2013 IEEE 10th International Conference on Services Computing

Towards Services Discovery based on Service Goal Extraction and Recommendation

Jian Wang, Neng Zhang, Cheng Zeng, Zheng Li, Keqing He
State Key Lab of Software Engineering,
Computer School, Wuhan University
Wuhan, China
{jianwang, zhangneng, zengc, zhengli_hope, hekeqing}@whu.edu.cn

***Abstract*—Faced with the increasing services and users' personalized requirements, it remains a big challenge for users to effectively and accurately discover and reuse interested services. Goal oriented requirements modeling has attracted more and more attentions in services discovery and modeling, but little work has focused on extracting intentional goals from service descriptions. In this paper, based on the ranked domain keywords, we investigate how to extract domain-specific service goals from service descriptions, which can contribute to services discovery and recommendation. ProgrammableWeb, a publicly accessible service repository, is selected as the testbed. Experiments show the feasibility of the proposed approach.**

Keywords- service goal extraction, service goal recommendation, servcies discovery

I. INTRODUCTION

With the rapid development of services computing and cloud computing, increasing amounts of software have been published as reusable Web APIs or Web services. The publicly available services can greatly improve the efficiency and quality of software development. However, there are also many obstacles in services discovery. For example, on one hand, these services are heterogeneous due to the fact that they follow diverse protocols such as SOAP, REST, or XML-RPC, and they are described in different description languages including WSDL, WADL, and natural language. On the other hand, different types of users including end users and developers may have various kinds of personalized requirements towards services. Faced with the increasing services and users' personalized requirements, it remains a big challenge for users to effectively and accurately discover and reuse their interested services.

In most non-UDDI service repositories including Seekda[1]and ProgrammableWeb[2], users can use one or more query items such as keywords and category during the services discovery process. However, such a services discovery mechanism is keyword-based in essential, and thus is usually low accuracy [24]. In fact, many activities on the Web such as Web search are driven by high-level goals of users, e.g., "plan a trip", "buy some product", and "coordinate appointments" [1]. Compared with the keyword-based query, the intentional queries according to users' goals can get more accurate results for users. Users are also assumed to greatly benefit from having access to the goals of other users [1]. Therefore, sharing and reusing users' goals will contribute to discovering more accurate services. More and more works represent and model users' service requirements in terms of goals [1, 2, 3]. In WSMO [4], the goal is one of the four top level elements, which is defined as the objective that a client may have when consulting a Web service. Although the goal has attracted more and more attentions in services discovery and modeling, little work has been done on extracting goals from service descriptions. In this paper, we will present how to mine goal knowledge from service descriptions, and discover services based on the mined service goal knowledge.

Our work will have the following potential contributions. Firstly, it can contribute to service-based requirements engineering. The extracted service goals, as a kind of domain assets, will help supplement users' requirements during requirements elicitation and analysis phase. Secondly, most existing researches on services discovery focus on discovering SOAP Web services described in WSDL, while only a few of them can discover RESTful Web services. Our work will contribute to services discovery on RESTful Web services described in natural language. Thirdly, social tagging of services is also a popular way in services discovery. The service goals extracted and recommended by our approach can also be used as service tags, which will be more meaningful than keywords used in most existing service tagging systems.

The remainder of this paper is structured as follows. In section II, we discuss the related work. Section III gives an overview of our approach. Section IV discusses how to extract service goals from service descriptions in detail. Section V introduces services discovery and service goals recommendation. In Section VI, we present experimental results and analysis. Section VII concludes the paper.

II. RELATED WORK

The work presented in this paper is closely related to Web services discovery. Many research results have been reported in the literature in this area.

In services discovery, most researches perform profile-based service signature (such as Input, Output, Precondition, and Postcondition) matching or exploit the structure of WSDL documents to categorize Web services. For example, in OWLS-MX [5], logic-based reasoning and syntactic concept similarity computations in OWL-S are combined for

[1] http://webservices.seekda.com/
[2] http:// http://www.programmableweb.com/

978-0-7695-5026-8/13 $26.00 © 2013 IEEE
DOI 10.1109/SCC.2013.16

services discovery. Junghans et al. propose a formalism to describe functionalities and service requests [6]. Many researches leverage bipartite graphs to calculate similarity between Web services based on WSDL documents [7, 8]. Chen et al. [9] propose a services clustering approach WTCluster, which uses both WSDL documents and service tags to more accurately discover services. In contrast, we extract service goals from service descriptions and leverage the extracted service goals to discover services.

There are also many researches on how to discover services to satisfy users' requirements. For example, Driss et al. propose a requirement-centric approach for services modeling, discovery and selection [2]. The paper mentions the important role played by modeling users' requirements in services discovery, which are represented as intentions using the MAP formalism. Cremene et al. present an approach on automatic matching of Web services to user queries written in natural language [10]. Wang et al. apply K-means algorithm to cluster both requirements and candidate services, which can effectively reduce the search space [11]. Compared with these work, our work focuses on how to extract the intentional requirements knowledge from service descriptions.

