaches 82.07. However, the precision of the "Search" category is low (57.14, highlighted in Table 3). Our investigation found that the category name *Search* appears in quite a number of APIs, showing that the APIs provide the ability of information retrieval. That is why some APIs from other domains were categorized

Table 3. Evaluation of categorization accuracy

Domain	Single-class SVM			Knowledge-Empowered SVM		
	Precision (%)	Recall (%)	f-Measure (%)	Precision (%)	Recall (%)	F-Measure (%)
Advertising	69.66	100	82.12	84.97	100	91.87
Education	67.66	100	80.71	84.96	100	91.86
Email	66.51	100	79.89	83.88	100	91.23
Enterprise	62.25	99.65	76.63	73.28	100	84.58
Financial	82.45	99.29	90.09	77.90	99.29	87.30
Games	74.07	99.00	84.74	84.74	99.00	91.32
Government	81.49	100	89.80	91.6	100	95.61
Internet	63.22	99.14	77.21	83.30	99.78	90.80
Mapping	59.90	99.62	74.82	75.64	100	86.13
Messaging	70.60	100	82.77	86.36	100	92.68
Music	88.04	100	93.64	93.10	100	96.42
Payment	75	100	85.71	86.53	100	92.78
Photos	71.18	99.40	82.96	81.95	99.40	89.83
Reference	55.60	99.63	71.37	83.28	100	90.87
Science	73.79	100	84.92	91.81	99.53	95.51
Search	35.81	97.52	52.39	57.14	99.00	72.46
Security	60.93	97.76	75.07	90.41	98.50	94.28
Shopping	69.02	100	81.67	86.98	100	93.04
Social	57.54	99.26	72.85	75.65	100	86.13
Telephony	51.10	99.56	67.54	72.95	100	84.36
Tools	54.18	99.78	70.23	83.74	100	91.15
Transportation	66.25	100	79.69	94.64	100	97.24
Travel	73.89	99.33	84.74	89.34	100	94.37
Utility	42.10	96.55	58.63	64.08	100	78.11
Video	66.41	100	79.81	73.64	100	84.81
Average	**65.55**	**99.42**	**78.40**	**82.07**	**99.78**	**89.79**

Copyright © 2012, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

into the *Search* domain. Such findings further prove that some APIs registered at the ProgrammableWeb naturally belong to multiple domains.

To evaluate the effectiveness of our domain knowledge-empowered service categorization approach, we applied the traditional SVM method to redo the experiments over the services on ProgrammableWeb. The results are listed in Table 3 as well. It is shown that our method exceeds the traditional SVM method in each of the categories comprising more than 100 APIs.

We further conducted experiments to evaluate the efficiency of our approach facing different scales of testing sets. We compared the precision/recall values of using our methodology with those of 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: 1000, 3000, 5000, and 7190. As explained, the experiments were repeated until the termination criterion is met (the precision rate remains). Table 4 summarizes our findings.

As shown in Table 4, 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 1000 services), our methodology always reaches higher than 98.03, meaning that our method is good at identifying all services containing significant domain-related keywords. For the Precision index, our method outperforms the traditional SVM even from the first iteration, and terminates quickly (no more than 5 times for the test set containing 7190 services). In general, using our method,

Table 4. Categorization accuracy comparisons

Test#		#test	1000	3000	5000	7190
Traditional SVM		precision	95.54	82.87	74.62	73.89
		recall	99.33	99.33	99.33	99.33
		F-measure	97.40	90.36	85.27	84.74
Knowledge-empowered SVM	Iteration1	precision	100	96.77	96.15	94.33
		recall	99.33	99.33	99.33	99.33
		F-measure	99.66	98.03	97.71	96.77
	Iteration2	precision	99.33	96.77	96.17	94.37
		recall	99.33	99.33	100	100
		F-measure	99.33	98.03	98.05	97.10
	Iteration3	precision	99.33	X	96.17	86.78
		recall	99.33		100	100
		F-measure	99.33		98.05	92.92
	Iteration4	precision	X	X	X	86.28
		recall				100
		F-measure				92.63
	Iteration5	precision	X	X	X	86.28
		recall				100
		F-measure				92.63

** "X" means that this iteration does not exist

Copyright © 2012, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

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 ProgrammableWeb as the reference points. 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 PW. 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 prove that our methodology has the ability to: 1) justify existing categorization of services; 2) identify services that belong to multiple domains; and 3) find services that should be categorized into different domains. For the second and third types of services, we choose to add tags to them, so that such information may support further services discovery.

