Optimizing Semantic Annotations for Web Service Invocation

Keman Huang, *Member, IEEE*, Jia Zhang, *Senior Member, IEEE*, Wei Tan, *Senior Member, IEEE*, Zhiyong Feng, *Member, IEEE*, and Shizhan Chen, *Member, IEEE*

Abstract—Semantic annotations play an important role in semantics-aware service discovery, recommendation and composition. While existing approaches and tools focus on facilitating the development of semantic annotations on web services, the validation of the quality of annotations is largely overlooked. Meanwhile, the refinement of semantic annotations mostly goes through manual processes, which not only is time-consuming but also requires significant domain knowledge. To enhance the Quality of Semantic Annotation (QoSA), we have developed a technique to incrementally assess and correct semantic annotations of web services. Aiming at supporting web service interoperation, we have formalized the QoSA of input and output parameters. Based on such formalism, test cases are automatically generated to validate service annotations. Learned semantic instances are then accumulated to iteratively validate semantic annotations of other services. Furthermore, a three-phase optimization methodology including local-feedback, global-feedback, and global-propagate is developed to improve the QoSA by incrementally correcting inaccurate annotations. Experiments over a real-world web services repository have demonstrated that our technique can effectively improve QoSA of services, gaining a 78.68 percent improvement in input parameters annotations and identifying 36.47 percent inaccurate output parameters annotations. The proposed technique can be equipped at various service repositories to enhance service discovery and recommendation.

Index Terms—Quality of semantic annotation (QoSA), web service invocation, trusted instance repository, evaluation of QoSA, optimization strategy for QoSA improvement

1 Introduction

With the rapid advancement of Services Computing techniques, more and more reusable software services have been published to the Internet on a daily basis. How to help service users find appropriate candidates in the sea of services becomes increasingly important. Semantic Web technology [1] has been proven effective in service discovery [2], [3]. The last decade has witnessed the emergence of a number of approaches and tools that leverage Semantic Web technology [4], [5], [6], [7] to support service discovery, composition and recommendation. These solutions often exploit various markup languages to annotate service elements (i.e., operations, inputs, and outputs) with the concepts defined in the ontologies. Some example markup languages include the LOD (Linked Open Data) [8], SAWSDL (Semantic Annotations for WSDL) [9], Meta-WSDL, WSML, OWL-S [10], and domain ontology bootstrapping from the web service description [11]. Such concepts in annotations can then be used to help consumers select more suitable services. Apparently, the annotation process plays a fundamental role in these scenarios. Therefore, several tools, such as Iridescent [12], Meteor-S [13], and Kino [14], have been developed by the Semantic Web community to assist the development of annotations for web services.

However, the current tools typically cannot validate how accurate the annotated ontologies reflect the web service semantics. In other word, the *Quality of Semantic Annotation* (QoSA) is not guaranteed and its verification is usually overlooked [15]. Meanwhile, most of the semantic annotation-based service discovery and composition mechanisms assume that the annotations generated by the tools are accurate while it is not always the case [16]. For example, consider an incorrect annotation for the input of operation "*Convert*" in web service "*Rates*"[1] is "*DBpedia: Currency*." It implies the input should be a currency code instance such as "*CNY*" or "*USD*" instead of a currency value. Such an inaccurate annotation will mislead users and fail the interoperation between "*Convert*" and other service operations. In Section 6, our empirical study over a real-world web service repository has revealed that, only 38.61 percent of the original annotations are accurate enough to support web service invocation. Obviously, the inaccuracy of annotations will significantly reduce the effectiveness of the related semantics-based service discovery, recommendation and composition [17]. As a result, in practical use, peer-inspection is oftentimes necessary before annotations are

- *K. Huang, Z. Feng, and S. Chen are with the School of Computer Science and Technology, Tianjin University, Tianjin 300071, China. E-mail: {keman.huang, zyfeng, shizhan}@tju.edu.cn.*
- *J. Zhang is with the Department of Electrical and Computer Engineering, Carnegie Mellon University–Silicon Valley, Moffett Field, CA 94035. E-mail: jia.zhang@ sv.cmu.edu.*
- *W. Tan is with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598. E-mail: wtan@us.ibm.com.*

Manuscript received 1 May 2016; revised 31 July 2016; accepted 19 Sept. 2016. Date of publication 22 Sept. 2016; date of current version 7 Aug. 2019. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TSC.2016.2612632

1. http://www.mondor.org/ces/rates.asmx

1939-1374 © 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: DA IICT. Downloaded on August 06,2024 at 17:01:09 UTC from IEEE Xplore. Restrictions apply.

published, which may be rather time-consuming and require significant domain knowledge and expertise.

How to systematically verify and improve semantic annotations has attracted attentions from both academia and industry in recent years. Belhajjame etc. [18] proposed a verification framework based on software testing technology to assess the adequacy of semantic annotations. Test cases are automatically constructed from workflow provenance logs and used to identify defects from the inputs and outputs annotations. Afterwards, human curators will examine and correct the errors discovered during the verification process. However, correcting inaccurate annotation is a time-consuming task, and it is after-the-fact makeup. Furthermore, when the large-scale and rapid-increasing Semantic Web knowledge base such as LOD is used for annotations, how to select more suitable annotations has become a non-trivial work.

Our hypothesis is that, *provenance data embedded in the past service invocation processes is valuable for QoSA improvement*. For example, the response of a successful service invocation contains accurate semantic information which can be used for QoSA optimization. Our pilot study [19] extends the verification framework proposed by Belhajjame et al. [18] and presents a technique to assist annotators, not only in evaluating but also in improving the QoSA of input parameters of web services.

In this paper, we significantly extend and improve our technique in three dimensions: (1) *supporting evaluation and enhancement of both inputs and outputs annotations*, (2) *supporting semi-automatic ontology bootstrapping*, *and* (3) *enhancing the optimization strategy*. Our goal is to automate the verification and improvement of the quality of semantic annotations for web services to enhance their interoperability. Overall, we have established a four-phase framework for annotation verification and optimization, including semantic annotation, invocation-oriented quality evaluation, feedback-based optimization, and annotation application. As the main motivation for semantic annotation is to support interoperation between services, the basic argument here is that "*the better the semantic annotation can support web service interoperation, the higher QoSA it owns.*" Hence, for the input parameters, their QoSA can be formally defined as "*the success rate of invocation*" while for the output parameters, their QoSA are defined as "*the semantic similarity between the successful output and the annotation.*" Based on such formal definitions, we propose an invocation-based technique to verify the QoSA. Furthermore, we believe the past execution information of the invocations during the assessment is useful for the QoSA improvement, not only the response for the "*successful invocations*" but also the "*rectification operations.*" Therefore, we have *developed a semi-automatic method to bootstrap the ontology* from the response information to form an incrementally growing trusted instance repository (TIR). On top of the TIR, we have designed a three-phase optimization mechanism to facilitate the QoSA optimization, including a *Local-feedback Strategy (LFS)* to improve the annotated instances, a *Global-feedback Strategy (GFS)* to improve the annotated concepts, and a *Global-propagation Strategy (GPS)* to enhance the annotation approach.

Our major contributions are as follows:

1. A formal definition of the Quality of Semantic Annotation is presented, associated with a verification framework.
2. An ontology bootstrapping method is presented to incrementally build a trusted instance repository based on automatic test case generation and invocation.
3. Based on the trusted instance repository, a three-phase optimization methodology is presented to improve the annotation quality, consisting of local-feedback, global-feedback and global-propagation strategies.

We have designed and conducted a set of experiments based on the semantic annotations of real-world web services. Our experiments state that our approach can achieve a significant improvement in QoSA, gaining a 78.68 percent improvement for input parameters annotations and identifying 36.47 percent inaccurate output parameters annotations.

The rest of this paper is organized as follows. Section 2 reviews the related work. Section 3 introduces the four-phase service annotation lifecycle framework. Section 4 details the formal definition of the QoSA associated with the verification framework. Section 5 discusses the three-phase optimization strategy for QoSA improvement. Section 6 reports the implementation as well as the empirical results. Section 7 concludes the paper.

2 Related Work

2.1 Semantic Annotation

Semantic annotation technologies have been widely used in many disciplines, such as biomedical engineering [20], product lifecycle management [15], emotion-oriented research [21], e-commerce [22], web services and RESTful APIs [23], [24], [25], etc. Many tools have been developed to assist human curators in annotating tasks. For example, Meteor-S [13] semi-automatically suggests to users the concepts from domain ontologies to facilitate their annotation task. SAWS [26] is a tool aiming to enhance the WSDL descriptions with semantic concepts provided by domain ontologies. Kino [14] automatically annotates web services based on the similarity between the service's descriptions and the vectors of available ontological terms. It also allows users to utilize the ontology of their own choice for annotation. Based on SAWSDL, the Iridescent tool [12] enables both expert and non-expert users to create semantic service annotations by matching elements and concepts and suggesting annotations. Protégé [27] is an open-source platform that assists consumers in constructing knowledge-based applications with ontologies. With the large volume and rapid growth of available Semantic Web knowledge base [28], some tools exploit ontologies to automate service annotation creation [8], [23], [29]. Hong et al. proposed a linked context model which applies the linked data to model and obtain context data from both users and services [8]. Zhang et al. employed the DBpedia knowledge base to automate the semantic annotation process [23].

However, a service annotation is valuable only when it can accurately reflect the web service's semantic meaning. Existing tools mainly focus on assisting the annotation process and suppose that all the annotations are correct for further usage, while it may not always be true [18]. Our empirical study over a real-world service repository shows that only about 38.61 percent of the original annotations are accurate enough for supporting web service interoperations. Therefore, how to verify and guarantee the quality of semantic annotations has become an important issue for the services computing community.

Authorized licensed use limited to: DA IICT. Downloaded on August 06,2024 at 17:01:09 UTC from IEEE Xplore. Restrictions apply.