Many researches on services organization and composition rely on the concept of service abstractions that represent semantically compatible services, that is, the services which provide the same functionality. Athanasopoulos et al. mine a hierarchy of service abstractions that represent alternative design options, via an agglomerative clustering algorithm that takes Web service descriptions as input [12]. Liu et al. propose an approach to fully automate the generation of abstract services from a service community that consists of a set of functionally similar services [13]. In contrast, our work mainly extracts similar knowledge to service abstractions from RESTful Web services in natural language, instead of SOAP Web services described in WSDL.

Mining functionality from natural language documents is widely investigated in other domains. Ghose et al. realize a toolkit named R-BPD which uses a syntax parser such as NLTk to identify business processes [14]. Friedrich et al. [15] present an automatic approach to generate BPMN models from natural language text using the natural language tool such as the Stanford Parser, FrameNet, and WordNet. Dumitru et al. [16] utilize text mining and a novel incremental diffusive clustering algorithm to discover domain-specific features from product descriptions, and use association rule mining and *k*-NN (*k*-Nearest Neighbor) to generate product specific feature recommendations. Compared with these work, our approach mainly focuses on extracting service goals from service descriptions in short text. Moreover, we adopt a different extraction strategy and make full use of the domain ranked keyword list.

III. AN OVERALL FRAMEWORK

In this section, we will give an overview of our approach. As shown in Fig. 1, our approach mainly consists of two phases: one is the service goal extraction, and the other is service matching and service goal recommendation.

More specifically, in the first phase, service descriptions will be crawled from the publicly accessible Web service repositories such as ProgrammableWeb and Seekda. These services will be categorized into different domains according to our prior work in [17]. A ranked keyword list of each domain will also be produced by our approach. Then we extract service goals from service descriptions based on the ranked domain keyword list. In the service matching and service goal recommendation phase, the user firstly provides an initial description of service requirements. According to the requirements, the domain that the user is interested in can be identified first. Afterwards, the goal related information will be extracted by parsing the requirements description. Then the extracted goals from the requirement text will be matched with the existing domain goals. Based on the service-by-service goal matrix, the services that can satisfy the user's requirements can be determined. In addition, more related service goals can be recommended to users to supplement their incomplete requirements.

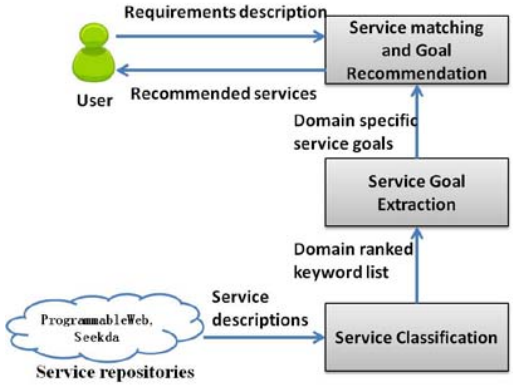

Figure 1. Service goal extraction and recommendation.

In our opinion, a service goal of a service is used to exhibit the intentional functionality of the service. However, there is still no standard way to characterize the service goal. In [18], a goal in the field of requirements engineering is expressed as a clause with a main verb and several parameters, where each parameter plays a different role with respect to the verb. Among the four types of parameters, the target is the most important one, which designates entities affected by the goal. The target can be either an object (e.g., receipt) or a result (e.g., the user's choice). Both of them can be represented as a noun or a noun phrase. Therefore, a goal can be generally represented in the form of "verb-noun" pair. In this paper, we basically follow this definition, and the service goal can be formally defined in Definition 1.

DEFINITION 1. (Service Goal) A service goal *sg* is defined as a triple <*sgv*, *sgn*, *sgp*>, where *sgv* is a verb or verb phrase, which denotes the action of the service goal, *sgn* is a noun or noun phrase, which denotes the entities affected by the action, and *sgp* is a parameter, which denotes the additional information such as how the action affects the entity, and the initial or final state of the entity. Please note that for a specific *sg*, *sgv* and *sgn* are mandatory, while *sgp* is optional. The service goals mentioned in this paper mainly consist of *sgv* and *sgn*.

IV. SERVICE GOAL EXTRACTION

Service goals can be extracted from both SOAP Web Services and RESTful Web services. The naming of interface and operations of SOAP Web services has been widely discussed. In [19], it shows that the definition of typical enterprise service interface and operation names usually consists of a verb such as *create*, *approve*, and *notify*, and a noun such as *order*. It is also suggested that the noun-verb naming convention for services and operations tends to help people focus on the functional cohesiveness of the service interface [20, 21]. As a matter of fact, most of the publicly accessible services are described in this style. Thus it is easy to extract service goals from SOAP Web services.

As for RESTful Web services, although both WSDL 2.0 and WADL can be used to describe them, according to the survey in [22], service providers often use a simple textual description on Web pages to explain their APIs, instead of using XML-based WADL or WSDL 2.0. Therefore, it is necessary to investigate how to extract service goals from textual descriptions of RESTful Web services.