Validation

We have collected all API data of ProgrammableWeb in September 2011, and December 2011, separately. We found that 29 APIs were moved from their original categories to different categories. We tested whether our classification method can support these changes. We also applied the traditional SVM method to conduct the same experiments. The results are listed in Table 5.

Among all APIs that changed their categories, three APIs are exceptions. Two APIs (openBmap, OpenGeoTracker) were moved from the *Mapping* category to the *Other* category by ProgrammableWeb, which means that their category information was removed. An API (43Things) is difficult to be automatically classified into any category *Goal Setting,* since there are only four APIs in the category. For the other 26 APIs, our method supported 16 cases of API re-categorization and suggested the same new categorizations as the manual process. For the other 10 APIs, according to the approach, our method can provide their similarity to the new categories, ranging from 0.09 to 0.39. As shown in Table 5, the traditional single-class SVM classification supported 14 API re-categorization. Two cases supported by our approach but not the traditional SVM approach are highlighted in Table 5. This study again proved the effectiveness of our approach.

Performance

We evaluated the performance of our method. Here we describe our results for the categories that have more than 100 APIs Using the same training set as mentioned, we repeated the experiments with different scales of testing sets including 1000, 3000, 5000, and 7190 (all) APIs. In each experiment, we recorded the average execution time of service classification, as summarized in Figure 9. The average execution time ranges from 79 to 174 seconds. The largest execution time is 424 seconds. This overhead is acceptable because service categorization only needs to be conducted periodically. Furthermore, our experiments evaluated every iteration (with different testing sets) separately, without considering the domain ontology built from smaller scale of testing data. In reality, domain ontology will be carried on to start a new service categorization process. As a result, the number of iteration may be decreased and the execution time will be shorter.

Furthermore, we examined the number of iterations that is needed to finish service categorizations. Again we focus on the 25 categories containing more than 100 APIs. For each category, the testing set was set to be 7190 (all) APIs. As shown in Figure 10, the average iteration number is 3.8. Only two domains iterated more than 5 times.

Copyright © 2012, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

Table 5. Validation results

API Name	Original Category	New Category	Our Method	SVM
Socialight	Mapping	Social	√	√
Spot2be	Mapping	Social	√	√
Veniu	Mapping	Social	√	√
waldstat	Mapping	Social	√	√
Gnip	Internet	Social	√	√
Safe2pee	Mapping	Search	√	√
Touch Local	Mapping	Search	√	√
Unlock	Mapping	Search	√	√
SingTelinSingBusiness Search	Telephony	Search	√	√
ShowMyIP	Mapping	Internet	√	√
VodoModo	Mapping	Video	√	√
ViaMichelin	Mapping	Travel	√	√
YourStreet	Mapping	News	√	√
Strike Iron Tax Service	Other	Reference	√	√
eBay	Shopping	Search	√	×
USGS Gazetteer Query	Government	Mapping	√	×
43Things	Other	Goal Setting	×	×
openBmap	Mapping	Other	×	×
OpenGeoTracker	Mapping	Other	×	×
Tellmewhere	Mapping	Recommendations	×	×
mite	Utility	Office	×	×
Prototype GeoIP	Mapping	Internet	×	×
Quova	Mapping	Internet	×	×
Where2GetIt Geospatial	Mapping	Tools	×	×
Unified Software AddressVal	Mapping	Tools	×	×
Tixik	Mapping	Reference	×	×
SitOrSquat	Mapping	Search	×	×
Wigle	Mapping	Search	×	×
Where2GetIt SlippyMap	Mapping	Search	×	×

CONCLUSION

The unique structural feature of service repositories and their hidden domain knowledge inspire us in extending the traditional SVM methodology to enhance the effectiveness and efficiency of automatic service categorization. Our proposed technique is particularly valuable in building service search engines oriented to small- to middle-scale service repositories. It will also help services discovery and recommendations.

