Soft Comput (2011) 15:929–938
DOI 10.1007/s00500-010-0548-4

FOCUS

A novel approach to annotating web service based on interface concept mapping and semantic expansion

Guobing Zou · Yang Xiang · Yanglan Gan · Yixin Chen

Published online: 17 February 2010

Abstract With the rapid development of web service technology in these years, traditional standards have been matured during the process of service registry and discovery. However, it is difficult for service requesters to discover satisfactory web services. The reason for this phenomenon is that the traditional service organization mode lacks semantic understanding ability for service function interface. This paper proposes a novel approach to annotating web services. We first adopt domain ontology as a semantic context, and give our general framework of service semantic annotation. Then, interface concept mapping algorithm and service interface expansion algorithm are respectively presented in detail. Finally, the generation process of semantic web service repository is presented based on preceding algorithms. Simulation experiment results demonstrate that annotated web services by the proposed method can more satisfy requirements for service requesters than traditional ones by service matchmaking engine. It can get better service discovery effectiveness.

Keywords Web service · Service semantic annotation · Domain ontology · Interface concept mapping · Service interface expansion

G. Zou (✉) · Y. Xiang · Y. Gan
Department of Computer Science and Technology,
Tongji University, Shanghai 201804, China
e-mail: guobing278@sina.com

Y. Chen
Department of Computer Science and Engineering,
Washington University, St. Louis, MO 63130, USA
e-mail: chen@cse.wustl.edu

1 Introduction

The Internet is emerging not only as an infrastructure for data, but also for a wide variety of information resources, which are increasing being made available as web services (Patil et al. 2004). Web services are modular, self-describing, self-contained applications that are accessible over the Internet (Curbera et al. 2001). Although traditional XML-based standards (e.g., UDDI, WSDL and SOAP) have been applied in the process of service registration and service discovery, it is very difficult for service requesters to discover their satisfactory web services. The reason for conducting this phenomenon is that the current service organization mode mainly concentrates on the implementation and execution of web services from the perspective of the syntax-based level. It limits the whole service discovery techniques in the keyword-based way. Therefore, it motivates our interest in exploring efficient approaches to implement service semantic understanding, discovery and interoperation.

Our investigation has shown that semantic web technique can help us solve disadvantages of keyword-based web service matching. Ontology defined in Gruber (1993) and Studer et al. (1998) is one of the most important components in semantic web architecture and has been identified as the basis for service semantic annotation that can be used for further service discovery. Therefore, by adopting domain ontology (DO) as the provider of semantic context, we can develop different new approaches to annotating syntax-based source web services and convert them into semantically described web services.

In summary, we investigate the problem of web service annotation under the condition of semantic context. Our main contributions are as follows. First, rather than constructing a dictionary or knowledge index, we model DO and visualize it as a directed hierarchy tree (DHT), which will be used as a semantic context for web service annotation. Secondly, on the basis of constructed DO and its corresponding DHT, we propose an overall architecture of service semantic annotation mainly consisting of seven closely correlative components. Thirdly, in order to generate and enrich semantic information for service I/O function interface, we respectively design an interface concept mapping algorithm based on comprehensive similarity calculation and service interface expansion algorithm based on semantic relations in DO and DHT, which maps source service function interface set into interface concept set from ontology concepts and expand interface concept set respectively. Finally, we present a semantic web service repository generation algorithm, which is in charge of annotating and yielding semantic web services based on the results of the above-designed algorithms.

The remainder of this paper is organized as follows. Section 2 reviews related work about web service annotation methods and gives the idea of our approach. In Sect. 3, domain ontology is modeled and visualized. In Sect. 4, the general architecture of service semantic annotation is presented. In Sect. 5, we first give a service semantic description model, and then interface concept mapping, service interface expansion, and semantic web service repository generation algorithms are proposed respectively. In Sect. 6, simulation experiments are conducted to validate the effectiveness of our proposed method. Finally, Sect. 7 concludes the paper and our future work plan.

2 Related work

Web service annotation is the first step but very critical to implement service registration, service discovery, service interoperation and service composition at the semantic level. The target of our work is to give a novel approach to automatically understanding and annotating web services by finding precise semantic information of service I/O function interface.

In this section, we discuss some related efforts that describe how to add semantics to web services. We also look into some semantic web service description languages because they are the foundation and useful for us to consider how to efficiently model web services.

Several web service annotation methods have been proposed from different ways in recent years. The work discussed in Patil et al. (2004) presented a METEOR-S web service annotation framework MWSAF, where domain ontologies are utilized to categorize web services into domains for semi-automatically marking up web services descriptions. The key to invoking and composing web services is to connect computer-understandable semantic information to services, a variety of machine learning techniques have been explored in Heβ and Kushmerick (2003) to semi-automatically create semantic metadata for describing semantics of web services, such as Bayesian learning and inference algorithm for classifying HTML forms into semantic categories, Navie Bayes and SVM algorithm to the task of service classification, etc. A web service semantic annotation method by workflow definitions has been presented in Belhajjame et al. (2008), where information can be inferred about the semantics of operation parameters based on their connections to other annotated operation parameters within tried-and-tested workflows.

There are mainly two kinds of semantic description languages for modeling semantic web services. DAML-S (Ankolekar et al. 2002; Martin et al. 2003) is a DAML-based web service ontology, which supplies web service providers with a core set of markup language constructs for describing properties and capabilities of their web services in unambiguous, computer-interpretable form (Martin et al. 2003). OWL-S (2004) is an OWL-based web service ontology and can provide similar but more functionality description than DAML-S.

From the above investigation to current web service annotation methods and main semantic web service description models, we can find out that these research works suffer from the following three problems.

