A Framework of Summarizing XML Documents with Schemas

Teng Lv[1] and Ping Yan[2]
[1]Teaching and Research Section of Computer, Army Officer Academy, China
[2]School of Science, Anhui Agricultural University, China

Abstract: *eXtensible Markup Language (XML) has become one of the de facto standards of data exchange and representation in many applications. An XML document is usually too complex and large to understand and use for a human being. A summarized XML document of the original document is useful in such cases. Three standards are given to evaluate the final summarized XML document: document size, information content, and information importance. A framework of summarizing an XML document based both on the document itself and the schema is given, which applies schema to summarize XML documents because there are many important semantic and structural information implied by the schema. In our framework, redundant data are first removed by abnormal functional dependencies and schema structure. Then tags and values of the XML document are summarized based on the document itself and schema. Our framework is a semi-automatic approach which can help users to summarize an XML document in the sense that some parameters must be specified by the users. Experiments show that the framework can make the summarized XML document has a good balance of document size, information content, and information importance comparing with the original one.*

Keywords: *XML, document summarization, schema, key, functional dependency.*

Received June 14, 2010; accepted March 1, 2011

1. Introduction

eXtensible Markup Language (XML) [15] has become one of the de facto standards of data representation over World Wide Web and elsewhere. More and more data are stored in XML format. To understand these XML documents with complex structure and abundant data, a human being must spend much time to read such documents. In some cases, it is impracticable, even not impossible, for a human being to read the whole XML document when the document is very large and complex. So it is necessary to present a human being with a summarized form of the original complex and large XML document. Such a summarized XML document is also useful in other applications: querying XML documents, comparing two XML documents, displaying or storing XML documents in a mobile or embedded device which has limited CPU processing ability, display screen and storage spaces, etc. Although, such a summarized XML document is very useful, it is difficult to generate a good summarized XML document. Although a human being has good ability of summarizing and analyzing, a computer is not good at doing such things. So the challenge of summarizing XML documents is how to generate such a summarized XML document by computers.

A summarized XML document should grasp the core information of the original document so that a human being can have a basic understanding of the original document. Of course, such a summarized XML document should have less size than the original document considering the storage space and complexity. A good summarized XML document can be evaluated by the following 3 standards:

1. *Document Size:* The first goal of summarizing XML document is to obtain an XML document with an acceptable size comparing with the original one according to specific applications. In general, an XML document of smaller size is more readable than a larger one for a human being. Suppose the sizes of the original and the summarized XML document are $S_{original}$ bytes and $S_{summarized}$ bytes, respectively, so the summarized ratio of a summarized XML document respect to the original one is $R_S=S_{summarized}/S_{original}$.
2. *Information Content:* A perfect summarized XML document should contain the entire information content of the original one, i.e., it is equivalent to the original one in the aspect of information content. But in reality, it is impossible for a summarized XML document with less size to contain the entire information content of the original document which has no redundant information. Suppose the number of tag values of the original XML document without redundant information and the summarized XML document are $C_{original}$ and $C_{summarized}$, respectively, so the information content ratio of a summarized XML document respect to the original one is $R_C=C_{summarized}/C_{original}$. For original XML document with redundant information, we can remove the

redundant information by methods proposed in section 2. It is obviously to see that an original XML document D_1 with redundant information is equivalent to an XML document D_2 with respect to information content if D_2 is obtained by removing the redundant information of D_1. Although it is difficult to generate a perfect summarized XML document, a good summarized XML document should contain more information in given size than a bad one.

3. *Information Importance:* As a summarized XML document can not contain the entire information content of the original one in most cases, it is necessary and practicable to contain the most important information of the original one. For how to determine which information is important to an XML document, we will give some guidelines in sections 3 and 4. Dalamagas T and alt .., [5] gives a measure of quality of XML schema document.

1.1. Related Works

Text summarization [1, 8] focused on free-flowing texts in text datasets, which is not always applicable to XML summarization as the structure information and semantic information are often important to XML. XML schema summarization [17] is one related topic which summarizes XML schemas rather than XML documents. XML structure summarization [3] is another related topic which summarizes XML structures rather than XML documents. Compression technique [9, 11] is another related topic to reduce the document size without considering the readability to human beings. Other works focused on constructing XML summarization for XML efficient query estimation: StatiX [7] explores schema transformation and schema validation to obtain statistics for query selectivity estimation in XML documents. Treesketch synopses [13] can produce fast, accurate approximate answers for XML documents. Bloom histogram [16] is a framework for XML path selectivity estimation in a dynamic environment. Xseed [18] is a method to estimate cardinality of Xpath queries. Mayorga V. and Polyzotis N., [12] proposed a method to summarize XML data streams other than XML documents. A semi-automatic method to summarize XML collections is proposed in [6], which applies a template to specify the user requirement and matching rules to extract the summarized XML collections.