2.2 Semantic Annotation Evaluation

Although the semantic annotation technologies have been proven effective for solutions in different disciplines, its effectiveness suffers from the low quality of semantic annotations [18]. This is partly because that it is extremely time-consuming and non-trivial to verify, collect and improve the annotation quality through manual efforts. Therefore, a few tools have been proposed to evaluate the quality of annotations. Mokarizadeh et al. [30] introduced two golden ontologies: one is constructed manually and the other is constructed by automatically learning from web service message element/part names. The difference between the annotation and the golden ontology is considered as the indicator for the QoSA. Meanwhile, the network properties such as small-world and scale-free of the web service network resulted from the semantic annotation are studied and discussed. Belhajjame et al. [18] adapted techniques from traditional software testing to verify the semantic annotations for web services' input and output parameters. An annotated instance pool is generated by trawling the workflow provenance logs [16]. Based on the instance pool, if an operation accepts a particular instance of a concept that is disjoint with the annotation, the annotation will be considered as incorrect. Hence, the QoSA can be evaluated before the annotation becomes publicly available.

These proposals describe a first step towards providing tools for QoSA evaluation. However, they strongly depend on the accuracy of the golden ontologies while the golden ontologies need to be previously constructed before the evaluation. Additionally, how to improve the QoSA is overlooked by these methodologies.

2.3 Semantic Annotation Optimization

To the best of our knowledge, there is no proposal for automatically or semi-automatically improve the semantic annotation quality for web services interoperation. Actually, most of the inaccurate semantic annotations are corrected through manual efforts. In [18], a human curator has to examine the errors discovered and then chooses a different concept for annotation. Due to the large-scale and rapid increasing Semantic Web knowledge base such as Linked Open Data (LOD), it is non-trivial to select correct ontology to improve annotations. Considering the specific feature of Semantic Web service annotation, our previous work [19] introduced a two-layer optimization strategy to improve the web service annotation. Our principle is that the response information from a service invocation is useful for the QoSA improvement, not only the response for the "successful invocation" but also the "rectification operations."

From a different perspective, crowdsourcing [31] has been used for linked data management [32], such as entity linking quality assurance [33], resource management [34] and ontology alignment [35]. However, the crowdsourcing may lead to information and/or cognitive overload which bring no benefit for annotation quality improvement, for example, the crowdsourcing disagreement for collecting semantic annotation [36].

Hence, given a collection of semantic annotations for web services, our goal is to verify and improve the annotations to support web service interoperations. We significantly extended our previous framework in [19] to *take the outputs annotations into account, semi-automatically bootstrap ontologies to recommend potential correct concepts from knowledge base, as well as to introduce the three-phase optimization framework to further enhance the performance.*

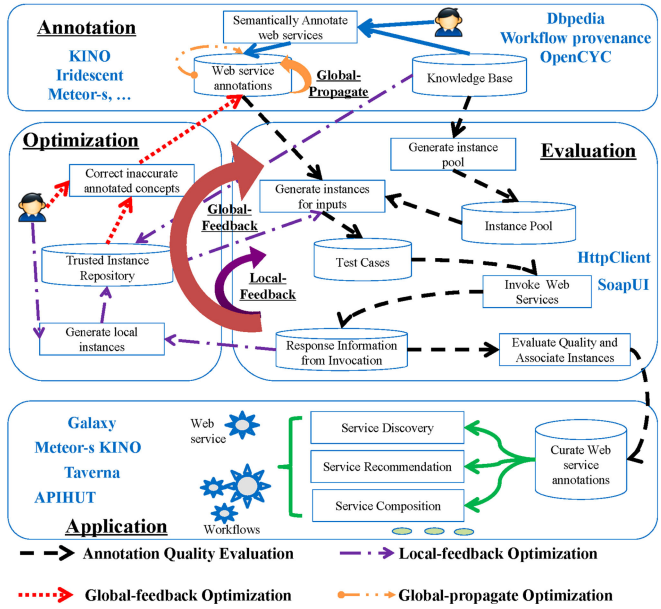


Fig. 1. Four-phase semantic annotation lifecycle management framework.

3 Service Annotation Lifecycle Framework

Significantly extending our previous work [18], we propose a four-phase service annotation lifecycle management framework as shown in Fig. 1. The lifecycle of a service annotation is divided into four phases: semantic annotation, annotation quality verification, annotation optimization and semantic-aware applications.

3.1 Semantic Annotation

In the semantic annotation phase, a user can *semantically annotate web services* either manually or semi-automatically using tools like KINO [14], Iridescent [12], and Meteor-S [13]. Typically, such tools can provide suggestions for annotation based on the *knowledge base* such as DBpedia [37], OpenCYC [38], workflow provenance [16], or BioCatalogue [39]. Based on the *web semantic annotations*, service elements (such as operation names, input/output parameters, and functional description) will be allocated with semantic concepts.

3.2 Annotation Evaluation

In order to evaluate the quality of semantic annotation, *instances for inputs* are drawn from the *instance pool* based on the *knowledge base*, or from the *trusted instance repository* constructed during the QoSA assessment and optimization process. Details about the construction of the TIR will be discussed in Section 5. The instances are treated as *test cases* and the web service operations will be *invoked* through tools such as SoapUI [40] or HttpClient. Therefore, we can get the *response information for each invocation*. Based on the response results, we can *evaluate the quality* of the annotations. Finally, each semantic concept for successfully invoked web service

Authorized licensed use limited to: DA IICT. Downloaded on August 06,2024 at 17:01:09 UTC from IEEE Xplore. Restrictions apply.

operations will be *associated with an instance* which can be used to support the semantics-aware solution.

3.3 Annotation Optimization

Based on the execution status from the evaluation phase, the *trusted instances will be generated* to construct the *TIR*. The *TIR* is empty at the beginning but it will become more comprehensive as time goes by. Then the *TIR* will be used to *generate instances for the invocation* which can improve the QoSA. As it only affects the instance generation but not the concept annotation, it will not impact the annotation phase. Therefore, we name it *Local-Feedback Strategy*. Furthermore, the *TIR* can be used to *correct the inaccurate annotated concepts* during the semantic annotation phase, both for the input and output parameters. Therefore, we name it *Global-Feedback Strategy*. Note that *LFS* and *GFS* both focus on improving the annotations for successful invocations. Therefore, in this paper, we *extract the rules from the annotation optimization log* to deal with annotations of the false invocations. We name it as *Global-propagation Strategy*. After the optimization phase, the annotations including the concepts and instances will become publicly available for further usage. As their QoSA being improved, such annotations will become much more valuable.

3.4 Annotation Application

Based on the evaluation and optimization phase, each semantic annotation for a web service will not only contain the concepts but also the suggested instance, which can enrich the semantic information for further semantic-aware solutions, such as service discovery, composition and recommendation [5], [6], [7], [41], [42]. Details about these technologies are out of scope in this paper.

Note that in the proposed lifecycle framework, different kinds of annotation algorithms and tools can be used to allocate the annotations. Apparently, how to verify and optimize the QoSA are the most critical processes.

4 Service Annotation Evaluation

4.1 Semantic Annotation

For service annotation verification, unlike other methods oriented to the annotations of individual services, our approach is oriented to the annotations of a repository of services. In other words, we learn and leverage peer services' effective annotations to verify and enhance incoming service annotations. During the semantic annotation phase, the service elements will be allocated with a semantic concept from the Semantic Web knowledge base. *The better the annotations can express the web service's input/output parameters' real semantics, the better that the annotations can support the interoperability of the service*. As the main motivation of the semantic annotation for web service is to facilitate the interoperation among different services, we define the basic principle to evaluate the annotation quality as follows:

Definition 1 (Quality of Semantic Annotation for Web Service Interoperation). *The quality of annotations of a service is measured by the success rate of the service interoperation.*

In order to evaluate service interoperability, we focus on the input and output parameters of each comprising operation $\hat{p}_i$ of a web service. Here we extend the traditional semantic annotation into the following tuple which not only considers the annotated concepts $c_{i,j}$ but also the instances $cin_{i,j}$ for each parameter $p_{i,j}$:

$$sa_i(\hat{p}_i) = <I, O> = <\{<ip_{i,j}, ic_{i,j}, icin_{i,j}> \mid 0 \le j \le n_i\}, \{< op_{i,k}, oc_{i,k}, ocin_{i,k}> \mid 0 \le k \le m_i\}>, \quad (1)$$

where $ic_{i,j}$ is the semantic concept and $icin_{i,j}$ is the semantic instance annotated for the input parameter $ip_{i,j}$, n_i is the number of input parameters for $\hat{p}_i$. $oc_{i,k}$ is the semantic concept, $ocin_{i,k}$ is the semantic instance annotated for the output parameter $op_{i,k}$ and m_i is the number of output parameters.

Both the concepts and instances are considered in our definition, because the associated instances can be considered as test cases thus to enrich the semantic information to improve the performance of later semantic-aware solutions. For each semantic annotation, there will be no instance associated with the input/output parameters to start with. Therefore $icin_{i,j}, 0 \le j \le n_i$ and $ocin_{i,k}, 0 \le k \le m_i$ are all null at the very beginning of the verification phase.

As the input and output of each operation play different roles in service interoperation, the evaluation methods should be different. Furthermore, the evaluation of the output parameters' annotations should only be processed for the successful invocations with correct input parameters' annotations. Therefore, we will first discuss the verification for input parameters and then present how to evaluate output parameters' annotations.

4.2 Quality Evaluation for Input Parameters

4.2.1 Annotation Quality

The semantic annotations for input parameters intend to represent what kinds of instances are acceptable for the input parameters. For example, "*DBpedia: Currency Code*" means that only an instance of currency code such as "*CNY*" is acceptable. Therefore, given the semantic annotation for an input parameter $ip_{i,j}$ and a collection of instances $in_{i,j} = < in_1, \ldots in_x >$ generated based on the semantic concept $ic_{i,j}$, if there exists at least one instance that is acceptable for the parameter, the annotation $ic_{i,j}$ for $ip_{i,j}$ is considered as acceptable.

