Advanced Tagging and Semantic-Annotation Methods for the Semantic-based OpenAPI Retrieval System

Seung-Jun Cha and Kyu-Chul Lee
Chungnam National University
{junii, kclee}@cnu.ac.kr

Abstract

The OpenAPI retrieval system has been developed applying semantic web technologies as the typical information retrieval method had. The semantic web technologies applied in the system are tagging and semantic-annotation. However, these methods are difficult for the non-technical general users to use and the probability of failure is high. To solve these problems we suggest improved methods for the tagging and semantic-annotation. Tagging turn human-readable service description to machine-readable one by adding tags into the page. Through the use of XHTML and DTD modification, auto-completion and auto-view are enabled. Semantic-annotation means to input related semantic information into parameters in the service description. The java-based tool providing methods of target parameters selection and the ontology viewer is developed for annotating easily. Tagging and semantic-annotation are applied and tested through the test-bed system. These improvements enabled non-technical general users to applying semantic information to the OpenAPI retrieval system.

Keywords: OpenAPI, Service Retrieval, Tagging, Semantic Annotation

1. Introduction

The new web trend, ‘Web 2.0’ is first appeared after the dot-com collapse; it has the distinguished characteristics of survived company’s in dot-com bubble. Web 2.0 is a term which is widely used to describe developments to the web which provide an emphasis on use of the Web to provide collaborative and communications services, as opposed to a previous environment in which the Web was used primarily as a one-way publishing tool.

Especially the first principle of Web 2.0, the web as a platform, means the environments for users to develop and deploy their own services. A lot of service vendors, i.e., Google and Facebook, open the API interface of their services; it is named the OpenAPI. Depending on such phenomenon, the number of the OpenAPI is enlarging rapidly. According to the web site ‘ProgrammableWeb’, the typical web site providing the information about the OpenAPIs, there already is 7,770 services and 324 services are added per month. Therefore, for the effective use of the OpenAPIs, the retrieval system ([1]) is developed.

The paper [2] suggested the integrated retrieval methods for the various protocols of OpenAPIs. Integrated service information model (ISIM) was defined for adopting various protocols of services to the single model. The keyword-based retrieval method was developed based on ISIM. The paper [3] applied semantic technologies, tagging and semantic annotation, to existing retrieval system for overcoming the limitation of the keyword-based retrieval method in paper [2]. The tagging and semantic-annotation mean to add semantic information to machine-readable service description. The retrieval methods are also enhanced using...
common-sense and domain ontologies. By use of the semantic technologies, the retrieval results are purified; the precision has increased. However, the administrator manually do tagging and semantic-annotation to every service. It is difficult for the non-technical general users to apply and the probability of failure is also high.

To solve these problems we improved the methods of tagging and semantic-annotation. Enhanced tagging method and the tool for semantic-annotation are developed. Furthermore, we make the proposed methods experiment in the prototype system.

The rest of the paper is organized as followed. In the Section 2 describe about the developed integrated retrieval system for OpenAPI as related work. Enhanced tagging method is shown in the Section 3, and semantic-annotation tool is shown in the Section 4. The Section 5 explains about the experimental test result, and conclusions and future works follow in the Section 6.

2. Integrated Retrieval System for OpenAPI

In the paper [3], various kinds of protocols for OpenAPIs (SOAP, REST, JavaScript, RSS, etc.) are collected and stored in the database using semantic technologies. To do this, the semantic annotated-integrated service information model for OpenAPI (SA-OpenAPI) was defined first for adopting various protocols as well as embedding semantic information. Tagging and semantic-annotation method is developed to convert the service description into SA-OpenAPI. The semantic based retrieval method was also developed.

2.1. SA-OpenAPI

SA-OpenAPI is the XML-based description based on the semantic integrated service information model defined in the paper [4]. The semantic related information is added (Figure 1).
Top level of the SA-OpenAPI is the BusinessEntity. It means the organization or company of the providing services. Below the BusinessEntity, the BusinessService and the Service come with. It means the services provided from the BusinessEntity. Port, the group of the operations, and Operation are come with the Service in order. Every element is basically composed of the name, the name of the service, and description (documentation), the detailed information of the service.