A. Services classification

In [17], we have implemented an SVM engine using the LIBSVM (http://www.csie.ntu.edu.tw/~cjlin/libsvm/) for services classification.

The basic procedure is as follows. Firstly, the textual service descriptions will be preprocessed, including word segmentation, stop word removal, stemming, verbs and nouns extraction, and term frequency count. Secondly, for each domain, we select a training set and a testing set from the service description sets, and use a formula named keyword frequency - inverse document frequency - domain frequency (KF-IDF-DF), an extension to TF-IDF, to construct the vector space. Afterwards, we leverage SVM to classify services. Our services classification approach adopts an iterative process to incrementally improve the accuracy of services classification. To achieve this, we use a formula named keyword frequency - inverse repository frequency (KF-IRF) to calculate the ranking of keywords in the domain after the SVM-based classification. Then a new iteration will initiate by reconstructing the vector space according to KF-IDF-DF. Since KF-IDF-DF is based on the ranking of keywords in the domain, once the ranked keyword list changes, the vector space will be modified accordingly. The termination criteria can be set when the resulting keyword ranking remains unchanged (e.g., when the top 50 keywords ranking remains unchanged in new iterations). When the iteration is terminated, we can obtain the classified services and a domain keyword list ranked by KF-IRF. More details can be found in [17].

B. Domain service goal identification

Next, we discuss how to extract candidate services goals from each service description, and identify domain service goals from these candidate service goals.

1. Extract candidate service goals

Given a textual description document of a service, we need to extract the verb-noun pairs according to the sentence structure. Many tools can be leveraged to parse the document in natural language. In this paper, the Stanford parser [23] is adopted. The first step is to split the document into individual sentences. And then the Stanford parser is able to generate Stanford Dependencies, which reflect the grammatical relationships between the words in a sentence. Among these dependencies, "dobj" and "nsubjpass" are the two most important ones that can be used to extract candidate service goals from active clauses and passive clauses, respectively. Other dependencies can be used to supplement the candidate service goals. Table I shows some of these important dependency relations.

TABLE I. USED STANFORD DEPENDENCY RELATIONS

Dependency relation	Definition	Usage
dobj	a noun phrase which is the (accusative) object of the verb	identify a "verb-noun" pair in an active clause
nsubjpass	a noun phrase which is the syntactic subject of a passive clause.	identify a "verb-noun" pair in a passive clause
prep	any prepositional phrase that serves to modify the meaning of the verb, adjective, noun, or even another prepositon	identify a "verb-noun" pair in an active clause by combining the dependency relation "nsubj"
conj	the relation between two elements connected by a coordinating conjunction, such as "and" and "or"	identify other poential "verb-noun" pairs that are coordinated with the original one
prt	identifies a phrasal verb, and holds between the verb and its particle	supplement the verb part of a "verb-noun" pair
nn	any noun that serves to modify the head noun	supplement the noun part of a "verb-noun" pair
amod	any adjectival phrase that serves to modify the meaning of the noun phrase	supplement the noun part of a "verb-noun" pair

For example, for the sentence "The API also lets users create a social travel graph by relating objects to each other," many dependencies including the following three ones will be generated:

amod(graph-10, social-8);
nn(graph-10, travel-9);
dobj(create-6, graph-10).

Based on these dependencies, the following candidate service goal set *csg*1 can be created: *csg*1={create graph, create travel graph, create social travel graph}.

For the sentence "The available hotel information can be retrieved and reserved by using this API," many dependencies including the following four ones will be generated:

amod(information-4, available-2);
nn(information-4, hotel-3);
nsubjpass(retrieved-7, information-4);
conj(retrieved-7, reserved-9).

Based on these dependencies, the following candidate service goal set *csg*2 can be created: *csg*2 = {retrieve information, retrieve hotel information, retrieve available hotel information, reserve information, reserve hotel information, reserve available hotel information}.

Besides these two basic patterns, we have also summarized other kinds of patterns that can be used for extracting candidate service goals. Here we only list these two due to the space limitation.

2. Identify domain service goals

In the previous step, we have extracted as more as possible candidate service goals. To make full use of them in services discovery, it is necessary to refine and merge them to obtain domain related service goals.

First of all, we leverage the ranked domain keyword list produced in services classification to filter out the domain independent candidate service goals. If one of the noun terms in a candidate service goal belongs to the top k (e.g., 100) keywords of the domain, then the candidate goal will be kept, otherwise, it will be removed. For example, in *csg*2, since the term "information" does not belong to the top 100 keywords of the travel domain, and the term "hotel" exists in the list, then the candidate goals "retrieve information" and "reserve information" will be removed. Then the new *csg*2={retrieve hotel information, retrieve available hotel information, reserve hotel information, reserve available hotel information} can be obtained.

