Mapping XML to RDF: An algorithm based on element classification and aggregation

Yuchun Liu¹, Zhiyu Hong¹*

¹School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, Jiangsu, China
E-mail: zhiyuhong@nuaa.edu.cn

Abstract. Many data in Web are exchanged and stored in XML. RDF achieves rapid response and reasoning by aggregating a large amount of knowledge. It has been widely used in intelligent field as a way of data organization. Building RDF based on XML is the focus of this article. This paper analyzes the tree structure of XML, divides the XML elements into three sub-models according to the structure, then aggregates and classifies XML elements, and abstracts the aggregates into three abstract structure models, and finally proposes related mapping rules. Based on the optimization of existing methods, this paper proposes an XML-to-RDF mapping algorithm based on element classification and aggregation, which is reflected in three aspects: optimized mapping rules to ensure the completeness and accuracy of data content and semantics; proposed general algorithms for different types of XML; identified equivalent elements of XML and solved the problem of RDF data redundancy caused by such equivalent elements.

1. Introduction

The emergence of RDF originated from the description of metadata. It was invented by Apple researcher Guha in 1997. The concept of the Semantic Web was proposed by Tim Berners-Lee [1]. The Semantic Web is based on the organization of graphs and "links", but what is linked is no longer a web page but an entity in the objective world. Semantic descriptions have also been added to the links. RDF is a data model used to organize semantic data in the Semantic Web. RDF Schema [2] gives it semantics by defining the vocabulary used in RDF. As a product of the development of the Semantic Web, the knowledge graph has become a hot research topic. Although not all knowledge graphs' framework is RDF, its essence is to describe entities and their attributes and relationships. In the knowledge graph, triples are called knowledge. RDF achieves rapid response and reasoning by aggregating a large amount of knowledge. It has been widely used in the field of intelligence as a way of data organization and has become the recommended standard of W3C.

There are a large number of data with different structures without semantic information in Web. Building RDF or knowledge graphs based on these data has become an important research direction. XML is a markup language [3], and its data model is called Document Object Model (DOM). The elements and attributes in XML documents store data information. XML is a unified data storage and exchange format in Web. Because of its flexible structure and strong portability, data described in XML is widely available in the Web. Building RDF based on XML is the focus of this article.

2. Related work

Some work has been devoted to researching methods for constructing RDF based on XML. Michel Klein proposed in [5] to map the data in an XML document to a sequence of RDF triples based on RDF
Schema. RDF Schema specifies the classes and attributes of the RDF data generated by the mapping. In the process of traversing the XML DOM tree deeply, select the relevant data in the XML document based on the classes and attributes and map it to the RDF components. However, the classes and attributes are completely customized manually and depend on the user's subjectivity, so the semantic information in the original data may be lost and some irrelevant semantics may be added. Moreover, the entire mapping process is not automatically constructed, and human factors affect the RDF data constructed by the mapping. Pham et al. proposed a method of constructing RDF Schema based on DTD and XML Schema in paper [8][10], and constructing RDF mapping algorithm based on the constructed RDF Schema. This method is more effective for XML documents constrained by structured documents. And it effectively retains the implicit semantics of the data. The structured document restricts the organization of the data in the XML document by specifying the data types used by the elements and attributes in the XML. The paper analyzes the structured document XML Schema or DTD and obtain the definition of the elements and attributes in the constrained XML document, map them as classes or attributes according to the data types and related mapping rules of the elements and attributes, and then traverse the XML DOM tree according to the constructed classes and attributes, and convert the XML elements and attributes to parts of RDF. But not all XML documents are restricted by DTD or XML Schema. For XML documents with relatively free structure, the above methods are obviously invalid. In addition, in paper [12], Pham et al. believe that elements with different tag names defined in XML Schema and modified by the same data type should describe similar objects. If they are mapped to different classes, the generated RDF data is redundant, so they proposed a method to calculate the semantic similarity of such elements. This similarity calculation is obtained on the basis of calculating the semantic similarity between the child element and the parent element of the related element. If the similarity exceeds a certain value, it is considered that these elements with different tags describe different objects, and they can be mapped to different classes. Paper [11] proposes a more general method. The paper divides XML documents into two types: the first type is without structural restrictions; the second type is bound by XML Schema or DTD. The method proposed in this paper traverse the DOM tree of the first type of XML document, and decide to map them as different components of RDF through the mutual nesting relationship of XML elements and attributes, but due to the element tags in the first type XML document is more flexible, there will be elements with the same tag name and different embedded structures. Elements with the same label should not be mapped to different components of RDF.

