CLASSIFICATION BASED PARAMETER ASSOCIATION FOR NON-REDUNDANT ANNOTATION

Xuehao Sun[1,2], Shizhan Chen[1,2], Zhiyong Feng[1,2,3], Keman Huang[1,2,*], Dongxiao He[1,2], Xiaocao Hu[1,2]
[1]Tianjin Key Laboratory of Cognitive Computing and Application, Tianjin, China
[2]School of Computer Science and Technology, Tianjin University, Tianjin, China
[3]School of Computer Software, Tianjin University, Tianjin, China
xuehaosun@sina.com,{shizhan, zyfeng, keman.huang, hedongxiao, huxiaocao}@tju.edu.cn

Abstract
Semantic annotation of Web Services can facilitate the automated service discovery and composition. At present, however, many solutions suffer from redundant annotations or imprecise derived annotations. To solve the problem, we need to find the parameters that have equivalent semantics in a large number of Web Services. This paper proposes a classification based parameter association, laying the foundation for non-redundant annotation and amendment. Then the paper presents the non-redundant annotation in reduced parameter space. And the annotation results are expanded to the original parameter space. Finally a prototype tool is made to show all Web Services' information and parameter association information, by which we can manually amend the annotation results. Moreover, the amendment results will be applied to all parameters that are semantically equivalent. To evaluate the final annotation results, the paper gives a method based on service discovery. The experimental results show that the parameter association approach can accurately identify the equivalence between parameters. It also suggest that non-redundant annotation and amendment can greatly reduce the workload, and can augment the semantics of Web Services with adequate accuracy, which can achieve similar performance in service discovery as OWL-S services do.
Keywords: Web Service; Parameter Association; Classification; Non-redundant Annotation; Amendment

1. INTRODUCTION

Nowadays, Web Services [1] is more and more popular in various fields, especially in specifying the semantic description of a service, which facilitates the automated service discovery and composition. Earlier research of specifying the semantics of services focus on manual annotation [2], which suffers from the following problems. Firstly, choosing relevant ontologies brings a tremendous burden to the user. Secondly, with the growth of the available web services, the potential number of concepts is increasing rapidly. Thirdly, the ontology used for annotation could be very large with correspondingly large number of concepts. As a consequence, manually specifying the semantics of services becomes a very tedious task. To deal with these difficulties, many studies have been carried out on the automatic semantic annotation of Web Service. The well-adopted approach is to associate Web Services, especially the inputs and outputs, with concepts from ontologies.

Most approaches on the automatic semantic annotation of Web Services can be classified into three main categories, annotation using schema matching techniques [3, 7-8, 10-11], annotation using machine learning algorithms [4-5, 9], and annotation using data-driven workflows [6]. For the first two categories, the process is performed whenever parameters of a new Web Service come, even when parameters of new service have same semantics with parameters of services that are already annotated, thus resulting in redundant annotations, which may be a large number of calculations. For the third category, annotation information on one side of a data link is derived by the other side of the data link, which is already annotated. In this case, although the redundancy is avoid, derived annotations are imprecise and exact annotations cannot be inferred.

Finding parameters that have same semantics in a large number of Web Service can efficiently reduce redundant annotations and guarantee exact annotations. We propose a classification based approach for associating parameters that are semantically equivalent in this paper. The approach lays the foundation for reducing the parameter space for annotation and amendment. Then the semantic annotation and amendment of Web Services is performed on the reduced parameter space instead of the original parameter space. To guarantee exact annotations, the annotation result of a parameter is directly applied to other parameters that have same semantics. A prototype tool is made to show us the annotation results of Web Services, by which we can manually amend the annotation that are not suit the semantic of the Web Service. For example, "apple" may not be a kind of fruit but a mobile phone. In this case, it is necessary to amend the annotation result of the parameter. In addition, this paper presents a methodology to evaluate the annotation results. The major contributions of our work are summarized as follows:

- An efficient approach for identifying parameters that have same semantically equivalent in different Web Services.

- An overall process of the non-redundant annotation and an approach for amendment of the annotation results.
- A methodology for evaluating the annotation results.
- A detailed experimental evaluation for the parameter association approach and the non-redundant annotation.
- A prototype tool to show the annotation results and the amendment results.

The rest of this paper is organized as follows. Section 2 looks back at the related work. Section 3 presents the details of the approach for parameter association. Section 4 introduces the overall process of the non-redundant annotation. This section also introduces an approach for amendment based parameter association. Section 5 describes a methodology for the evaluation of annotation results. Section 6 conducts experiments while Section 7 draws the conclusion.

2. RELATED WORK

Many solutions have been developed in the field of semantic annotation of Web Service. Patil [3] presents a framework, METEOR-S Web Service Annotation Framework (MWSAF), to semantically annotate WSDL descriptions of services with relevant ontologies. Firstly, WSDL files and ontologies are represented by the same format through two translators. Secondly, the best mapping between elements in the WSDL files and concepts of an ontology is selected.

Hess [4] proposes ASSAM, a tool that assists a user in creating semantic metadata for Web Services. Firstly, developing an iterative relational classification algorithm for semantically classifying Web Services, their operations, and input and output messages. Secondly, to aggregate the data returned by multiple semantically related Web Services, a schema mapping algorithm that is based on an ensemble of string distance matrix.

Lerman [5] presents an approach that automatically recognizes semantic types of the data used by Web Services. To recognize input data types using only the terms extracted from a WSDL file, a metadata-based classification method is described. And then the classifier's predictions are verified by invoking the service with some sample data of that type.

Belhajjame [6] explores the potential uses of trusted data-driven workflows. If a workflow is known to generate sensible results, it must be the case that the operation parameters that are connected within the workflow are compatible with one another. Therefore, if one side of a data link is annotated, annotation information for the parameter on the other side of the link is derived.

Salomie [7] presents SAWS, as a new tool for semantic annotation of Web services with data and protocol semantics aiming the enhancement of the WSDL descriptions with semantic concepts provided by domain ontologies. The tool combines string matching algorithms and the Levenshtein distance concept to attain a high degree of correctness in ranking domain ontology concepts relative to an abstract and purely syntactic element.

Kungas [8] proposes a practical method for semantically annotating collections of XML Schemas and Web service interfaces. Hierarchical structures of WSDL interfaces and XML schemas are exploited to extract contextual information, and XML schema elements that have either a built-in XSD type or a "simpleType" are annotated on the basis of contextual information.

Chifu [9] describes an unsupervised model for classifying Web services datatypes into a large number of classes specified by an ontology. The classification starts from a representation of datatypes built as a result of analyzing WSDL files of services. The framework is based on an unsupervised training of an extended model of hierarchical self-organizing maps.

Bouchiha [10] proposes an approach for semi-automatically annotating WSDL Web services descriptions. The annotation approach consists of two main processes: categorization and matching. Categorization process consists in classifying WSDL service description to its corresponding domain and matching process consists in mapping WSDL entities to pre-existing domain ontology. Both categorization and matching rely on ontology matching techniques.

Hu [11] proposes an approach for augmenting the semantics of Web Services based on public open ontology. The utilization of public open ontology addresses the issue that domain ontologies used in previous works are mostly created manually and are not comprehensive enough to contain all the concepts in the domain.

Though there are different approaches utilized in [3-11], major drawbacks of these solutions are redundant annotations and imprecise derived annotations. For works in [3-5, 7-11], once a new Web Service comes, the annotation process is performed on the new service, even though the semantics of the new service can refer to the one of other annotated services. And lacking of strategies that efficiently access large number of ontology concepts result in that a new service is compared against all the ontology concepts. As a consequence, redundant annotations may cause a large number of calculations. For work in [6], when one side of a data link is annotated, the information on this side is utilized to derive annotation information for the parameter on the other side of the link. Though redundant annotations are avoided, however, since data links in the workflows do not necessarily contain every possible connection of compatible parameters, derived annotations tend to be a looser form of annotation that specifies constraints on the semantics of parameters.