Afterwards, it is necessary to merge the semantically equivalent or similar service goals. The similarity of two service goals sg_1 and sg_2 can be calculated based on Equation (1). As mentioned before, each service goal *sg* consists of two mandatory parts: the verb part *sgv* and the noun part *sgn*. *VProportion* and *NProportion* denote the proportions of *sgv* and *sgn*, respectively. Since the noun part is more important, here we set *VProportion* and *NProportion* as 0.4 and 0.6, respectively. The *NounSim* of two service goals is obtained by referring to the similarity of two terms in Wordnet (http://www.wordnet.princeton.edu). Since each *sgn* may consist of one or more noun terms, we will consider the average maximal similarity of each term in a smaller term set of *sgn*, as shown in Equation (2). The calculation of *VerbSim* is similar to Equation (2).

$$GoalSim(sg_1, sg_2) = VProportion \times VerbSim(sg_1, sg_2) + NProportion \times NounSim(sg_1, sg_2) \quad (1)$$

$$NounSim(sg_1, sg_2) = \begin{cases} \frac{\sum_{k=1}^{|sgn_1|} \max\{sim(t_k, t_j) | t_k \in sgn_1, t_j \in sgn_2\}}{|sgn_1|} & |sgn_1| \le |sgn_2| \\ \frac{\sum_{k=1}^{|sgn_2|} \max\{sim(t_k, t_i) | t_k \in sgn_2, t_i \in sgn_1\}}{|sgn_2|} & |sgn_1| > |sgn_2| \end{cases} \quad (2)$$

If the similarity of two service goals exceeds a predefined similarity threshold, then the two service goals can be viewed as semantically equivalent. In this way, we can get many service goal clusters where the service goals in the same cluster are semantically equivalent to each other. For example, the following service goal set {reserve hotel information, reserve available hotel information, book accommodation, book hotel, reserve lodging} is a service goal cluster.

The frequency of occurrence of a service goal in a domain is defined as the number of services that contain the service goal. The frequency of occurrence of a service goal cluster in a domain is defined as the number of services that contain at least one of its members of the cluster. If the frequency of occurrence of a service goal or a service goal cluster exceeds a predefined threshold, then the service goal or the service goal cluster can be defined as a domain service goal. Please note that for a service goal cluster, the simplest or the most abstract service goal will be the representative. For example, "reserve hotel" will be the representative goal of the above mentioned goal cluster. The service goal cluster can also be used to generate the concept hierarchy. For example, "hotel information", "available hotel information", "accommodation", and "lodging" will be the potential subclass or equivalent class of the concept "hotel". The service goal cluster will be represented as "reserve hotel [hotel information, available hotel information, accommodation, lodging]." In this way, the domain knowledge can be further enhanced.

V. Service Matching and Goal Recommendation

Once service goals in each domain have been identified, these domain service goals can be used as a bridge to match users' requirements and existing services. In this section, we will present how to discover services based on domain service goals, and recommend related service goals to supplement users' requirements.

A. Service matching

We propose a two-phase method to match services according to users' requirements. In phase 1, we intend to identify the domain that the requirement belongs to by calculating the similarity between the requirement and candidate domains. In phase 2, we calculate the similarity between the requirement and candidate services in the domain.

A user can propose his requirement in natural language. The requirement may consist of one or several sentences, not just keywords, e.g., "I want to search flight and hotel information for my trip". The user's requirement will be automatically preprocessed first, including word segmentation, stop word removal, stemming, and term frequency count. Afterwards, the similarity between the requirement and candidate domains can be calculated using the cosine similarity, as shown in Equation (3). When constructing the weighted term vectors, we consider both term frequency and the corresponding values of $kf\text{–}irf$ mentioned in Section IV.A. The basic idea of Equation (3) is that if a requirement carries the representative keywords (i.e., ranked high) in a domain, and such keywords appear multiple times (term frequency), then it is likely that the requirement has higher relevance to the domain.

$$domSim(req, D_j) = \frac{\sum_{k=1}^{N}(W_{req,t_k} \times W_{D_j,t_k})}{\sqrt{(\sum_{k=1}^{N} {W_{req,t_k}}^2)(\sum_{k=1}^{n} {W_{D_j,t_k}}^2)}} \quad (3)$$

$$W_{req,t_k} = kf\text{–}irf_{D_j,t_k} \times log(1 + F_{req,t_k}) \quad (4)$$

$$W_{D_j,t_k} = \begin{cases} kf\text{-}irf_{D_j,t_k} \times log(1 + F_{req,t_k}) & if\ F_{req,t_k} \geq 1 \\ kf\text{-}irf_{D_j,t_k} & otherwise \end{cases} \quad (5)$$

where $kf\text{-}irf_{D_j,t_k}$ denotes the $kf\text{-}irf$ value of term $t_k (k \leq N)$ in the ranked keyword list of domain D_j; F_{req,t_k} denotes the term frequency of term t_k in the user's requirement; and the logarithm operation aims to minimize the effect of the term frequency. In other words, the top N keywords in domain D_j will be used to examine the term list in requirement *req*. If t_k does not exist in requirement *req,* then $W_{req,t_k} = 0$, and $W_{D_j,t_k} = kf\text{-}irf_{D_j,t_k}$, otherwise, both of them equal to $kf\text{-}irf_{D_j,t_k} \times log(1 + F_{req,t_k})$.