Paper [13][14] designed a data semantic model based on XQuery and XPath query languages, and provided a unified semantic model for both XML and RDF data models. Paper [15][16] integrated XML query language XPath into RDF query language SPARQL, the query of data in the XML document is realized in the process of querying RDF triples. These methods are based on the relevant query language to output the query results in the form of RDF. Paper [17][18] is based on the mapping language to map various types of data into RDF triples, which involves the part of building RDF based on XML. This method has better results when XML data is frequently updated, but different mapping requirements require different mapping documents, and the construction of mapping documents is cumbersome.

After comprehensive analysis of the existing methods, we found the following shortcomings: the mapping rules are unreasonable and cannot truly reflect the semantic information contained in XML; the type of XML that can be processed is single, and the method's versatility and portability are poor; it fails to effectively identify equivalence Element, it is impossible to avoid redundancy in the constructed RDF. This article starts with the structure and content of XML, and proposes the automatic construction of RDF based on XML element classification and aggregation.
3. Classification and aggregation of XML elements

3.1. Tree model for parsing XML

Based on the characteristics of XML, a well-formed XML document can be visualized as a tree model. Assuming that \( E \) and \( A \) respectively represent the collection of elements and attributes in XML document, the elements and attributes in XML are divided into three models.

**Definition 1:** There is a type of element in XML where the embedded content is text value, and the starting tag does not contain attributes. Assuming that all such elements in XML document constitute \( E_1 \). We take the text node of the XML document and the corresponding element tags node as a simple sub-model. In XML document, the start tag of some elements contains several attributes, the map of attribute name node—attribute value node is also a simple sub-model.

- Assuming \( N_1 \) is the collection of the element's label names of \( E_1 \), and \( N_1 \) is the collection of the attribute's names of \( A_1 \):
  
  \[
  f_1 : E_1 \rightarrow N_1
  \]
  \[
  f_2 : A_1 \rightarrow N_1
  \]

  - Assuming \( V_1 \) is a collection of text values contained in all simple sub-models.
  - \( s = (n, v) \in (N_1 \cup N_1) \times V_1 = S \) is a mathematical expression of a simple sub-model, assuming that the set of all simple sub-models in XML document is \( S \).

**Definition 2:** There is a type of element embedded with several children-elements, and the start tag of the element can contain several attributes. It is assumed that all such elements in XML document constitute \( E_2 \). The map of element tags node—{children-element modules, attribute modules} is abstracted as complex sub-models.

- Assuming \( N_2 \) is a collection of the label names of all elements in \( E_2 \):
  
  \[
  f_3 : E_2 \rightarrow N_2
  \]

- \( \forall e \in E_2 \), the set of its sub-elements is \( CE_2 \), the set of attributes in start tag is \( EA_2 \), the set of sub-models corresponding to all sub-elements is \( MOD_{e_2} \), and the set of sub-models corresponding to all attributes in \( EA_2 \) is \( MOD_{e_2} \).

- \( c = (n, MOD) \) is a mathematical abstract expression of complex sub-models, the set of all complex sub-models is \( C_n \), \( n \in N_2 \), \( MOD = MOD_{e_2} \cup MOD_{e_2} \).

**Definition 3:** There is a type of element that contains not only a few sub-elements but also text values. At the same time, the start tag of the element can also contain a number of attributes, assuming that all such elements in XML document constitute \( E_3 \). The map of element tags node—{children-element modules, attribute modules, text node} is abstracted as compound sub-model.

- Assuming \( N_3 \) is a collection of the label names of all elements in \( E_3 \):
  
  \[
  f_4 : E_3 \rightarrow N_3
  \]

- \( \forall e \in E_3 \), the set of its sub-elements is \( CE_3 \), the set of attributes in start tag is \( EA_3 \), the text content embedded in the element is \( v_e \), the set of sub-models corresponding to all sub-elements is \( MOD_{e_3} \), and the set of sub-models corresponding to all attributes in \( EA_3 \) is \( MOD_{e_3} \).

- \( m = (n, [MOD, v]) \) is a mathematical abstract expression of compound sub-models, the set of all compound sub-models is \( M_n \), \( n \in N_3 \), \( MOD = MOD_{e_3} \cup MOD_{e_3} \).