The semantic information is defined in the input/output parameter below the Operation. The modelReference is the reference information to the class in the ontology having related meaning with the parameter. The liftingSchemaMapping and the loweringSchemaMapping are the converting information to/from the ontology. They should be used together with the modelReference.

2.2. Tagging

Tagging is the method for embedding machine-readable tags into the human-readable service description for extracting necessary elements from the service description automatically. The RDFa [4], the W3C’s standard for adding a set of attribute-level extensions to XHTML for embedding rich metadata within web documents, is used for the element.
In order to apply RDFa to the research, RDF schema applied the structure of the SA-OpenAPI should be defined. BusinessEntity, BusinessService, Service, Port, and Operation is define with the class, and the object property, relation between class, and the data property, the value of the class, is also defined. The whole structure of the schema is shown in Figure 2.

Tagging is classified into three types; tagging about the class, tagging about the class and object property, and tagging about the data property. Elements (‘div’, ‘span’) and attributes (‘src’, ‘rel’, ‘about’, and ‘typeof’) are used for tagging complying with the RDFa standard.

Tagging about the class is only used to the root class BusinessEntity. The element ‘div’ is defined for the name of the element. The attribute ‘typeof’ is for the name of the class and ‘about’ is for the ID value for the specific class. Followed example is about the BusinessEntity tagging having ID ‘BE-01’.

```html
<div typeof="us:BusinessEntity" about="BE-01" />
```

Tagging about the class and object property is used to the parent-associated class with BusinessService, Port, and Operation. The element ‘div’ is defined for the name of the element either and the element ‘span’ is used to define the associated class. In the ‘div’ element, the attribute ‘src’ is for the ID of the parent’s and ‘rel’ is for the name of the object property. In the ‘span’ element, the attribute ‘typeof’ is for the name of the class, and ‘about’ is for the ID value for the specific class. Followed example is about the BusinessService tagging having ID ‘BS-01’ and associated class ‘BE-01’ as ‘hasBusinessService’ relation.

```html
<div src="BE-01" rel="us:hasBusinessService">
  <span typeof="us:BusinessService" about="BS-01" />
</div>
```

Tagging about the data property is used to define the value of the class. The element ‘span’ is defined for the class. The attribute ‘src’ is for the associated class, and ‘property’ is for the name of the object property. Followed example define name of the BusinessEntity ‘BE-01’ valued ‘google’.

```html
<span src="BE-01" property="us:hasBusinessEntityName">google</span>
```

The additional information as ‘category’, ‘accesspoint’, and ‘protocol’ is defined using ‘span’ element. Those elements may not appear in the service description page. The attribute ‘src’ is for the associated class, ‘property’ is for the name of the object property, and ‘content’ is for the value. Followed example define the category and the accesspoint of the BusinessService ‘BS-01’ valued ‘weather’, and ‘…url…’.

```html
<span src="BS-01" property="us:hasCategory" content="weather" />
<span src="BS-01" property="us:hasAccessPoint" content="...url..." />
```

The values of each attributes are fixed based on the RDF schema in Figure 2. Table 1 shows the description of the attribute values for every attribute.
### Table 1. Attribute Values