3. PARAMETER ASSOCIATION

Generally speaking, the first step of the parameter association approach is to parse documentations of Web

Services, and construct the hierarchy of parameters. The next step is to measure the similarities of both semantic and adjacent information of parameters in Web Services. And the last step is to employ classifiers in machine learning to determine what pairs of parameters in the two given parameter hierarchies are semantically equivalent. The architecture of the approach is illustrated in *Figure 1*. Details of the three major modules, i.e. the parameter hierarchy construction, similarity generator, and equivalence classification, are discussed in the following sections.

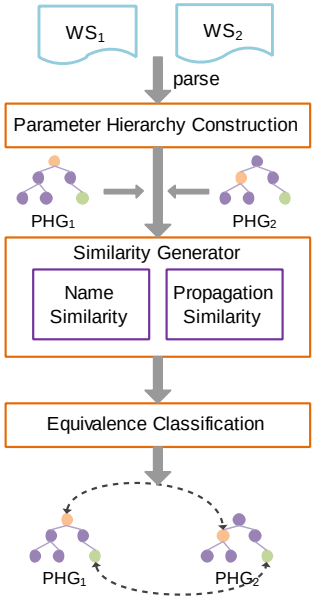


Figure 1. Architecture of the Parameter Association.

3.1 Parameter Hierarchy

It is inconvenient to match two parameter models directly because of the diversity in expressiveness of parameters in Web Services. For example, an alternative syntax may be used to specify the composite structure of parameters by using the type system directly. And multiple part elements may be defined to represent several parameters. Therefore, an efficient solution is to convert different expressiveness of parameters to a common representation format.

The construction of parameter hierarchy is guided by several rules specified in *Table 1*, which are inspired by the WSDL2Schema conversion rules in [3].

```
<definitions ...>
  <types>
    <schema ...>
      <element name="PO" type="tns:POType" />
      <complexType name="POType">
        <all>
          <element name="id" type="string" />
          <element name="name" type="string" />
          <element name="items">
            <complexType>
              <all>
                <element name="item" type="tns:Item" minOccurs="0" maxOccurs="unbounded" />
              </all>
            </complexType>
          </element>
        </all>
      </complexType>

      <complexType name="Item">
        <all>
          <element name="quantity" type="int" />
          <element name="product" type="string" />
        </all>
      </complexType>

      <element name="Invoice" type="tns:InvoiceType" />
      <complexType name="InvoiceType">
        <all>
          <element name="id" type="string" />
        </all>
      </complexType>
    </schema>
  </types>

  <message name="PO">
    <part name="po" element="tns:PO" />
    <part name="invoice" element="tns:Invoice" />
  </message>
</definitions>
```

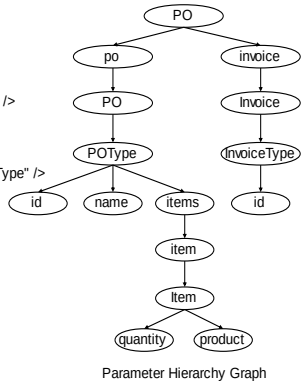


Figure 2. Example of Parameter Hierarchy

For example, a parameter hierarchy Construction is illustrated in *Figure 2*. In the WSDL files, there is one message named PO, which is the root of the parameter hierarchy Construction. Part po and invoice are the children

Table I. Parameter hierarchy construction rules

WSDL Element	Parameter Hierarchy
Message	Node with the message name, root of the parameter hierarchy
Part with elementary data type	Node with the part name, edge pointing from the root node to this node
Part with complex type	Node with the part name, edge pointing from the root node to this node, node with the element name, and edge pointing from the former node to the latter node
ComplexType	Node
Element, defined under complex type, with elementary data type	Node with the element name, and edge pointing from the complex type node to the former node, and edge pointing from the former node to the latter node
SimpleType	Node
Element	Node with the element name

of message PO. Element PO is the child of po. ComplexType POType is the child of element PO, which have the same name as message PO but not the one. Element id, name and items are the children of POType. Element item is a complex type, which is also the child of items. ComplexType item is the child of element item, which has the children named quantity and product. Element invoice is constructed as element po do. As we can see, the parameter hierarchy is clear at a glance to us.

3.2 Similarity Generator

Two hierarchy of the parameters are the inputs of similarity generator. The outputs are two similar matrixes. The one is name similarity matrix NSM, and the other is propagation similarity matrix PSM, as written in (1-2).

$$NSM = [ns_{ij}]_{m\times n} \quad (1)$$

$$PSM = [ps_{ij}]_{m\times n} \quad (2)$$

where ns_{ij} and ps_{ij} are respectively the name similarity and the propagation similarity between parameter p_i in the former hierarchy graph and parameter p_j in the latter hierarchy graph.

3.2.1 Name similarity

To measure the semantic similarity between two parameters based on their names, we defined the name similarity.

Parameter names are always defined by typical, compound words. Therefore, it is necessary to preprocess each parameter by several steps, which we summarize to five steps:

(1)Process the parameter name in the form of "get/set...By...". The parameter named getXXByYY means that "get YY through XX". So it come to two cases. One is that the parameter is as the input, thus we get the YY. Another case is that parameter is as the output, so we get the XX.

(2)Process the parameter name in the form of "get/set...Result(s)..." or "get/set...Return(s)...". In this way, we get the character between "set/get" and "Result(s)/Return(s)".

(3)Tokenize based on special characters and capitalization. Stop words are removed from the set of tokens.

(4) Reduce the tokens to their stem by using a stemmer .

(5) Replace abbreviations by words in full.

As a consequence, each parameter is described by a collection of terms.

The name similarity between two parameters is calculated based on their term collections. Calculate the semantic similarity of any two terms from the collections by using WordNet [13]. And then there is a semantic similarity matrix produced, which displays semantic similarities for each pair of terms from these two collections. The task is turned into finding an optimal assignment in the semantic similarity matrix without assigning a term more than once instead of calculating the name similarity. Therefore, the task is reduced to an assignment problem.

Assignment problem is an important subject discussed in real physical world. There are various methods presented for assignment problem. And different articles have been published on the subject [14-15]. A considerable number of methods have been so far presented for assignment, in which the Hungarian method [16] is a well-adopted methodology among them. The iterative method is based on adding or subtracting a constant to every element of a row or column of the matrix, and creating some zeros in the given matrix and then trying to find a complete assignment in terms of zeros. By a complete assignment for a matrix $n\times n$, an assignment plan is defined as containing exactly n assigned independent zeros, one in each row and one in each column. Finding the optimum allocation of a number of resources to an equal number of demand points is the main concept of assignment problem. An assignment plan is optimal if optimizes the total cost or effectiveness of doing all the jobs.

Considering that term collections of two different parameters may have different sizes, thus the semantic similarity matrix of the two collections may not be a square matrix. Therefore, it is necessary to transformed the semantic similarity matrix to square matrix by adding several rows or columns. Of course, the elements in new-added rows and columns are equal to 0. Then the Hungarian method is applied to the semantic similarity matrix to calculate the maximum similarity between the two term collections. The resultant similarity is regarded as the name similarity between the two parameters.

3.2.2 Propagation similarity

To measure the match similarity between two parameters based on their adjacent parameters, we defined the propagation similarity.

To calculate the propagation similarity, we need to find the correspondences between elements of parameter hierarchy, namely matching between parameter hierarchies. That is, the task is reduced to a matching problem.