3.2. Aggregation class and abstract structure model

The elements in the XML document correspond to different sub-models. The elements with the same tag name are classified and aggregated, and the sub-models of the elements of the same aggregate type are integrated to construct an abstract structure model.

**Definition 4:** Aggregate all elements in XML document into several classes according to the name of the element's start tag. Assume that the set of names is \( N_4 \), the set of elements whose starting tag name of elements are \( n \in N_4 \) (i=1,2,...,n) is aggregation class \( E_4 \).
Definition 5: The set of sub-models corresponding to all elements in aggregate class $E_\alpha$ is defined as $MOD_\alpha$, the sub-model corresponding to $e \in E_\alpha$ may be $mod_\alpha \in (S | C | M)$. Integrate the sub-models contained in $MOD_\alpha$ to obtain abstract structure model $smod_\alpha$. Assume that the abstract structure model of the aggregation class is $smod_\alpha = (n_\alpha, \{CN_\alpha, AN_\alpha, v_\alpha\})$, $v_\alpha$ is a logical variable. There are three specific abstract structure models, as follows:

- If the sub-model of $\forall e \in E_\alpha$ is simple sub-model, and $CN_\alpha$, $AN_\alpha$ does not exist, $v_\alpha$ exists, the abstract model of aggregation class is $smod_\alpha = (n_\alpha, v_\alpha)$, which is called simple abstract structure model.
- If the sub-model of $\forall e \in E_\alpha$ is complex sub-model, $CN_\alpha$ is set of names of sub-elements, and $AN_\alpha$ is set of names of attributes contained in the element's start tag, $v_\alpha$ does not exist. The abstract model of aggregate class is $smod_\alpha = (n_\alpha, \{CN_\alpha, AN_\alpha\})$, which is called complex abstract structure model.
- If the sub-model of $\forall e \in E_\alpha$ is a compound sub-model or there are both simple sub-model and complex sub-model elements, the abstract model of aggregate type is $smod_\alpha = (n_\alpha, \{CN_\alpha, AN_\alpha, v_\alpha\})$, which is called a compound abstract structure model, and $v_\alpha$ exists.

3.3. Aggregation class and abstract structure model

Equivalent elements refer to elements with the same embedded content that belong to the same aggregation class. These equivalent elements may cause data redundancy.

Definition 6: The sub-models of all elements in the current XML document include three types, so there are three types for determining whether the two elements are equivalent:

- When $e$ and $e_i$ are both simple sub-model, the equivalent condition is:
  \[
  \begin{align*}
  n_i &= n_i, \\
  v_i &= v_i 
  \end{align*}
  \]

- When $e$ and $e_i$ are both complex sub-models, the equivalent condition is:
  \[
  \begin{align*}
  n_i &= n_i, \\
  \{e_1, e_2, \ldots, e_n\} &= \{e_1, e_2, \ldots, e_n\}, \\
  \{a_1, a_2, \ldots, a_n\} &= \{a_1, a_2, \ldots, a_n\}
  \end{align*}
  \]

- When $e$ and $e_i$ are both compound sub-models, the equivalent condition is:
  \[
  \begin{align*}
  n_i &= n_i, \\
  v_i &= v_i 
  \end{align*}
  \]

4. Construct RDF mapping rules

4.1. Build aggregation class and its abstract structure model

The aggregation class is the basis for building RDF. This section will introduce the process of obtaining aggregated classes. XML documents can be divided into two categories: the tags of the elements of the first type of XML can be customized by the user, which has poor versatility among different Web applications; the elements and attributes of the second type of XML are restricted by the XML Schema.

(1) Processing of first type XML

The processing has following steps:

I. Traverse the DOM tree of XML, classify the elements according to their names, and get the aggregation class collection $XSD = \{E_\alpha, E_{\alpha_2}, \ldots, E_{\alpha_n}\}$:

\[
\begin{align*}
  f : e(e \in E) &\rightarrow XSD
\end{align*}
\]
II. Aggregate the sub-models of elements belonging to the same aggregation class to obtain the abstract structure model $smod$.