Definition 2 (Annotation Correctness for Input Parameter, ACIP). *Given an operation $\hat{p}_i$ and one of its input parameter $ip_{i,j}$, the annotation $ic_{i,j}$ for $ip_{i,j}$ is correct iff there exists at least an instance $in \in in_{i,j} = <in_1, \ldots in_x>$ generated by $ic_{i,j}$ that is acceptable:*

$$ic_{i,j} \overset{correct}{\rightarrow} <\hat{p}_i, ip_{i,j}> \overset{iff}{\Leftarrow} in \in in_{i,j} = <in_1, \ldots in_x>, \quad in \overset{accepted}{\rightarrow} <\hat{p}_i, ip_{i,j}> . \quad (2)$$

If and only if the annotations for all the comprising input parameters are acceptable, the operation can be successfully invoked. Therefore, we define acceptable annotation for a given operation as follows:

Definition 3 (Annotation Correctness for Input Parameters of Operation, ACIO). *Given an operation $\hat{p}_i$ and its input parameters' semantic annotation $sa_i.I$, the annotation*

Authorized licensed use limited to: DA IICT. Downloaded on August 06,2024 at 17:01:09 UTC from IEEE Xplore. Restrictions apply.

is correct iff the annotation for all of its input parameters are correct:

$$sa_i.I \overset{correct}{\rightarrow} \hat{p}_i \overset{iff}{\Leftarrow} \forall\ <ip_{i,y}, ic_{i,y}> \in \{<ip_{i,j}, ic_{i,j}>\}, \quad ic_{i,y} \overset{correct}{\rightarrow} <\hat{p}_i, ip_{i,y}>\ . \tag{3}$$

Based on the discussions above, we define the QoSA for the input parameters as follows:

Definition 4 (Quality of Semantic Annotation for Input Parameter of Operations, $QoSA_I$). *Given a collection of operations $\{\hat{p}_i, 0 < i \leq N\}$ for a web service and the semantic annotations$sa_i.I$ for the input parameters of each operation $\hat{p}_i$. The $QoSA_I$ is defined as follows:*

$$QoSA_I(\{\hat{p}_i, 0 < i \leq N\}) = \frac{|ACIO|}{N} = \frac{|\{sa_i.I \overset{correct}{\rightarrow} \hat{p}_i\}|}{|\{sa_i\}|}. \tag{4}$$

Obviously $QoSA_I \in [0, 1]$. The larger the $QoSA_I$ is, the better quality the semantic annotation owns. If $QoSA_I$ is equal to 1, all the annotations are correct. If $QoSA_I$ is equal to 0, no annotation is correct.

Algorithm 1. Instance Generation for Input (*IGI*)

Input: $<ip_{i,j}, ic_{i,j}>$: annotated concept for input
x: instances number
TIR: the trusted instance repository
IP: the instance pools from knowledge base
Output: $in_{i,j} = <in_1, \ldots in_x>$: generated instances
Procedure:
01. $in_{i,j} \leftarrow \phi$
02. $query \leftarrow genSPARQL(ic_{i,j}, x)$// Generate SPARQL query with annotation $ic_{i,j}$
03. **IF** $TIR \neq \phi$ **THEN**
04. $lin_{i,j} = executeSPARQL(TIR, query, x)$;
05. $in_{i,j} \leftarrow in_{i,j} \cup lin_{i,j}$;
06. **IF**$|lin_{i,j}| < x$ **THEN**
07. $x \leftarrow x - |lin_{i,j}|$;
08. **ELSE**
09. $x \leftarrow 0$;
10. **ENDIF**
11. **ENDIF**
12. **IF** $x > 0$ **THEN**
13. $pin_{i,j} = executeSPARQL(IP, query, x)$; // execute the query to generated x instances from IP
14. $in_{i,j} \leftarrow in_{i,j} \cup pin_{i,j}$;
15. **ENDIF**

4.2.2 Quality Evaluation

Instance Generation for Input Parameters. From Fig. 1, it can be seen that there exist two sources for instance generation: *an instance pool* (*IP*) from the Semantic Web knowledge base and *a trusted instance repository*. As TIR is constructed based on the execution provenance information from successful invocations during the assessment and optimization phase, the instances in TIR can reveal the specific domain knowledge so that they can be used to guide future invocations. The TIR construction will be detailed in Section 5. As shown in Algorithm 1 we generate the instances from the *TIR* with a higher priority.

Line 02 generates a *SPARQL* query[2] with the given annotation and the candidate instance number. Lines 03∼11 execute the generated query in *TIR* to get the relevant instances. Lines 12∼15 execute the query in the *IP* if the number of instances generated from *TIR* is not enough for the evaluation.

Note that at the beginning, the *TIR* will be empty and all the instances are generated from the *IP* coming from the knowledge base such as DBpedia. Specially, the two files "*mappingbased_properties_en.nt*"[3] and "*infobox_proper ties_en.nt*"[4] which both contain specific instances are used to generate the *IP*. As the evaluation going on, the *IP* will be fleshed out and more instances will be generated from the *TIR*. Hence, the solution in [18] can be considered as a special case in our approach.

Algorithm 2. QoSA Evaluation for Input Parameters (***EIP***)

Input: $SAI = \{sa_i.I\}$: Annotations for Input
Output: $QoSA_I$: QoSA for Input Parameters
$EIR = \{eir_z\}$: Execution information records
$sp = \{sp_i\}$: successful invocations
Procedure:
01. $EIR \leftarrow \phi$; $ACIO \leftarrow \phi$;
02. $sp \leftarrow \phi$; $fp \leftarrow \{\hat{p}_i\}$;
03. **FOR** $sa_i.I \in SAI$
04. **FOR** $< ip_{i,j}, ic_{i,j}, icin_{i,j} > \in sa_i.I$
05. $in_{i,j} \leftarrow IGI(sa_i.I, x, TIR, IP)$; // Use Algorithm 1 to generate the instances for invocation
06. **ENDFOR**
07. **FOR**$\{< ip_{i,j}, in_{i,j,k_{i,j}} > |0 \leq j \leq n, 0 \leq k < x\}$
08. $eir_z \leftarrow invoke(\hat{p}_i, \{< ip_{i,j}, in_{i,j,k_{i,j}} >\})$
09. $EIR \leftarrow EIR \cup eir_z$
10. **IF** $eir_z.st = true$
11. $AICO \leftarrow AICO \cup sa_i.I$
12. $sa_i.icin_{i,j} \leftarrow in_{i,j,k_{i,j}}$; //update the instances
13. $sp \leftarrow sp \cup \hat{p}_i$;
14. **BREAK**;
15. **ENDIF**
16. **ENDFOR**
17. **ENDFOR**
18. $QoSA_I \leftarrow \frac{|ACIO|}{|SA|}$

Evaluation for Input Parameters Annotation. For each operation of a web service, given the combination of instances generated from Algorithm 1 for the input parameters, the invocation will generate the execution information including the invocation status as well as the result. We formally define each *execution information record* as the following tuples:

$$eir =< sa_i, \{< ip_{i,j}, in_{i,j,k_{i,j}} >\}, st, er >, \tag{5}$$

where $\{< ip_{i,j}, in_{i,j,k_{i,j}} >\}$ is the instance combination for each input parameter $st \in \{true, false\}$ is the invocation status, er is the response from the invocation. For description

2. SPARQL is a semantic query language for RDF databases. An example of SPARQL query is shown in Supplementary File Figure A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TSC.2016.2612632.
3. http://wiki.dbpedia.org/Downloads2015-04#dbpedia-ontology
4. http://wiki.dbpedia.org/Downloads2015-04#dbpedia-ontology

Authorized licensed use limited to: DA IICT. Downloaded on August 06,2024 at 17:01:09 UTC from IEEE Xplore. Restrictions apply.

convenience, we denote the successful invocation as $sp = \{sp_i\}$ and the fault invocation as $fp = \{fp_j\}$. Obviously $\{p_i\} = sp \cup fp, sp \cap fp = \phi$.

Therefore, we can get the details of the evaluation process for the input parameters annotations, reported in Algorithm 2. Lines 04∼06 generate the instances for each input parameter; Lines 07∼17 invoke the operation, get the execution information record and then separate the initial operations into the successful invocations and false invocations based on the status. Line 16 calculates the QoSA for input parameters.

If the *TIR* is set as null, it will not be used for instance generation. The evaluation will reflect the annotation quality of the original input parameters annotations.

4.3 Quality Evaluation for Output Parameters

4.3.1 Annotation Quality

The semantic annotations for the output parameters of an operation represent the output values which can be consumed by other services. Many tools, such as Taverna [43] and Galaxy [44], are developed to assist designers in constructing service compositions. Specially, these tools will employ the semantic annotations of the output parameters to search for services that are able to consume them and recommend the interoperable services to the designers. Apparently, the more similar between the semantic annotations and the real output of the operation is, the better the constructed composition is. Hence the quality of the annotation for the output parameters can be considered as *the semantic similarity between the annotations and the output values*.

Definition 5 (Annotation Quality for Output Parameters of Operation, AQOO). *Given a successful operation sp_i, its output parameters' semantic annotation $sa_i.O$, its output values $\widehat{sa_i.O} = \{< op_{i,k}, bc_{i,k}, bcin_{i,k} >\}$, the annotation quality for output parameters is defined as:*

$$AQOO(sa_i) = sim(sa_i.O, \widehat{sa_i.O}) = \sum correct(oc_{i,k}, bc_{i,k}). \tag{6}$$

In this paper, only the situation that the annotated concept $oc_{i,k}$ is the same to the output value $bc_{i,k}$ will be considered as correct:

$$correct(oc_{i,k}, bc_{i,k}) = \begin{cases} 1 & oc_{i,k} = bc_{i,k} \\ 0 & oc_{i,k} \neq bc_{i,k} \end{cases}. \tag{7}$$

Note that only the successfully invoked services can generate the output values. Here, we can define the annotation quality of output parameters as the ratio of the corrected annotations:

Definition 6 (Quality of Semantic Annotation for Output Parameter of Operations, $QoSA_O$). *Given a collection of successfully invoked operations $\{sp_i, 0 < i \leq N_s \leq N\}$ for a web service and the semantic annotations $sa_i.O$ for the output parameters of each operation sp_i. The $QoSA_o$ is defined as follows:*

$$QoSA_o(\{sp_i\}) = \frac{\sum_{i=1}^{N_s} |AQOO(sa_i)|}{\sum_{i=1}^{N_s} |sa_i.O|}, \tag{8}$$

```
<soap:Body>
  <GetCurrencyListResponse xmlns="http://mondor.org/">
    <GetCurrencyListResult>
      <Currency><Code>AED</Code><Name>UAE
Dirham</Name></Currency>
      <Currency><Code>ALL</Code><Name>Albanian
Lek</Name></Currency>
      ……
      <Currency><Code>ZAR</Code><Name>South African
Rand</Name></Currency>
      <Currency><Code>ZMW</Code><Name>Zambian
Kwacha</Name></Currency>
    </GetCurrencyListResult>
  </GetCurrencyListResponse>
</soap:Body>
```

Fig. 2. Execution result from the invocation of the operation "GetCurrencyList" in web service "Rates."

where $|sa_i.O|$ refers to the number of output parameters for sp_i.