- Most of these methods can only categorize or classify web services by using ontologies so that it has not made the most use of the semantic functionality of DO. Therefore, it lacks semantic understanding for specific I/O function interface of web service, which may leads to low service discovery effectiveness.
- These service annotation methods have shown that no good algorithm is known to automatically annotate web services. It may result in a problem that service annotation process is a time consuming and thus difficult task.
- Currently popular semantic web service description languages are complicated and thus difficult for service providers to master and publish their service registration information. Meanwhile, it is inconvenient for service requesters to submit their service requirements.

In order to address above issues, we give a three-step solution. We first map service I/O interface into ontology concept as interface concept that can best define its basic semantic information. This idea has also been mentioned in Sivashanmugam et al. (2003). Then service interface expansion is considered in order to further enrich semantic meaning of mapped interface concept based on our previous work about semantic expansion search (Zou et al. 2008). Finally, we present a synthetic process of semantic web service repository generation that can effectively and efficiently annotate source web services and convert them into semantic web services. Therefore, we can discover web services for service requesters at the semantic level.

3 Domain ontology

At present, many definitions have been given about what is ontology. However, the widely accepted one shows that ontology is a formal, explicit specification of a shared conceptualization (Studer et al. 1998). On the basis of the above description of DO, we respectively give the formal definitions of DO and its semantic relation set.

Definition 1 (*Domain ontology*) DO is defined in a specific domain as a five tuple:

$$\mathrm{DO} = \{C, P^C, R, I, A\} \tag{1}$$

- C represents all ontology concepts in the specific domain;
- P^C includes all properties attached to C;
- R denotes all kinds of semantic relations among ontology concepts, properties and instances;
- I consists of all instances that belong to ontology concepts;
- A is composed of all axioms in the specific domain.

In terms of the characteristic of semantic relation among ontology concepts, properties and instances, four different kinds of semantic relations are considered here as it appears in the following definition.

Definition 2 (*Semantic relation set*) In a domain ontology (DO), its semantic relation set R is formally denoted as a four tuple:

$$R = \{compose\text{−}\mathrm{of}, \mathrm{kind}\text{−}\mathrm{of}, \mathrm{attribute}\text{−}\mathrm{of}, instance\text{−}\mathrm{of}\} \tag{2}$$

where, *compose-of* depicts the whole and part semantic relation between two ontology concepts. *kind-of* represents the successive relation between two ontology concepts. *attribute-of* describes the relation between ontology concept and its corresponding properties. *instance-of* denotes the relation between ontology concept and its subordinate instances.

According to the above two definitions, we utilize node to represent an ontology concept, property or instance; joint edge is also used as one of the semantic relations in R. Therefore, a directed hierarchy tree (DHT) is constructed as the structure of domain ontology. A part of DHT for the city traffic domain is shown in Fig. 1.

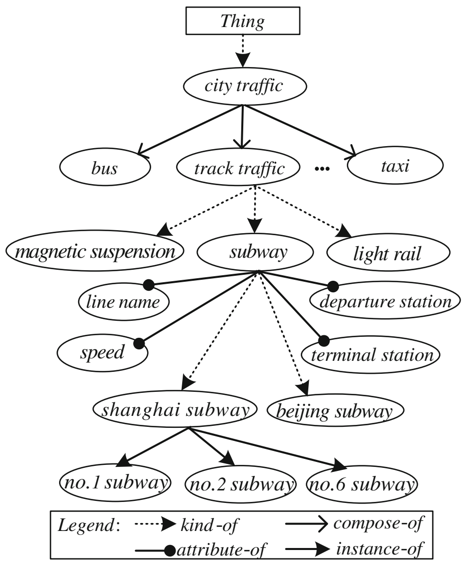

Fig. 1 A part of DHT for the city traffic domain

From the DHT in Fig. 1, for instance, we can see that 'subway' is an ontology concept; *'line name'* and *'speed'* are the properties affiliated to *'subway'*; *'no.1 subway'* is one of the instances that belong to ontology concept *'shanghai subway'*; ontology concept *'city traffic'* is composed of *'bus'*, *'track traffic'*, *'taxi'*, etc; *'subway'* is a kind of *'track traffic'*.

4 Architecture of service semantic annotation

We give the general framework of service semantic annotation called WS-SAF as it appears in Fig. 2. It consists of an interface extractor, concept set mapper, interface semantic expander, domain ontology, DHT, service annotator, and semantic web service repository.

Interface extractor is in charge of acquiring service file modeled by WSDL description language from WSDL document pool, and then extracting initial input and output interface parameter set from web service operation with the analysis of XML schema. The function of concept set mapper is to find optimal matching interface concept set for the initial I/O interface set from DO. Therefore, it can define basic semantic information for the extracted interface parameter set. In order to further enrich semantic meaning of the mapped interface concept set, we carry out interface semantic expander to expand interface concept set by use of semantic analysis and reasoning based on DO and its corresponding DHT.

