A Novel Lifecycle Framework for Semantic Web Service Annotation Assessment and Optimization

Juan Chen, Zhiyong Feng, Shizhan Chen, Keman Huang*
Tianjin Key Laboratory of Cognitive Computing and Application
School of Computer Science and Technology,
Tianjin University,
Tianjin, China
{juanchen, zyfeng, shizhan, keman.huang}@tju.edu.cn

Wei Tan
IBM Thomas J. Watson Research Center
Yorktown Heights, NY 10598, USA
wtan@us.ibm.com

Jia Zhang
Carnegie Mellon University
Silicon Valley
jia.zhang@sv.cmu.edu

***Abstract*—Semantic annotation plays an important role for semantic-aware web service discovery, recommendation and composition. In recent years, many approaches and tools have emerged to assist in semantic annotation creation and analysis. However, the *Quality of Semantic Annotation* (QoSA) is largely overlooked despite of its significant impact on the effectiveness of semantic-aware solutions. Moreover, improving the QoSA is time-consuming and requires significant domain knowledge. Therefore, how to verify and improve the QoSA has become a critical issue for semantic web services. In order to facilitate this process, this paper presents a novel lifecycle framework aiming at QoSA assessment and optimization. The QoSA is formally defined as the success rate of web service invocations, associated with a verification framework. Based on a local instance repository constructed from the execution information of the invocations, a two-layer optimization method including a local-feedback strategy and a global-feedback one is proposed to improve the QoSA. Experiments on real-world web services show that our framework can gain 65.95%~148.16% improvement in QoSA, compared with the original annotation without optimization.**

Keywords- Quality of Semantic Annotation; Semantic Annotation Lifecycle; Web Service Invocation; Local Instance Repository; Quality Assessment; Annotation Optimization

I. INTRODUCTION

Semantic web technology [1] has been proven effective to service integration and interaction [2, 3]. Many semantic-aware approaches have been proposed to facilitate service discovery, recommendation and composition [4-7]. Generally speaking, the solutions exploit semantic web knowledge base such as LOD (Linked Open Data) [8], SAWSDL (Semantic Annotations for WSDL) [9] or domain ontology bootstrapping from the web service description [10] to annotate service elements with concepts. Afterwards, the semantic information is utilized to optimize the solution's performance. Apparently, the semantic annotation plays a foundational role in these solutions. Therefore, how to assist users in the annotation process is one of the most important issues for semantic web services.

Recently, several tools, such as Iridescent [11], Meteor-S [12], and Kino [13], have been developed by the semantic web service community to assist curators annotating the services. These tools can effectively facilitate the annotation task. However, the following two issues have been overlooked:

1) *Annotation Quality Evaluation*. The more accurate the semantic annotation can reflect the web services' semantic meaning, the more valuable it is for the semantic-aware solution. Most of the solutions assume that all annotations are accurate, while it is usually not the case [14, 15]. This is because web services are often annotated by third parties and it is difficult to guarantee the quality of annotations. Hence, how to help the annotator evaluate the annotation quality should be a fundamental functionality for the annotation tool.

2) *Annotation Quality Optimization.* The annotation of services, especially improving the services' annotation, is an extremely time-consuming and non-trivial task. It typically requires application domain knowledge and expertise. It is also difficult to identify inaccurate annotation for the web services. Thus, how to facilitate the annotation improvement is also a critical issue for semantic annotation.

There exists only few preliminary approaches for annotation verification [14, 16], and no tools are available for annotation optimization. In this paper, a four-phase semantic annotation lifecycle framework is presented to bridge the gap, including the semantic annotation, invocation-oriented quality evaluation, feedback-aware optimization and annotation usage. Our research is based on a basic hypothesis, "*the better the semantic annotation can support the web service invocation, the higher quality it owns.*" We formally define the *Quality of Semantic Annotation (QoSA)* as the success rate of invocations. The evaluation framework aims to verify the QoSA. Furthermore, the execution information of the invocations during the assessment is used to semi-automatically construct the local instance repository. Finally, we have developed a two-layer optimization mechanism to facilitate the QoSA optimization, including the Local-Feedback Strategy (LFS) and the Global-Feedback Strategy (GFS). Specifically, the LFS is used to improve the annotated instances during the assessment while the GFS is employed to facilitate the annotated concepts improvement in the annotation phase.

The major contributions of this paper are summarized as follows:

978-1-4673-7272-5/15 $31.00 © 2015 IEEE
DOI 10.1109/ICWS.2015.55

• A four-phase service annotation lifecycle framework is presented to assess and optimize the QoSA.

• A two-layer feedback-aware mechanism is proposed to facilitate the QoSA optimization.

• Experiments show that our approach can gain a 65.95%~148.17% improvement in QoSA, compared with the original annotation without optimization.