4.3.2 Annotation Evaluation

Bootstrapping from Invocation Response. For each successful invocation, the web service operation will reveal information about the execution. The response represents the values output by the services which can be consumed by other services. Fig. 2 illustrates the invocation result from the operation "*GetCurrencyList*" in web service "*Rates*."

It can be seen that each item "*<Currency>…</Currency>*" in the resulting list refers to one currency, "*AED*" in "*<Code> AED</Code>*" is an ISO currency code, and "*UAE Dirham*" in "*<Name>UAE Dirham</Name>*" is a currency name. Such information can be used to generate the instances which are accurate for the invocation. Additionally, we can extract the concepts representing the output of the web services. Similar to [11], [45], we extract the tokens from the execution result. As the execution result is in XML format, each value of the XML schema's leaf element will be considered as a *trusted instance (tin)*; each element name list from the finest granularity to the general levels with operation name will be considered as the *candidate concept (cc)*. The granularity level is used to build the concept hierarchy. For example, the possible concepts in Fig. 2 are "*Currency Code*," "*Currency Name*" and "*Currency*" while "*Currency Code*," "*Currency Name*" are the sub classes of "*Currency*."

For each candidate concept, we consider it as the key word and then query from the following three sources "*OWL*," "*Resource*" and "*Property*" in DBpedia[5]:

OWL (dbpedia_2015-04.owl[6]): This dataset consists of the information about the ontology concepts in DBpedia, including the class, datatype property and object property.

Resource (mappingbased_properties_en.nt[7]): This dataset contains triples extracted from the respective Wikipedia

5. http://oldwiki.dbpedia.org/Datasets#h18-19
6. http://wiki.dbpedia.org/Downloads2015-04#dbpedia-ontology
7. http://wiki.dbpedia.org/Downloads2015-04#dbpedia-ontology

Authorized licensed use limited to: DA IICT. Downloaded on August 06,2024 at 17:01:09 UTC from IEEE Xplore. Restrictions apply.

TABLE 1
Trusted Instance Records for "*GetCurrencyList*"

cc_k	bc_k	$\{tin_q \mid 0 < q \leq M_k\}$
Currency	DBpedia: Currency	/
Currency Code	DBpedia: Currency Code	EUR, AUD, BRL, CAD, CNY, CUP, EUR, EGP, etc.
Currency Name	DBpedia: Currency Name	Dollar, Taka, Franc, Pound, Yen, Krona, Baht, Lira, etc.

whose subject and object resource have an equivalent English article.

Property (infobox_properties_en.nt[8]*)*: This dataset contains all properties extracted from all infoboxes and templates within all Wikipedia articles.

Based on the quality of these data sources, we use the priority ordering: "*OWL*">"*Resource*">"*Property*." This means that the generated concepts from the higher priority data sources will have a higher ranking. In order to guarantee the accuracy of the trusted instance repository, the bootstrapping process will be semi-automatic and the annotator will participate in the "*getMappingConceptFromOWL*," "*getMappingConceptFromResource*," "*get-MappingConceptFromProperty*" process to pick the final concepts from the query result. How to improve the recommendation performance, for example, using the hierarchy to generate better query result, is out of scope and we will leave it as the future work.

Finally, from the response information, we can get the trusted instance records modeled as the following tuples:

$$tir(sp_i) = \{ < cc_k, bc_k, \{tin_q | 0 < q \leq M_k\} > \}, \quad (9)$$

where sp_i is the successfully invoked operation, cc_k is the candidate concept, bc_k is the bootstrapping concept generated from the bootstrapping process, $\{tin_q | 0 < q \leq M_k\}$ is the extracted instance list and M_k is the trusted instances number.

Algorithm 3 details the bootstrapping process. Line 02 extracts the instances and candidate concepts from the response information. Line 04∼Line 10 query the bootstrapping concepts from the three DBpedia sources and the annotators will participate to select the bootstrapped concepts from the recommended list. Lines 11∼13 add the generated records into the trusted instance repository. Table 1 shows the generated result for the response from "*GetCurrencyList*".

Evaluation for Output Parameters Annotation. Based on the definition shown above, the evaluation of the output parameter is straightforward. For example, as shown in Table 2, the original annotations for the output parameters of "*GetCurrencyList*" are "*DBpedia: Currency*," "*DBpedia: PostalCode*" and "*DBpedia: GivenName*" while the generated concepts are "*DBpedia: Currency*," "*DBpedia: Currency Code*" and "*DBpedia: Currency Name*." Then the $QoSA_o$ for "*GetCurrencyList*" is only 1/3.

Details about the evaluation of $QoSA_o$ are reported in Algorithm 4. Line 03 uses Algorithm 3 to extract the bootstrapping concepts from the response information. Lines 04∼06 compare the original annotated concepts with the bootstrapping concepts. Line 08 calculates the quality of the output parameters annotations.

TABLE 2
$QoSA_o$ for "*GetCurrencyList*"

bc_k	oc_k
DBpedia:Currency	*DBpedia: Currency*
DBpedia:Currency Code	*DBpedia: PostalCode*
DBpedia:Currency Name	*DBpedia: GivenName*

Algorithm 3. Bootstrapping Response Information (*BRI*)

Input: sp_i: successfully invoked operation
$er(sp_i)$: response from the invocation
Output: $tir(sp_i) = \{ < cc_k, bc_k, \{tin_q | 0 < q \leq M_k\} > \}$
Procedure:
01. $tir = \phi$;
02. $\{ < cc_k, \{tin_q\} > \} \leftarrow tokenExtraction(er)$; // extract the candidate concepts and trusted instance list from the execution response;
03. **FOR** $cc_k \in \{ < cc_k, \{tin_q\} > \}$
04. $bc_k \leftarrow getMappingConceptFromOWL(cc_k)$;
05. **IF** $bc_k = null$
06. $bc_k \leftarrow getMappingConceptFromResource(cc_k)$;
07. **IF** $bc_k = null$
08. $bc_k \leftarrow getMappingConceptFromProperty(cc_k)$;
09. **ENDIF**
10. **ENDIF**
11. **IF** $bc_k \neq null$
12. $tir \leftarrow tir \cup \{ < cc_k, bc_k, \{tin_q\} > \}$;
13. **ENDIF**
14. **END FOR**

Algorithm 4. QoSA Evaluation for Output Parameters (*EOP*)

Input: $\{sp_i\}$: the annotations for the output parameters of the successful invocations.
$\{tir(sp_i)\}$: the concepts generated from the response
Output: $QoSA_o$
Procedure:
01. $AQOO \leftarrow 0$; $SAO \leftarrow 0$;
02. **FOR** $sp_i \in \{sp_i\}$
03. $bc(sp_i) \leftarrow \{ < cc_k, bc_k > \} \in tir(sp_i)$;// bootstrapping concepts from the response information
04. $oc(sp_i) \leftarrow \{ < oc_{i,k} > \}$; //original annotated concepts
05. $AQOO \leftarrow AQOO + sim(bc(sp_i), oc(sp_i))$;
06. $SAO \leftarrow SAO + |sa_i.O|$//number of output parameters
07. **END FOR**
08. $QoSA_o = \frac{AQOO}{SAO}$

5 Service Annotation Optimization

During the verification phase, the quality of the semantic annotation is evaluated, and the inaccurate annotations are also identified so that we can improve them. As discussed above, each semantic annotation for web service consists of the concepts and the instances, the optimization methodology should improve both the associated concepts and instances. Additionally, we can use the optimization log to

8. http://wiki.dbpedia.org/Downloads2015-04#dbpedia-ontology

Authorized licensed use limited to: DA IICT. Downloaded on August 06,2024 at 17:01:09 UTC from IEEE Xplore. Restrictions apply.

further improve the annotations for the false operations. Therefore, our annotation optimization methodology consists of the following three folds:

Local-feedback strategy to improve the annotated instances for the input parameters annotations;
Global-feedback strategy to improve the annotated concepts for both input and output parameters annotations;
Global-propagation strategy to improve annotations for false operations.

5.1 Local-feedback strategy for Input Parameter

As we extract the response information of the successful invocations to generate the trusted instance records to enrich the trusted instance repository, more accurate instances can be generated during the "*Instance Generation*" phase. It is possible that some of the unsuccessful invocations can become successful with the new instances. Therefore, the local-feedback strategy for the input parameter annotation optimization is straightforward.

Definition 7 (Local-Feedback Strategy for Input Parameter Annotation Optimization). *For each successful invocation during evaluation, get the execution response, extract the trusted instance records and update the trusted instance repository.*

The trusted instance repository starts with no instance. The instances generated for the first-round evaluation all come from the knowledge base. However, as time goes by, the successful invocations will enrich the trusted instances in *TIR*. Due to the fact that the trusted instance records in *TIR* are more accurate, some false invocations due to the illegal input instances will become successful during the local-feedback optimization. Here we name them as the *rectification operations* for convenience. The optimization process will stop when there exist no more *rectification operations*. Details are reported as Algorithm 5.