(2) Processing of second type XML

Different from the first type XML, the second type XML is based on XML Schema. XML Schema specifies the elements and attributes used in XML. The steps are as follows:

I. Get collections of various elements in XML Schema, which are embedded sub-elements of the root tag of XML Schema:

$$f : \text{Schema} \rightarrow \text{GXSE}, \text{GXSA}, \text{GSTE}, \text{GCTE}$$  \hspace{1cm} (9)

$\text{GXSE}, \text{GXSA}, \text{GSTE}, \text{GCTE}$ respectively represent the global elements, global attributes, global simple data type collection, and global complex data type collection;

II. Traverse $\text{GXSE}, \text{GXSA}$, get the aggregation class collection $\{\text{Es, Es}, \ldots \}$;

III. Obtain the abstract model of the aggregation class.

4.2. Mapping rules for RDF Schema

The rules for constructing RDF Schema based on the abstract model are as follows:

I. The abstract model of $Es$ is $smod = (n, v_\text{s})$, properties can be constructed based on the name of aggregation class:

$$\begin{align*}
\{ f_o : n \rightarrow pi_s , (n \in N, pi_s \in PI) \\
\text{Type}(pi_s, \text{Property}) \\
\text{PropVal}(\text{range}, ? \text{pi}_s, \text{DataTypeURI})
\end{align*}$$  \hspace{1cm} (10)

$PI$ represents a collection of properties in RDF and $\text{DataTypeURI}$ represents a built-in data type.

II. The abstract model of $Es$ is $smod = (n, \{CN_s, AN_s\})$, class can be constructed based on the name of aggregate class:

$$\begin{align*}
\{ f_o : n \rightarrow ci_o , (n \in N, ci_o \in CI) \\
\text{Type}(ci_o, \text{Class})
\end{align*}$$  \hspace{1cm} (11)

$CI$ represents classes in RDF. Assuming $cn \in CN_s$, properties can be constructed:

$$\begin{align*}
\{ f_o : \{cn_1, \ldots, cn_k\} \rightarrow \{pi_{cn_1}, \ldots, pi_{cn_k}\} \\
\text{Type}(?pi_{cn}, \text{Property}) \\
\text{PropVal}(\text{domain}, ?pi_{cn}, ?ci_o)
\end{align*}$$  \hspace{1cm} (12)

There are two cases for the value range of $pi_i$: when the abstract model is $smod = (cn, v_\text{cn})$, the range of $pi_i$ follows Rule I; when the abstract model is the other two, the range of $pi_i$ is constructed based on the name of $E_n$:

$$\begin{align*}
\{ f_o : cn \rightarrow ci_o , (cn \in CN_s, ci_o \in CI) \\
\text{PropVal}(\text{range}, ?pi, ?ci_o)
\end{align*}$$  \hspace{1cm} (13)

If $an \in AN_s$, there are the following mapping rules:

$$\begin{align*}
\{ f_o : \{an_1, an_2, \ldots, an_k\} \rightarrow \{pi_{an_1}, pi_{an_2}, \ldots, pi_{an_k}\} (an \in AN_s, pi_{an} \in PI) \\
\text{Type}(?pi_{an}, \text{Property}) (k = 1, 2, \ldots, n) \\
\text{PropVal}(\text{domain}, ?pi_{an}, ?ci_o) \\
\text{PropVal}(\text{range}, ?pi_{an}, ?\text{DataTypeURI})
\end{align*}$$  \hspace{1cm} (14)

III. The abstract model of $Es$ is $smod = (n, \{CN_s, AN_s, v_\text{cn}\})$, the rules of constructing classes and attributes in RDF Schema are similar to rule II. For logical variables $v_\text{cn}$, we use the inherent RDF property $\text{rdf: value}$ to describe the relationship between resources and $v_\text{cn}$.

$$\text{Type(value, Property)}$$  \hspace{1cm} (15)

4.3. Mapping rules for RDF triples

The rules and steps for constructing RDF triples based on XML data entities are as follows:
I. Traverse the elements and attributes of XML and attach different ID;
II. Traverse the DOM tree and adjust the elements and attributes to attach the same ID to equivalent elements and attributes;
III. Traverse the adjusted DOM tree and store the mapped elements to OID, the rules of constructing RDF sequence:

◆ If $e_i$ is a simple sub-model, the rules for constructing RDF triples are as follows:

$$f: (n, ID_\alpha) \rightarrow r_\alpha (r_\alpha \in R) $$

$$PropVal(value, ?r_\alpha, ?v)$$

$$Triple(r_\alpha, rdf : value, v)$$

(16)

Among them $R$ is the collection of resources in RDF triples;