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Attribute Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>typeof</strong></td>
<td>businessEntity</td>
</tr>
<tr>
<td></td>
<td>businessService</td>
</tr>
<tr>
<td></td>
<td>port</td>
</tr>
<tr>
<td></td>
<td>operation</td>
</tr>
<tr>
<td><strong>rel</strong></td>
<td>hasBusinessService</td>
</tr>
<tr>
<td></td>
<td>hasPort</td>
</tr>
<tr>
<td></td>
<td>hasOperation</td>
</tr>
<tr>
<td><strong>property</strong></td>
<td>hasBusinessEntityName</td>
</tr>
<tr>
<td></td>
<td>hasBusinessEntityDescription</td>
</tr>
<tr>
<td></td>
<td>hasServiceName</td>
</tr>
<tr>
<td></td>
<td>hasServiceDescription</td>
</tr>
<tr>
<td></td>
<td>hasCategory</td>
</tr>
<tr>
<td></td>
<td>hasAccessPoint</td>
</tr>
<tr>
<td></td>
<td>hasPortName</td>
</tr>
<tr>
<td></td>
<td>hasPortDocumentation</td>
</tr>
<tr>
<td></td>
<td>hasProtocol</td>
</tr>
<tr>
<td></td>
<td>hasOperationName</td>
</tr>
<tr>
<td></td>
<td>hasOperationDocumentation</td>
</tr>
<tr>
<td></td>
<td>hasInput</td>
</tr>
<tr>
<td></td>
<td>hasOutput</td>
</tr>
</tbody>
</table>

#### 2.2. Semantic-annotation


The methods of semantic-annotation are to put additional metadata, a `modelReference`, `liftingSchemaMapping`, and `loweringSchemaMapping`, into input/output parameters as an attribute.

A `modelReference` is the reference information of the ontology class having same meaning with a selected input/output parameter. A `liftingSchemaMapping` refer to the conversation information from the datatype of the parameter to the datatype of the ontology class. A `loweringSchemaMapping` is vice versa. Followed example is about the semantic-annotation in input parameter ‘city’ `modelReferenced` the ontology class ‘address’ and `lifting/loweringSchemaMapping` mapped each ‘lifting.xslt’ and ‘lowering.xslt’.

```xml
<sahtml:modelReference="http://url/gis.owl#address"
sahtml:loweringSchemaMapping="http://url/lowering.xslt">
city</a>
```

#### 3. Enhanced Tagging Method

The proposed tagging method has a limitation that administrators always analyze every HTML page having service description and tag the metadata into them manually. Therefore,
non-programmers hardly analyze the document as well as tag the metadata. Furthermore, it is
not guarantee the fault-tolerant as the metadata items are too complex.
To solve these problem, we propose enhanced the RDFa tagging method using the
XHTML. XHTML apply the grammatical structure of XML to the HTML [7, 8]. By applying
the RDFa structure to XHTML, tagging has functions of auto-completion and auto-view.
Auto-completion is the function when input specific alphabet, related keywords are
automatically completed, and auto-view suggests attribute names related the input element.
For the use of these functions, non-programmer can easily tag and also guarantee fault-
tolerant.
To do this, first we modify the DTD for XHTML supporting the attribute names and values
defined for the RDF schema, and then examine tools can adopt the modified DTD and
support auto-completion and auto-view.

3.1. Modification of the HTML DTD

The XHTML provides the DTD for structuring. Necessary elements for tagging the RDFa
might add in the DTD for having structure of the SA-OpenAPI. Only the elements ‘div’ and
‘span’ are used for tagging.
Table 3 show the modified DTD adding attribute names and values defined in Table 1.

<table>
<thead>
<tr>
<th>Table 2. Modified DTD for XHTML (Partial)</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;!ELEMENT div %Flow;&gt; &lt;!- generic language/style container - &gt;</td>
</tr>
<tr>
<td>&lt;!ATTLIST div typeof (us:BusinessEntity) #IMPLIED&gt;</td>
</tr>
<tr>
<td>&lt;!ATTLIST div about CDATA #IMPLIED&gt;</td>
</tr>
<tr>
<td>&lt;!ATTLIST div rel (us:hasBusinessService</td>
</tr>
<tr>
<td>&lt;!ATTLIST div src CDATA #IMPLIED&gt;</td>
</tr>
<tr>
<td>&lt;!ELEMENT span %Inline;&gt; &lt;!- generic language/style container - &gt;</td>
</tr>
<tr>
<td>&lt;!ATTLIST span property (us:hasBusinessEntityName</td>
</tr>
<tr>
<td>&lt;!ATTLIST span content CDATA #IMPLIED&gt;</td>
</tr>
<tr>
<td>&lt;!ATTLIST span src CDATA #IMPLIED&gt;</td>
</tr>
<tr>
<td>&lt;!ATTLIST span about CDATA #IMPLIED&gt;</td>
</tr>
<tr>
<td>&lt;!ATTLIST span typeof (us:BusinessService</td>
</tr>
</tbody>
</table>