Since some services might belong to several domains [17], similarly a requirement may also have close relations with multiple domains. If the similarity between a requirement and a domain exceeds a given threshold, the requirement is viewed as belonging to the domain.

Once the domain that the requirement belongs to has been determined, we can extract the candidate service goals (e.g., "search flight" and "search hotel") from the requirement description. Using these extracted service goals, we will discover suitable services within the domain. As shown in Equation (6), the goal similarity $serviceSim(req, s_i)$ between requirement *req* and each existing service s_i in the domain is computed using the cosine similarity.

$$serviceSim(req, s_i) = \frac{n(SG_{req} \cap SG_{s_i})}{\sqrt{n(SG_{req}) \times n(SG_{s_i})}} \quad (6)$$

where SG_{req} denotes the set of candidate service goals of the requirement, SG_{s_i} denotes the set of domain service goals contained in service s_i, and $n(A)$ denotes the number of elements in set A. It is obvious that the range of the similarity value is between 0 and 1. If a service shares the same service goals with the requirement, then the service similarity scores 1. The top k services that exceeds a given similarity threshold will be returned to the user. For example, the services named Skyscanner and Despegar in ProgrammableWeb can be returned for the requirement example of this section.

B. Service goal recommendation

Since users' requirements are usually incomplete, it is also an important task to recommend suitable service goals to supplement their requirements. For example, if most services that can satisfy the above-mentioned requirement consist of other service goals such as "reserve hotel" and "reserve flight", these services goals will be recommended to the user. We adopt a standard *k*-Nearest Neighbor (*k*-NN) strategy to realize a service goal recommendation algorithm.

The *k*-NN algorithm calculates a service goal-based similarity measure between a requirement and each existing services in the domain. The top *k* most similar services are viewed as neighbors of the new requirement. It then uses information from these neighbors to infer other potentially interested service goals and to make recommendations for requirements supplementation.

Once service goals in each domain have been identified, a service-by-service goal matrix for each domain can be generated: $M_d = (m_{i,j})_{S \times SG}$, where S represents the number of services in domain d, SG denotes the number of extracted service goals in domain d, and $m_{i,j} \in \{0,1\}$ indicates whether service s_i contains service goal sg_j. When the domain of a new requirement *req* is determined and service goals of *req* are identified, *req* can be added as a new row in the matrix M_d of domain d and a new matrix M_d' will be generated.

The top k most similar services can be selected as the neighborhood of the requirement using Equation (6). Then, we can predict the likelihood that the user will be interested in a new service goal *sg*. We just adopt the standard prediction formula in Equation (7).

$$predict(req, sg) = \frac{\sum_{s \in neighbor(req)} serviceSim(req,s) \cdot m_{s,sg}}{\sum_{s \in neighbor(req)} serviceSim(req,s)} \quad (7)$$

where $s \in neighbor(req)$ depicts that s is a neighbor of *req*, and $m_{s,sg}$ is the value of row s and column sg in matrix M_d', which describes whether service goal *sg* belongs to service *s*. Only the service goals belonging to at least one of the neighbors of *req* excepting those emerged in *req* will be viewed as candidate service goals to be recommended. The prediction scores of each candidate service goal will be calculated, and the service goals with the highest prediction score will be recommended to the user.

VI. Experiments and Discussions

We have conducted a series of experiments to evaluate our approach. 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.

A. Experimental preparation

We use the publically available repository ProgrammableWeb (PW) as our testbed. PW provides Web APIs for users to get descriptive data about the registered services, such as API name, tags, summary, and description. We combine the open APIs in PW and a crawler to collect such descriptive data of all registered services from PW. We have collected 7,190 service description documents from 63 domains (data gathered on Sep. 7, 2012). Afterwards, the collected service documents are preprocessed and classified according to our SVM-based service classification approach. The detailed document preprocessing and services classification results can be found in [17]. In this paper, we mainly discuss the experiments on service goal extraction and recommendation.

B. Service goal extraction

As mentioned before, descriptions of Web services in ProgrammableWeb are in the form of short text. Firstly, we

show this characteristic of these service description documents.

Table II shows the average number of sentences contained by each service and the average length of each sentence. The top 10 domains ranked by its number of services are listed, and each domain has more than 150 services. We can see that most service description documents consist of about 4 sentences, and each sentence consists of about 19 words. Since the length of most service descriptions is no more than 100, we mainly concentrate on the quality of service goal extraction, and do not pay more attention to the efficiency of the approach.