To the best of our knowledge, the most related work is [14] which proposed a method of XML document summarization based on document itself alone. As we know that a schema defines the structure and semantic of an XML document which implies many important information of an XML document. From the former observation, we propose a method of summarizing an XML document based on both document itself and the schema. Such method can make the summarized XML document have a good balance of document size, information content and information importance. More specifically, our approach have the advantages over [14] in the following aspects:

1. Some data redundancies introduce by abnormal functional dependencies and nested structures are first removed in the summarizing process which can make the XML document concise and meaningful section 2.
2. It can reserve key information of the XML document section 3.1
3. It can deal with the difficult situation when a tag occurs many times but with little importance in an XML document section 3.2
4. It can deal with another difficult situation when same tag occurs many times under the same parent tag in an XML document section 4.1

1.2. Contributions

In this paper, we give a framework of summarizing an XML document based on the document itself and the schema of the document. We apply schemas to summarize XML documents because there are many important semantic and structural information implied by the schemas including functional dependencies, keys, and structure information. Our framework can help users to summarize an XML document in the sense that some parameters must be specified by the users according to the different requirement such as intended summarized document size, intended information content, intended information importance, and specific XML documents. The framework is worked as follows:

1. First, abnormal functional dependencies and nested structure of schema are used to remove redundant data of the XML document. After this preprocess, the XML document contain no redundant data caused by abnormal functional dependencies and structure.
2. Second, we summarize tags in an XML document. Keys (with values) of the document are preserved in the summarized document as keys are important to understand and query the document in most cases. Other tags are determined by their occurrences in the context of document or a set of documents with same schema.
3. Finally, we summarize tag values in the XML document. For a same tag with multiple values, only the first tag value is reserved in the summarized XML document. For long tag values, we truncate them to a fixed length according to specific applications.

   Preliminary experiments show that the summarized XML document has a good balance of the above three standards (i.e., document size, information content, and information importance) comparing

with the original document. Another advantage is that such method is more useful as the summarized XML document can contain some important information such as keys and functional dependencies.

1.3. Organization

The rest of the paper is organized as following: section 2 illustrate the first two steps to summarize an XML document by removing data redundancies according to functional dependencies and structure. Sections 3 and 4 are the third and fourth steps to summarize an XML document by summarizing tags and values in the XML document. The entire framework of summarizing an XML document is given in section 5 which applies the above 4 steps. Experiments and discussions are also given in the section. We conclude the paper and give the future work in section 6.

2. Removing XML Data Redundancies by Schema

In real XML documents, there are many data redundancies as the causal design of XML schemas and documents. We focus on two kinds of XML data redundancies: data redundancies caused by functional dependencies and data redundancies caused by structure.

2.1. Removing XML Data Redundancies by Functional Dependencies

The first kind of XML data redundancies is caused by abnormal XML functional dependencies proposed in our previous work [10]. We do not give the formal definitions of functional dependencies and normal forms here considering the space. Detailed descriptions can be referred to [10]. In this paper, we focus on the third normal form of XML document considering the simplicity and applicability in real applications. Higher normal forms may be too complex to understand and apply in real XML documents. The details of removing data redundancies in an XML document by functional dependencies will be given in section 5.1.

2.2. Removing XML Data Redundancies by Structure

The second kind of XML data redundancies is caused by the nested structure of XML documents. Considering the following XML document D_4 which contains a nesting tag orders:

```
<customerorders>
 <orders>
   <orders>
     <order>
     <orderID>10643</orderID>
     <customerID>9232</customerID>
     <order date>2010-03-25</order date>
       </order>
       </orders>
 <order>
     <order ID>10692</order ID>
     <customer ID>9349</customer ID>
     <order date>2009-10-03</order date>
   </order>
     <company name>A futterkiste</company name>
   </orders>
</customer orders>
```

To reduce such redundancies, the tags in sub-tree rooted on the nested tag Orders (i.e., the second Orders tag here) are moved up as sub-tags of the nesting tag A (i.e., the first Orders tag). Of course, the nested tag Orders is removed in the moving-up process. The above XML document D_4 can be transformed to the following XML document D_5 by above method.

```
<customerorders>
  <orders>
     <order>
      <orderID>10643</orderID>
      <customerID>9232</customerID>
      <orderdate>2010-03-25</orderdate>
  </order>
  <order>
     <orderID>10692</orderID>
     <customerID>9349</customerID>
     <orderdate>2009-10-03</orderdate>
  </order>
     <companyname>A futterkiste</companyname>
  </orders>
</customerorders>
```

The detailed method to do this will be given in section 5.2.