In many application scenarios, matching problem is an indispensable task. Various approaches have been proposed for different scenarios because that matching problem often differ a lot due to the variety of mapping models. The similarity flooding algorithm [17] provides a more general technique that is usable across various scenarios. First, the models should be converted into directed labeled graphs, which are used in an iterative fix-point computation. An intuition is kept in mind for computing the similarities, that is, elements of two distinct models are similar if their adjacent elements are similar. In other words, a part of the similarity of two elements propagates to their respective neighbors. The similarities spread in the matched models are reminiscent to the way how IP packets flood the network in broadcast communication. Because of the results of fix-point computation, it comes to the conclusion that one node

in the first graph is similar to one in the second graph. A subset of the resulting mapping is chosen by using adequate filters according to the particular matching goal.

Benefits from the parameter hierarchy construction module, we directly employ the parameter hierarchy graphs in the iterative fix-point computation. Meanwhile, an initial similarity between parameters of the two parameter hierarchy graphs is chosen as a starting point for the iterative computation. In most of cases, the initial similarity is acquired in two ways. One way is to assume that no initial mapping between the two graphs is available. In this case setting the initial similarity between any two nodes from the two graphs to 1.0. The other is to use different matching algorithms, such as a simple string matcher that compares common prefixes and suffixes of literals. The latter way is well-adopted to obtain the initial similarity, more specifically, the name similarity matrix NSM is utilized as the initial similarity for the iterative computation.

Since the parameter hierarchy graphs and the initial similarity are available, then perform the similarity flooding algorithm to propagate the initial similarity of any two parameters through the graphs over a number of iterations. And select a subset of parameter pairs as the best match results. The resultant similarity of a parameter pair is regarded as the propagation similarity between the two parameters in the parameter pair.

3.3 Equivalence Classification

Based on the name similarity and the propagation similarity, it is necessary to utilize multiple similarity measures to determine which parameter is associated with the another, that is, whether one parameter is semantically equivalent with the another.

Typically, the weighted strategy is the better way to aggregate multiple similarities. At present, systems that adopt the weighted strategy need to manually set aggregation weights on the basis of experience for different similarities. However, it is a major limitation that set aggregation weights arduously by hand. And manually predefined weights cannot be generalized to adapt to different situations.

This paper proposes a strategy to aggregate different similarities without manually setting weights to cope with the limitation. Specifically, we adopt several distinguished machine learning approaches, such as Na ïve Bayes, decision trees and support vector machines. The name similarity and propagation similarity are directly passed to the classifiers, which are regarded as the features. Due to the classification, it is determined that whether two parameters are associated with each other.

4. Non-Redundant Annotation And Amendment

In this section, we firstly introduce the overall process of non-redundant annotation, then we present a approach for amendment based parameter association.

4.1 Non-Redundant Annotation

The process of non-redundant annotation is illustrated in *Figure 3*. The parameter association constitutes an essential step for the non-redundant annotation, which lays the foundation for reducing the parameter space. First of all, the documentations of Web Service is parsed to get the parameter information. In previous works, every parameter need to passed to the concept mapping, which result in the redundant annotations. To deal with this issue, the parameter association is performed to determine the equivalence between parameters. An edge is established between two parameters when the parameter association classifiers the two parameters as equivalent. As a result, a number of subgraphs are separated from the original parameter space, and each separate subgraph is a gathering of parameters that are semantically equivalent. The original parameter space is easily reduced based these separate subgraphs. Finally, each parameter in the reduced parameter space is passed to match the concept of ontologies. And the mapping results for the reduced parameter space are expanded to the original parameter space.

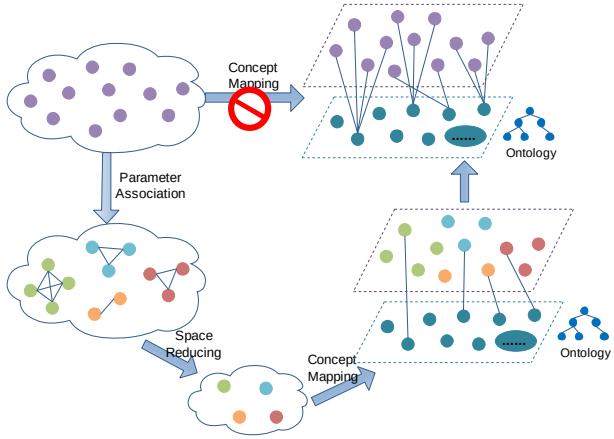


Figure 3. Overall Process of the Non-Redundant annotation

4.1.1 Annotation based on OpenCyc

To calculate the match degree between parameter and the ontology concept in the process of concept mapping based on OpenCyc, two similarities are defined including linguistic similarity and cosine similarity. The linguistic similarity is calculated based on WordNet and the Hungarian method, which is the name similarity between parameter and concept. And the cosine similarity is between descriptions of the parameter and the concept. It is calculated in a vector space model, as written in (3-5).

$$\overrightarrow{pv} = (e_1, e_2, \cdots, e_n) \quad (3)$$

$$\overrightarrow{cv} = (f_1, f_2, \cdots, f_n) \quad (4)$$

$$ds = \frac{\overrightarrow{pv} \bullet \overrightarrow{cv}}{\left\|\overrightarrow{pv}\right\| \bullet \left\|\overrightarrow{cv}\right\|} \quad (5)$$

where $\overrightarrow{pv}$ is calculated based on the description of p_i, $\overrightarrow{cv}$ is calculated based on the comment of ccs_j (concept candidates), ds is the result of cosine similarity. Finally, the similarity degree is a weighted average of the linguistic similarity and the cosine similarity.

The process of mapping parameters with ontology concepts is given in Algorithm 1. This process is performed in the reduced parameter space. For each parameter, the first step is to find ontology concept candidates whose denotation matches the parameter (Line 2). The aim is to provide an strategy that can efficiently access large number of ontology concepts. To achieve this, the parameter is compared against labels of ontology concepts. And then for each concept candidate, a comparison is performed between the parameter and the concept from two aspects. One is linguistic similarity, and the other is cosine similarity (Line 6-8). A weighted average of the linguistic similarity and the cosine similarity is calculated (Line 9). Finally, the parameter is mapped to the concept with the biggest weighted average (Line 10-14).

Algorithm1:ConceptMapping

Input: P: Parameter Set
OC: Ontology Concept Set
Output: PCM: Parameter-Concept Mapping Set

1 **for** i← 1 to P.size
2 ccs = concept candidates whose denotation matches p_i in OC
3 max ← 0; oc ← ""
4 **for** j = 1 to ccs.size
5 ns ← calculate the name similarity between p_i and ccs_j
6 $\overrightarrow{pv}$ ← (e_1, e_2, …, e_n) based on the description of p_i
7 $\overrightarrow{cv}$ ← (f_1, f_2, …, f_n) based on the comment of ccs_j
8 $ds \leftarrow \frac{\overrightarrow{pv} \bullet \overrightarrow{cv}}{\left\|\overrightarrow{pv}\right\| \bullet \left\|\overrightarrow{cv}\right\|}$
9 s ← w_1*ns + w_2*ds
10 **if** (s > max)
11 max ← s
12 oc ← ccs_j
13 **if** (oc ≠ "")
14 PCM ← PCM⋃ {$p_i \rightarrow ccs_j$}
15 **endif**
16 **endif**
17 **endfor**
18 **endfor**

4.1.2 Annotation based on DBpedia

We can also do the annotation work based on DBpedia, thus we can contrast the experiment results. It is sure that the annotation work based on DBpedia is performed in the reduced parameter space, and then the results are expanded to the original space.