The rest of this paper is organized as follows. Section II reviews the related work. Section III introduces the four-phase service annotation lifecycle framework. Section IV details the QoSA evaluation and Section V presents the two-layer feedback-aware optimization mechanism. Section VI reports the empirical results based on the implemented prototype. Section VII concludes the paper.

II. Related Work

Semantic annotation plays an important role for semantic web services. In this section, we review the relevant literatures for the annotation task.

A. Semantic Annotations Tools

Several tools have emerged to assist human curators in annotating web services. For example, Meteor-S [12] semi-automatically suggests to users the concepts from domain ontologies to facilitate their annotation task. SAWS [17] is a tool aiming to enhance the WSDL descriptions with semantic concepts provided by domain ontologies. Kino [13] 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 [11] enables both expert and non-expert users to create semantic service annotations by matching elements and concepts and suggesting annotations. With the large volume and rapid growth of available semantic web knowledge base [18], some tools exploit ontologies to automate service annotation creation [8, 19, 20]. Hong et al. proposed the linked context model which applies the linked data to model and obtain context data from both users and services [8]. Zhang et al. employed DBpedia knowledge base to automate the semantic annotation process [20].

The service annotation is valuable only when it can accurately reflect the web services' semantic meaning. Existing tools all suppose that all the annotations are correct for further usage, while it may not always the case [14]. Therefore, how to evaluate and guarantee the quality of the semantic annotation has become an important issue for semantic web services.

B. Quality of Semantic Annotation

The verification of QoSA begins to attract attentions from the academic and industry. Mokarizadeh et al. [16] 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. [14] adapted techniques from conventional 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 [15]. 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. Therefore, the QoSA can be evaluated before the annotation is publicly available.

These proposals described a first step towards providing tools for QoSA verification. However, they strongly depend on the accuracy of the golden ontologies and the golden ontologies need to be previously constructed before the evaluation. Additionally, none of them considered how to improve the QoSA based on the verification. From a different perspective, this paper introduces a novel four-phase annotation lifecycle framework to evaluate and improve the QoSA.

III. Service Annotation Lifecycle Framework

As shown in Figure 1, this paper extends the annotation lifecycle model presented in [14] and proposes a four-phase service annotation lifecycle, consisting of the semantic annotation, the annotation quality evaluation, the annotation optimization and the semantic-aware usage.

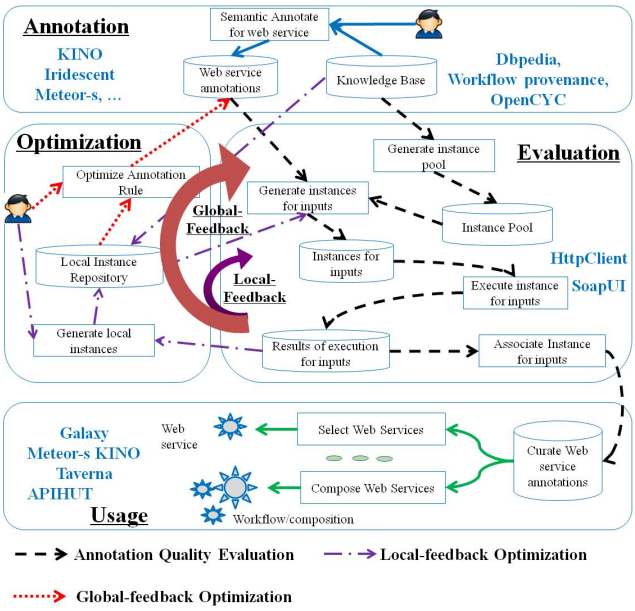

Figure 1. Four-phase Semantic Annotation Lifecycle Framework.

A. Semanic Annotation

In the semantic annotation phase, a user can annotate the web services manually or semi-automatically using tools such as KINO [13], Iridescent [11], Meteor-S [12] etc. Typically, such tools can provide suggestions for the annotation based on the *semantic web knowledge base* such as DBpedia [21], OpenCYC [22], workflow provenance [15], or Biocatalogue [23] etc. Based on a *semantic annotation*,

service elements such as operation names, input/output parameters, functional description etc. will be allocated with a semantic concept.

B. Annotation Quality Evaluation

In order to evaluate the quality of semantic annotation, instances for the input parameters are generated from the *instance pool* based on the *semantic web knowledge base*, or from the *local instance repository (LIR)* constructed during the QoSA assessment. Then the instances are considered as the test cases for input parameters and the web service operations will be invoked through tools such as SoapUI [24] or HttpClient [1]. Therefore, we can get the execution status for each invocation. Finally, each semantic concept for web service operations will be associated with an instance which can be used to support the semantic-aware solution.