◆ If $e_i$ is a complex sub-model, the rules are as follows:

$$f: (n, ID_\alpha) \rightarrow r_\alpha (r_\alpha \in R) $$

$$CE_\alpha, EA_\alpha$$ are respectively the sets of child elements and the set of attributes in the start tag. The name of $e_\alpha \in CE_\alpha$ is $n$. If the element is a simple sub-model, the embedded text value of the element is $v_\alpha$, the rules are as follows:

$$PropVal(?p_\alpha, ?r_\alpha, ?v_\alpha)$$

$$Triple(r_\alpha, p_\alpha, v_\alpha)$$

(18)

If $e_i$ is a complex sub-model or composite sub-model, the construction rules are as follows:

$$f: (n, ID_\alpha) \rightarrow r_\alpha (r_\alpha \in R) $$

$$PropVal(p_\alpha, ?r_\alpha, ?r_\alpha)$$

$$Triple(r_\alpha, p_\alpha, r_\alpha)$$

(19)

◆ If $e_i$ is a compound sub-model, the rules are basically similar to those of Rule II, but because $e_i$ have embedded text values, the mapping of text values is as follows:

$$f: (n, ID_\alpha) \rightarrow r_\alpha (r_\alpha \in R) $$

$$PropVal(value, ?r_\alpha, ?v)$$

$$Triple(r_\alpha, rdf : value, v)$$

(20)

5. Experiment and result analysis

5.1. XML Data Repository

The mapping method proposed in this paper has good effects for both types of XML. This chapter conducts qualitative and quantitative analysis of the effect of the mapping algorithm. To test the effectiveness of the algorithm, we validated it on the XML Data Repository dataset. The XML Data Repository collects publicly available data in XML form and provides statistical information about the dataset. As shown in table 1.

| NO. | Test Data                          | Number of samples | Max-depth | Avg-depth |
|-----|------------------------------------|-------------------|-----------|-----------|
| DS1 | ACM SIGMOD Record                  | 30                | 5         | 3.55671   |
| DS2 | Partially-encrypted treebank       | 20                | 36        | 7.87279   |
| DS3 | Georgetown Protein Information Resource | 30             | 5         | 3.75641   |
| DS4 | ExPASy - SWISS-PROT and TrEMBL     | 30                | 5         | 3.76608   |
| DS5 | Astronomical Data                  | 23                | 8         | 5.58314   |
| DS6 | Courses from Reed College          | 22                | 4         | 3.19979   |
| DS7 | Courses from UWM                   | 20                | 5         | 3.95243   |
| DS8 | Courses from WSU                   | 10                | 4         | 3.15787   |
5.2. Comparison of different algorithm versatility

The mapping algorithm can realize the mapping to RDF for both types of XML documents. Compared with several representative mapping algorithms in paper [8][10], the algorithm in this chapter can handle a wider range of XML data. The algorithm in paper [8][10] only implements mapping for XML data with XML Schema constraints. Figure 1 shows the test Results of several algorithms on the test data in table 1. The results show that the mapping algorithm in this chapter is more versatile.

![Comparison of different algorithms](image)

Figure 1. Comparison of test cases passed by different algorithms

5.3. Comparison of redundancy elimination effect

There are some equivalent elements in XML. The statistics of equivalent elements of data set in table 1 are shown in figure 2. The proportion of equivalent elements refers to the ratio of the XML equivalent element statistics to the total number of elements.

![The Proportion of equivalent elements](image)

Figure 2. The proportion of equivalent elements in the test data set
The equivalent elements in XML cause data redundancy in RDF triples. The mapping algorithm in [11] does not identify the equivalent elements in XML, so the RDF data generated by the mapping algorithm will have redundant data. Use the data set in table 1 to test the mapping algorithm in this paper and the mapping algorithm in [11], and compare the statistical total number of RDF triples generated by different mapping algorithm. It can be found from figure 3 that since the mapping algorithm in this chapter has identified the equivalent elements in XML, the equivalent elements that have been mapped are pruned during the process, data redundancy in RDF is well avoided. The redundancy rate refers to the ratio of the eliminated redundant data statistics to the RDF triples statistics generated by the unredundant mapping.

6. Conclusions
The key to constructing RDF triples required for knowledge engineering based on XML in WEB lies in the construction of mapping rules. Compared with various existing mapping algorithms, this article constructs mapping rules based on the classification and aggregation of XML elements. The mapping rules for mapping elements to RDF triples ensure the completeness and accuracy of the constructed RDF through the mapping algorithm. At the same time, through the identification of duplicate data, it is ensured that redundancy data will not appear in the constructed RDF triples. Finally, the algorithm can realize the process of mapping different types of XML to RDF. The experimental analysis proves that the mapping algorithm has reached corresponding requirements.