We use the DBpedia Spotlight to annotate the parameter name in this approach, by which we can automatically find the effective and appropriate resources from the massive correlation data set. The approach of using Spotlight in this paper is using the efficient, feature-rich client toolkit HttpClient based on Apache Jakarta Common. Call the Spotlight on http://spotlight.dbpedia.org/rest/annotate service, the parameter is as the input passed to the Spotlight. After the Spotlight response, the annotation information in XML format is returned by HttpResponse. Finally parse the XML information by Dom4j, thus, the annotation concept is available. But not all the parameters can find the concepts by Spotlight. The all process of Spotlight is illustrated in *Figure 4.*

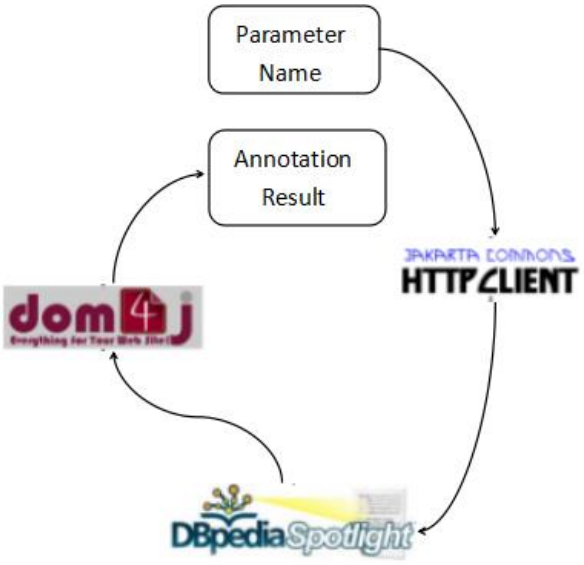


Figure 4. The Process of Using DBpedia Spotlight

To deal with the problem that some parameters can not get the ontology concepts by DBpedia Spotlight. This paper adopt the DISCO algorithm to calculate the similarity between the parameter or resource and concept.

For each parameter, the flow diagram of the DBpedia annotation is illustrated in *Figure 5.* First of all, the parameter name is preprocessed, which is similar to calculate the name similarity of two parameters in section 3. And then as the input, the preprocessed name is passed to the DBpedia Spotlight. There are two cases after call Spotlight service. One case is there is no result, that is, it is fail to call Spotlight service as well as there is no resource. Then we need to calculate the similarity between the parameter and all the DBpedia ontology concepts, and select the biggest similarity degree of the concept as the annotation result. Another case is that there is a returned information in

XML format, by which we can get the parsed result including resource of the parameter. The following there are two cases as well. One case is that the parsed result include the ontology concept, we can get the concept directly. On the contrary, the other case don't have the ontology concept. So we need to calculate the similarity between resource and all the DBpedia ontology concepts, and select the biggest similarity degree of the concept as the annotation result. Therefore, we can get the annotate results in most of reduced parameter space. Then expand the annotation results to original parameter space.

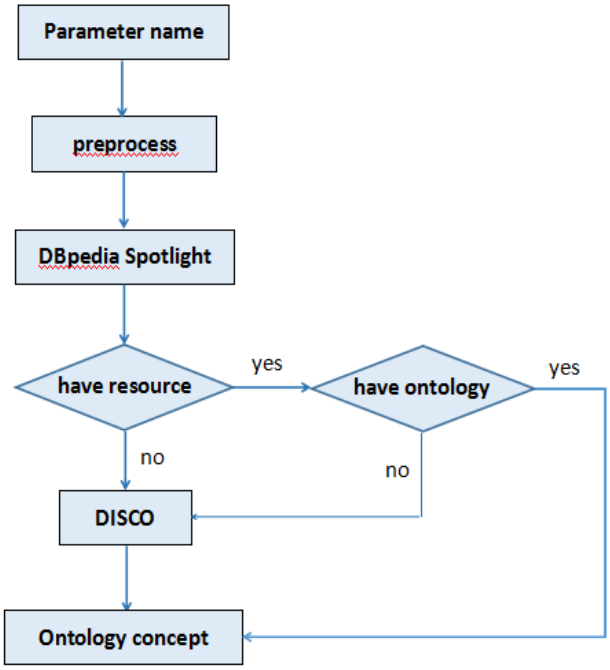


Figure 5. Flow Diagram of DBpedia Annotation

4.2 Non-Redundant Amendment

It is sure that not all annotation results is corresponding with the parameter, though we have high quality through the non-redundant annotation. Manual amendment is necessary for us to ensure the appropriate annotation. Since we have the classification based parameter association, the amendment of a parameter will be applied to all equivalent parameters.

The process of application with amendment is given in Algorithm 2. For an amendment, the first step is to get the id of original parameter and amend the ontology concept (Line 1-2). Then find the association of original parameter (Line 3). On the basis of parameter association, we can get that whether two parameters are semantically equivalent. As a result, a collection of parameters is constructed, which is used to amend the annotation. To show in the prototype tool, we made an array of parameter association information, and each equivalent parameter information is consist of five parts, i.e. pid, service name, operation name, parameter name and io. An example of parameter association collection is illustrated in *Figure 4*. If the array length of association is not zero, get the pid of all parameters in array and amend the ontology concept (Line 4-7).

Algorithm2:AmendmentApplication

Input: P: a object of a parameter
s: string of amendment annotation

```
1   pid ← P.pid
2   amend( pid, s )
3   pAssociation[ ][ ] ← findParameterAssociation (pid)
4   pas ← pAssociation.length
5   if (pas ≠ 0 )
6       for pa = 0 to pas
7           pid ← pAssociation[pa][0]
8           amend( pid, s);
9       endfor
10  endif
```

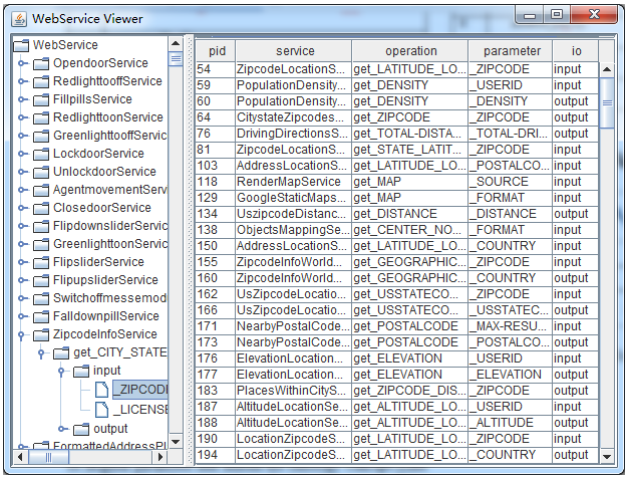


Figure 6. Collection of Parameter Association

As is seen in *Figure 6*, all Web Services, operations and parameters is shown on the left. The parameter named _ZIPCODE is selected. Then the parameters that are semantically equivalent are shown on the right, which include five parts of the information. As a result, when the parameter named _ZIPCODE is amended manually, the parameters on the right are all amended.

5. Evaluation of Annotation Results

In this section, we present the evaluation of annotation results. Since semantic annotation of Web Services is performed to get higher automation of service discovery and composition, results of service discovery to a certain extent

reveal the performance of semantic annotation. An intuitive idea is to assume that there are a number of Web Services and their corresponding semantic Web Services. The semantic annotation of Web Services is performed to be evaluated by the comparison of service discovery that are conducted on the annotated Web Services and on the semantic Web Services respectively. Since our goal is to evaluate the performance of annotation rather than to propose a novel discovery approach, a simple algorithm is utilized for simplicity.