The main function of service annotator is to first create an empty service instance on the basis of WS-SDM discussed in the following Sect. 5.1, and then fill its service operation field with the interface semantic meaning

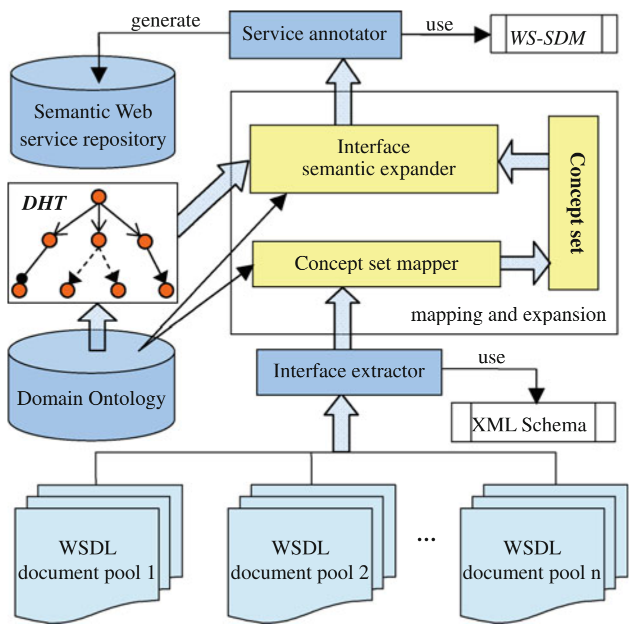

Fig. 2 The general framework of service semantic annotation

generated by interface semantic expander. Semantic web service repository stores all annotated web services that will be discovered and invoked at the semantic level. Domain ontology is in the format of an OWL document, which provides semantic context for interface concept mapping and semantic expansion. DHT is formed by DO discussed in Sect. 3 and used to yield semantic information for the module of interface semantic expander.

5 Web service annotation

5.1 Service semantic description model

Service description model plays an important role in the process of web service discovery. In order to make the web service machine-understandable and interoperated, it is necessary to model the web service for the purpose of adding semantics to its I/O function interfaces. We give a light web service semantic description model called WS-SDM, which consists of web service model and service operation model.

Definition 3 (*Web service model*) A web service can be defined as a four tuple:

$$ws = \{wsId, wsName, wsDesp, OprSet\} \tag{3}$$

where, *wsId* is the unique identifier all over the web service repository. *wsName* is the web service name. *wsDesp* is the functional text description of web service. *OprSet* is the service operation set, which is denoted as $OprSet = \{opr_1, opr_2, \ldots, opr_s\}$.

OprSet includes a series of correlative web service operations. Especially, each $opr_i (1 \le i \le s)$ can be executed for a special function. From the analysis of process model in OWL-S (2004), we can conclude that an output interface can map into one or multiple input interfaces. Considering the parameter binding relationship between I/O function interfaces in a service operation, we give its definition as follows.

Definition 4 (*Service operation model*) In a web service operation set $OprSet = \{opr_1, opr_2, \ldots, opr_s\}, \forall opr_i \in OprSet (1 \le i \le s)$, its formalization is modeled as a four tuple:

$$opr_i = \{oprName, InSet, OutSet, IOMap\} \tag{4}$$

- *oprName* is the service operation name;
- $InSet = \{inP_1, inP_2, \ldots, inP_m\}$ denotes input interface set, which has m parameters and each inP_i $(i = 1, 2, \ldots, m)$ is an input function interface;
- $OutSet = \{outP_1, outP_2, \ldots, outP_n\}$ is output interface set, which contains n parameters and each $outP_j$ $(j = 1, 2, \ldots, n)$ represents an output function interface.
- *IOMap* is the I/O mapping relationship between output interface and input interfaces. It can be denoted as $IOMap$: $outP_j \propto InSet'$, where $InSet' \subseteq InSet$. i.e., $\forall outP_j (j = 1, 2, \ldots, n)$, there exists a corresponding binding subset of *InSet*.

5.2 Interface concept mapping

For a web service in the WSDL document pool, its input and output interface set in each source service operation cannot provide explicit semantic information. Therefore, it restricts the whole process of web service discovery at the keyword-based level. In this subsection, we map I/O service interface set into optimal ontology concepts as interface concept set that can express basic semantic information of source interface set.

Definition 5 (*Interface concept mapping*) In domain ontology $\mathrm{DO} = \{C, P^C, R, I, A\}$, $\mathrm{SIS} = \{inE_1, inE_2, \ldots, inE_u\}$ is the source interface set, where $inE_i (1 \le i \le u)$ is an interface element. $\forall inE \in \{InSet \cup OutSet\}$, there exists a concept mapping and a corresponding ontology concept $mc \in C$ that can best match inE and represent its basic semantic information.

In order to find the optimum mapping concept from DO, similarity calculation is used to perform comparison between service interface element and corresponding ontology concept. Here, we take a comprehensive similarity calculation to evaluate mapping relativity and select optimal mapping concept for each interface element, which contains linguistic similarity and structural similarity.

We calculate linguistic similarity between interface element and ontology concept by Hamming distance. The standard definition of Hamming distance can be found in Forney (1992), in an appropriately general finite-alphabet setting A, all sequences (words) of length N can be described as A^N. For two sequences $a = \{a_k, 1 \le k \le N\}$ and $a' = \{a'_k, 1 \le k \le N\}$, The Hamming distance $d_H(a, a')$ between a and $a' \in A^N$ is defined as the number of components in which they differ ($a_k \neq a'_k$).

More specially, the Hamming distance $d_H(f, g)$ between two code words f and g of length n is simply the number of places in which they differ (Forney 1966), where it gives the formal description in the following equations.

$$d_H(f_i, g_i) = \begin{cases} 0, & f_i = g_i \\ 1, & f_i \neq g_i \end{cases} \tag{5}$$

$$d_H(f, g) = \sum_{i=1}^{n} d_H(f_i, g_i) \tag{6}$$

where, $f = \{f_1, f_2, \ldots, f_n\}$, $g = \{g_1, g_2, \ldots, g_n\}$. The function of $d_H(f_i, g_i)$ judge single place in f and g. $d_H(f, g)$ computes all the different places. $\forall inE \in \{InSet \cup OutSet\}$ and mapping concept $mc \in C$, we can calculate their linguistic similarity $Sim_{\text{ling}}(inE, mc)$ by the Hamming distance as follows:

$$Sim_{\text{ling}}(inE, mc) = 1 - \frac{||inE| - |mc||}{\max(|inE|, |mc|)} - \frac{\sum_{k=1}^{\min(|inE|,|mc|)} d_H(inE[k], mc[k])}{\max(|inE|, |mc|)} \tag{7}$$

where, $|x|$ is the character number of code word x, $inE[k]$ and $mc[k]$ are the character in the k place of the code word. From the above Eq. 7, we can see that the bigger the Hamming distance between the minimum length of code word inE and mc becomes, the smaller it is for the linguistic similarity between inE and mc.