Copyright © 2012, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

Figure 9. Performance evaluation

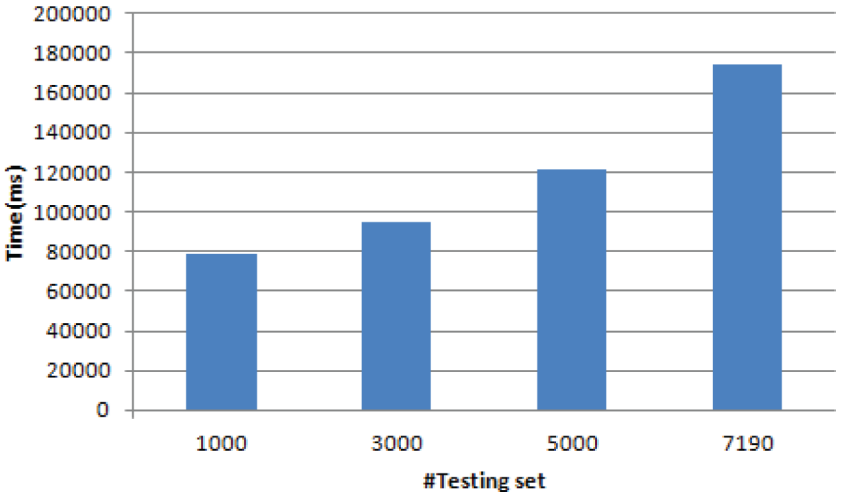
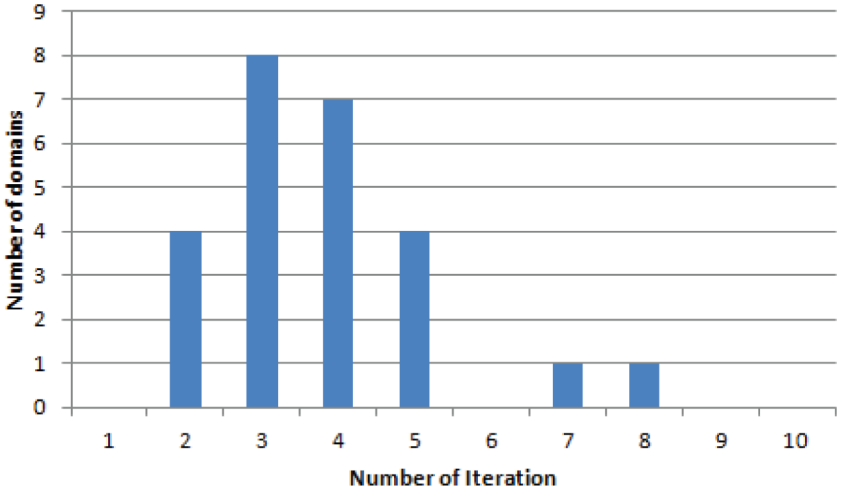


Figure 10. Iteration numbers



As the first step, we focus on one feature of domain knowledge: keyword ranking that indicates the significance of a term to represent a domain. If a service is tagged, manually or systematically, by significant terms of a domain, it is more likely that the service belongs to the domain. Consequently, a service may also represent multiple overlapped domains. Semantic relationships between terms may further help to extract domain knowledge. We will study this topic in our future research.

We plan to continue our research in the following directions. First, we will study the effectiveness and efficiency of our approach on other popular service repositories such as Tech-News and TechFreaks. Second, we will

Copyright © 2012, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

encapsulate our technique and develop a Web service to monitor and categorize services on the ProgrammableWeb. Third, we will explore to enhance existing service repository-oriented service search engine.

ACKNOWLEDGMENT

The work described in this paper is partially supported by the National Natural Science Foundation of China under Grant No. 61202031, 60970017, 61100017, the National Science & Technology Pillar Program of China under grant No.2012BAH07B01, the central grant funded Cloud Computing demonstration project of China undertaken by Kingdee Software (China), and the National Science Foundation of USA under grant IIS-0959215.

REFERENCES

Akkiraju, R., Farrell, J., Miller, J., Nagarajan, M., Schmidt, M., Sheth, A., & Verma, K. (2005). *Web Service Semantics – WSDL-S.* Retrieved September 5, 2012, from http://www.w3.org/Submission/WSDL-S/

Arabshian, K., Danielsen, P., & Afroz, S. (2012, March). LexOnt: Semi-automatic ontology creation tool for programmable web. In *Proceedings of the AAAI Spring Symposium on Intelligent Web Services Meet Social Computing*, Palo Alto, CA.

Battle, R., & Benson, E. (2008). Bridging the Semantic Web and Web 2.0 with Representational State Transfer (REST). *Journal of Web Semantics*, *6*, 61–69. doi:10.1016/j.websem.2007.11.002

Bianchini, D., Antonellis, V., Pernici, B., & Plebani, P. (2006). Ontology-based methodology for e-service discovery. *Information Systems*, *31*(4-5), 361–380. doi:10.1016/j.is.2005.02.010

Boser, B. E., Guyon, I., & Vapnik, V. (1992). A training algorithm for optimal margin classiers. In *Proceedings of the 5th ACM Annual Workshop on Computational Learning Theory* (pp. 144-152).