Algorithm3: ServiceDiscovery

```
Input:  S:  Service Set
        q:  Service Query q
Output: RS: Relevant Service Set
1   qis = q.inputs
2   qos = q.outputs
3   for each service s ∈ S
4       ops = s.operations
5       for each operation op ∈ ops
6           ois = op.inputs
7           oos = op.outputs
8           if (ois = Φ)
9               is = 1
10          else if (qis = Φ)
11              is = 0
12          else
13              is = Hungarian(qis, ois)
14          if (qos = Φ)
15              os = 1
16          else if (oos = Φ)
17              os = 0
18          else
19              os = Hungarian(qos, oos)
20          if (is + os >= 2 * threshold)
21              RS = RS ⋃ {s}
22              break
23          endif
24      endfor
25  endfor
```

The process of service discovery is given in Algorithm 2. First of the all process is to get the inputs and outputs of the Web Services and semantic Web Services. Then the general idea is to match inputs and outputs of the query with inputs and outputs of the service operation respectively (Lines 8-19). The match is performed on the assumption that inputs and outputs are already associated with ontology concepts. In other words, ontology concepts are utilized in the match. For the case that inputs of an operation is empty, the input similarity between the query and the operation is 1 (Lines 8-9). For the case that inputs of an operation is not empty but inputs of the query is empty, the input similarity is 0 (Lines 10-11). For the case that inputs of the query and an operation are both not empty, the match is reduced to an assignment problem, thus the input similarity is calculated by employing Hungarian method (Lines 12-13). The match between outputs is similar to the one between inputs (Lines 14-19). And then a threshold is maintained to filter out relevant services (Lines 20-21).

6. EXPERIMENTS

6.1 Experimental Settings

In this section we perform the methodology of parameter association. We also annotate and amend the parameter in reduced space. In addition, in order to evaluate the parameter association approach and the non-redundant annotation, we conduct several experiments with MyEclipse 10.6, MySQL 5.5, WordNet 2.1, Weka 3.7 and OpenCyc 4.0, DBpedia Spotlight, en-wikipedia-20080101. We utilize the fourth version of OWL-S service retrieval test collection (OWLS-TC4) as the experimental data source, providing 1076 Web Services and 1083 Semantic Web Services from nine different domains. In the first experiment, we employ the Naïve Bayes classifier, the J48 classifier, and the SMO classifier implemented in the Weka toolkit [18] for the equivalence classification. In the second experiment, we utilize the fourth version of OpenCyc [19] as the semantic support for the semantic annotation. In the third experiment, we utilize the DBpedia as the semantic support for the semantic annotation. In fourth the experiment, we implement a prototype tool, and manually amend the ontology concept several times to ensure the usability of the tool. Through the prototype tool, it is obvious for us to see which two parameters are semantically equivalent.

6.2 Metrics

Four measures is utilized to evaluate the parameter association approach and the non-redundant annotation, including accuracy, precision, recall and F-Measure. Accuracy is defined as "the fraction of correct records in all records". Precision is defined as "the fraction of retrieved records that are relevant". Recall is defined as "the fraction of relevant records that are retrieved". F-Measure integrates precision and recall into a single, composite harmonic mean. Formally:

$$Accucacy = \frac{|\{correct\}|}{|\{all\}|} \tag{6}$$

$$Precision = \frac{|\{relevant\} \cap \{retrieved\}|}{|\{retrieved\}|} \tag{7}$$

$$Recall = \frac{|\{relevant\} \cap \{retrieved\}|}{|\{relevant\}|} \tag{8}$$

$$F - Measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{9}$$

6.3 Parameter Association Evaluation

Because classification is a supervised learning task, we extract training sets with 225 Web Services from two domains (60 from the geography domain and 165 from the travel domain). And the 225 Web Services provide 225 operations and 766 parameters. Note that the propagation similarity between two parameters may be null, that is, the two parameters are not regarded as a match pair. For convenience, we get rid of the records of two parameters like the above case from training sets. In other words, we only kept the record of two parameter in the training sets if the propagation similarity between the two parameter is not null.

As a result, 80909 records are required. There are 6027 records are regarded as positive examples among all the records. In other words, only 6027 records of two parameters are semantically equivalent. One amazing phenomenon that captures our attention is that equivalent parameters form complete subgraphs. We find the reason behind this is the transitivity of semantic equivalence. More specifically, if parameter P_1 and parameter P_2 are semantically equivalent, parameter P_2 and parameter P_3 are semantically equivalent, it is evident that P_1 and P_3 are semantically equivalent. Another phenomenon is that there are much more negative examples than positive examples, that is, there is a high degree of imbalance between the two classes. In order to deal with the imbalanced data sets, we adopt the random over-sampling strategy [20]. In this way we randomly select examples from the minority class (i.e. the one with fewer cases) and re-adding them in this class, until the two classes are equal in numbers.

First of all, we evaluate the equivalent classification using 10-fold cross-validation. In order to get more reliable evaluation, we repeat the 10-fold cross-validation by ten times. The average accuracy for the equivalence classification with the three classifiers is shown in *Table 2*.

Table 2. Average Accuracy Via 10-fold Cross-validation.

	Naïve Bayes	J48	SMO
Average accuracy	97.54%	99.06%	98.03%

It is obvious that all the three classifiers achieve outstanding classification accuracy, which indicates the feasibility and efficiency of the parameter association approach.We also notice that the Naïve Bayes classifier achieves the lowest accuracy among the three. The depper reason is that the Naïve Bayes classifier is on the basis of the independence assumption for distinct features, however, the independence assumption is clearly violated in the two features.

The average of precision and recall for the positive class and the negative class is illustrated in *Table 3*. It is obvious that all the three classifiers achieve adequate precision and recall for the two classes. We can also get the conclusion that the J48 classifier performs the best among the three, not only with the highest classification accuracy, but also with the highest precision and recall for both of the two classes. Another important conclusion is that the precision and recall for the negative class are higher than the ones for the positive class. The deeper reason is that the costs of false positives are higher than ones of false negatives in the parameter association task. In other words, for incorrectly classified instances, we prefer to the case in which the two parameters are semantically equivalent but classified as not equivalent, rather than the case in which the two parameters are not equivalent but classified as equivalent.

Table 3. Average Precision and Recall for Two Classes Via 10-fold Cross-validation.

Classifier	Positive Class		Negative Class	
	Precision	Recall	Precision	Recall
Naïve Bayes	0.784	0.924	0.994	0.980
J48	0.945	0.928	0.994	0.996
SMO	0.833	0.920	0.994	0.985

To measure the quality of the results, we extract several training sets of different sizes. For example we extract the training sets ranging from much prior knowledge of the parameter association instances (90% training set size) to poor prior knowledge (10% training set size). Using the same method gets the reliable evaluation. we repeat the percentage split for every training set size by five times. The average accuracy of different training set sizes for the three classifiers is illustrated in *Figure 7*. The encouraging conclusion is that as less training data is provided the accuracy remains stable. The result suggests that the size of training sets has little impact on the three classifiers, making them suitable for classifying large unknown sets. That is, all the three classifiers can be applied to the parameter association no matter what size of parameter. So it is proved that the all three classifiers are suitable to the classification based on parameter association.