In the structural level, structural similarity is calculated based on semantic dictionary WordNet (Miller 1995) developed by Princeton University. In Xun and Yan (2006), the authors compute word similarity according to the sum of sense similarity, which is based on three dimension semantic characteristics of each word in WordNet. $\forall inE \in \{InSet \cup OutSet\}$ and mapping concept $mc \in C$, we can calculate their structural similarity $Sim_{\text{struc}}(inE, mc)$ based on Xun and Yan (2006) as follows:

$$Sim_{\text{struc}}(inE, mc) = \frac{\sum_{i=1}^{|inE|} \max_{j=1}^{|mc|}(S(inE(i), mc(j)))}{|inE| + |mc|} + \frac{\sum_{i=1}^{|mc|} \max_{j=1}^{|inE|}(S(mc(i), inE(j)))}{|inE| + |mc|} \tag{8}$$

where, $|x|$ is the sense number of word x, $inE(k)$ and $mc(k)$ are the sense element of the word inE and mc extracted from the WordNet. $S(w_1(i), w_2(j))$ is in charge of computing sense similarity between two sense elements, which has the same functionality of sense similarity calculation in Xun and Yan (2006).

In order to facilitate the description of interface concept mapping algorithm, we first give the definition of synonym set in domain ontology.

Definition 6 (*Synonym set*) $\text{DO} = \{C, P^C, R, I, A\}, C = \{c_1, c_2, \ldots, c_v\}. \forall c \in C$, its synonym set is denoted as:

$$Syn(c, \text{DO}) = \{c_i \mid c_i \in C \wedge (c_i \simeq c)\} \tag{9}$$

where, $c_i (1 \le i \le v)$ is one of the elements in concept set C and has the same meaning with concept c, which can be inferred by axioms. For example, With regard to concept '*magnetic suspension*', *Syn*('*magnetic suspension*', DO) = {'*magnetic levitation*', '*Maglev*'}.

The description of interface concept mapping algorithm called ICMA is shown in Algorithm 1.

In Algorithm 1 (Fig. 3), we loop each interface element $inE_k (1 \le k \le u)$ in source interface set (SIS) and yield its optimum interface concept $c_j (1 \le j \le |C|)$ from ontology concept set C, which has the maximum comprehensive similarity with inE_k. More specifically, for each $inE_k (1 \le k \le u)$, we first generate synonym set of $c_j \in C$ and judge whether inE_k is included in the *Syns*. If it is one of the elements in *Syns*, the current loop index j is appointed to the optimal concept index *maxIndex* and we jump out current loop (*line* 4–16) to *line* 17, where inE_k's optimal mapping ontology concept is inserted to ICS. Otherwise, we calculate linguistic similarity *lingSim* and structural similarity *strucSim* between inE_k and c_j respectively in Eqs. 9 and 11, and then get its comprehensive similarity *comSim* by given weighting values (*line* 10–12). Meanwhile, if generated *comSim* is bigger than maximum similarity *maxSim*, we update its latest value and corresponding *maxIndex* (*line* 13–15). Finally, after getting the final *maxIndex*, we insert optimal mapping concept $c_{maxIndex}$ for inE_k into ICS.

```
Algorithm 1: Interface_concept_mapping($DO, SIS$)

Input: $DO$ and $SIS = \{inE_1, inE_2, ..., inE_u\}$
Output: Interface concept set $ICS = \{si_1, si_2, ..., si_u\}$
1  $ICS \leftarrow \emptyset$;
2  foreach $inE_k \in SIS$ do
3    $maxSim \leftarrow 0$; $maxIndex \leftarrow 0$;
4    foreach $c_j \in C$ do
5      $Syns \leftarrow Syn(c_j, DO) \cup \{c_j\}$; /*generate synonym set*/
6      if ($inE_k$ is an element in $Syns$) then
7        $maxIndex \leftarrow j$; /*set optimal concept index*/
8        break; /*finish $inE_k$'s optimum concept mapping*/
9      else if ($inE_k$ is not in $Syns$) then
10       compute $lingSim \leftarrow Sim_{ling}(inE_k, c_j)$;
11       compute $strucSim \leftarrow Sim_{struc}(inE_k, c_j)$;
12       $comSim \leftarrow w_{ling} \times lingSim + w_{struc} \times strucSim$;
13     if ($comSim > maxSim$) then
14       $maxSim \leftarrow comSim$; /*update maximum similarity*/
15       $maxIndex \leftarrow j$; /*update optimal concept index*/
16   end
17   $ICS \cdot insert(c_{maxIndex}, k)$; /*add optimal concept*/
18 end
19 return $ICS$;
```

Fig. 3 Interface concept mapping algorithm ICMA

5.3 Service interface expansion

For an ontology concept $c \in C$, its semantic context is yielded by semantic reasoning based on DO and its directed hierarchy tree (DHT).