C. Annotation Quality Optimization

Based on the execution status from the evaluation phase, the LIR will be constructed. Note that the LIR is empty at the beginning but it will become more comprehensive as time goes by. Then the LIR 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 influence the annotation phase. Therefore, we name it *Local-Feedback Strategy (LFS)*. Furthermore, the LIR can be used to guide the annotation during the semantic annotation phase. Therefore, we name it *Global-Feedback Strategy (GFS)***.** After the optimization phase, the QoSA for the web services will be improved, making it more valuable for the further solution.

D. Semantic-aware Usage

Based on the evaluation and optimization phase, each semantic annotation for a web service will not only contain the concept but also the suggested instance, which can enrich the semantic information for the further semantic-aware usage such as service discovery, composition and recommendation [5-7, 25, 26]. Details about these technologies are out of scope of this paper.

In our proposed four-phase semantic annotation lifecycle framework, the evaluation and optimization phase are critical and we will discuss their details in the following sections.

IV. QUALITYOF SEMANTIC ANNOTATION EVALUATION

A. Quality of Semanic Annotation (QoSA)

During the semantic annotation phase, the service elements will be allocated with a semantic concept from the semantic web knowledge base. As we intend to evaluate the QoSA from the invocation perspective, we only consider the semantic annotation for the input parameters. The rationale is that the annotation for the output parameters will not affect the invocation, the evaluation of the output parameter annotation is similar to the input annotation, and their verification should only be processed with the correct input parameter annotation [14]. We will leave the evaluate of the output parameter annotation quality for the future work. In this paper, each semantic annotation is modeled as the following tuple:

$$sa_i =< p_i, c_{pi}, \{< ip_{i,j}, c_{i,j}, cin_{i,j} > | \, 0 \le j \le n_i\} > \quad (1)$$

where p_i refers to an operation for the web service, c_{pi} is the concept annotated with the operation, n_i is the number of input parameters for p_i, $c_{i,j}$ is the semantic concept and $cin_{i,j}$ is the semantic instance annotated with the input parameter $ip_{i,j}$. $cin_{i,j}, 0 \le j \le n_i$ will be null before entering the evaluation phase.

In order to verify the semantic annotation for an input parameter $ip_{i,j}$, a collection of instances $in_{i,j} =< in_1, \ldots in_x >$ will be generated based on the semantic concept $c_{i,j}$. Here x is the number of instances generated for evaluation. Details about how to generate the instance will be discussed in the next subsection IV.B. If there exist at least one instance which is acceptable for the operation, the annotation $c_{i,j}$ for $ip_{i,j}$ is considered as acceptable.

Definition 1 *(Annotation Correct for Parameters, ACP): Given the operation p_i and one of its input parameter $ip_{i,j}$, the annotation $c_{i,j}$ for $ip_{i,j}$ is correct iff there exists an instance $in \in in_{i,j} =< in_1, \ldots in_x >$ that is accepted:*

$$c_{i,j} \overset{correct}{\rightarrow} < p_i, ip_{i,j} > \overset{iff}{\Leftarrow} \exists \, in \in in_{i,j} =< in_1, \ldots in_x >, in \overset{accepted}{\rightarrow} < p_i, ip_{i,j} > \quad (2)$$

When and only when the annotations for all the input parameters are correct, the operation will be successfully invoked. Therefore, we can derive the definition of the correct annotation for a given operation as follow:

Definition 2 *(Annotation Correct for Operations, ACO): Given the operation p_i and its semantic annotation sa_i, the annotation is correct iff all the annotations for its input parameters are correct:*

$$sa_i \overset{correct}{\rightarrow} p_i \overset{iff}{\Leftarrow} \forall < ip_{i,y}, c_{i,y} > \in \{< ip_{i,j}, c_{i,j} >\}, c_{i,y} \overset{correct}{\rightarrow} < p_i, ip_{i,y} > \quad (3)$$

Based on the discussions above, it is straightforward to formally define the QoSA for web services as follows:

Definition 3 *(Quality of Semantic Annotation for web Service, QoSA): Given a collection of operations $\{p_i, 0 < i \le N\}$ for web services and the semantic annotations sa_i for each operation p_i. The QoSA is defined as follows:*

$$QoSA = \frac{|ACO|}{N} = \frac{|\{sa_i \overset{correct}{\rightarrow} p_i\}|}{|\{sa_i\}|} \quad (4)$$

Obviously, $QoSA \in [0,1]$. The larger the QoSA, the better quality the semantic annotation owns. If QoSA is 1, all the

[1] http://hc.apache.org/httpcomponents-client-ga/httpclient/

annotations are correct. If QoSA is 0, no annotations are right.

B. Instance Generation for Input Parameter

From Figure 1, it can be seen that there exists two sources for instance generation: *the instance pools* (*IP*) from the semantic web knowledge base and *LIR*. As LIR is constructed based on the execution information, the instances in LIR are supposed to be more dedicated for the invocation. Therefore, just as shown in Algorithm 1 we generate the instances from the LIR with a higher priority.