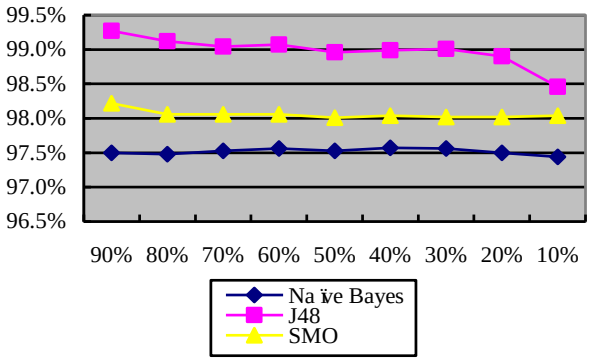


Figure 7. Accuracy of Different Training Set Size

The average precision and recall of different training set sizes for the positive class and the negative class is

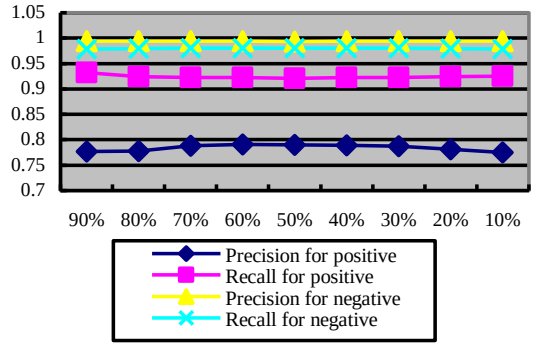


Figure 8. Average Precision and Recall for Two Classes with Different Training Set Size by Naïve Bayes

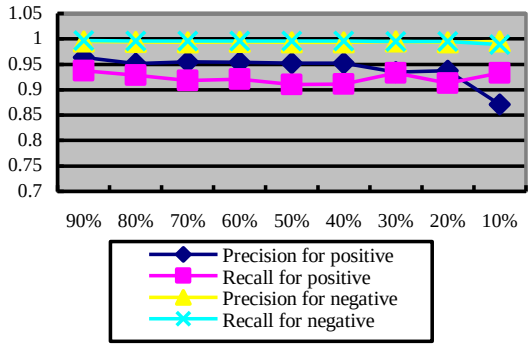


Figure 9. Average Precision and Recall for Two Classes with Different Training Set Size by J48

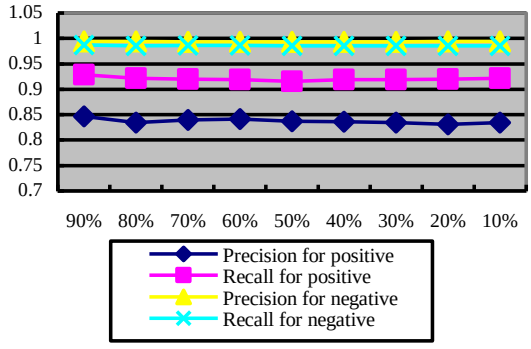


Figure 10. Average Precision and Recall for Two Classes with Different Training Set Size by SMO

illustrated in *Figure 8-10*. It is easy to see that for all the three classifiers, the precision and recall for the negative class are basically constant no matter what size of training set, while the precision and recall for the positive class change slightly as the size of training set differs. For the Naïve Bayes classifier, the precision and recall for the negative class are basically 0.994 and 0.98, while the precision for the positive class ranges from 0.775 (with 10% training set size) to 0.791 (60% training set size), and the recall for the positive class ranges from 0.932 (with 90% training set size) to 0.921 (with 50% training set size). For the J48 classifier, the precision and recall for the negative class are basically 0.994 and 0.996, while the precision for the positive class ranges from 0.964 (with 90% training set size) to 0.871 (with 10% training set size), and the recall for the positive class ranges from 0.937 (with 90% training set size) to 0.911 (with 40% training set size). For the SMO classifier, the precision and recall for the negative class are basically 0.993 and 0.985, while the precision for the positive class ranges from 0.847 (with 90% training set size) to 0.831 (with 20% training set size), and the recall for the positive class ranges from 0.929 (with 90% training set size) to 0.916 (with 50% training set size).

6.4Non-redundant Annotation Performance

6.4.1 Annotation based on OpenCyc

OWLS-TC4 provides 1076 Web Services, in which there are 1076 operations and 3132 parameters. In other words, there are 3121parameters in the original parameter space. With the parameter association, the parameter space is reduced to 383 parameters, that is, the reduced parameter space is merely 12.23 percent of the original space. As a consequence, there is no need to annotate all the 3132 parameters. Instead, merely annotating the 383 parameters will be enough. Therefore, it is suggested that the non-redundant annotation can greatly reduce redundant annotations.

By checking each parameter manually, we acquire the statistics of appropriate and inappropriate mappings, as shown in *Table 4*. The number of parameters that are mapped to appropriate concepts is 2818, accounting for 89.97 percent of all parameters, and 314 parameters are mapped to inappropriate concepts, accounting for 10.03 percent of all parameters. Pay attention to the fact that a concept is regarded as appropriate for a parameter in two cases, one of which is that the concept describes the precise meaning of the parameter, the another is that the concept represents the general meaning of the parameter. Compared with our previous work [11], the accuracy decreases by 6 percent. The deeper reason is that we adopt the different version of OpenCyc, and the different measure of match degree. So we can come to the conclusion, the statistical results indicate that the non-redundant annotation can still achieve adequate accuracy.

Table 4. Statistics of Appropriate and Inappropriate Mappings.

Total	Appropriate	Inappropriate
3121	2818	314

Since OWLS-TC4 provides 42 test queries which are associated with relevance sets, service discovery is performed to conduct performance evaluation. Firstly, service discovery is performed on the annotation results. And then with the same test queries and algorithm, service discovery is performed on the OWL-S services.

For the aforementioned service discovery algorithm, we have maintained a threshold for discovering relevant services. With the increases of the threshold, the precision becomes higher while the recall becomes lower. the comparison between service discovery on annotation results and on OWL-S services as the threshold increases is illustrated in *Figure 11*. More specifically, the average F-Measure as the threshold ranges from 0 to 1 is illustrated in *Figure 11(a)*, and the average precision and recall is illustrated in *Figure 11(b)*. It is easy to see that service discovery on annotation results have similar performance with the service discovery on OWL-S services, in some cases, the former even achieve slightly better performance than the latter. As a result, it can be inferred that Web Services that are annotated by the non-redundant annotation based parameter association can achieve the same effect as OWL-S services do.

6.4.2 Annotation based on DBpedia

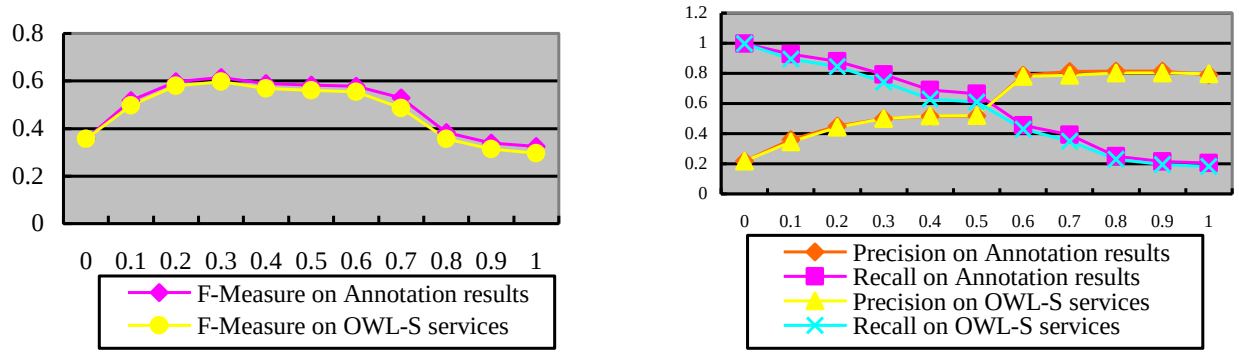
To ensure that the annotation results are not effected by the annotation tools. Another experiment is performed to annotate the parameter by DBpedia. In this experiment, we use the real service on the Internet, which provides 298 Web Service including 1910 operations and 20603 parameters. But not all parameter can be annotated. There are 20467 parameters have been annotated, accounting for 99.3 percent of all parameters. In this way, we call the Web Services to evaluate the annotation results. But the accuracy is only 28.1 percent. It is shown to us that the accuracy is lower than the result in OWLS-TC4. Reasons behind this are summarized to four aspects. First, note that the parameter name is defined casually in the real Web Service, so the parameter name can not reflect the function well, thus the parameter name interfere the annotation results. Second, the DBpedia Spotlight has the limitation that sometimes there isn't a corresponding resource returned. Therefore, the annotation results may be inappropriate with the parameter. Third, a great number of abbreviations are unknown, which interfere the annotation results. Finally, it is the most important reason that not all Web Services can be called successfully. According to the statistics, there are only 158 Web Service, which is alive, that is, the alive Web Services are only about half of all Web Services. And the 158 Web Services include 1046 operations. The statistics results are shown in *Table 5. T*here are 294 operations are called successfully, while there are 752 operations are failed.