In order to facilitate the description of service interface expansion algorithm, we first give several related definitions as follows:

Definition 7 (*Direct subclass set*) DO = $\{C, P^C, R, I, A\}$, DHT is the directed hierarchy tree of DO. $\forall\ c \in C$, its direct subclass set is denoted as:

$$Dss(c, \mathrm{DHT}) = \{f_l \mid f_l \in C \wedge (KR(f_l, c) \vee CR(c, f_l))\} \tag{10}$$

where, $f_l(1 \le l \le v)$ is one of the elements in concept set C. $KR(f_l, c)$ denotes that f_l and c satisfy $kind - of$ semantic relation. $CR(c, f_l)$ describes $compose - of$ semantic relation between c and f_l. Taking ontology concept *'track traffic'* for example shown in Fig. 1, *Dss('track traffic'*, DHT)={*'magnetic suspension', 'subway', 'light rail'*}.

Definition 8 (*Instance set*) DO = $\{C, P^C, R, I, A\}$, $I = \{Inst_1, Inst_2, \ldots, Inst_r\}$, DHT is the directed hierarchy tree corresponding to DO. $\forall c \in C$, its instance set is denoted as the following equation:

$$Ins(c, \mathrm{DHT}) = \{inst_k \mid inst_k \in I \wedge (inst_k \dashv c)\} \tag{11}$$

where, $inst_k(1 \le k \le r)$ is one of the elements in instance set I and subordinate to concept c. Taking ontology concept *'shanghai subway'* as an example, *Ins*(*'shanghai subway'*, DHT)={*'no.1 subway', 'no.2 subway', 'no.6 subway'*}.

Definition 9 (*Property set*) DO = $\{C, P^C, R, I, A\}$, $P^C = \{p_1, p_2, \ldots, p_w\}$, *DHT* is the directed hierarchy tree of DO. $\forall c \in C$, its property set is defined as the following equation.

$$Prop(c, \mathrm{DHT}) = \{p_j \mid p_j \in P^C \wedge (p_j \lhd c)\} \tag{12}$$

where, $p_j(1 \le j \le w)$ is one of the elements in set P^C and also attached to concept c. For instance, Prop(*'subway'*, DHT) = {*'speed', 'line name', 'departure station', 'terminal station'*}.

Service interface expansion algorithm called SIEA is described in Algorithm 2.

In Algorithm 2 (Fig. 4), we loop each interface concept si_k and generate its interface expansion set IES[k] based on domain ontology and corresponding DHT. For each $si_k(1 \le k \le u)$, we first append itself to interface expansion set IES, and then its synonym set *Syns* is yielded and attached to IES (*line* 3–6). If generated direct subclass set *Dss* is not empty, we insert it into IES and finish its service interface expansion (*line* 7–10). Similarly, if generated instance set *Inst* is not empty, it is added to IES and finishes its service interface expansion (*line* 11–14). Under the condition of both empty interface expansion result of direct subclass set and instance set, we further get property set *Pset* and expand it to IES.

5.4 Semantic web service repository generation

By adopting DO as the provision of semantic context for web service annotation, we can convert source web services from the WSDL document pool into the semantic web service repository based on interface concept mapping and service interface expansion. Semantic web service (SWS) repository generation algorithm called SWS-RG is represented in the following Algorithm 3.

In Algorithm 3 (Fig. 5), we loop each web service $ws_i(1 \le i \le t)$ in the WSDL document pool and convert it into semantically annotated web service *aws*. For each $ws_i(1 \le i \le t)$, if it is an empty set of its service operation set *OprSet*, we finish its annotation process (*line* 4–5); If there is at least one service operation in *OprSet*, we generate its corresponding annotated web service *aws* (*line* 6), which is discussed in Algorithm 4 in detail. Then, we

```
Algorithm 2: Service_interface_expansion($DO, DHT, ICS$)
Input: $DO = \{C, P^C, R, I, A\}$,$DHT$,and $ICS = \{si_1, si_2, ..., si_u\}$
Output: Interface expansion set $IES$
1  $IES \leftarrow \emptyset$;
2  foreach $si_k \in ICS$ do
3      $IES[k] \cdot insert(si_k)$; /*append $si_k$ to $IES$*/
4      $Syns \leftarrow Syn(si_k, DO)$;
5      if ($Syns \neq \emptyset$) then /*judge $si_k$'s synonym set*/
6          $IES[k] \cdot insert(Syns)$;
7      $Dss \leftarrow Dss(si_k, DHT)$; /*get direct subclass set*/
8      if ($Dss \neq \emptyset$) then
9          $IES[k] \cdot insert(Dss)$;
10         continue; /*finish $si_k$ service interface expansion*/
11     $Inst \leftarrow Ins(si_k, DHT)$;
12     if ($Inst \neq \emptyset$) then
13         $IES[k] \cdot insert(Inst)$;
14         continue;
15     $Pset \leftarrow Prop(si_k, DHT)$; /*get property set*/
16     if ($Pset \neq \emptyset$) then
17         $IES[k] \cdot insert(Pset)$; /*expand property set to $IES$*/
18 end
19 return $IES$;
```

Fig. 4 Service interface expansion algorithm SIEA

```
Algorithm 3: SWS_repository_generation($DO, DHT, WSDL\_P$)
Input: $DO$,$DHT$,and $WSDL\_P = \{ws_1, ws_2, ..., ws_t\}$
Output: Semantic web service repository $SWS\_R$
1  define $aws$ as annotated web service;
2  $SWS\_R \leftarrow \emptyset$;
3  foreach $ws_i \in WSDL\_P$ do
4      if ($ws_i \cdot OprSet$ is an empty set) then
5          continue;
6      $aws \leftarrow SSA(DO, DHT, ws_i)$; /*invoke Algorithm 4*/
7      $aws \cdot wsId = ws_i \cdot wsId$;
8      $aws \cdot wsName = ws_i \cdot wsName$;
9      $aws \cdot wsDesp = ws_i \cdot wsDesp$;
10     $SWS\_R \cdot insert(aws, i)$; /*add $aws$ to $SWS\_R$*/
11 end
12 return $SWS\_R$;
```