Chang, C.-C., & Lin, C.-J. (2011). *LIBSVM: A library for support vector machines*. Retrieved from http://www.csie.ntu.edu.tw/~cjlin/libsvm/

Church, K. W., & Hanks, P. (1990). Word association norms, mutual information, and lexicography. *Computational Linguistics*, *16*(1), 22–29.

Coalition, O. S. (2004). *OWL-S: Semantic markup for web services*. Retrieved from http://www.w3.org/Submission/OWL-S/

Corella, M. Á., & Castells, P. (2006). A heuristic approach to semantic web services classification. In *Proceedings of the 10th International Conference on Knowledge-Based & Intelligent Information & Engineering Systems* (pp. 598-605).

Dasgupta, S., Bhat, S., & Lee, Y. (2011). Taxonomic clustering and query matching for efficient service discovery. In *Proceedings of the IEEE International Conference on Web Services* (pp. 363-370).

Dong, X., Halevy, A., Madhavan, J., Nemes, E., & Zhang, J. (2004). Similarity search for web services. In *Proceedings of the 30th International Conference on Very Large Data Bases.*

Gomadam, K., Ranabahu, A., Nagarajan, M., Sheth, A. P., & Verma, K. (2008). A faceted classification based approach to search and rank web APIs. In *Proceedings of the IEEE International Conference on Web Services*, Beijing, China (pp. 177-184).

Hadley, A. (2009). *Web Application Description Language (WADL)*. Retrieved from http://www.w3.org/Submission/wadl/

Hess, A., & Kushmerick, N. (2003). Learning to attach metadata to web services. In D. Fensel, K. Sycara, & J. Mylopoulos (Eds.), *Proceedings of the International Semantic Web Conference* (LNCS 2870, pp. 258-273).

Jiang, Y. C., Liu, J. X., & Tang, M. D. (2011). An effective web service recommendation method based on personalized collaborative filtering. In *Proceedings of the 9th IEEE International Conference on Web service* (pp. 211-218).

Jones, K. S. (1972). A statistical interpretation of term specificity and its application in retrieval. *The Journal of Documentation*, *28*(1), 11–21. doi:10.1108/eb026526

Junghans, M., Agarwal, S., & Studer, R. (2010). Towards practical semantic web service discovery. In L. Aroyo, G, Antoniou, E. Hyvonen, A. Teije, & H. Stuckenschmidt (Eds.), *Proceedings of the 7th International Conference on The Semantic Web: Research and Applications - Volume Part II* (LNCS 6089, pp. 15-29).

Copyright © 2012, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

Klusch, M., Fries, B., & Sycara, K. (2006, May 8-12). Automated semantic web service discovery with OWLS-MX. In *Proceedings of the ACM International Conference on Autonomous Agents*, Hakodate, Japan (pp. 915-922).

Klusch, M., & Kaufer, F. (2009). WSMO-MX: A hybrid semantic web service matchmaker. *Web Intelligence and Agent Systems*, *7*(1), 23–42.

Kopecky, J., Vitvar, T., Bournez, C., & Farrell, J. (2007). SAWSDL: Semantic annotations for WSDL and XML schema. *IEEE Internet Computing*, *11*(6), 60–67. doi:10.1109/MIC.2007.134

Lee, Y.-J., & Kim, C.-S. (2011). A learning ontology method for restful semantic web services. In *Proceedings of the IEEE International Conference on Web Services* (pp. 251-258).

Liu, X., Zhao, Q., Huang, G., Mei, H., & Teng, T. (2011). Composing data-driven service mashups with tag-based semantic annotations. In *Proceedings of the IEEE International Conference on Web Services* (pp. 243-250).

Martin, D., Burstein, M., Mcdermott, D., Mcilraith, S., Paolucci, M., & Sycara, K. (2007). Bringing semantics to web services with OWL-S. *World Wide Web (Bussum)*, *10*(3), 243–277. doi:10.1007/s11280-007-0033-x

NIST. (2011). *Cloud computing definition*. Retrieved from csrc.nist.gov/publications/drafts/800-145/Draft-SP-800-145_cloud-definition.pdf

Paolucci, M., Kawamura, T., Payne, T. R., & Sycara, K. P. (2002). Importing the semantic web in UDDI. In *Proceedings of the International Workshop on Web Services, E-Business, and the Semantic Web* (pp. 225-236).

Patil, A. A., Oundhakar, S. A., Sheth, A. P., & Verma, K. (2004, May 17-22). Meteor-s web service annotation framework. In *Proceedings of the 13th International Conference on World Wide Web* (pp. 553-562).