3. Summarizing Tags in An XML Document

After the previous summarizing procedures, an XML document has no data redundancies caused by abnormal functional dependencies and nested structures proposed in sections 2. In this section, we will focus on the problem of determining the important tags in the original XML document to be included in the summarized XML document.

3.1. Keys of XML Documents

As keys [2] are important for querying and understanding an XML document (an example is given in section 5.5), the keys have priority over other tags to be contained in the summarized XML document. As how to deal with other tags that are not a key or a part of a key, i.e., whether or not they are contained in the summarized XML document, we will propose the method in section 3.2. For example, considering the following XML document D_6.

```
<book>
   <title>database</title>
   <author>Peter Lee</author>
   <price>5USD</price>
      ... ...
</book>
```

Suppose {title, author} is a key (a book is uniquely determined by the combination of its title and author), the summarized XML document must contain the key information of a book: title and author. This is also a common sense that the book title and its author is the priority information when we browse a book.

3.2. Other Tags

For a tag which is not a key or a component part of a key, we can determine the tag importance by its occurrence in the XML document. In general, a tag A is more important than another tag B if A's occurrence is higher than that of B. But in some special cases, it is not always true. Consider the following XML document D_7:

```
<book>
   <title>database </title>
   <author>Peter Lee</author>
   <comment>This book introduces the basic concepts of...
</comment>
   <comment>Normal forms are discussed ...</comment>
   <comment>Query optimization is ...</comment>
   <comment>The author publish ...</comment>
</book>
```

If we just consider tag importance in the context of the given single XML document D_7, tag *comment* is more important than tags *title* and *author*, because tag *comment* occurs 4 times while each of tags *title* and *author* only occurs once in the above XML document D_7. To deal with this situation, we must expand the context to determine the tag importance, i.e., we must use a set of XML documents with same schema to determine the tag importance other than the single XML document itself. Although we extend the context of tag importance, our work is still focused on how to summarize a single XML document. Suppose we can get a statistics from a set of XML documents (e.g., 100 XML documents) conforming to the same schema as Table 1:

Table 1. A Statistics of tag occurrences and importance.

Tags	Occurrences of Tags	Tag Importance
Book	100	1
Title	100	1
Author	100	1
Comment	30	2

From the statistics of Table 1, tag importance of tags book, title, and author is higher than that of tag comment according to the occurrences of the 4 tags. The detailed method of summarizing tags in XML documents is given in section 5.3.

4. Summarizing the Values of Tags in XML Documents

After the previous summarizing procedures in sections 2, 3, the summarized XML document contains no redundancies proposed in section 2 and only the important tags with corresponding values of the original document are preserved in the summarized XML document. In this section, we will focus on summarizing the values of the tags in the XML document.

4.1. Summarizing the Values of Tags in XML Documents

If the tags with same tag name have multiple values and all the tags have a same parent tag in the XML document, we just include the first tag with its value in the summarized document and a mark is left to indicate that there is more information about the same tag. If the reader is interested in that information, he or she can unfold the mark to browse the information when necessary. Of course, this will not increase the size of summarization, but only increase the process time when unfold the mark to browse the hide information. The intuitive motivation of this treatment is that the information of the first tag and its value is more important than subsequent tags with same tag names as the first tag and their values under the same parent tag in general. For example, the first author is more interested than the co-authors of a book for a reader in general. If it is not the case in extreme situations, the reader can browse the interested information when necessary. For example, a book has four authors in an XML document D_8 as shown in following:

```
<book>
   <title>database</title>
   <author>author1</author>
   <author>author2</author>
   <author>author3</author>
   <author>author4</author>
</book>
```

We just contain the first author information in the summarized XML document as following XML document D_9, where the mark ○,m indicates that there is more information about authors of the book:

```
<book>
   <title>Database</title>
   <author>author1</author>○,m
</book>
```