Fig. 5 Service repository generation algorithm SWS-RG

respectively append *wsId*, *wsName* and *wsDesp* of ws_i to *aws* (*line* 7–9). Finally, we insert annotated web service *aws* to *SWS_R*.

```
Algorithm 4: Service_semantic_annotation($DO, DHT, ws$)
Input: $DO$,$DHT$,and $ws = \{wsId, wsName, wsDesp, OprSet\}$
Output: Annotated web service $aws$
1  $aws \leftarrow \emptyset$;
2  define $a\_opr, mappedIS, mappedOS$;
3  foreach $opr_i \in OprSet$ do
4      $a\_opr \cdot oprName \leftarrow opr_i \cdot oprName$;
5      $mappedIS \leftarrow ICMA(DO, opr_i \cdot InSet)$;
6      $a\_opr \cdot InSet \leftarrow SIEA(DO, DHT, mappedIS)$;
7      $mappedOS \leftarrow ICMA(DO, opr_i \cdot OutSet)$;
8      $a\_opr \cdot OutSet \leftarrow SIEA(DO, DHT, mappedOS)$;
9      add $IOMap$ to $a\_opr$; /*generate I/O mapping relation*/
10     $aws \cdot OprSet \cdot insert(a\_opr)$; /*insert $a\_opr$ into $aws$*/
11 end
12 return $aws$;
```

Fig. 6 Service semantic annotation algorithm SSA

Service semantic annotation algorithm called SSA is shown in the following Algorithm 4, which is in charge of adding semantic information to I/O interface set for each service operation.

In Algorithm 4 (Fig. 6), we respectively get each service operation opr_i in *ws* and yield its annotated service operation. For each opr_i ($1 \leq i \leq s$), its operation name is first acquired and set to annotated operation *a_opr* (*line* 4). Subsequently, we use ICMA algorithm to get mapped input set *mappedIS*, and then SIEA algorithm is invoked to get input interface expansion set, which is attached to *a_opr* (*line* 5–6). Similarly, output interface expansion set is generated and appended to *a_opr* (*line* 7–8). Meanwhile, we generate new *IOMap* in *a_opr* according to original I/O mapping relationship of opr_i. Finally, we insert annotated operation *a_opr* into *aws*.

6 Evaluation

6.1 Experiment environment

In order to compare the effectiveness of our proposed framework WS-SAF with the other existing methods, we have constructed a domain ontology in the area of city traffic by ontology editor Protégé 3.3.1 designed by Stanford University, which includes approximate 300 concepts, properties and instances. At the same time, a WSDL document pool containing about 500 web service files in the area of city traffic has been collected for the experiment data set. The experiment is conducted on a IBM Server with Xeon 2.13 GHZ CPU and 4G RAM. Moreover, we have utilized integrated development environment (IDE) Eclipse and Java, JSP programming languages as prototype development platform to implement function modules (e.g., Concept set mapper, Interface semantic expander, Service annotator) and user interaction interface shown in Fig. 2.

Three different kinds of web service index repositories have been generated based on the collected WSDL web services for comparing service discovery effectiveness among three service organization modes. They are JAXR service repository denoted as *Keyword* by using the standard UDDI registration and discovery mechanism, JUDDI extended registry (Srinivasan et al. 2004) denoted as OWL-S by use of OWL-S service description ontology, and SWS repository denoted as WS-SAM by adopting the given service semantic description model, respectively.

We mainly adopt web service recall ratio S_{recall} and web service precision ratio $S_{\mathrm{precision}}$ as the evaluation standards to validate service discovery effectiveness. S_{recall} refers to the proportion of matched correlative service number S_{mcn} out of total correlative service number S_{tcn} in web service repository. $S_{\mathrm{precision}}$ is defined as the proportion of matched correlative service number S_{mcn} relative to total matched service number S_{tmn}.

$$S_{\mathrm{recall}} = \frac{\text{matched correlative number } \mathrm{S}_{\mathrm{mcn}}}{\text{total correlative number } S_{\mathrm{tcn}}} \tag{13}$$

$$S_{\mathrm{precision}} = \frac{\text{matched correlative number } \mathrm{S}_{\mathrm{mcn}}}{\text{total matched number } S_{\mathrm{tmn}}} \tag{14}$$

6.2 Simulation experiments

Two simulation experiments have been conducted on service discovery effect and service annotation performance respectively as follows:

6.2.1 *Experiment 1: service discovery effect*

We take a set of service requests $Req = \{req_1, req_2, req_3, req_4, req_5\}$ extracted from the city traffic domain to calculate service discovery effectiveness under the uniform service matchmaking engine. Especially, the I/O interface number of each service request $req_i(1 \le i \le 5)$ ranges from three to five. According to the experiment condition, we set the parameters in the interface concept mapping algorithm as follows: (1) w_{ling}=0.5; (2) w_{struc}=0.5. Service recall ratio and service precision ratio among three different service organization modes are shown in Figs. 7 and 8.

From the experiment results of three different modes, we can see that the service organization mode by our proposed method can outperform other two related methods for the service discovery effect in a specific service request set.