TABLE II. THE TERM COUNT OF SERVCIE DOCUMENTS IN PROGRAMMABLEWEB

Domain	Average number of sentences in a service	Average length of a sentence
Enterprise	4.2	19.2
Financial	3.6	18.9
Internet	4.0	18.6
Mapping	3.9	20.5
Music	3.8	18.8
Search	4.2	18.5
Shopping	4.3	19.1
Tools	4.1	18.2
Travel	4.4	19.2
Video	3.9	18.0
Average	4.0	18.9

We have designed experiments to evaluate the quality of extracting service goals using our approach. Since there is no related work on extracting service goals, we have to design the evaluation experiments by ourselves. Five master students are selected to make an evaluation. Each student selects 10 service description documents from the domain of Travel, Music, and Financial, respectively. They are requested to manually extract service goals that can embody the semantics of the domain from these documents. These manually extracted service goals will be viewed as the baseline compared with the automatically extracted service goals to evaluate the quality of service goal extraction.

We adopt two indexes, *Precision* and *Recall* to evaluate the performance. *Precision* is the fraction of the automatically extracted service goals that are consider as relevant to manually extracted ones; *Recall* is the fraction of the manually extracted service goals that can be automatically extracted. *Precision* and *Recall* are formally defined as follows:

$$\text{Precision} = \frac{|CSG|}{|SG_A|};\ \ Recall = \frac{|CSG|}{|SG_M|}.$$

where *CSG* represents the set of automatically extracted service goals that are correct, in other words, the intersection of automatically extracted service goals and manually extracted service goals; $|CSG|$ denotes the number of service goals in *CSG*. SG_A represents the set of automatically extracted service goals; $|SG_A|$ indicates the number of service goals in SG_A. SG_M represents the set of manually extracted service goals; $|SG_M|$ indicates the number of service goals in SG_M.

The evaluation results are shown in Table III. The average precision in three domains is 77.37%, while the average recall in three domains is 82.46%. The recall is better than the precision, which indicates that our approach can extract most of the targeted service goals although some of them may be inaccurate. However, there is still much space to improve the extraction quality. For example, the service goal "retrieve market data" is expected to be extracted from the sentence containing "retrieval of market data". But currently our approach cannot deal with this case. We still need to summarize more patterns and strategies to improve the extraction results.

TABLE III. EVALUATION OF EXTRACTED SERVICE GOALS

Domain	Precision	Recall
Travel	0.7982	0.8761
Financial	0.7386	0.7846
Music	0.7843	0.8131
Average	0.7737	0.8246

Based on our approach, the top 10 extracted service goals ranked by its frequency in the domain of Travel, Music, and Financial are shown in Table IV. Please note that the noun phrases within the square brackets denote that they are semantically equivalent to the preceding noun or noun phrase. We can see that most of these extracted service goals are meaningful, although some of them are less meaningful such as "provide listing" and "earn commission" in Travel domain, "provide market" and "provide management" in Financial domain, and "license music" in Music domain. However, domain experts can manually adjust these service goals to make them more meaningful without much effort. Our approach can be used to alleviate domain experts' burden to extract such kind of domain service goals.

TABLE IV. EXTRACTED DOMAIN SERVICE GOALS

Domain	Domain service goal list(TOP 10)
Travel	provide travel [travel booking, booking travel accommodation, travel agency] provide booking [booking car rental, hotel booking, travel booking, booking travel accommodation, booking flight] provide listing [listing destination, listing property] make review accommodation search hotel[hotel website] get trip [trip plan, travel experience, travel activity] provide trip insurance [travel insurance] share travel plan earn commission cancel booking [reservation]
Financial	provide rate [exchange rate, currency exchange rate, rate quote, rate feed, tax rate] provide quote [stock quote, rate quote, price quote] provide market [market industry, stock market] provide management[management function, expense management} provide price [price data, sale stock price] calculate tax [tax rate, sale tax] provide payment [payment experience, payment gateway, payment number] create invoice access accounting entry [account, invoice] provide credit [credit report, credit score]
Music	provide music [music lyric] share music [sheet music, song]

	search music [music blog, music playlist, song] search artist sell ticket[concert ticket] upload music[music collection] play song license music create music[music notation, music playlist, music station] search track[list track, artist track]

C. *Service matching and service goal recommendation*

It is also not an easy task to evaluate the quality of service matching and service goal recommendation. An ideal approach might be letting users propose their requirements and check whether the users will select and use the matched services. However, there is no open data set that can support the experiment.

We discuss first how to evaluate the service goal recommendation. In [16], the authors adopted a leave-one-out cross validation method for evaluating recommended software product features. In this paper, we borrow a similar strategy to evaluate our approach. That is, we will partition the services in a specific domain into four complementary subsets, perform the analysis on one of them that is called the training set, and validate the analysis on the remainder called testing set. Multiple rounds of cross-validation are performed using different partitions. In each round, firstly a random service goal will be removed from each service; afterwards, our approach can generate a recommendation service goal set; and finally whether the removed service goal belongs to the recommendation set will be determined.

For each service in the test set, m service goals are randomly selected and kept as the content of a requirement document. The remaining service goals will be removed from the requirement document, and one of them is randomly selected as a target service goal. In each case, our method will be evaluated about whether it can recommend this target service goal. Note that in each test, only these services that have more than m service goals will be tested.