4.2. Treating the Length of Tag Values

For different tags, tag values may vary greatly in length. Some tag values just contain several characters "short values" and others may contain thousands of characters "long values". It is obviously that a summarized XML document should not contain such

long values without any change because it occupies too much space comparing to its provided simple information. To deal with the long values, it is sufficient that a summarized XML document contains some fixed length of the characters of the whole long values with a mark left to indicate that there is more information about the same tag. Considering the following XML document D_{10}:

```
<book>
  <author>Peter Lee<authors/>
  <comment> I found it to be a very useful text-book.The concepts are easy to understand and the authors provide plenty of examples for better understanding. The book is detailed, which means that if you want to go into detail you can, e.g., on what normal form is and how to use it, of the difference between relational and ... </comment>
</book>
```

The length of *author* value is 9 characters ("Peter Lee"), but the length of *comment* value is thousands of characters. We can just choose a fixed length, for example 41 characters, to be include in the summarized XML document with the mark ○,m indicates that there is more information about comment of the book. The final summarized XML document D_{11} is following:

```
<book>
   <author>Peter Lee<authors/>
   <comment> I found it to be a very useful text-book.○,m
   </comment>
</book>
```

The problem of the above method is how to decide whether a tag value is a long value or not, and if it is a long value, how to decide the appropriate length of a long value to be contained in a summarized XML document. One solution is to analyze the XML schema and document to get a statistics about the length of all tag values. Then uses can specify a fixed length to determine whether a tag value is long or not and another fixed length to determine how many characters of the long tag value should be contained in the final summarized XML document according to the specific needs such as intended document size, intended document information, etc.,. The detailed method of summarizing XML tag values is given in section 5.4.

5. A Framework of Summarizing XML Documents

Based on the above discussion of sections 2, 3 and 4, we give the following framework of summarizing an XML document. Summarizing_XML_Document (D, S, FD, K, I, L, M):

- *Input*: An XML schema S, an XML document D conforming to S, a set of keys K and functional dependencies FD of D and S, and three constants I, L, and M (M≤L), which determines the threshold values of tag importance, long tag value, and how many characters a long tag value should be contained in the summarized XML document, respectively.
- *Output:* A summarized XML document D_4 of D.
- *Method:*

  1. (D_1, S_1, FD_1, K_1) = removing_data_ redundancies_ FD (D, S, FD, K).
  2. (D_2, S_2, FD_2, K_2) = removing_data_ redundancies_ structure(D_1, S_1, FD_1, K_1).
  3. (D_3, S_3, FD_3, K_3)=summarizing_tags(D_2, S_2, FD_2, K_2, I).
  4. D_4 = summarizing_values(D_3, S_3, L, M).

The framework applies 4 steps in turn to summarize an XML document: Step 1 removes data redundancies by functional dependencies; Step 2 removes data redundancies by schema structure; Step 3 summarizes the tags in the XML document; and Step 4 summarizes the tag values in the XML document. We will elaborate the details of the 4 steps in the following sections 5.1, 5.2, 5.3 and 5.4.

5.1. Function Removing Data Redundancies FD

Function removing_data_redundancies_FD is to remove data redundancies by Functional Dependencies (FDs) of the XML document and the schema, which is similar to previous work proposed in [10]. We just give the outline of the method here with a brief explanation. Rule 1 is to remove data redundancies by moving up the sub-tree regarding to Partial Functional Dependencies (PFD) [10] and rule 2 is to remove data redundancies by creating new elements regarding to Transitional Functional Dependencies (TFD) [10]. After applying rules 1 and 2, the XML document, the schema, the FDs and keys of the XML document and schema must be reformed as the structure and tags are changed in the process. More details about the 2 rules and how to change the forms of the XML document, the schema, the set of FDs and keys can be found in [10].

- *Function 1*: (D_1, S_1, FD_1, K_1)=removing_data_ redundancies_FD (D, S, FD, K).
- *Input:* An XML schema S, an XML document D conforming to S, and a set of functional dependencies FD and keys K of D and S.
- *Output:* An XML schema S_1 and an XML document D_1 conforming to S_1, and a set of functional dependencies FD_1 and keys K_1 of D_1 and S_1.
- *Method:*

  1. Apply rule 1 sub-tree moving up transformation.
  2. Apply rule 2 creating a new element transformation.
  3. Rewrite D, S, FD, and K as D_2, S_2, FD_2, and K_2 respectively.