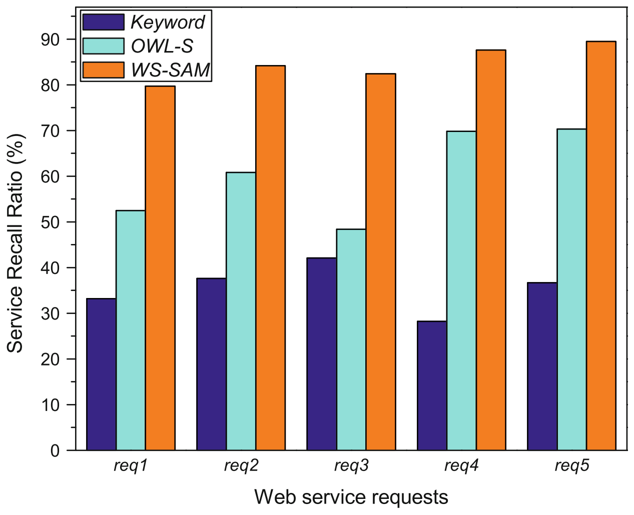

Fig. 7 Service recall ratio of three organization modes

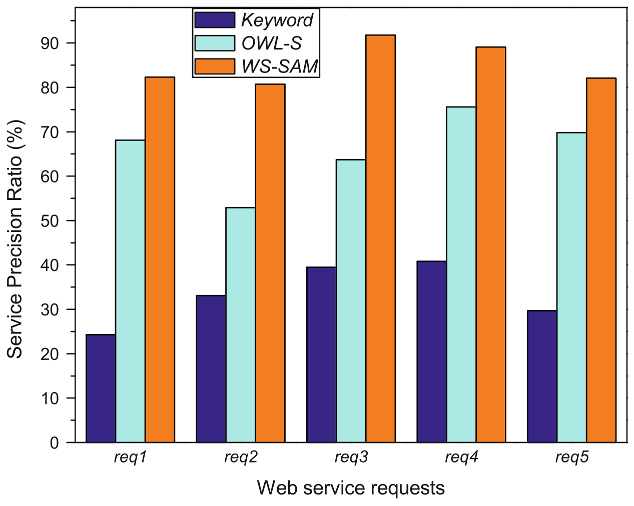

Fig. 8 Service precision ratio of three organization modes

Moreover, we additionally evaluate the whole effect of our method from the perspective of service discovery average value. Here, service average recall ratio (SA_R) and precision ratio (SA_P) can be got in a group of fixed weights of similarity calculation for interface concept mapping. For a specific service request set *Req*, SA_R and SA_P are formalized as follows:

$$\mathrm{SA}_R(Req) = \frac{\sum_{i=1}^{|Req|} S_{\mathrm{recall}}(req_i)}{|Req|} \tag{15}$$

$$\mathrm{SA}_P(Req) = \frac{\sum_{i=1}^{|Req|} S_{\mathrm{precision}}(req_i)}{|Req|} \tag{16}$$

where, each service recall ratio and precision ratio are calculated in Eqs. 13 and 14, respectively. Then, we add up their average value. Three groups of weight parameters are set as: (1) w_{ling}=0.7, w_{struc}=0.3; (2) w_{ling}=0.5, w_{struc}=0.5; (3) w_{ling}=0.2, w_{struc}=0.8. The outcome of SA_R and SA_P are shown in Table 1.

From the results of Table 1, we can conclude that different weighting values influence the average service discovery effect as the change of the parameter w_{ling} and w_{struc}. More specifically, SA_R and SA_P will gradually increase when the structural weight holds more and more quotient but less than a boundary value.

6.2.2 Experiment 2: service annotation performance

The goal of this experiment is to calculate time performance for service annotation process as the number of web services changes. In the initial sample, there are 40 source web service files, and then the service number of each sample increases by 40 web services. Therefore, we can accumulate 10 test samples and respectively measure each time consumption taken by service annotation. The time performance trend of service sample annotation is shown in Fig. 9.

From the performance trend graph in Fig. 9, we can come to the conclusion that time consumption of service annotation has an incremental trend with the persistent growing of web service number. However, it is an approximately linear growth trend so that service annotation process can be accepted within limited time consumption for the existing multiple web services.

6.3 Evaluation analysis

On the basis of the previous simulation experiments, we reach a conclusion that our new approach WS-SAF can not only effectively understand and annotate source web service, but also its annotation performance can be controlled within an acceptable time for web service repository. However, we should still conduct some in-depth analysis here to interpret why our new approach is useful for service annotation.

– An interface concept mapping scheme has been introduced to the semantic expression of service I/O interface. Especially, linguistic similarity calculation is combined with structural similarity so as to map service interface parameter into optimal ontology concept. This process will apparently specify the basic semantic information of service interface in comparison with the traditional syntax-based service organization modes. e.g., when a service provider publishes '*rail transit*' as one of the service interface parameters, '*track traffic*' will be calculated by the interface concept mapping as its mapped ontology concept, which is considered as the standard expression way in a specific domain. Therefore, it can increase service precision ratio $S_{\text{precision}}$ and its average effectiveness SA_P.
– Domain ontology in the area of city traffic has been modeled, constructed and visualized as a DHT, which plays an important role in providing useful semantic information for understanding service I/O function interface thoroughly in our new approach. Although some other related methods also constructed and employed DO but only used to roughly classify web services, they do not make the most use of its strong semantic context for web service annotation.
– We have given an efficient interface expansion scheme based on the semantic relations in the DO. This scheme will help us expand mapped interface concept set and enrich semantic information of service function interface. e.g., if '*track traffic*' is one of the mapped concept of a service input parameters, then its corresponding interface expansion set will be composed of '*track traffic*', '*magnetic suspension*', '*subway*' and '*light rail*' by our semantic expansion scheme. In this case, it will be beneficial for service requesters to have more opportunities to find their closely related web services but it is potential and difficult for other service annotation methods to retrieve them because they do not take enough semantic analysis and reasoning for service interface. Therefore, the proposed scheme can