Pautasso, C., Zimmermann, O., & Leymann, F. (2008, April 21-25). Restful web services vs. "big"' web services: Making the right architectural decision. In *Proceedings of the ACM 17th International Conference on World Wide Web*, Beijing, China (pp. 805-814).

Porter, M. (1980). An algorithm for suffix stripping program. *Automated Library and Information Systems*, *14*(3), 130–137. doi:10.1108/eb046814

Ranganathan, S. (1962). *Elements of library classification*. New York, NY: Asia Publishing House.

Sabou, M., Wroe, C., Goble, C., & Mishne, G. (2005a). Learning domain ontologies for web service descriptions: An experiment in bioinformatics. In *Proceedings of the 14th ACM International Conference on World Wide Web* (pp. 190-198).

Sabou, M., Wroe, C., Goble, C., & Stuckenschmidt, H. (2005b). Learning domain ontologies for semantic web service descriptions. *Journal of Web Semantics*, *3*(4). doi:10.1016/j.websem.2005.09.008

Sbodio, M. L., Martin, D., & Moulin, C. (2010). Discovering semantic web services using SPARQL and intelligent agents. *Web Semantics: Science. Services and Agents on the World Wide Web*, *8*(4), 310–328. doi:10.1016/j.websem.2010.05.002

Segev, A., & Sheng, Q. Z. (2010). Bootstrapping ontologies for web services. *IEEE Transactions on Services Computing*, *5*(1), 33–44. doi:10.1109/TSC.2010.51

Sheth, A. P., Gomadam, K., & Lathem, J. (2007). SA-REST: Semantically interoperable and easier-to-use services and mashups. *Internet Computing*, *11*(6), 91–94. doi:10.1109/MIC.2007.133

Sivashanmugam, K., Verma, K., Sheth, A., & Miller, J. (2004). Discovery of web services in a federated registry environment. In *Proceedings of the IEEE International Conference on Web Services*, San Diego, CA (pp. 270-277).

Vitvar, T., Zaremba, M., Moran, M., Zaremba, M., & Fensel, D. (2007). SESA: Emerging technology for service-centric environment. *IEEE Software*, *24*(6), 56–67. doi:10.1109/MS.2007.178

Yang, Y., & Liu, X. (1999). A re-examination of text categorization methods. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 42-49).

Zhang, J., Madduri, R., Tan, W., Deichl, K., Alexander, J., & Foster, I. (2011, July 4-9). Toward semantics empowered biomedical web services. In *Proceedings of the IEEE International Conference on Web Services* (pp. 371-378).

Zhang, L., & Li, B. (2004). Requirements driven dynamic business process composition for web services solutions. *Journal of Grid Computing*, *2*, 121–140. doi:10.1007/s10723-004-4202-1

Copyright © 2012, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

Zhang, L.-J., Zhang, J., & Cai, H. (2007). *Services computing*. New York, NY: Springer.

ENDNOTES

[1] This issue has been resolved by ProgrammableWeb in 2012. However, the approach how we graph an image of metadata of a service repository can be applied to other resources without full API support.

*Jia Zhang, PhD, is an Associate Professor at Carnegie Mellon University – Silicon Valley. Her current research interests center on Services Computing, with a focus on collaborative scientific workflow, service-oriented architecture, and semantic service discovery. She has co-authored one textbook titled "*Services Computing*" and published over 120 journal articles, book chapters, and conference papers. Zhang is an associate editor of* IEEE Transactions on Services Computing *(TSC) and of* International Journal of Web Services Research *(IJWSR). She is a member of the IEEE .*

Jian Wang, PhD, is a Lecturer of State Key Lab of Software Engineering, Computer School, Wuhan University, China. His current research interests focus on web services. He has published over 20 journal articles and conference papers.

Patrick C.K. Hung, PhD, is an Associate Professor of University of Ontario Institute of Technology, Canada. His current research interests focus on web services and business process management. He has published over 100 papers. Hung is an associate editor of IEEE Transactions on Services Computing *(TSC) and* International Journal of Web Services Research *(IJWSR), and* International Journal of Business Process Integration and Management *(IJBPIM). He is a member of the IEEE.*

Zheng Li is a PhD student at State Key Lab of Software Engineering, Computer School, Wuhan University, China. Her current research interests focus on service discovery. She has published 5 journal articles and conference papers.

Neng Zhang is an undergraduate student at Computer School, Wuhan University, China. His current research interests focus on services discovery.

Keqing He, PhD, is a Professor and Chief Scientist of State Key Lab of Software Engineering, Computer School, Wuhan University, China. His current research interests focus on Software Engineering. He has published over 100 journal articles and conference papers.

Copyright © 2012, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.