5.2. Function Removing Data Redundancies Structure

Function removing_data_redundancies_structure is to remove data redundancies by XML structures of the XML document and schema. The function traverses the tree using pre-order traversal to detect tags which have an ancestor with the same tag name. It reduces the nested tags in the XML document. Other tags in the sub-tree rooted on a nested tag are moved up as sub-tags of the nesting tag which is the ancestor of the nested tag.

- *Function 2:* (D_2, S_2, FD_2, K_2)=Removing_data_redundancies_Structure(D_1, S_1, FD_1, K_1).
- *Input:* An XML schema S_1, an XML document D_1 conforming to S_1, and a set of functional dependencies $FD_1$and keys K_1 of D_1 and S_1.
- *Output:* An XML schema S_2, an XML document D_2 conforming to S_2, and a set of functional dependencies FD_2 and keys K_2 of D_2 and S_2.
- *Method:*

  1. For each tag n, traverse the path from n to the root tag r, if there exists a tag such that ancestor(n)=n, then move up the tags of sub-tree rooted on n (n itself is not included) as sub-tags of ancestor(n). //ancestor(n) is an ancestor tag of n.
  2. Rewrite D1, S1, FD1, and K1 as D2, S2, FD2, and K2 respectively. //After moving up the sub-tree, the XML document, the schema, the functional dependencies and keys of the XML document and schema must be reformed as the structure and tags are changed in the process.

5.3. Function Summarizing_Tags

Function summarizing_tags is to summarize the tags in an XML document. The function traverses the XML document and contains three kinds of tags in the summarized XML document:

1. The tags with values when available which are keys or a component part of a key.
2. Tags with values when available whose importance are equal to or higher than the specified threshold.
3. Tags with values when available which are the ancestors of the first two kinds of tags. Ancestor tags are included in the summarized XML documents because we must guarantee that the relative tags to be connected each other.

- *Function 3:* (D_3, S_3, FD_3,K_3)=summarizing_tags(D_2, S_2, FD_2, K_2, I).
- *Input:* An XML schema S_2, an XML document D_2 conforming to S_2, a set of functional dependencies FD_2 and keys K_2 of D_2 and S_2, a constant "I" indicates that only the tags whose tag importance is equal to or greater than "I" are included in the summarized XML document D_3.
- *Output:* An XML schema S_3, an XML document D_3 conforming to S_3, and a set of functional dependencies FD_3 and keys K_3 of D_3 and S_3.
- *Method:* Traverse D_3 from leaf node to the root:

  1. For each tag *n*, it *n* is a key or a component part of a key, then *n* (with *value*(*n*)) and *ancestor_set*(*m*) (with *value*(*ancestor_set*(*n*)) when available) are included in D_3. //*value*(*n*) is the value of tag *n*, *ancestor_set*(*n*) is the set of ancestor tags of *n*.
  2. For other tag *m*, if *Importance* (*m*)≥I, then *m* (with *value*(*m*) when available) and *ancestor_set*(*m*) (with *value*(*ancestor_set*(*m*)) when available) are included in D_3. //*Importance*(*m*) is the function to calculate a tag importance in the context of XML document or a set of XML documents with the same schema.
  3. Rewrite D_2, S_2, FD_2, and K_2 as D_3, S_3, FD_3, and K_3 respectively. //The document structure is changed as some tags are not included in the summarized document D_3. It is necessary to rewrite the corresponding items.

5.4. Function Summarizing_Values

Function summarizing_values is to summarize the tag values (i.e., leaf nodes) of an XML document. For each tag value, the function verifies that there is only one occurrence of the tag under the same parent tag. If it is not so, only the first tag vale is contained in the summarized XML document. Then for each long tag value (the length of a tag value is longer than a specified length), only the specified length of characters is contained in the summarized XML document.

- *Function 4:* D_4=summarizing_values(D_3, S_3, L, M)
- *Input:* An XML schema S_3, an XML document D_3 conforming to S_3, and two constants "L" and "M" (M≤L), which determines whether a tag value is long or not and how many characters a long tag value should be contained in the summarized XML document, respectively.
- *Output:* A summarized XML document D_4 of D_3.
- *Method:*

  1. For each tag *n*, if *child_set*(*n*)$\supseteq m^+$ and *child*(*m*)= NULL, then *value*(*first*(*m*, *child_set*(*n*))) is included in D_4. //*child_set*(*n*) indicates the set of child tags of *n*, m^+ means there are one or more *m* tags, NULL means that there is no child tag, i.e., *m* is leaf node, and *first*(*m*, *child_set*(*n*)) is the first *m* tag of *child_set*(*n*).
  2. For each value *v*=*value* (*n*), if *length* (*v*)>L, then *value* (*n*)=M-truncation(*v*, M). //value of *n* is a long tag value and is truncated to length M.