Algorithm 1: Instance Generation for Input Parameter

Input: $< ip_{i,j}, c_{i,j} >$: annotated concept for each parameter
x : instances number
LIR : the local instance repository
IP : the instance pools from the knowledge base
Output: $in_{i,j} =< in_1, \ldots in_x >$: generated instances
Procedure:
01. $in_{i,j} \leftarrow \phi$
02. $query \leftarrow genSPARQL(c_{i,j}, x)$; // Generate SPARQL query with annotation $c_{i,j}$
03. **IF** $LIR \neq \phi$ **THEN**
04. $lin_{i,j} = excuteSPARQL(LIR, query, x)$; //execute the $query$ to generate x instances from LIR
05. $in_{i,j} \leftarrow in_{i,j} \bigcup lin_{i,j}$;
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. $pin_{i,j} = excuteSPARQL(IP, query, x)$; // execute the $query$ to generate x instances from IP
14. $in_{i,j} \leftarrow in_{i,j} \bigcup pin_{i,j}$
15. **ENDIF**

Line 02 generates the SPARQL query with the given annotation and the number of the required instances. Table I shows an example with $c_{i,j}$ = <*http://dbpedia.org/ontology /Currency*> and $x = 6$. Lines 03~11 execute the generated query in LIR to get the relevant instances. Lines 12~15 execute the query in the IP if the number of instances generated from LIR is not enough for the evaluation.

Note that at the beginning, the *LIR* will be null and all the instances are generated from the knowledge base. This scenario is discussed in [14]. As the evaluation going on, the *LIR* will be fleshed out and more instances will be generated from the LIR. Hence, the solution in [14] can be considered as a special case for our approach.

TABLE I. SPARQL QUERY GENERATED FOR GIVEN ANNOTATION AND INSTANCE NUMBER

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-
syntax-ns#>
SELECT DISTINCT *
WHERE {
?subject
rdf:type ?<http://dbpedia.org/ontology/Currency>
} LIMIT 6
```

C. QoSA Evaluation

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 tuple:

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

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.

Algorithm 2 details the process to evaluate the Quality of the semantic annotation.

Algorithm 2: QoSA Evaluation

Input: $SA = \{sa_i\}$: Annotations for Service Operations
x : instances number
LIR : the local instance repository
IP : the instance pools from the knowledge base
Output: $QoSA$: Quality of Semantic Annotation
$EIR = \{eir_z\}$: Execution information records
Procedure:
01. $EIR \leftarrow \phi$; $ACO \leftarrow \phi$;
02. **FOR** $sa_i \in SA$
03. **FOR** $< ip_{i,j}, c_{i,j} > \in sa_i$
04. $in_{i,j} \leftarrow ICIP(sa_i, x, LIR, IP)$; //use Algorithm 1 to generate instances for invocation
05. **ENDFOR**
06. **FOR** $\{< ip_{i,j}, in_{i,j,k_{i,j}} > | 0 \le j \le n, 0 \le k < x\}$
07. $eir_z \leftarrow invoke(p_i, \{< ip_{i,j}, in_{i,j,k_{i,j}} >\})$
08. $EIR \leftarrow EIR \bigcup eir_z$
09. **IF** $eir_z.st = true$
10. $ACO \leftarrow ACO \bigcup sa_i$;
11. $sa_i.cin_{i,j} \leftarrow in_{i,j,k_{i,j}}$; //update the instances
12. **BREAK**;
13. **ENDIF**
14. **ENDFOR**
15. **ENDFOR**
16. $QoSA \leftarrow \frac{| ACO |}{| SA |}$

Lines 03~05 generate the instances for each parameters; Lines 06~15 invoke the operation and get the execution information record. Line 16 calculates the QoSA. Obviously,

if the *LIR* is set as null, then the local instance repository will not be used for instance generation. This means that no optimization strategies will be employed and the result reflects the QoSA of the original semantic annotation.

V. TWO-LAYER ANNOTATION OPTIMIZATION

Supported by the local instance repository generated based on the execution results from the operation invocation, we propose a local-feedback strategy and a global-feedback strategy to assist annotators in optimizing the semantic annotation.

A. Local-feedback Strategy for QoSA optimization

For each invocation, the web service operation will reveal information about the execution. Some is meaningless while some contain critical content which can be used to improve the QoSA. Figure 2 illustrates the invocation result from the operation "*GetCurrencyList*" in web service "*Rates.*" It can be seen that "*AED*" in "*<Code>AED</Code>*" is an ISO currency code, and "*UAE Dirham*" in "*<Name>UAE Dirham</Name>*" is a currency name. Obviously, such kind of information can be used to generate the instances which are accurate for the invocation.

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
  <GetCurrencyListResponse
xmlns="http://mondor.org/">
    <GetCurrencyListResult>
      <Currency><Code>AED</Code><Name>UAE
Dirham</Name></Currency>
      <Currency><Code>ALL</Code><Name>Albania
n Lek</Name></Currency>
      ……
      <Currency><Code>ZAR</Code><Name>South
African Rand</Name></Currency>
      <Currency><Code>ZMW</Code><Name>Zambi
an Kwacha</Name></Currency>
    </GetCurrencyListResult>
  </GetCurrencyListResponse>
</soap:Body>
</soap:Envelope>
```

Figure 2. Execution Result from the Invocation of the operation "GetCurrencyList" in web service "Rates".