Algorithm 5. Local-Feedback Strategy

Input: $SAI(\widehat{p}_i) = \{sa_i.I\}$: Input Parameters Annotations
Output: $sp = \{sp_i\}$: successful invocations
$fp = \{fp_j\}$: fault-invocations
Procedure:
01. $sp = \phi$;$fp = \{\widehat{p}_i\}$;
02. **Repeat**
03. $< QoSA_I, EIR, nsp > \leftarrow EIP(SAI(fp), x)$;
04. **IF** $nsp \neq null$
05. **FOR** $sp_i \in nsp$
06. $BRI(sp_i, eir(sp_i))$;
07. **ENDFOR**
08. $sp \leftarrow sp \cup nsp$;
09. $fp \leftarrow fp - nsp$;
10. $QoSA_I = \frac{|sp|}{|\widehat{p}_i|}$ // calculate the total QoSA
11. **ELSE**
12. **BREAK**;
13. **ENDIF**
14. **Until Convergence**

Line 03 invokes the initial or unsuccessful operations to get the accurate instances. Lines 05∼07 bootstrap the concepts and instances from the successful responses to update the trusted instance repository. Lines 08∼09 update the successful and false invocations based on the response information. Line 10 calculates the QoSA for the whole original annotations. Line 12 stops the process if there is no more successful invocation.

5.2 Global-feedback Strategy for Input

In the local-feedback strategy, the instances generated for invocations will be updated for each round until the LFS reaches a convergence. Obviously, the original semantic annotations for the *rectification operations* are inaccurate. These *reclaimed records (RR)* can be used to help the annotators correct their annotations. Here we formally define them as the following tuple:

$$rr = < sa_i, \{ < ip_{i,j}, tin_{i,j,k_{i,j}} > \ | 0 \leq j \leq n \} >, \quad (10)$$

where $\{ < ip_{i,j}, tin_{i,j,k_{i,j}} > \}$ is the instance combination which makes earlier unsuccessful invocation successful.

Each instance $tin_{i,j,k_{i,j}}$ is allocated with a concept bc_k. If the original annotated concept $ic_{i,j}$ for the parameter $ip_{i,j}$ is different from bc_k, the original annotation is considered as inaccurate and it should be replaced by bc_k. Hence, the global-feedback strategy for the annotation optimization can be described as follows:

Definition 8 (Global-Feedback Strategy for Input Parameter Annotation Optimization, GFSI). *For each reclaimed record generated during the LFS, identify the inaccurate annotation for the input parameter and update it to the new concept.*

The original inaccurate concepts will be replaced by the new correct annotations. Here we name these optimization logs as the *optimized annotation (OA)* for convenience and formally define them as follows:

$$oa_k = < p_{i,k}, c_{i,k}, bc_{i,k}, tin_{i,k} >, \quad (11)$$

where $p_{i,k}$ refers to the parameter for the operation $p_{i,k} = < name_{i,k}, type_{i,k} >$, $name_{i,k}$ refers to the parameter's name, $type_{i,k}$ refers to its data type, $c_{i,k}$ is the original annotated concept and $bc_{i,k}$ is the bootstrapping concept. $tin_{i,k}$ is the list of the trusted instances.

Algorithm 6. Global-Feedback Strategy for Input (*GFSI*)

Input: $RR = \{rr_s\}$// the reclaimed records from LFS
Output: $OA = < oa_k >$;//the optimized log
Procedure:
01. **FOR** $rr_s \in RR$
02. **FOR** $< ip_{i,j}, tin_{i,j,k_{i,j}} > \in rr_s$
03. $bc_{i,j} \leftarrow getConcept(tin_{i,j,k_{i,j}}, TIR)$;
04. **IF** $sa_i.ic_{i,j} \neq bc_{i,j}$
05. $sa_i.ic_{i,j} \leftarrow bc_{i,j}$;
06. $sa_i.icin_{i,j} \leftarrow tin_{i,j,k_{i,j}}$;
07. $oa_k = < ip_{i,j}, sa_i.c_{i,k}, bc_{i,k}, tin_{i,j,k_{i,j}} >$
08. $OA \leftarrow OA \cup oa_k$
09. **ENDIF**
10. **ENDFOR**
11. **ENDFOR**

Authorized licensed use limited to: DA IICT. Downloaded on August 06,2024 at 17:01:09 UTC from IEEE Xplore. Restrictions apply.

TABLE 3
Annotation Update during GFSI (Part)

Original Annotation	Instance ($tin_{i,j,k_{i,j}}$)	Replace Annotation
DBpedia:Currency	Dollar	DBpedia: Currency Name
DBpedia:Currency	CAD	DBpedia: Currency Code
DBpedia:Programming Language	English	DBpedia: Language Name
DBpedia:endDate	2014-12-16T08: 00:00	DBpedia:Datetime
DBpedia:Country	CN	DBpedia: Country Code

Algorithm 6 details the global-feedback strategy for input parameters. Line 03 gets the bootstrapping concept of the instance from the trusted instance repository. Lines 05∼06 update the inaccurate annotation into the new one. Lines 07∼08 generate the optimized log. Table 3 shows a snapshot of annotations correction during the *GFSI*.

5.3 Global-feedback Strategy for Output

As discussed in Section 4.3, we extract the bootstrapping concepts from all the successfully invoked responses. Then we consider the similarity between the original output parameters annotations and the bootstrapping concepts as the quality of the annotation. Straightforwardly, the original annotations which are different from the bootstrapping concepts are considered as the inaccurate annotations. Therefore we can use the bootstrapping concepts to replace them.

Definition 9 (Global-Feedback Strategy for Output Parameter Annotation Optimization, GFSO). *For each successful invocation, extract the bootstrapping concepts from the response and update the inaccurate output parameters annotations.*

Algorithm 7. Global-Feedback Strategy for Output (*GFSO*)

Input: $\{sp_i\}$: the annotations for the output parameters of the successful invocations.
$\{tir(sp_i)\}$: the concepts generated from the response
Output: $OA = < oa_k >$;//the optimized log
Procedure:
01. **FOR** $sp_i \in \{sp_i\}$
02. $oc(sp_i) \leftarrow \{ < oc_{i,k} > \}$; //original annotated concepts
03. $bc(sp_i) \leftarrow \{ < cc_k, bc_k > \} \in tir(sp_i)$;// bootstrapping concepts from the response information
04. **FOR**$oc_{i,j} \in oc(sp_i)$
05. **IF** $oc_{i,j} \neq bc_{i,j}$
06. $sp_i.oc_{i,j} \leftarrow bc_{i,j}$;
07. $sp_i.ocin_{i,j} \leftarrow tin_{i,j,k_{i,j}}$;
08. $oa_k = < op_{i,j}, oc_{i,j}, bc_{i,j}, tin_{i,j,k_{i,j}} >$;
09. $OA \leftarrow OA \cup oa_k$;
10. **ENDIF**
11. **ENDFOR**
12. **ENDFOR**

Considering the example shown in Table 2 again, the original annotations "*DBpedia: PostalCode*" and "*DBpedia: GivenName*" for the output parameters of "*GetCurrencyList*" will be replaced by the bootstrapping concepts "*DBpedia: Currency Code*" and "*DBpedia: Currency Name*" during the global-feedback strategy for the output parameters. Algorithm 7 details the GFSO. Line 06∼07 replace the inaccurate annotated concept with the bootstrapping concept. Lines 08∼09 update the optimized log. Table 4 shows a snapshot of annotations correction during the *GFSO*.

TABLE 4
Annotation Update during GFSO (Part)

Invocation Result	Original	Replace
<Code>AD </Code>	DBpedia: postalCode	DBpedia: currency Code
<Name>UAE Dirham </Name>	DBpedia: GivenName	DBpedia: currencyName
<State>NY </State>	DBpedia: highestState	Dbpedia: stateName
<City>Albany </City>	DBpedia: BroadcastNetwork	Dbpedia: cityName

5.4 Global-propagation Strategy for False-Operations

As discussed above, the LFS, GFSI, GFSO all focus on the successful invocations. For the false-operations, fortunately, the optimized log can be used to update the annotations. The basic idea here is that "*if the annotations for fp_j is similar to oa_k, then the annotation $c_{j,k}$ for fp_j can be updated by $bc_{i,k}$.*" Hence, the strategy to further improve the annotations for the false-operations can be defined as:

Definition 10 (Global-propagation Strategy for false-operation annotation optimization, GPS). *For the annotation for a false-operation, if there exist a similar optimized annotation (oa) record, then update the annotation with the bootstrapping concept in oa.*

Algorithm 8. Global-Feedback Strategy (*GPS*)

Input: $\{fp_i\}$: the annotations for the fault invocations.
$\{oa_k\}$: the optimized annotations
Procedure:
01. **FOR** $fp_i \in \{fp_i\}$
02. **FOR** $< p_{i,j}, c_{i,j} > \in fp_i$
03. **IF** $sim(< p_{i,j}, c_{i,j} >, oa_k) \geq T$
04. $fp_i.c_{i,j} \leftarrow oa_k.bc_{i,k}$;
05. $fp_i.cin_{i,j} \leftarrow oa_k.tin_{i,k}$;
06. **ENDIF**
07. **ENDFOR**
08. **ENDFOR**

The calculation of the similarity between two parameters with semantic annotations has been widely discussed [46], [47], [48], [49]. In order to guarantee the accuracy of the optimization, we first calculate the similarity between the original annotations for the parameters and the optimized annotations, and then annotators need to participate to select the recommended revisions. Specially, if the parameter name, data type and the associated annotation are all the same, we will directly replace the original concept by the bootstrapping concept. Actually, the empirical study shows that 58.78 percent revisions fall into this scenario.

Authorized licensed use limited to: DA IICT. Downloaded on August 06,2024 at 17:01:09 UTC from IEEE Xplore. Restrictions apply.

TABLE 5
Benchmark Datast

# service	#operations	#inputs	#output
115	790	8,036	6,290

Algorithm 8 details the GPS to improve the annotations for the false operations. Lines 03∼06 update the annotations of the fault-operation if there exists a similar optimized annotation. In this paper, we just set $T = 0.8$ and then the annotators will participate in the similarity calculating process to select one for the replacement.