5.5. Experiments

A preliminary test is performed to evaluate the framework. We choose XML document D_1 in [10] and DBLP [4] records as tested XML documents. The summarized XML documents are evaluated by 100 evaluators, where 50 evaluators have computer research background and another 50 do not have such background. Some interesting and useful results are given in following:

1. *Data redundancies:* Considering an XML document D_{12} with data redundancies as following:

```
<courses>
  <course cno=”c10”>
    <title>db</title>
    <takenby>
      <student sno=”s10”>
        <sname>Joe</sname>
        <teacher tno=”t10”>
          <tname>John</tname>
        </teacher>
      </student>
      <student sno=”s20”>
        <sname>Smith</sname>
        <teacher tno=”t10”>
          <tname>John</tname>
        </teacher>
      </student>
    </takenby>
  </course>
  <course cno=”c20”>
    <title>at</title>
    <takenby>
      <student sno=”s20”>
        <sname>Smith</sname>
        <teacher tno=”t10”>
          <tname>John</tname>
        </teacher>
      </student>
      <student sno=”s30”>
        <sname>Jane</sname>
        <teacher tno=”t10”>
          <tname>John</tname>
        </teacher>
      </student>
    </takenby>
  </course>
  <course cno=”c30”>
    <title>os</title>
    <takenby>
      <student sno=”s10”>
        <sname>Joe</sname>
        <teacher tno=”t20”>
          <tname>Mary</tname>
        </teacher>
      </student>
      <student sno=”s30”>
        <sname>Smith</sname>
        <teacher tno=”t20”>
          <tname>Mary</tname>
        </teacher>
      </student>
    </takenby>
  </course>
</courses>
```

we apply function 1 to get a summarized XML document D_{13} as follwong:

```
<courses>
  <course cno="c10">
      <title>db</title>
      <takenby>
        <student sno="s10">
          <sname>Joe</sname>
        </student>
        <student sno="s20">
          <sname>Smith</sname>
        </student>
      </takenby>
      <teacher tno="t10"></teacher>
</course>
    <course cno="c20">
      <title>at</title>
      <takenby>
        <student sno="s20">
          <sname>Smith</sname>
        </student>
        <student sno="s30">
          <sname>Jane</sname>
        </student>
      </takenby>
      <teacher tno="t10"></teacher>
    </course>
    <course cno="c30">
      <title>os</title>
      <takenby>
        <student sno="s10">
          <sname>Joe</sname>
        </student>
        <student sno="s30">
          <sname>Jane</sname>
        </student>
      </takenby>
      <teacher tno="t20"></teacher>
    </course>
    <teacherinfo tno="t10">
      <tname>John</tname>
    </teacherinfo>
    <teacherinfo tno="t20">
      <tname>Mary</tname>
    </teacherinfo>
</courses>
```

we analyze the summarized XML document according to the proposed 3 standards:

a. Size of the summarized XML document. The size of XML document D_{12} is 893 bytes and the summarized XML document D_{13} is 781 bytes which is about 88% of that of D_1 (i.e., the summarized ration R_S=88%). So removing data redundancies alone can get a smaller summarized XML document, which is just the first step of summarizing an XML document in our approach.
b. Information content of the summarized XML document. As this step only removes data redundancies in the original XML document, the information content of the summarized XML document is unchanged comparing with the original one.
c. Information importance of the summarized XML document. Another advantage is that removing data redundancies can make the summarized XML document more readable because the document is re-organized in structure according to the implied semantic (functional dependency) when data redundancies are removed, which can be seen from D_{12} and D_{13}. This point is agreed by all the 100 evaluators. For removing data redundancies by schema structure, it has the similar effects as

removing data redundancies by functional dependencies and we do not give the analysis anymore.