Similar to [10, 27], we extract the tokens from the execution result. As the execution result is in the XML format, each value of the XML schema leaf element will be considered as a *local instance (lin)*; each element name list from the finest granularity to the general levels with operation name will be considered as the *possible concept (pc)*. Given the concept annotated with the operation and its parameters, their sub-concepts and relations in the knowledge base such as Dbpedia will be used to construct the *concept candidate pool*. Finally, for each possible concept, we map it to the concept with the largest semantic similarity in the concept candidate pool. Therefore, the local instance record can be modeled as the following tuple:

$$lir = \{< p_i, pc_o, c_{lir}, \{lin_q \mid 0 < q \le M\} >\} \quad (6)$$

where p_i is the operation, pc_o is the possible concept, c_{lir} is the concept with the largest similarity to pc_o in the concept candidate pool, $\{lin_q \mid 0 < q \le M\}$ is the local instance list generated from the execution result and M is the local instance number.

Algorithm 3 details the process and Table 2 shows a snapshot of the local instance records we constructed based on the invocation result. Note that in order to guarantee the accuracy of the local instance repository, the process will be semi-automatic and the annotator will participate during the generation process.

Algorithm 3: Local Instance Record Generation

Input: $sa_i =< p_i, c_{pi}, \{< ip_{i,j}, c_{i,j}, cin_{i,j} >\mid 0 \le j \le n_i\} >$: annotation for the operation;
er : response from the invocation

Procedure:

01. $li = \phi$;
02. $ccp = \phi$; //the concept candidate pool
03. $\{< p_i, pc_o, \{lin_q \mid 0 < q \le M\} >\} \leftarrow tokenExtraction(er)$; // extract the possible concepts and local instance list from the execution response
04. $ccp \leftarrow candidateConceptGeneration(c_{pi}, \{c_{i,j}\})$; // get the sub-concepts to form concept candidate pool
05. **FOR** $pc_o \in \{< p_i, pc_o, \{lin_q \mid 0 < q \le M\} >\}$
06. $c_{lir} \leftarrow getMostSimilarConcept(pc_o, ccp)$;// get the most similar concept from the concept candidate pool
07. $LIR \leftarrow LIR \cup \{< p_i, pc_o, c_{lir}, \{lin_q \mid 0 < q \le M\} >\}$ // add the local instance records into the *LIR*.
08. **END FOR**

TABLE II. GENERATED LOCAL INSTANCE RECORDS (PART)

c_{lir}	$\{lin_k \mid 0 < k \le M\}$
DBpedia:Currency Code	EUR, AUD, BRL, CAD, CNY, CUP, EUR, EGP, etc.
DBpedia:Currency Name	Dollar, Taka, Franc, Pound, Yen, Krona, Baht, Lira, etc

Based on the local instance records, the local-feedback strategy for the annotation optimization is straightforward:

Definition 4 *(Local-feedback Strategy for QoSA optimization) For each successful invocation during the evaluation, get the execution response, and extract the local instance records to update the local instance repository.*

The local instance repository starts from no instance. The instances generated for the evaluation come from the knowledge base. However, as time goes by, the successful

invocations will enrich the local instances in *LIR*. Due to the fact that the local instance records in *LIR* are more accurate, some fault invocation because of the illegal input instances will become successful during the local-feedback optimization. Therefore, the QoSA will be improved until there exists no more error invocation because of incorrect input instances.

Algorithm 4: Local-feedback Strategy (LFS)

Input: $SA = \{sa_i\}$: Annotations for Service Operations
x : instances number
LIR : the local instance repository
IP : the instance pools from the knowledge base
Output: $QoSA$: Quality of Semantic Annotation
$EIR = \{eir_z\}$: Execution information records
Procedure:
01. **Repeat**
02. $< QoSA, EIR > \leftarrow Evaluation(SA, x, LIR, IP)$; //use Algorithm 2 to evaluate the QoSA
03. **FOR** $eir_z \in EIR$
04. $LIRG(eir_z.sa_i, eir_z.er)$; // use Algorithm 3 to update the local instance records in LIR
05. **ENDFOR**
06. **Until Convergence**

B. Global-feedback Strategy for QoSA optimization

In the local-feedback strategy, the instances generated for invocations will be updated for each round until the LFS reaches convergence. Some illegal invocations would become successful and we name them as the *rectification operations* for convenience. Obviously, the original semantic annotations for these operations are incorrect and these *reclaimed records (RR)* can be used to help the annotators correct their annotations. Here we formally define them as the following tuple:

$$rr =< sa_i, \{< ip_{i,j}, lin_{i,j,k_{i,j}} > | 0 \le j \le n\} > \quad (7)$$

where $\{< ip_{i,j}, lin_{i,j,k_{i,j}} > | 0 \le j \le n\}$ is the instance combination which succeeds the original fault invocation.