5.5 Annotations Optimization

Based on the strategies presented above, in order to optimize the annotation, the LFS is first used to construct the trusted instance repository and allocate the instances for the input parameters, and then the GFSI and GFSO are used to semi-automatically correct the inaccurate concepts for the successful invocations. The GPS is further employed to update the annotations for the false-operations. The optimized annotations process will stop until no more new successful invocations happen. Finally, the optimized annotations, consisting of the concepts and instances, will be publicly published for further application. Algorithm 9 details the optimization process.

Line 03 uses Algorithm 1 to invoke the operations. Lines 05∼07 use Algorithm 2 to extract the bootstrapping concepts. Line 08 organizes the reclaimed records. Line 10 uses Algorithm 6 to update the concepts for input parameters and Line 11 uses Algorithm 7 to correct the output parameters concepts. Line 12 uses Algorithm 8 to update the annotations for the false-invocations.

6 Experiments

6.1 Data and Prototype

In order to prove the effectiveness of the presented framework, we have implemented a prototyping system based on our proposed lifecycle model.

Similar to [23], we employed the dataset consisting of 300 real-world web services with WSDL documents[9], relevant to travel and weather domains collected from the web. Since only the web services available for invocation can be used for the QoSA evaluation, we removed the services whose endpoints or WSDL references are inactive. Afterwards, we further filtered the services with errors because they are not actually available. Error messages we used include *"Endpoints refer to other websites or services," "Service data has been ported," "Endpoints turn out to be other URLs while accessing,"* and *"Services have no truly useful content."* Additionally, our previous work [19] found that 151 operations are always failed with the return information *"There is a problem with the resource you are looking for and it cannot be displayed."* Thus, those operations were removed from the dataset. Finally, we received a dataset summarized in Table 5, consisting of 115 services, 790 operations and 14,326 parameters (6,290 outputs parameters and 8,036 inputs parameters) as the benchmark.

9. The WSDL documents used in the experiment have been publicly available in our website: http://colmanzf.com/colman/?page_id=25

Algorithm 9. Annotation Optimization Process

Input: $SA(\widehat{p}_i) = \{sa_i\}$: Original Annotations
Output: $sp = \{sp_i\}$: successful invocations
$fp = \{fp_j\}$: fault-invocations
Procedure:
01. $sp = \phi$;$fp = \{\widehat{p}_i\}$;
02. **Repeat**
03. $< QoSA_I, EIR, nsp > \leftarrow EIP(SAI(fp), x)$;
04. **IF** $nsp \neq null$
05. **FOR** $sp_i \in nsp$
06. $BRI(sp_i, eir(sp_i))$;
07. **ENDFOR**
08. $RR \leftarrow constuctRR(nsp)$;
09. $OA \leftarrow \phi$;
10. $OA \leftarrow GFSI(RR)$;
11. $OA \leftarrow OA \cup GFSO(nsp, LIR)$;
12. $GPS(fp, OA)$;
13. $sp \leftarrow sp \cup nsp$;
14. $fp \leftarrow fp - nsp$;
15. **ELSE**
16. **BREAK**;
17. **ENDIF**
18. **Until Convergence**

The experimental environment is a Windows 7 machine equipped with 3 GB memory, Myeclipse 9 and Mysql 5.5. The annotation approach presented in [23] is used to generate the original semantic annotations as the baseline. Dbpedia 2015-04 is used as the semantic knowledge base. An Httpclient-based simulator was developed to automatically invoke web service operations.

To facilitate the concepts bootstrapping from the successful response information, we have developed a semi-automatic bootstrapping tool, reported its screenshot in supplementary Fig. B, available online, to help annotators to select suitable concepts from the semantic knowledge base. As discussed in Section 4.3.2, the response file is loaded into the tool and then the candidate concepts will be extracted from the file. Furthermore, for each candidate concept, we query the concepts from the three knowledge bases in DBpedia so that the annotators can pick the most suitable ones as the bootstrapping concepts.

Note that our framework is generic and it can be further extended in the following aspects: 1) the annotation approach can be replaced by other techniques such as KINO [14], Iridescent [12], Meteor-S [13] etc.; 2) the knowledge base can be switched to OpenCYC [38], workflow provenance [16], or BioCatalogue [39] etc.; and 3) the dataset can be substituted by other WSDL-based service dataset [50]. Since semantic annotation technologies have been recently developed for the RESTful services [24], [52], our framework can also be used for RESTful services' annotations.

6.2 Experiments and Discussions

6.2.1 Optimization for Input Parameters

Comparison Methodology. To evaluate the effectiveness of our optimization framework for input parameters annotations, we set the instances for evaluation as six and then considered the following methodologies:

Authorized licensed use limited to: DA IICT. Downloaded on August 06,2024 at 17:01:09 UTC from IEEE Xplore. Restrictions apply.

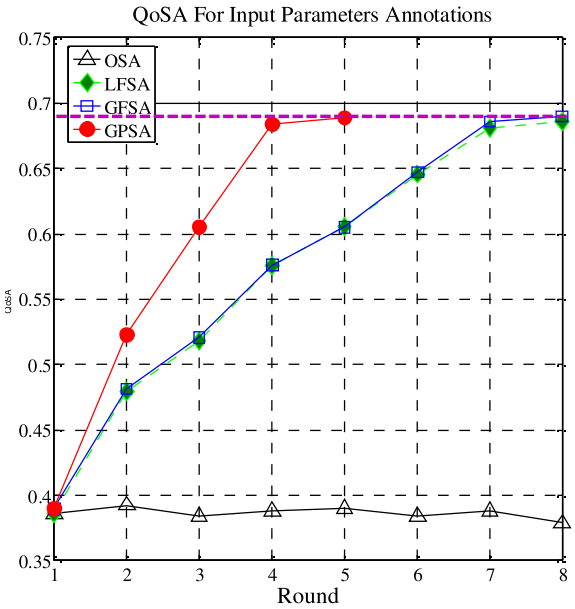


Fig. 3. Effectiveness of the optimization framework for input.

Original Semantic Annotation (OSA): all instances for each annotation are generated from Dbpedia and no feedback optimization strategy is employed.

Local-feedback Strategy for Annotation Optimization (LFSA): instances are generated from both Dbpedia and the trusted instance repository. The local-feedback strategy is used to optimize the semantic annotation.

Global-feedback Strategy for Annotation Optimization (GFSA): based on the LFSA, the global-feedback strategies GFSI is used to update the inaccurate concepts.

Global-Propagate Strategy for Annotation Optimization (GPSA): based on the GFSA, the global-propagation strategy is further used to correct the inaccurate concepts for input parameters annotations of the fault invocations.

Results. To remove the random effect from the dynamic environment, we constructed four clients using the same network environment for each experiment. For each round, we ran eight times to get the average result.

Fig. 3 reports the result of the $QoSA_I$ for each methodology. It can be seen that the $QoSA_I$ for the original annotations was no more than 40 percent, about 38.61 percent. It proves that it is not suitable to suppose that the semantic annotations for web services are always accurate. This result is consistent with the conclusion in [16] which motivates our research. On the contrary, our optimization framework can effectively improve the quality of the input parameters annotations. The $QoSA_I$ for the three methods all reached 0.6899, gaining a 78.68 percent improvement in $QoSA_I$. The GFSA and the LFSA gained the similar performance because the GFSA only corrects the annotation for the successful invocations which will not be tested in the next round.

Additionally, comparing with the LFSA and the GFSA, the GPSA gained a faster convergence speed. It reached a convergence in the fifth round while the LFSA and GFSA both need eight rounds. This may be because GPSA will update the inaccurate annotations for the false invocations so that more false invocations can be successfully invoked in the next round.

TABLE 6
Benchmark Datast

	Without	With
OSA	OSA	OSAW
LFSA/GFSA	LGFSA	LGFSAW
GPSA	GPSA	GPSAW

Furthermore, we conducted a depth analysis about the reclaimed records in which the annotations were inaccurate at the beginning but were corrected after the optimization. It can be seen that the *percent of exceptions from the illegal invocation* (*IIE*) "*Server was unable to process request. Object reference not set to an instance of an object*" is rapidly decreasing. After the optimization process, the *IIE* will reduce from 44.18 to 14.30 percent. This means that our optimization framework can help to annotate more accurate instances for the operations, which can enhance the effectiveness of the semantic-aware solutions.

6.2.2 Optimization for Output Parameters

Comparison Methodology. To evaluate the performance of our optimization framework to update the output parameters annotations, based on whether the global-propagation strategy is used to identify the inaccurate output parameters annotations, we considered the six methodologies shown in Table 6.

Result. Fig. 4 shows the result of the $QoSA_o$ for each methodology. It can be seen that the propagation methodology gained significant improvement, about 36.66∼43.61 percent, for all the three approaches.

Furthermore, comparing OSAW, LGFSAW, and GPSAW, it can be seen that LGFSAW and GPSAW both gained significant improvement than OSAW, from 7.06 to 36.47 percent. Similarly, GPSAW gained a faster convergence speed than the LGFSAW. Therefore, we can conclude that our framework can significantly improve the quality of the output parameters annotations.

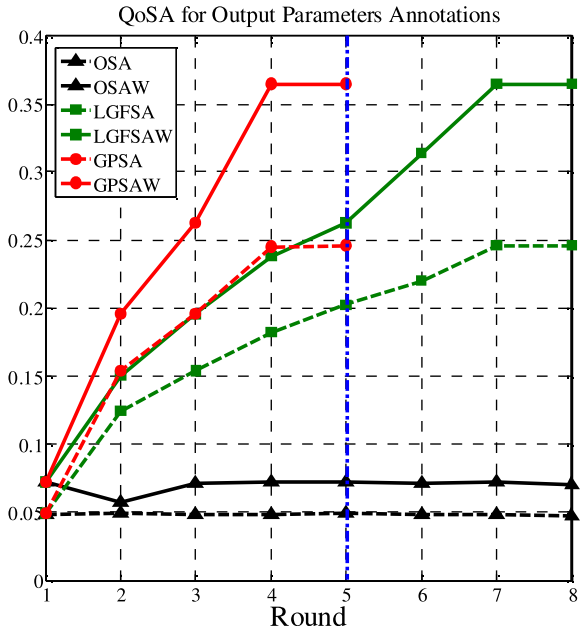


Fig. 4. Effectiveness of the optimization framework for output.