2. *Multiple Occurring Tags and Long Tag Values*: For a tag occurring many times in a DBLP record, the size of summarized XML file can be very small comparing with the original one. For example, consider the following DBLP record of all publications of an author whose name is "Moira C. Norrie" (http://dblp.uni-trier.de/rec/pers/n/Norrie:Moira_C=/xk):

```
<dblpperson name="Moira C. Norrie">
   <dblpkey type ="personrecord"> homepages/n/
                  MoiraCNorrie
   </dblpkey>
   <dblpkey>conf/chi/WeibelISN08</dblpkey>
   <dblpkey>conf/cscw/IgnatPON08</dblpkey>
    ... ...
</dblpperson>
```

we analyze the summarized XML document according to the proposed 3 standards:

a. Size of the summarized DBLP record. The size of the original DBLP record is 4535 bytes. As tag *dblpkey* occurs multiple times in the record, the summarized DBLP record is only 185 bytes which is approximately 4% of the original one (i.e., the summarized ration R_S=4%).
b. Information content of the summarized DBLP record. The information content ratio of the summarized XML document respect to the original one is R_C=3%. Although only the first *dblpkey* tag is reserved in the summarized record, our approach provides a method to expand and browse other interested *dblpkey* information when necessary to resolve this problem.
c. Information importance of the summarized DBLP record. 90 (44 with computer research background and 46 without computer research background) out of 100 evaluators are basically agreed to our approach, i.e., only the first *dblpkey* tag is reserved in the summarized record. This means that most of the evaluators think that the summarized DBLP record contains the most interested and necessary information to them. The method of processing long tag values is similar to the above method in the framework, so it is also acceptable and reasonable.

3. *XML Key*. It is important and useful to understand the original XML document for a human being when key information reserved in the summarized XML document. It is one major difference between our approach and other related work. Most of the 50 evaluators with computer research background agreed that key information is helpful to understand DBLP records. Most of another 50 evaluators without computer research background also agreed that key information is helpful to them to understand DBLP records when we explain the meaning of the key to them. For example, consider the following DBLP record with key information (http://dblp.uni-trier.de/rec/bibtex/conf/er/Norrie08.xml):

```
<dblp>
    <inproceedings key="conf/er/Norrie08"
    mdate="2008-10-20">
     0<author>Moira C. Norrie</author>
          <title>PIM Meets Web 2.0.</title>
          <pages>15-25</pages>
          <year>2008</year>
          <booktitle>ER</booktitle>
          <ee>http://dx.doi.org/10.1007/978-3-540-
                87877-3_3</ee>
          <crossref>conf/er/2008</crossref>
         <url>db/conf/er/er2008.html#Norrie08</url>
         </inproceedings>
</dblp>
```

The key value of *inproceedings* "conf/er/Norrie08" indicates that the paper is a "conference" paper appearing in "ER 2008" and the author is "Norrie". Of course, if the evaluator is not familiar to related computer science or the original DBLP record is inconsistency or inaccurate, key information may be not much helpful as that in this situation. But this exception is resolved by other methods such as adjusting or correcting key and key values. It is another research topic beyond the current research of the paper.

5.6. More Discussions of the Framework

More discussions of the framework are given in the following aspects:

1. A balance of the three proposed standards. From above observation, we can see that the framework can obtain a good balance of document size, information content, and information importance, which are the three standards proposed in section 1 to evaluate a "good" summarized XML document. As the three evaluation standards are contradictory and inconsistency in most cases, it is impossible to get a perfect summarized XML document to satisfy all the three standards. So it is also acceptable if we can obtain a relative good summarized XML document by our method.
2. Other useful information. The framework can output more useful information about the summarized XML document D_4. This can be obtained by outputting some results of function 3 Summarizing_Tags, which includes: a schema S_3, a set of functional dependencies FD_3 and a set of keys K_3 of the summarized XML document. The information is useful to understand, query, and re design the XML document. The above characteristic is one of advantages of our approach.

6. Conclusions and Future Work

This paper proposed a semi-automatic framework to help users summarize XML documents, which means

that some parameters of the method must be specified by uses according to different requirements such as intended summarized document size, intended information content, intended information importance, and specific XML documents. The contribution of the method is that the summarized XML document can get a good balance of document size, information content, and information importance of the original document. Another contribution is that the summarized XML document can help a human being to understand the original large and complex document well and present a clear and core information of the original one to the user. The future work should be done on the following:

1. *Tag importance:* Although we propose the method to determine tag importance by keys and tag occurrences in the context of XML document or a set of XML documents with the same schema, there are still some cases that the methods proposed here can not deal with. For example, if there is only one single XML document and tag importance can not be determined well in the context of XML document, we must turn to the help of human beings, semantic ontology, or artificial intelligence. That is beyond the current scope of the paper and is an interesting work to be done in future.
2. *To generate a summarized XML document from a summarized XML schema:* If we have already had a summarized XML schema, than an interesting work is how to generate a summarized XML document based on such schema. It is another approach to summarizing XML document. The technique used in such case maybe different to the method proposed here and is a challenge in future work.