Apparently, each instance $lin_{i,j,k_{i,j}}$ is allocated with a concept c_{lir} , if the original annotated concept $c_{i,j}$ for the parameter $ip_{i,j}$ is different from c_{lir} . This means that the original annotation is inaccurate and we can replace it into c_{lir} to update the annotation. Therefore, the global-feedback strategy for the annotation optimization can be described as follows:

Definition 5 *(Global-feedback Strategy for QoSA optimization) For each reclaimed records generated during the LFS, identify the inaccurate annotation for the input parameter and update it into the new concept.*

Algorithm 5 details the global-feedback strategy. Line 03 gets the concept of the instance from the local instance repository. Lines 04~07 identify the inaccurate annotation and update it into the new one. Line 10 reevaluates the QoSA to prove the effectiveness of the GFS. Table III shows a snapshot of annotation correction during the GFS.

Algorithm 5: Global-feedback Strategy (GFS)

Input: $RR = \{rr_s\}$ // the reclaimed records from LFS
Output: $QoSA$: Quality of Semantic Annotation
Procedure:
01. FOR $rr_s \in RR$
02. FOR $< ip_{i,j}, lin_{i,j,k_{i,j}} > \in rr_s$
03. $c_{lir} \leftarrow getConcept(lin_{i,j,k_{i,j}}, LIR)$; //get the concept of $lin_{i,j,k_{i,j}}$ in *LIR*
04. IF $sa_i.c_{i,j} \neq c_{lir}$
05. $sa_i.c_{i,j} \leftarrow c_{lir}$;
06. $sa_i.cin_{i,j} \leftarrow lin_{i,j,k_{i,j}}$;
07. ENDIF
08. ENDFOR
09. ENDFOR
10. $< QoSA, EIR > \leftarrow Evaluation(SA, x, LIR, IP)$; // Evaluation the QoSA based on Algorithm 2.

TABLE III. ANNOTATION UPDATE DURING GFS (PART)

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

Note that the GFS will be used after the LFS reaches convergence. This is because the LFS will update the LIR and generate the reclaimed records which can enable the GFS. However, after the annotation updated by GFS, the LFS can be used to optimize the quality.

VI. PROTOTYPE AND EXPERIMENTS

A. Prototype Implementation

In order to prove the effectiveness of the presented framework, we have implemented a prototyping system based on our proposed lifecycle model. The annotation approach presented in [20] is used to generate the original semantic annotation as a baseline. Dbpedia is used as the semantic web knowledge base and the Httpclient is integrated to invoke the web service operation.

Similar to [20], we employed a dataset consisting of 300 real-world web services with WSDL documents. 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"*.. Finally, we received a dataset summarized in Table IV, consisting of 121 services, 941 operations and 15,888 parameters as the benchmark.

TABLE IV. BENCHMARK DATAST

#web service	*#operations*	*#parameters*
121	941	15,888

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 [13], Iridescent [11], Meteor-S [12] etc.; 2) the knowledge base can be switched by OpenCYC [22], workflow provenance [15], or Biocatalogue [23] etc.; 3) the dataset can be substituted by other WSDL-based service dataset [28].

B. Experiment Results and Discussions

1) Local-feedback for Annotation Optimization

In order to evaluate the effectiveness of our local-feedback strategy for annotation optimization, we set the instances for evaluation as 6 and then considered the following two methodologies:

- Original Semantic Annotation (OSA): all instances for each annotation are generated from Dbpedia and no feedback strategy is employed.
- Semantic Annotation with Local-feedback (LF-SA): instances are generated from both Dbpedia and the local instance repository. Additionally, our local-feedback strategy is used to optimize the semantic annotation.

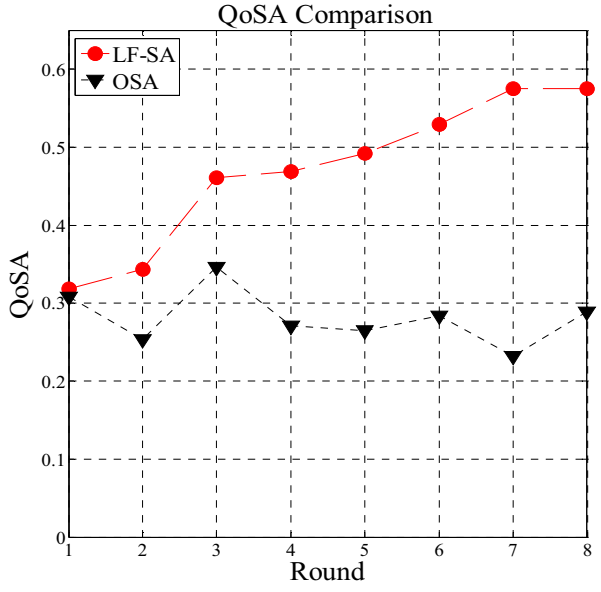

Figure 3. QoSA comparison between the original semenatic annotation and the semantic annotation with local-feedback strategy.