3.2. Evaluation

We use the application XML-SPY[9] for evaluating the proposed tagging method. XML-
SPY basically provides the text editing interface and also support functions for auto-
completion and auto-view.
For tagging, first open new XHTML document and apply SAHTML DTD file. Then define
the namespace for tagging and semantic-annotation. Table 4 indicates the modified
namespace. 3 namespaces, for using rdfs, tagging, and semantic-annotation, is defined.
The function auto-view is used as shown in Figure 3(a). User want to input ‘div’ element
for defining the class, then input just ‘<div’ text. The possible attribute names ‘about’, ‘rel’,
‘src’, ‘typeof’ are automatically viewed on the screen.
The function auto-completion is used as shown in Figure 3(b). User want to input
relationship about ‘BE-01’, ‘rel=’ keyword are inputted automatically using auto-view. The
possible values of the attribute ‘rel’ are shown on the screen and autocompleted.
4. Tool for the Semantic Annotation

The paper [5] proposed the semantic-annotation method that manually input semantic related metadata in the web page of the service description. It is difficult for non-specialists who never deal with the semantic information or ontology to annotate directly.

To solve this problem, we develop semantic annotation tool for annotating easily. It simply shows parameter information required annotation, and provide ontology viewer for effective navigation of the ontology.

The java applet based tools is shown in Figure 4. It classified in 4 sections; Operation information, Semantic-annotation, Element information, and Ontology viewer.

4.1. Ontology Viewer

In the OWL specification, the relationship among classes in the ontology is defined ‘subClass’, ‘equivalentClass’, ‘disjointClass’. Based on this definition, we design the ontology viewer analyze the ontology then output the relationship of classes on the tree and graphic interface. The OWL API [10] is used for analyzing ontology. As the top class of the ontology is ‘Thing’, the tree and graphic view expanded from ‘Thing’. Users traverse the nodes and select the related class with the meaning of the parameter.

The tree viewer is structured tree from the top to the bottom class of the ontology (Figure 5). Users easily notify the classes what they want to use.
The tree viewer provide two functions using just clicking arrow left of the class name or double clicking: Expand Node, Collapse Node. Expand Node expand nodes just one level below from the selected class. Collapse Node collapse sub-nodes from the selected class.

The essential part of the graphic viewer (Figure 5) is based the open source project TouchGraph [11]. It is to output related information as a graph. It also provided basic screen and node configuration. The ontology information analyzed using the OWL API is graphically shown using the TouchGraph reflecting the characteristics of the semantic.

The graphic viewer provides additional functions of controlling nodes. Expand Node, Collapse Node, Hide Node, Center Node, and Apply functions are supported by use of the mouse left button shown in Figure 6. Radius, Zoom, Rotate, and Hyperbolic functions are supported below the tool.

Expand Node shows all sub-trees of selected class, and Collapse Node is vice versa. Hide Node hides the selected class only. Center Node moves the selected class on the center of the screen. Apply applies the selected class to the tool. Radius is the setting value for the output depth from the selected class. It can be set from 1 to 10. Zoom control the size of the tree, and Rotate control the rotation of the graph. Hyperbolic controls the length of the connected line between nodes.
4.2. Process of the Semantic Annotation

The process of the semantic-annotation is as follows. First, service descriptive information (SAHTML), having tagging metadata is loaded. Operation information is structured in the table classified in input and output. Second, user select the specific parameter want to annotate. Next, verify the class of the ontology related the selected parameter and load using the ontology viewer. Then, do semantic-annotation to the selected parameter; add `modelReference`, `liftSchemaMapping`, and `lowerSchemaMapping` information. Finally, store semantic-annotated service description in the database, then utilize for retrieving services and the ontology matching. Figure 8 describe the completing status of the annotation.