Acknowledgements

The work is supported by Natural Science Foundation of Anhui Province (No.1208085MF110), National Natural Science Foundation of China (No. 11201002), and Foundation of Introduction of Talents of Anhui Agricultural University (No.YJ201012).

References

[1] Amini M., Tombros A., Usunier N., and Lalmas M., "Learning-Based Summarisation of XML Documents," *Information Retrieval*, vol. 10, no. 3, pp. 233-255, 2007.

[2] Buneman P., Davidson S., Fan W., Hara C., and Tan W., "Keys for XML," *Computer Networks*, vol. 39, no. 5, pp. 473-487, 2002.

[3] Dalamagas T., Cheng T., Winkel K., and Sellis T., "A Methodology for Clustering XML Documents by Structure," *Information Systems*, vol. 31, no. 3, pp. 187-228, 2006.

[4] DBLP, available at: http://dblp.uni-trier.de/xml, last visited 2011.

[5] Dilek B. and Sanjay M., "Entropy as a Measure of Quality of XML Schema Document," *The International Arab Journal of Information Technology*, vol. 8, no. 1, pp. 75-83, 2011.

[6] Fischer G. and Campista I., "A Template-Based Approach to Summarize XML Collections," *in Proceedings of Lernen, Wissensentdeckung and Adaptivit* , Germany, pp. 103-108, 2005.

[7] Freire J., Haritsa J., Ramanath M., and Simon J., "StatiX: Making XML Count," *in Proceedings of the International Conference on Management of Data*, USA, pp. 181-191, 2002.

[8] Hahn U. and Mani I., "The Challenges of Automatic Summarization," *Journal of Computer*, vol. 33, no. 11, pp. 29-36, 2000.

[9] League C. and Eng K., "Type-Based Compression of XML Data," *in Proceedings of Data Compression Conference*, USA, pp. 273-282, 2007.

[10] Lv T., Gu N., and Yan P., "Normal forms for XML Documents," *Information and Software Technology*, vol. 46, no. 12, pp. 839-846, 2004.

[11] Maneth S., Mihaylov N., and Sakr S., "XML Tree Structure Compression," *in Proceedings of the 3rd International Workshop on XML Data Management Tools and Techniques*, Italy, pp. 243-247, 2008.

[12] Mayorga V. and Polyzotis N., "Sketch-Based Summarization of Ordered XML Streams," *in Proceedings of IEEE 25th International Conference on ICDE*, China, pp. 541-552, 2009.

[13] Polyzotis N., Garofalakis M., and Ioannidis Y., "Approximate XML Query Answers.," *in Proceedings of SIGMOD International Conference on Management of Data*, France, pp. 263-274, 2004.

[14] Ramanath M. and Kumar K., "A Rank-Rewrite Framework for Summarizing XML Documents," *in Proceedings of 2nd International Workshop on Ranking in Databases, ICDE Workshop*, México, pp. 540-547, 2008.

[15] W3C, "Extensible Markup Language," available at: http://www.w3.org/XML/, last visited 2011.

[16] Wang W., Jiang H., Lu H., and Yu J., "Bloom Histogram: Path Selectivity Estimation for XML Data with Updates," in Proceedings of the 30th International Conference on Very Large Data Bases VLDB, Canada, pp. 240-251, 2004.

[17] Yu C. and Jagadish H., "Schema Summarization," *in Proceedings of the 32nd International Conference on Very Large Data Bases VLDB*, Korea, pp. 319-330, 2006.

[18] Zhang N., Ozsu T., Aboulnaga A., and Ilyas I., "Xseed: Accurate and Fast Cardinality Estimation for XPath Queries," *in Proceedings of the 2nd International Conference on ICDE*, USA, pp. 61, 2006.

Teng Lv received his PhD degree from Fudan University, China. His research interests include database and XML data management. He is the author or coauthor of more than 50 journal papers or reviewed conference papers. He is the reviewers or PC members of several journals and conferences both at home and abroad.

Ping Yan received her PhD degree from Fudan University, China. Her research interests include partial differential equations and their applications in neural network and epidemic diseases, databases, and XML data management.