Hit ratio is adopted to validate the results of recommendation, which represents the probability that a given service goal is recommended as one of the top n recommendations. It is calculated by the ratio of the number of services that the target service goal can be correctly recommended back towards the number of all services in the testing set. The hit ratio increases when the number n of recommended services increases. A good recommendation result means that the hit ratio is relatively high even when the recommendation set remains very small.

The experiment is conducted with the requirements' size m = 3, 4, and 5. As shown in Fig. 2, if the size of the initial requirement is larger, the recommended results will become better, which corresponds to our expectation. For these three experiments, over 75% service goals can be returned within the top 20 recommendations. In the recommendation process, our approach can get relative high hit ratio due to the fact that we leverage the semantically similarity set of service goals. The recommendation quality is acceptable for domain experts to supplement users' requirements.

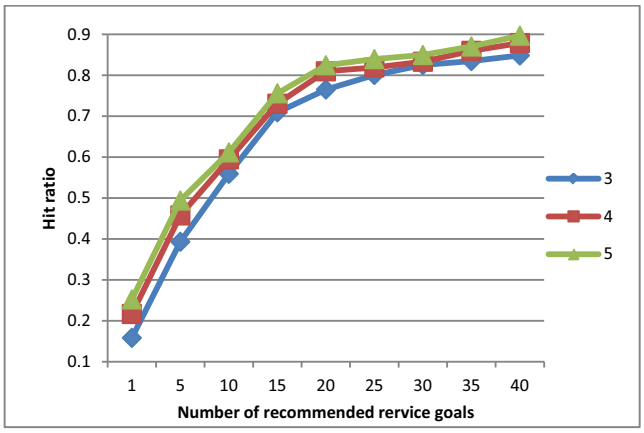

Figure 2. Hit ratio of service goal recommendation.

In this way, the effectiveness of the service goal recommendation is tested. Besides the service goal recommendation, we also propose a service matching approach, which consists of two phases: matching with domains, and matching with services in the domain. For the validation of the first phase, we also adopt a similar cross validation strategy, that is, we randomly select 10 services for each domain, use the description of each service as a requirement, and calculate the similarity of the requirement and the domains. As shown in Table V, when the similarity threshold is set 0.8, all the services in the testing set can be correctly matched to the domain. What's more, for almost 8 services in each domain, the similarity between their descriptions and their original domains are higher than that of any other domains. In this way, we can guarantee that users' requirements can be correctly matched to the domains. For the second phase, since the service goal recommendation is based on this phase, the experiment of the service goal recommendation can also indirectly validate this phase.

TABLE V. EVALUATION OF EXTRACTED SERVICE GOALS

Domain	Number of services that belong to the domain	Number of services with this domain as their closest domain
Travel	10	7
Financial	10	8
Music	10	10
Average	10	8

VII. CONCLUSIONS

In this paper, we present an approach on how to extract domain-specific service goals from service descriptions based on the ranked domain keywords, which can provide support for services discovery. Furthermore, we propose an approach for service matching and service goal recommendation. We conduct experiments using the data of ProgrammableWeb to show the feasibility of our approach.

We have developed a service supermarket named as CloudCRM [25], where users can select services according to their personalized requirements and receive recommended services. The service goal extraction and recommendation approach has been applied in this platform. CloudCRM

provides a collection of requirement templates to help elicit users' requirements. Users can propose their requirements in natural language in the template. According to the approach presented in this paper, the domain that the users' requirements belong to can be determined first, and then corresponding service goals can be identified. Based on the identified service goals, the users may have two choices: on one hand, corresponding services can be directly recommended to them; on the other hand, the users can go to the process customization template to customize their process models based on the relationship between goal and process.

In future, we plan to improve our approach in the following directions. Firstly, we need to summarize more patterns and strategies to improve the accuracy of the service goal extraction. Secondly, the current form of goal is a verb-noun pair, which just consists of the mandatory parts in the definition of service goal. Service goal parameter, an optional part, will be considered during service goal extraction in future. Thirdly, our approach can be only used to identify domain-specific goals, while neglect domain-independent goals. We still need to consider how to discover the domain-independent goals.

ACKNOWLEDGMENT

The work is supported by the National Science & Technology Pillar Program of China under grant No.2012BAH07B01, the National Natural Science Foundation of China under grant No. 61202031, U1135005, and 61100017, and grant of Nokia Research Center (Beijing).

REFERENCES

[1] M. Strohmaier, M. Lux, M. Granitzer, P. Scheir, S. Liaskos, and E. Yu, "How Do Users Express Goals on the Web? - An Exploration of Intentional Structures in Web Search," Proc. of the 2007 international conference on Web information systems engineering, 2007, pp. 67-78.

[2] M. Driss, N. Moha, Y. Jamoussi, JM J´ez´equel, and H. Ben Gh´ezala, "A Requirement-Centric Approach to Web Service Modeling, Discovery, and Selection," Proc. of 2010 International Conference on Service-Oriented Computing, 2010, pp. 258-272.

[3] Z. Ma, L. Liu, H. Yang, and J. Mylopoulos, "Adaptive Service Composition Based on Runtime Requirements Monitoring," Proc. of 2011 International Conference on Web Services, 2011, pp.339-346.