Table 1 Service average effect under three group of parameters

w_{ling}	w_{struc}	$SA_R(Req)$	$SA_P(Req)$
0.7	0.3	0.7926	0.7534
0.5	0.5	0.8468	0.8520
0.2	0.8	0.8539	0.8851

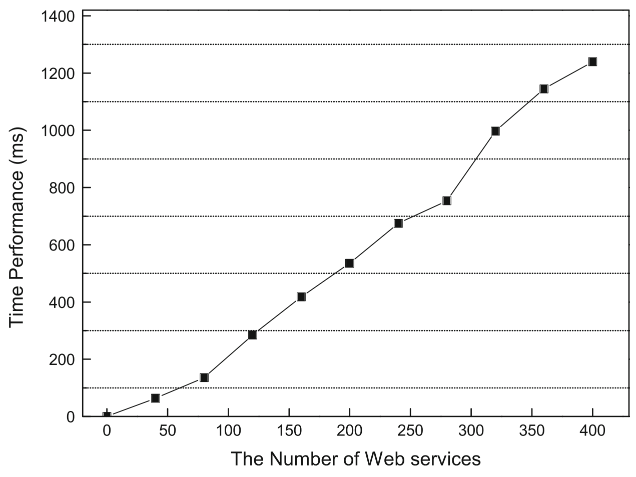

Fig. 9 Time performance trend of web service annotation

effectively improve web service recall ratio S_{recall} and its average effectiveness SA_R.

From the above specific evaluation analysis, we can demonstrate that our new approach for web service annotation can refine service organization mode in an acceptable time performance. However, it must satisfy several conditions for achieving its best effectiveness. Therefore, we further show additional evaluation analysis on when our new approach will be useful for web service annotation.

The new approach will be more useful when it is subject to two restrictions: (1) service providers publish service I/O function interface parameters approximate to normative ontology concepts so that we can best map them into corresponding concepts and (2) service requesters give their requirements explicitly and precisely within three to five words, which is important for them to discover highly related web services.

7 Conclusion and future work

In this paper, we have discussed and proposed a novel approach to annotating web service on how to add semantic information to service I/O function interface. We first give the general framework of service semantic annotation, which is based on our given service semantic description model WS-SDM. Then, we gave an interface concept mapping algorithm to find optimum ontology concept as basic semantic information for each function interface and service interface expansion algorithm to further expand semantic scope of mapped interface concept set. Finally, we presented the process of semantic web service repository generation based on the results of preceding algorithms. Annotated web services by the proposed service annotation method contain abundant semantic information and can be semantically discovered by service matchmaking engine. Experiment results that validate the effectiveness of our proposed approach were also presented.

As the future work plan, we will mainly focus on improving the precision of interface concept mapping, and the optimization of service interface expansion algorithm considering more semantic relations. Meanwhile, we will also consider how to design a scheme for distributing semantic weights to annotated service interfaces so that it can promote service discovery effect.

Acknowledgments We would like to respectively thank the anonymous reviewers and editors of Soft Computing Journal for giving insightful review comments on this manuscript and in charge of our submission. This work was supported in part by National Natural Science Foundation of China under Grant No. 70771077, and National High-Tech Research and Development Plan of China under Grant No. 2008AA04Z106.

References

Ankolekar A, Burstein M, Hobbs J et al (2002) DAML-S: web service description for the semantic web. In: Proceedings of the first international semantic web conference, pp 348–363

Belhajjame K, Embury S, Paton N, Stevens R, Goble C (2008) Automatic annotation of web services based on workflow definitions. ACM Trans Web 2(2):1–34

Curbera F, Nagy W, Weerawarana S (2001) Web services: why and how. In: Proceedings of the OOPSLA 2001 workshop on object-oriented Web services, Tampa, Florida, USA

Forney G Jr (1992) On the hamming distance properties of group codes. IEEE Trans Inform Theory 38(6):1797–1801

Forney G Jr (1966) Generalized minimum distance decoding. IEEE Trans Inform Theory 12(2):125–131

Gruber T (1993) A translation approach to portable ontology specifications. Knowl Acquis 5(2):199–220

Heβ A, Kushmerick N (2003) Learning to attach semantic metadata to web services. In: Proceedings of the second international semantic web conference, pp 258–273

Martin D, Burstein M, Denker G et al (2003) DAML-S (and OWL-S) 0.9 draft release. The draft release report. http://www.daml.org/services/daml-s/0.9/

Miller G (1995) WordNet: a lexical database for English. Commun ACM 38(11):39–41

Patil A, Oundhakar S, Sheth A, Verma K (2004) METEOR-S web service annotation framework. In: Proceedings of the 13th international world wide web conference, pp 553–562

Sivashanmugam K, Verma K, Sheth A, Miller J (2003) Adding semantics to web services standards. In: Proceedings of the international conference on web services, pp 395–401

Srinivasan N, Paolucci M, Sycara K (2004) An efficient algorithm for OWL-S based semantic search in UDDI. In: Proceedings of the first international workshop on semantic web services and web process composition, pp 96–110

Studer R, Benjamins V, Fensel D (1998) Knowledge engineering, principles and methods. Data Knowl Eng 25(12):161–197

The OWL Services Coalition (2004) OWL-S: semantic markup for web services. The OWL-S white paper. http://www.daml.org/services/owl-s/1.0/owl-s.html

Xun E, Yan W (2006) English word similarity calculation based on semantic net. J China Soc Sci Tech Inform 25(1):43–48

Zou G, Zhang B, Gan Y, Zhang J (2008) An ontology-based methodology for semantic expansion search. In: Proceedings of the 5th international conference on fuzzy systems and knowledge discovery, vol 5, pp 453–457