Authorized licensed use limited to: DA IICT. Downloaded on August 06,2024 at 17:01:09 UTC from IEEE Xplore. Restrictions apply.

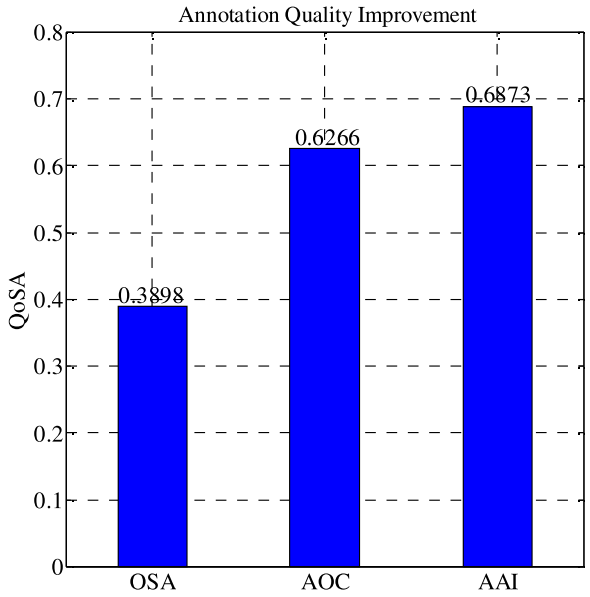


Fig. 5. Effectiveness of the optimization framework for output.

6.2.3 Annotation Publication Performance

Comparison Methodology. The semantic annotations will finally be published associated with the corresponding web services. To evaluate the quality of the published annotations, we consider the following two strategies:

Annotation with Optimized Concept (AOC): Annotations are optimized by the optimized framework and only the concepts are published.

Annotation with Attached Instance (AAI): Annotations are optimized by the optimized framework and the concepts as well as the instances are published.

Results. Here we consider the quality of the input parameters annotations for each strategy. As shown in Fig. 5, comparing with the original annotations, the *AOC* gained a 60.71 percent improvement in $QoSA_I$. This means that the annotated concepts for the web service annotations become more accurate after optimization, so that more accurate instances can be generated from the semantic knowledge database. Furthermore, if the attached instances were available, the $QoSA_I$ reached 0.6873, showing a 76.30 percent improvement to the original annotations.

Thus, we can conclude that our framework can effectively improve the accuracy of the annotated concepts and the associated instances can help to facilitate the interoperations between web services.

7 Conclusions

In this paper, we have presented a technique capable of verifying and improving the quality of semantic annotations for web services to support their interoperations. QoSA is formally defined as how well the semantic annotations can support the interoperation among web services. Based on a trusted instance repository incrementally constructed during the verification and optimization process, a three-phase optimization framework is presented to improve the QoSA, including the *Local-feedback strategy* to optimize annotated instances; *Global-feedback strategy* to correct inaccurate annotated concepts for successful invocations; and *Global-propagation strategy* to update the annotations for false operations.

Based on the presented framework, given a collection of semantic annotations for real-world web services, our empirical study shows that:

Comparing with the original annotations, the framework gains 78.68 percent improvement in QoSA of the input parameters annotations.

The framework can effectively identify and improve the outputs parameters annotations, from 7.06 to 36.47 percent.

Comparing with the original annotations, the optimized annotations can gain 60.71 percent improvement in quality. If the trusted instance repository is published, we gain 76.30 percent improvement, reaching 0.6873.

We believe that our presented technique can be applied at a service repository to help evaluate and improve the semantic annotations of its comprising services. A service will be published only after its QoSA reaches a predefined threshold.

In this paper, the bootstrapping process and the optimized process are both semi-automatic, which means that the candidate concepts are recommended for annotators to select. In the future, we will focus on recommendation performance improvement to further facilitate the annotation quality optimization. Additionally, our current experiments are based on SOAP-services; we plan to further employ our framework to improve RESTful services' semantic annotation in our following work. Finally, as the service context such as the way how services could be retrieved and used will also affect the service interoperations, we will further extend our framework to consider the service context as well as to support the shimming [51] between services to improve the interoperation performance.

Acknowledgments

The authors thank Juan Chen for the assistance in the implementation of the prototype. Shizhan Chen is the corresponding author. This work is supported by the National Natural Science Foundation of China grants 61373035, 61572350, 61502333.

References

[1] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Sci. Amer.*, vol. 284, pp. 28–37, 2001.

[2] B. Medjahed, A. Bouguettaya, and A. K. Elmagarmid, "Composing web services on the Semantic Web," *Int. J. Very Large Data Bases*, vol. 12, pp. 333–351, 2003.

[3] S. A. McIlraith, T. C. Son, and H. Zeng, "Semantic Web services," *IEEE Intell. Syst.*, vol. 16, no. 2, pp. 46–53, 2001.

[4] A. V. Paliwal, B. Shafiq, J. Vaidya, H. Xiong, and N. Adam, "Semantics-based automated service discovery," *IEEE Trans. Serv. Comput.*, vol. 5, no. 2, pp. 260–275, Apr.-Jun. 2012.

[5] K. P. Joshi, Y. Yesha, and T. Finin, "Automating cloud services life cycle through semantic technologies," *IEEE Trans. Serv. Comput.*, vol. 7, no. 1, pp. 109–122, Jan.-Mar. 2014.

[6] J. Zhang, J. Wang, P. Hung, Z. Li, N. Zhang, and K. He, "Leveraging incrementally enriched domain knowledge to enhance service categorization," *Int. J. Web Serv. Res.*, vol. 9, pp. 43–66, 2012.

[7] J. Zhang, R. Madduri, W. Tan, K. Deichl, J. Alexander, and I. Foster, "Toward semantics empowered biomedical web services," in *Proc. 2011 IEEE Int. Conf. Web Serv.*, 2011, pp. 371–378.

[8] Q. Y. Hong, Z. Xia, S. Reiff-Marganiec, and J. Domingue, "Linked context: A linked data approach to personalised service provisioning," in *Proc. 2012 IEEE 19th Int. Conf. Web Serv.*, 2012, pp. 376–383.

Authorized licensed use limited to: DA IICT. Downloaded on August 06,2024 at 17:01:09 UTC from IEEE Xplore. Restrictions apply.