Figure 3 reports the QoSA over the two methodologies. It can be seen that LF-SA gains a higher QoSA than OSA. The QoSA in LF-SA is increasing as time goes on and after 7 rounds, the LF-SA reaches the convergence at 0.5749. However, the QoSA for OSA is oscillating around 0.3. The LF-SA brings a 65.95% ~ 148.16% improvement for the QoSA.

Furthermore, we conducted a depth analysis over the service annotations which were inaccurate at the beginning but were corrected by the local-feedback strategy. The result shows that the exception for the illegal invocation is "*Server was unable to process request. Object reference not set to an instance of an object.*" Such an error message means the instances generated from the knowledge base were not accurate.

Additionally, for the service operations that are still incorrect after the local-feedback optimization, we find 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.*" This means that there is an internal error of the services thus no approach can verify or improve the QoSA. After excluding such operations, the QoSA for LF-SA reached 0.6848.

2) Global-feedback for Annotation Optimization

Based on the reclaimed records generated from the LFS, we updated the original semantic annotations for web services and then reevaluated the QoSA for the *Updated Semantic Annotation* (USA). From Figure 4, it can be seen that the GFS can effectively identify the inaccurate annotations and assist annotators in improving the QoSA. Comparing with the *vintage QoSA* from the original semantic annotation, the GFS gains a 60.45% improvement in QoSA. If we further filter the operations with internal error, the QoSA for USA will reach 66.20%. Such an enhancement will make the semantic annotation more valuable for applications.

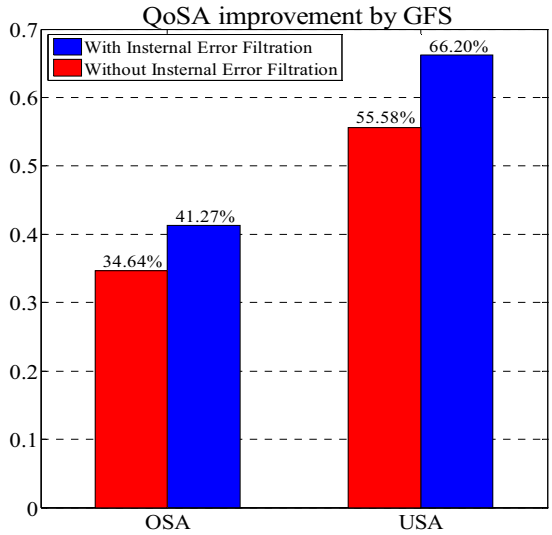

Figure 4. QoSA comparison between the original semenatic annotation and the semantic annotation after the global-feedback optimization.

VII. CONCLUSIONS

Semantic annotation plays a foundational role for semantic-aware solutions to improve web service discovery, recommendation and composition. Many tools have been developed to assist users in annotating web services. However, annotation quality evaluation and optimization have not gained sufficient attention.

This paper presents a four-phase annotation lifecycle framework to assist annotators in evaluating and improving the QoSA, including semantic annotation, annotation quality assessment, annotation optimization and semantic-aware usage. Specially, QoSA is formally defined as the success rate of invocations and the evolution framework acts as an instrument to verify the QoSA for the given annotations. Based on the local instance repository consisting of the local instances and reclaimed records generated from the invocation response information, a local-feedback strategy is presented to optimize the annotated instance and a global-feedback strategy is presented to correct the inaccurate annotation. Our empirical study based on real-world web services shows that comparing with the original annotation based on the methodology presented in [20], our framework gains a 65.95% ~ 148.16% QoSA improvement during evaluation and a 60.45% improvement for annotation.

In the future, we will further extend our framework into a benchmark platform which can evaluate the performance of different semantic annotation approaches such as Iridescent, Meteor-S, Kino and so on. Moreover, additional services will be imported in our framework to strengthen the results of the evaluation.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China grant 61070202, 61173155 and National High-Tech Research and Development Pro-gram of China grant 2013AA013204.

REFERENCES

[1] T. Berners-Lee, J. Hendler and O. Lassila, "The semantic web," Scientific American, vol. 284, pp. 28--37, 2001.

[2] B. Medjahed, A. Bouguettaya and A. K. Elmagarmid, "Composing web services on the semantic web," The VLDB journal, vol. 12, pp. 333-351, 2003.