Table 5. Statistics of Successful and Failed Operations.

Total	Successful	Failed
1046	294	752

6.5 Prototype Tool

In order to verify the effectiveness of the framework proposed in this paper, an prototype tool is implemented. A screenshot of the prototype is illustrated in *figure 13*. As is shown to us, all Web Services are the branches of the root named "WebService", and they also include the operations as the branch. Under each operation, there are two elements named "input" and "output". The input parameters and output parameters are the respective branches of "input" and "output". Every parameter can be selected, and then the detailed information is shown on the right of the window, which include the service name, operation name, parameter



(a) *(b)*

Figure 11. Comparison of Service Discovery on Annotated Services and OWL-S Services: (a) F-Measure; (b) precision and recall.

name, io, concept of annotation, match by people, match degree, and all parameters that is semantically equivalent. Moreover, the parameters in association can also be selected to shown their own information, which is illustrated in *Figure 12*. As it is shown to all, there is a button can be pressed to manually amend the annotation which is not corresponding with the parameter. The results of the experiment indicate that we can implements all the approach proposed in this paper. When we amend the annotation of the parameter, the annotation results of all the semantically equivalent parameters are also changed.

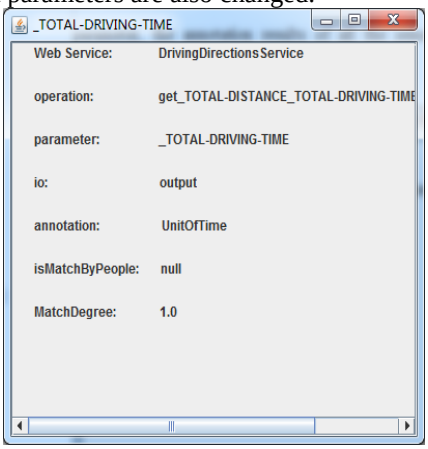


Figure 12. Screenshot of the parameter information.

7. CONCLUSIONS

In this paper, we propose a classification on the basis of parameter association approach. There are both benefits of the approach. On one hand, it is not necessary to rely on external auxiliary resources to find the equivalence between parameters by this approach, but just use the documentations of Web Services. On the other hand, the classification technique employs three distinguished machine learning approaches when attempting to aggregate multiple similarities, which avoids manually setting weights. And then we proposed the non-redundant annotation and amendment for Web Services. The benefits of non-redundant annotation and amendment are self-evident. First, it avoids redundant annotations and greatly reduces the workload. Then it guarantees the exact annotations. To evaluate the performance of annotation, we further present a methodology based on service discovery. In the end, we implement a prototype tool to show all the results including parameter association, concepts of annotation, and non-redundant amendment.

We verify the feasibility and efficiency of the parameter association approach with the support of Weka, and prove the effectiveness of the non-redundant annotation with the support of OpenCyc and DBpedia. The experimental results of the former indicate that the parameter association approach can achieve outstanding accuracy, even when the

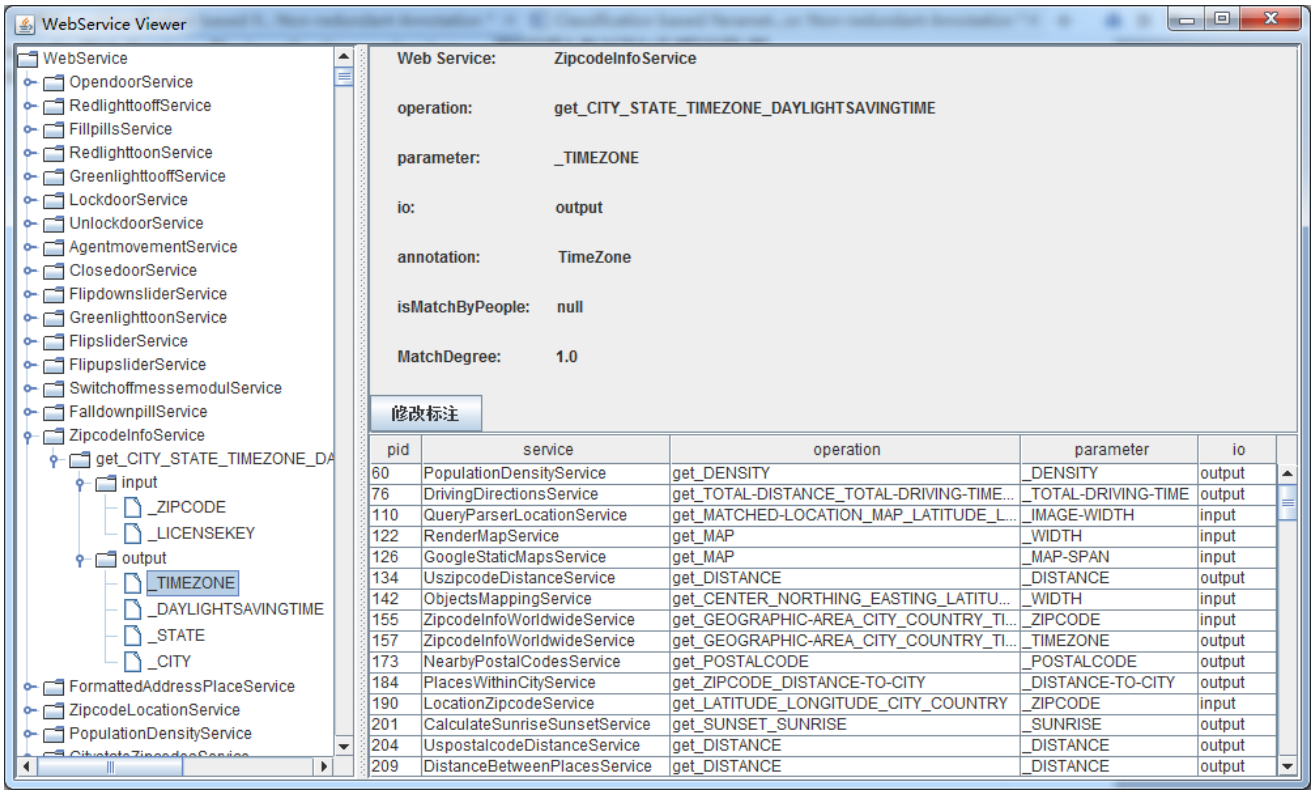


Figure 13. Screenshot of the Prototype Tool.

training set size is small, in finding parameters that have same semantics. The experimental results of the latter suggest that the non-redundant annotation can greatly reduce redundant annotations as the parameter space is heavily reduced, and can augment the semantics of Web Services with adequate accuracy, which can achieve similar performance in service discovery as OWL-S services do.

In the future, we plan to explore the semantics refinement of Web Services based on feedback and parameter association. Since all current solutions on automatic semantic annotation cannot guarantee the absolute accuracy, feedback mechanism is supposed to assist in improving the accuracy of semantic annotation. Meanwhile, since resources for manual annotation are both scarce and expensive, the parameter association is expected to expand manual annotations to more Web Services for precise annotations. Thus we can acquire the precise annotations when a new Web Service comes with the semantically equivalent parameters that available.

8. ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China grant 61373035, 61502333, 61502334, 61572350, and the Tianjin Research Program of Application Foundation and Advanced Technology grant 14JCYBJC15600. Keman Huang is the corresponding author.