[4] J. Domingue, D. Roman, and M. Stollberg, "Web Service Modeling Ontology (WSMO) - An Ontology for Semantic Web Services," Position paper at the W3C Workshop on Frameworks for Semantics in Web Services, 2005.

[5] M. Klusch, B. Fries, and K. Sycara, "OWLS-MX: A Hybrid Semantic Web Service Matchmaker for OWL-S Services," Journal of Web Semantics, vol.7, no.2, 2009, pp. 121-133.

[6] M. Junghans, S. Agarwal, and R. Studer,. "Towards Practical Semantic Web Service Discovery," Proc. of 2010 Extended Semantic Web Conference, 2010, pp. 15-29.

[7] J. Zhang, R.Madduri, W. Tan, K. Deichl, J. Alexander, and I. Foster, "Toward Semantics Empowered Biomedical Web Services," Proc. of 2011 International Conference on Web Services, 2011, pp. 371-378.

[8] F. Liu, Yuliang Shi, and Jie Yu, "Measuring Similarity of Web Services Based on WSDL," Proc. of 2010 International Conference on Web Services, 2010, pp.155-162.

[9] L. Chen, L. Hu, Z. Zheng, J. Wu, J. Yin, Y. Li, and S. Deng, "WTCluster: Utilizing Tags for Web Services Clustering," Proc. of 2011 International Conference on Service-Oriented Computing, 2011, pp. 204-218.

[10] M. Cremene, J.Y. Tigli, S. Lavirotte, F.C. Pop, M. Riveill, and G. Rey, "Service Composition based on Natural Language Requests," Proc. of 2009 International Conference on Services Computing, 2009, pp. 486-489.

[11] X. Wang, Z. Wang, and X. Xu, "Semi-empirical Service Composition: A Clustering Based Approach," Proc. of 2011 IEEE International Conference on Web Services, 2011, pp. 219-226.

[12] D. Athanasopoulos, A. V. Zarras, P. Vassiliadis, and V. Issarny, "Mining Service Abstractions," Proc. of 2011 International Conference on Software Engineering, 2011, pp.944-947.

[13] X. Liu and H. Liu, Automatic Abstract Service Generation fromWeb Service Communities, Proc. of 2012 IEEE International Conference on Web Services, 2012, pp. 154-161.

[14] A. K. Ghose, G. Koliadis, and A. Chueng, "Rapid business process discovery (R-BPD), " Proc. of 2007 International Conference on Conceptual Modeling, 2007, pp. 391-406.

[15] F. Friedrich, J. Mendling, and F. Puhlmann, "Process Model Generation from Natural Language Text," Proc. of 2011 International Conference on Advanced Information Systems Engineering, 2011, pp. 482-496.

[16] H. Dumitru, M. Gibiec, N. Hariri, et al., "On-demand Feature Recommendations Derived from Mining Public Product Descriptions," Proc. of 2011 International Conference on Software Engineering, 2011, pp. 181-190.

[17] J. Zhang, J. Wang, P. C. K. Hung, Z. Li, N. Zhang, and K. He, "Leveraging Incrementally Enriched Domain Knowledge to Enhance Service Categorization," International Journal of Web Services Research (IJWSR), vol. 9, no. 3, 2012, pp. 43-66.

[18] C. Rolland, C. Souveyet, and C. B. Achour, "Guiding Goal Modeling Using Scenarios," IEEE Transactions on Software Engineering, vol. 24, 1998, pp.1055-1071.

[19] M. Roy, B. Suleiman, and I. Weber, "Facilitating Enterprise Service Discovery for Non-technical Business Users," Proc. of 2010 International Conference on Service-Oriented Computing, 2010, pp. 100-110.

[20] D. J. N. Artus, "SOA realization: Service design principles Service design to enable IT flexibility," Available: http://www.ibm.com/developerworks/webservices/library/ws-soa-design/.

[21] TIBCO, "Web Services Naming Standards: A Guid for Enterprise Architects," white paper, Available: https://www.tibcommunity.com/servlet/JiveServlet/previewBody/2327-102-1-2221/Web%20Services%20Naming%20Standards.pdf.

[22] W. Jiang, D. Lee, and S. Hu, "Large-scale Longitudinal Analysis of SOAP-based and RESTful Web Services," Proc. of 2012 International Conference on Web Services, 2012, pp.218-225.

[23] M. de Marneffe, B. MacCartney and C. D. Manning, "Generating Typed Dependency Parses from Phrase Structure Parses," Proc. of LREC 2006, 2006.

[24] Y. J. Lee and C. S. Kim, "A Learning Ontology Method for RESTful Semantic Web Services," Proc. of 2011 International Conference on Web Services, 2011, 251-258.

[25] D. Yu, J. Wang, B. Hu, J. Liu, X. Zhang, K. He, L. Zhang, "A Practical Architecture of Cloudification of Legacy Applications," Proc. of 7th IEEE World Congress on Services, 2011, pp. 17-24.