Acknowledgments
Authors wishing to acknowledge that this work was supported by General Project of Jiangsu Natural Science Foundation (BK20191274).

References
[1] Berners-Lee T, Hendler J, Lassila O. The Semantic Web: A New Form of Web Content That is Meaningful to Computers Will Unleash a Revolution of New Possibilities[J]. entific American, 2001, 284(5):34-43.
[2] Manola F, Miller E, McBride B. RDF primer[J]. W3C recommendation, 2004, 10(1-107): 6.https://www.w3.org/TR/rdf-primer/.
[3] W3C R. Extensible markup language (XML) 1.0 (Third edition): W3C Recommendation 04 February 2004[J]. http://www.w3.org/TR/2004/REC-xml-20040204/.
[4] Decker S, Melnik S, Harmelen F V, et al. The semantic web: the roles of XML and RDF[J]. IEEE Internet Computing, 2000, 4(5).
[5] Klein M. Interpreting XML via an RDF Schema[C]// ECAI workshop on Semantic Authoring, Annotation & Knowledge Markup (SAAKM 2002). 2002.
[6] Kumar B H, Babu M S P. Study and Constructing RDF model for a well formatted Valid XML document[J]. International Journal on Computer Science & Engineering, 2013, 5(7).

[7] Rodrigues K R , Mello R D S. A Faceted Taxonomy of Semantic Integrity Constraints for the XML Data Model.[C]// International Conference on Database & Expert Systems Applications. Springer-Verlag, 2007.

[8] Thuy P T T, Lee Y K , Lee S , et al. Transforming Valid XML Documents into RDF via RDF Schema[M]. 2007.

[9] Kharrat M , Jedidi A , Gargouri F. A semantic approach for transforming XML data to RDF triples[C]// International Conference on Computer & Information Science. IEEE, 2015.

[10] Thuy P T T, Lee Y K , Lee S, et al. Exploiting XML Schema for Interpreting XML Documents as RDF[C]// IEEE International Conference on Services Computing. IEEE, 2008.

[11] Yang Y , Dai M , Chen H. An Automatic Semantic Extraction Algorithm for XML Document[C]// International Conference on Machine Vision & Human-machine Interface. IEEE, 2010.

[12] Thuy P T T, Lee Y K , Lee S. A Semantic Approach for Transforming XML Data into RDF Ontology[J]. Wireless Personal Communications, 2013, 73(4):1387-1402.

[13] Patel-Schneider P F , Simeon A. The Yin/Yang Web: a unified model for XML syntax and RDF semantics[J]. IEEE Transactions on Knowledge & Data Engineering, 2003, 15(4):797-812.

[14] Patel-Schneider P. The Yin/Yang web: XML syntax and RDF semantics[C]// International world wide web conference WWW2002. Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ, USA, pfps@research.bell-labs.com; Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ, USA, simeon@research.bell-labs.com;, 2002.

[15] Droop M, Flarer M, Groppe J, et al. Bringing the XML and semantic web worlds closer: transforming XML into RDF and embedding XPath into SPARQL[C]//International Conference on Enterprise Information Systems. Springer, Berlin, Heidelberg, 2008: 31-45.

[16] Bischof S, Decker S, Krennwallner T, et al. Mapping between RDF and XML with XSPARQL[J]. Journal on Data Semantics, 2012, 1(3): 147-185.

[17] Dimou A, Vander Sande M, Slepicka J, et al. Mapping hierarchical sources into RDF using the RML mapping language[C]//2014 IEEE International Conference on Semantic Computing. IEEE, 2014: 151-158.

[18] Michel F, Djimeno L, Zucker C F, et al. xR2RML: Relational and non-relational databases to RDF mapping language[J]. 2017.

[19] Breitling F. A standard transformation from XML to RDF via XSLT[J]. Astronomische Nachrichten: Astronomical Notes, 2009, 330(7): 755-760.

[20] Prud'hommeaux E. SPARQL query language for RDF, W3C recommendation[J]. http://www.w3.org/TR/rdf-sparql-query/.

[21] Ma Z , Zhao Z, Yan L. Heterogeneous fuzzy XML data integration based on structural and semantic similarities[J]. Fuzzy Sets and Systems, 2018, 351(NOV.15): 64-89.

[22] Yang D , Yan L. Transforming XML to RDF(S) with Temporal Information[J]. Journal of Computing and Information Technology, 2018, 26(2): 115-129.