[3] S. A. McIlraith, T. C. Son and H. Zeng, "Semantic web services," IEEE intelligent systems, vol. 16, pp. 46--53, 2001.

[4] A. V. Paliwal, B. Shafiq, J. Vaidya, H. Xiong, and N. Adam, "Semantics-based automated service discovery," IEEE Transactions on Services Computing, vol. 5, pp. 260-275, 2012.

[5] K. P. Joshi, Y. Yesha and T. Finin, "Automating cloud services life cycle through semantic technologies," IEEE Transactions on Services Computing, vol. 7, pp. 109-122, 2014.

[6] J. Zhang, J. Wang, P. Hung, Z. Li, N. Zhang, and K. He, "Leveraging incrementally enriched domain knowledge to Enhance service categorization," International Journal of Web Services Research (IJWSR), vol. 9, 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 IEEE International Conference on Web Services (ICWS): 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 2012 IEEE 19th International Conference on Web Services (ICWS), Honolulu, HI, 2012, pp. 376-383.

[9] J. Kopecky, T. Vitvar, C. Bournez, and J. Farrell, "SAWSDL: Semantic Annotations for WSDL and XML Schema," IEEE Internet Computing, vol. 11, pp. 60 - 67, 2007.

[10] A. Segev and Q. Z. Sheng, "Bootstrapping ontologies for web services," IEEE Transactions on Services Computing, vol. 5, pp. 33-44, 2012.

[11] T. G. Stavropoulos, D. Vrakas and I. Vlahavas, "Iridescent: A tool for rapid semantic annotation of web service descriptions," in Proceedings of the 3rd International Conference on Web Intelligence, Mining and Semantics, 2013, p. 12.

[12] A. A. Patil, S. A. Oundhakar, A. P. Sheth, and K. Verma, "Meteor-s web service annotation framework," in Proceedings of the 13th international conference on World Wide Web, 2004, pp. 553-562.

[13] 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 2011 Fifth IEEE International Conference on Semantic Computing (ICSC), 2011, pp. 205-208.

[14] K. Belhajjame, S. M. Embury and N. W. Paton, "Verification of Semantic Web Service Annotations Using Ontology-Based Partitioning,". IEEE Transactions on Services Computing, vol. 7, pp. 515-528, 2014.

[15] K. Belhajjame, S. M. Embury, N. W. Paton, R. Stevens, and C. A. Goble, "Automatic annotation of web services based on workflow definitions," ACM Transactions on the Web (TWEB), vol. 2, p. 11, 2008.

[16] 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 2011 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT), 2011, pp. 388-395.

[17] I. Salomie, V. R. Chifu, I. Giurgiu, and M. Cuibus, "SAWS: A tool for semantic annotation of web services," in IEEE International Conference on Automation, Quality and Testing, Robotics, 2008, pp. 387-392.

[18] B. Liu, K. Huang, J. Li, and M. Zhou, "An Incremental and Distributed Inference Method for Large-Scale Ontologies Based on MapReduce Paradigm," IEEE Transactions on Cybernetics, vol. 45, pp. 53-64, 2015.

[19] 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.: Springer Berlin Heidelberg, 2010, pp. 266-276.

[20] Z. Zhang, S. Chen and Z. Feng, "Semantic Annotation for Web Services Based on DBpedia," in 2013 IEEE 7th International Symposium on Service Oriented System Engineering (SOSE), 2013, pp. 280-285.

[21] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "Dbpedia: A nucleus for a web of open data,": Proceedings of the 6th international The semantic web and 2nd Asian conference on Asian semantic web conference, 2007, pp. 722-735.

[22] C. Matuszek, J. Cabral, M. J. Witbrock, and J. DeOliveira, "An Introduction to the Syntax and Content of Cyc.," in AAAI Spring Symposium: Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering, 2006, pp. 44-49.

[23] J. Bhagat, F. Tanoh, E. Nzuobontane, T. Laurent, J. Orlowski, M. Roos, K. Wolstencroft, S. Aleksejevs, R. Stevens, and S. Pettifer, "BioCatalogue: a universal catalogue of web services for the life sciences," Nucleic acids research, p. gkq394, 2010.

[24] C. Kankanamge, Web services testing with soapUI: Packt Publishing Ltd, 2012.

[25] D. Repchevsky and J. L. Gelpi, "BioSWR--semantic web services registry for bioinformatics," PLoS One, vol. 9, p. e107889, 2014.

[26] S. N. Han, G. M. Lee and N. Crespi, "Semantic context-aware service composition for building automation system," IEEE Transactions on Industrial Informatics, vol. 10, pp. 752-761, 2014.

[27] 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: Springer, 2010, pp. 401-410.

[28] Z. Zheng, Y. Zhang and M. R. Lyu, "Investigating QoS of real-world web services," IEEE Transactions on Services Computing, vol. 7, pp. 32-39, 2014.