9. REFERENCES

[1] S.J. Vaughan-Nichols, "Web services: beyond the hype," Computer, vol. 35, Feb. 2002, pp. 18-21, doi: 10.1109/2.982908.

[2] S. Agarwal, S. Handschuh, and S. Staab, "Surfing the Service Web," the 2nd International Semantic Web Conference (ISWC 03), Oct. 2003, pp. 211-226, doi: 10.1007/978-3-540-39718-2_14.

[3] A.A. Patil, S.A. Oundhakar, A.P. Sheth, and K. Verma, "METEOR-S Web Service Annotation Framework," the 13th International Conference on World Wide Web (WWW 04), ACM, 2004, pp. 553-562, doi: 10.1145/988672.988747.

[4] A. Hess, E. Johnston, and N. Kushmerick, "ASSAM: A Tool for Semi-Automatically Annotating Semantic Web Services," in LNCS, vol. 3298, S.A. Mcllraith, D. Plexousakis, and F. van Harmelen, Eds. Heidelberg: Springer, 2004, pp. 320-334.

[5] K. Lerman, A. Plangprasopchok, and C.A. Knoblock, "Automatically Labeling the Inputs and Outputs of Web Services," the 21st National Conference on Artifical Intelligence (AAAI 06), AAAI Press, 2006, pp. 1363-1368.

[6] K. Belhajjame, S.M. Embury, N.W. Paton, R. Stevens, and C.A. Goble, "Automatic Annotation of Web Services based on Workflow Definitions," the 5th International Semantic Web Conference (ISWC 06), Nov. 2006, pp. 116-129, doi: 10.1007/11926078_9.

[7] I. Salomie, V.R. Chifu, I. Giurgiu, and M. Cuibus, "SAWS: A Tool for Semantic Annotation of Web Services," IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR 08), IEEE Press, May. 2008, pp. 387-392, doi: 10.1109/AQTR.2008.4588773.

[8] P. Kungas and M. Dumas, "Cost-Effective Semantic Annotation of XML Schemas and Web Service Interfaces," IEEE International Conference on Services Computing (SCC 09), IEEE Computer Society, Sep. 2009, pp. 372-379, doi: 10.1109/SCC.2009.17.

[9] E.S. Chifu and I.A. Letia, "Unsupervised Semantic Annotation of Web Serivce Datatypes," IEEE International Conference on Intelligent Computer Communication and Processing (ICCP 10), IEEE Press, Aug. 2010, pp. 43-50, doi: 10.1109/ICCP.2010.5606464.

[10] D. Bouchiha and M. Malki, "Semantic Annotation of Web Services," the 4th International Conference on Web and Information Technologies (ICWIT 12), Apr. 2012, pp. 60-69.

[11] X.C. Hu, Z.Y. Feng, and S.Z. Chen, "Augmenting the Semantics of Web Services based on Public Open Ontology," IEEE International Conference on Services Computing (SCC 13), IEEE Computer Society, Jun. 2013, pp. 304-311, doi: 10.1109/SCC.2013.96.

[12] M.F. Porter, "An algorithm for suffix stripping," Program, vol. 14, 1980, pp. 130-137, doi: 10.1108/eb046814.

[13] C. Fellbaum, WordNet(s). Encyclopedia of Language & Linguistics, 2nd ed., 2006, pp. 665-670, doi: 10.1016/B0-08-044854-2/00946-9.

[14] Fellbaum C. WordNet: An electronic lexical database MIT Press[J]. Journal of biomedical informatics, 2009, 48: 40~59.

[15] M.S. Bazaraa, J.J. Jarvis, and H.D. Sherali, Liner Programming and Network Flows. Wiley, 2009.

[16] H. Basirzadeh, "One Assignment Method for solving Assignment Problems," Applied Mathematical Sciences, vol. 6, 2012, pp. 2345-2355.

[17] H.W. Kuhn, "A tale of three eras: The discovery and rediscovery of the Hungarian Method," Eur. J. Oper. Res., vol. 219, Jun. 2012, pp. 641-651, doi: 10.1016/j.ejor.2011.11.008.

[18] S. Melnik, H. Garcia-Molina, and E. Rahm, "Similarity flooding: a versatile graph matching algorithm and its Application to schema matching," the 18th International Conference on Data Engineering, IEEE, Feb. 2002, pp. 117-128, doi:10.1109/ICDE.2002.994702.

[19] I.H. Witten and E. Frank, Data Mining: Practical Machine Learning Tools and Techniques. 2nd ed., Elsevier, 2005.

[20] OpenCyc. http://www.cyc.com/platform/opencyc.

[21] G.E.A.P.A. Batista, R.C. Prati, and M.C. Monard, "A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data," Sigkdd Explorations, vol. 6, issue 1, 2004, pp. 20-29, doi: 10.1145/1007730.1007735.

[22] Bizer C, Lehmann J, Kobilarov G, et al. DBpedia--A Crystallization Point for the Web of Data[J], Journal of Web Semantics. 2009, 7(3): pp154-165.

[23] DBpedia Application[EB/OL]. http://wiki.dbpedia.org/Applications, 2012-5-31

[24] DBpedia Spotlight [EB/OL]. http://dbpedia.org/spotlight.

[25] Mendes P.N., Jakob M., García-Silva A., et al. DBpedia spotlight: shedding light on the web of documents[A]. In Proceedings of I-SEMANTICS[C]. 2011, 1-8.

[26] Gombotz R. and Dustdar S. On Web services workflow mining[A]. In Proceedings of the BPI Workshop (LNCS)[C], co-located at BPM, Nancy, France. Berlin: Springer-Verlag.2005. p. 216-228.

[27] Solé-Ribalta A, Sánchez D, Batet M, et al. Towards the estimation of feature-based semantic similarity using multiple ontologies[J]. Knowledge-Based Systems, 2014, 55: 101~113.

[28] Mokarizadeh S, Küngas P, Matskin M. Ontology learning for cost-effective large-scale semantic annotation of web service interfaces[M]. Knowledge Engineering and Management by the Masses. Springer Berlin Heidelberg, 2010: 401~410.

[29] Harispe S, Sánchez D, Ranwez S, et al. A framework for unifying ontology-based semantic similarity measures: A study in the biomedical domain[J]. Journal of biomedical informatics, 2014, 48: 38~53.

[30] Mokarizadeh S, Küngas P, Matskin M. Ontology learning for cost-effective large-scale semantic annotation of web service interfaces[M]. Knowledge Engineering and Management by the Masses. Springer Berlin Heidelberg, 2010: 401~410.

Authors

Xuehao Sun is a master candidate in School of Computer Science and Technology, Tianjin University, China. Her research interests are in the areas of service computing and Mobile Internet.

Shizhan Chen received the PhD degree in School of Computer Science and Technology, Tianjin University in 2010. He is currently an associate professor in Tianjin University. His research interests are in the areas of service computing and Mobile Internet.

Zhiyong Feng is a Professor at the school of Computer Soft ware, Tianjin University. He received the PhD degree from Tianjin University in 1996. His research interests include knowledge engineering, services computing, social computing and security soft-ware engineering.

Keman Huang is an assistant professor with School of Computer Science and Technology, Tianjin University, China. He received his PhD degree in automation from Tsinghua University, China in 2014. His research interests focus on the evolution and promotion of service ecosystem.

Dongxiao He is an assistant professor with School of Computer Science and Technology, Tianjin University, China. She received the PhD degree in College of Computer Science and Technology, Jilin University, China in 2014. Her current research interests include data mining, and analysis of complex networks.

Xiaocao Hu is a PhD candidate in School of Computer Science and Technology, Tianjin University, China. Her research interests are in the areas of service computing.