5. Experimental Test

5.1. Test Environment

The environment for the experimental test is followed;

- Operation System: Fedora release 8
- System Development: JSP, Tomcat 5.0, MySQL 5.0
- Tagging Tool: XML-SPY(applying SAHTML.dtd)
- Semantic-Annotation Tool: Java-based applet

Fedora 8 is used for the operating system, and entire system is developed in java language. Therefore, JSP is used for web page and Tomcat 5.0 is used for servlet container. MySQL 5.0 is used for storing the metadata. The tool ‘XML-SPY’ is used for the RDFa tagging, the semantic-annotation tool is developed using java-based applet. The semantic-annotation is provided shaped java applet in the HTML page.

5.2. Interface

The interface for the tagging and the semantic-annotation is shown is Figure 8. Ⓐ is a button for tagging and Ⓡ is a button for semantic-annotation. Registered services are listed in each table.

In the SOAP-based Web Services, services are fetched from the UDDI, web service repository. If semantic information has already adopted using SAWSDL, semantic-annotation is not required.

In the RESTful web service and others, there are three ways to register services to the system. One is to input service description manually according to the SA-OpenAPI. Another is to upload services. This method is used when SA-OpenAPI shaped service descriptions are already created and want to upload bundle of services. The other is to get the service description from the HTML page. Tagging and semantic-annotation is required.
5.3. Tagging

Tagging is applied only in the RESTful web service and others. Figure 9 show the interface for Tagging. The source code of HTML for Target service information is stored on the clipboard by using ‘Original’ button. After do tagging, the document is saved in the file, then uploaded to the system. ‘Save’ button is reflecting the file to the system.

![Figure 8. Interface for the Tagging](image)
5.4. Semantic Annotation

The semantic-annotation can be applied only services which are already tagged. By use of ① button, the java applet based semantic-annotation tool is executed and user can do semantic annotate. After annotating, modified service description is reflected in the system. Figure 12 show the result after semantic-annotation.

![Figure 9. Service Description after Semantic Annotation](image)

The effective for suggest method of tagging and semantic-annotation is verified by the testbed prototype system.

6. Conclusion

The purpose of this paper is to develop enhance methods of tagging and semantic-annotation. Tagging and semantic-annotation is required for processing automate of service discovery. Current methods are manual, inefficient, and time-consume. To solve these problems, we studied methods as followed.

Tagging methods are enhanced to input complex defined element and attribute accurately by using suggested auto-completion and auto-view function. Java applet based semantic-annotation tool is also developed. User easily found and adopted the parameter required semantic-annotation.

Developed methods for tagging and semantic-annotation will be integrated other related studies. If related studies, SAWSDL as well as hREST, SA-REST are handled together, various services are utilized integrated.

Acknowledgements

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology(2012R1A1A4A01015696).

References


Authors

Seung-Jun Cha finished a M.E. degree in computer engineering from Chungnam National University, Korea, 2008. He is currently in Ph.D candidate in department of computer science and engineering in Chungnam National University and researcher in the Software Research Center. He is also His research interests include database, XML, Web Services, and GIS.

Kyu-Chul Lee received B.E., M.E., and Ph.D. degrees in Computer Engineering from Seoul National University in 1984, 1986, and 1996, respectively. In 1994 he worked as a visiting researcher at the IBM Almaden Research Center, San Jose, California. From 1995 to 1996, he worked as a Visiting Professor at the CASE Center at Syracuse University, Syracuse, New York. He is currently a Professor in the Department of Computer Engineering at Chungnam National University, Daejeon, Korea.

His current areas include Multimedia Database System, Hypermedia Systems, Object-Oriented Systems, and Digital Libraries. He has published over 100 technical articles in various journals and conferences. He is a member of ACM, IEEE Computer Society, and Korea Information Science Society.