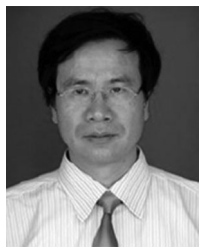
[9] J. Kopecky, T. Vitvar, C. Bournez, and J. Farrell, "SAWSDL: Semantic annotations for WSDL and XML Schema," *IEEE Int. Comput.*, vol. 11, no. 6, pp. 60–67, Nov./Dec. 2007.
[10] S. S. Alonso, O. S. Martinez, and L. J. Aguilar, "RAWS: Reflective engineering for web services," in *Proc. IEEE Int. Conf. Web Serv.*, 2004, pp. 488–495.
[11] A. Segev and Q. Z. Sheng, "Bootstrapping ontologies for web services," *IEEE Trans. Serv. Comput.*, vol. 5, no. 1, pp. 33–44, Jan.-Mar. 2012.
[12] T. G. Stavropoulos, D. Vrakas, and I. Vlahavas, "Iridescent: A tool for rapid semantic annotation of web service descriptions," in *Proc. 3rd Int. Conf. Web Intell. Mining Semantics*, 2013, Art. no. 12.
[13] A. A. Patil, S. A. Oundhakar, A. P. Sheth, and K. Verma, "Meteor-s web service annotation framework," in *Proc. 13th Int. Conf. World Wide Web*, 2004, pp. 553–562.
[14] A. Ranabahu, P. Parikh, M. Panahiazar, A. Sheth, and F. Logan-Klumpler, "Kino: A generic document management system for biologists using SA-REST and faceted search," in *Proc. 5th IEEE Int. Conf. Semantic Comput.*, 2011, pp. 205–208.
[15] Y. Liao, M. Lezoche, H. Panetto, N. Boudjlida, and E. R. Loures, "Semantic annotation for knowledge explicitation in a product lifecycle management context: A survey," *Comput. Ind.*, pp. 24–34, 2015.
[16] K. Belhajjame, S. M. Embury, N. W. Paton, R. Stevens, and C. A. Goble, "Automatic annotation of web services based on workflow definitions," *ACM Trans. Web*, vol. 2, no. 2, 2008, Art. no. 11.
[17] L. Derczynski, D. Maynard, N. Aswani, and K. Bontcheva, "Microblog-genre noise and impact on semantic annotation accuracy," in *Proc. 24th ACM Conf. Hypertext Social Media*, 2013, pp. 21–30.
[18] K. Belhajjame, S. M. Embury, and N. W. Paton, "Verification of Semantic Web service annotations using ontology-based partitioning," *IEEE Trans. Serv. Comput.*, vol. 7, pp. 515–528, 2014.
[19] J. Chen, Z. Feng, S. Chen, K. Huang, J. Zhang, and W. Tan, "A novel lifecycle framework for Semantic Web service annotation assessment and optimization," in *Proc. 22nd Int. Conf. Web Serv.*, 2015, pp. 361–368.
[20] Z. Xiang, J. Zheng, Y. Lin, and Y. He, "Ontorat: Automatic generation of new ontology terms, annotations, and axioms based on ontology design patterns," *J. Biomed. Semantics*, vol. 1, no. 6, 2015, Art. no. 4.
[21] L. Duan, S. Oyama, M. Kurihara, and H. Sato, "Crowdsourced semantic matching of multi-label annotations," in *Proc. 24th Int. Joint Conf. Artif. Intell.*, 2015, pp. 3483–3489.
[22] J. Kopecký and E. P. B. Simperl, "Semantic Web service offer discovery for e-commerce," in *Proc. 10th Int. Conf. Electron. Commerce*, 2008, pp. 29:1–29:6.
[23] Z. Zhang, S. Chen, and Z. Feng, "Semantic annotation for web services based on DBpedia," in *Proc. IEEE 7th Int. Symp. Serv. Oriented Syst. Eng.*, 2013, pp. 280–285.
[24] D. Roman, J. Kopecký, T. Vitvar, J. Domingue, and D. Fensel, "WSMO-Lite and hRESTS: Lightweight semantic annotations for web services and RESTful APIs," *Web Semantics Sci. Serv. Agents World Wide Web*, vol. 31, pp. 39–58, 2015.
[25] D. Tosi and S. Morasca, "Supporting the semi-automatic semantic annotation of web services: A systematic literature review," *Inf. Softw. Technol.*, vol. 61, pp. 16–32, 2015.
[26] I. Salomie, V. R. Chifu, I. Giurgiu, and M. Cuibus, "SAWS: A tool for semantic annotation of web services," in *Proc. IEEE Int. Conf. Autom. Quality Testing Robot.*, 2008, pp. 387–392.
[27] Protege, 2015. [Online]. Available: http://protege.stanford.edu/products.php.
[28] B. Liu, K. Huang, J. Li, and M. Zhou, "An incremental and distributed inference method for large-scale ontologies based on MapReduce paradigm," *IEEE Trans. Cybern.*, vol. 45, pp. 53–64, Jan. 2015.
[29] F. Daniel, F. M. Facca, V. Saquicela, L. M. Vilches-Blázquez, and Ó. Corcho, "Semantic annotation of RESTful services using external resources," in *Current Trends in Web Engineering*, F. Daniel and F. M. Facca, Eds. Berlin, Germany: Springer, 2010, pp. 266–276.
[30] S. Mokarizadeh, P. Kungas, and M. Matskin, "Evaluation of a semi-automated semantic annotation approach for bootstrapping the analysis of large-scale web service networks," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell. Intell. Agent Technol.*, 2011, pp. 388–395.
[31] Y. Zhao and Q. Zhu, "Evaluation on crowdsourcing research: Current status and future direction," *Inf. Syst. Frontiers*, vol. 16, pp. 417–434, 2014.
[32] H. M. Alonso and L. Romeo, "Crowdsourcing as a preprocessing for complex semantic annotation tasks," in *Proc. 9th Int. Conf. Language Resources Eval.*, 2014, pp. 229–234.
[33] G. Demartini, D. E. Difallah, and P. Cudré-Mauroux, "ZenCrowd: Leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking," in *Proc. 21st Int. Conf. World Wide Web*, 2012, pp. 469–478.
[34] C. Sarasua, E. Simperl, and N. F. Noy, "Crowdmap: Crowdsourcing ontology alignment with microtasks," in *Proc. Int. Semantic Web Conf.*, 2012, pp. 525–541.
[35] J. Wang, T. Kraska, M. J. Franklin, and J. Feng, "CrowdER: Crowdsourcing entity resolution," in *Proc. VLDB Endowment*, vol. 5, pp. 1483–1494, 2012.
[36] A. Dumitrache, "Crowdsourcing disagreement for collecting semantic annotation," in *Semantic Web. Latest Advances and New Domains*. Switzerland: Springer, 2015, pp. 701–710.
[37] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "DBpedia: A nucleus for a web of open data," in *Semantic Web*. Berling, Germany: Springer, 2007.
[38] C. Matuszek, J. Cabral, M. J. Witbrock, and J. DeOliveira, "An introduction to the syntax and content of Cyc," in *Proc. AAAI Spring Symp. Formalizing Compiling Background Knowl. Its Appl. Knowl. Representation Question Answering*, 2006, pp. 44–49.
[39] J. Bhagatet al., "BioCatalogue: A universal catalogue of web services for the life sciences," *Nucleic Acids Res.*, vol. 38, no. Suppl. 2, pp. W689–W694, 2010.
[40] C. Kankanamge, *Web Services Testing with SoapUI*. Packt Publishing Ltd, Birmingham, UK, 2012.
[41] D. Repchevsky and J. L. Gelpi, "BioSWR-Semantic Web services registry for bioinformatics," *PLoS One*, vol. 9, Jan 20, 2014, Art. no. e107889.
[42] S. N. Han, G. M. Lee and N. Crespi, "Semantic context-aware service composition for building automation system," *IEEE Trans. Ind. Informat.*, vol. 10, no. 1, pp. 752–761, Feb. 2014.
[43] D. Hull, et al., "Taverna: A tool for building and running workflows of services," *Nucleic Acids Res.*, vol. 34, pp. W729–W732, 2006.
[44] A. Dhamanaskar, M. E. Cotterell, J. Zheng, J. C. Kissinger, C. J. Stoeckert Jr, and J. A. Miller, "Suggestions for galaxy workflow design using semantically annotated services," in *Proc. 7th Int. Conf. Formal Ontology Inf. Syst.*, 2012, pp. 29–42.
[45] S. Mokarizadeh, P. Küngas and M. Matskin, "Ontology learning for cost-effective large-scale semantic annotation of web service interfaces," in *Knowledge Engineering and Management by the Masses*. New York, NY, USA: Springer, 2010, pp. 401–410.
[46] X. Hu, Z. Feng, K. Huang, and S. Chen, "Classification based parameter association for non-redundant annotation," in *Proc. IEEE Int. Conf. Serv. Comput.*, 2015, pp. 688–695.
[47] C. Fellbaum and G. Miller, *WordNet: An Electronic Lexical Database*: Cambridge, MA, USA: MIT Press, 1998.
[48] S. Harispe, D. Sánchez, S. Ranwez, S. Janaqi, and J. Montmain, "A framework for unifying ontology-based semantic similarity measures: A study in the biomedical domain," *J. Biomed. Inform.*, vol. 48, pp. 38–53, 2014.
[49] A. S. Ribalta, D. Sánchez, M. Batet, and F. Serratosa, "Towards the estimation of feature-based semantic similarity using multiple ontologies," *Knowl.-Based Syst.*, vol. 55, pp. 101–113, 2014.
[50] Z. Zheng, Y. Zhang and M.R. Lyu, "Investigating QoS of real-world web services," *IEEE Trans. Serv. Comput.*, vol. 7, no. 1, pp. 32–39, Jan.-Mar. 2014.
[51] A. Kashlev, S. Lu, and A. Chebotko, "Typetheoretic approach to the shimming problem in scientific workflows," *IEEE Trans. Serv. Comput.*, vol. 8, pp. 795–809, 2015.
[52] C. Luo, Z. Zheng, X. Wu, F. Yang, and Y. Zhao, "Automated structural semantic annotation for RESTful services," *Int. J. Web Grid Serv.*, vol. 12, pp. 26–41, 2016.

Authorized licensed use limited to: DA IICT. Downloaded on August 06,2024 at 17:01:09 UTC from IEEE Xplore. Restrictions apply.

Keman Huang received the dual BS degrees from the Department of Automation and School of Economics and Management from Tsinghua University, China, in 2009, and the PhD degree from the Department of Automation, Tsinghua University, China, in 2014. He is currently an assistant professor in the School of Computer Science and Technology, Tianjin University and a visiting scholar in the Sloan School of Management, MIT, Cambridge, Massachusetts. His research interests include service ecosystem, service recommendation, mobile service, and Semantic Web. He has published more than 30 journals and conference proceedings papers. He received the Best Paper Runner-up Award from IEEE SCC 2016, Best Student Paper Award from IEEE ICWS 2014 and ICSS 2013. He is currently an associate editor of the *International Journal of Services Computing*. He was in the program committees of many conferences and the publicly chair of IEEE ICWS/SCC/MS/ BIGDATA Congress 2016. He is a member of the ACM and the IEEE.

Jia Zhang received the MS and BS degrees in computer science from Nanjing University, China, and the PhD degree in computer science from the University of Illinois at Chicago. She is currently an associate professor in the Department of Electrical and Computer Engineering, Carnegie Mellon University. Her recent research interests center on service oriented computing, with a focus on scientific workflows, net-centric collaboration, and big data. She has co-authored one textbook titled "*Services Computing*" and has published more than 140 refereed journal papers, book chapters, and conference papers. She is currently an associate editor of the *IEEE Transactions on Services Computing* and the *International Journal of Web Services Research*, and editor-in-chief of the *International Journal of Services Computing*. She is a senior member of the IEEE.

Wei Tan received the BS and PhD degrees from the Department of Automation, Tsinghua University. He is currently a research staff member with the IBM T. J. Watson Research Center, New York. From 2008 to 2010, he was a researcher in the Computation Institute, University of Chicago and Argonne National Laboratory. At that time, he was the technical lead of the caBIG workflow system. His research interests include GPU accelerated machine learning, NoSQL, big data, cloud computing, service-oriented architecture, business and scientific workflows, and petri nets. He has published more than 70 journal and conference papers, and a monograph "*Business and Scientific Workflows: A Web Service-Oriented Approach*" (272 pages, Wiley-IEEE Press). He received the IEEE Peter Chen Big Data Young Researcher Award (2016), Best Paper Award from ACM/IEEE CCGrid (2015), Best Student Paper Award from IEEE ICWS (2014), Best Paper Award from IEEE SCC (2011), the Pacesetter Award from the Argonne National Laboratory (2010), and caBIG Teamwork Award from the National Institute of Health (2008). He is a member of the ACM and a senior member of the IEEE.



Zhiyong Feng received the PhD degree from Tianjin University. He is currently a full professor in the School of Computer Science and Technology, Tianjin University, China. He is the author of one book, more than 130 articles, and 39 patents. His research interests include knowledge engineering, services computing, and security software engineering. He is a member of the IEEE and the ACM.

Shizhan Chen received the BS, MS, and PhD degrees in computer science and technology from Tianjin University, China, in 1998, 2004, and 2010, respectively. He is currently an associate professor in the School of Computer Science and Technology, Tianjin University. His research interests include services computing and mobile Internet. He is a reviewer of some international conferences and journals. He is currently leading a research project supported by National Natural Science Foundation of China. He is a member of the ACM and the IEEE.

Authorized licensed use limited to: DA IICT. Downloaded on August 06,2024 at 17:01:09 UTC from IEEE Xplore. Restrictions apply.