SqueezeX: Synthesis and Compression of XML Data

Mario Cannataro  
ISI Institute  
National Research Council  
Via P. Bucci, 41/C  
Rende, Italy  
cannataro@isi.cs.cnr.it

Carmela Comito  
DEIS Department  
University of Calabria  
Via P. Bucci  
Rende, Italy  
comcar@tiscali.it

Andrea Pugliese  
DEIS Department  
University of Calabria  
Via P. Bucci  
Rende, Italy  
apugliese@si.deis.unical.it

Abstract

XML is emerging as the “universal” language for semistructured data description/exchange, and new issues regarding the management of XML data, both in terms of performance and usability, are becoming critical. The application of knowledge-based synthesization and compression methods (i.e., derivation of synthetic views and lossless/lossy approximation of contents) can be extremely beneficial in this scenario. In this paper we present SqueezeX, a system for the synthesization/compression of XML data based on a multidimensional/classification-based interpretation of their schema and on the application of semantic- and type-oriented compression techniques.

1 Introduction

Nowadays a large amount of semistructured data [1] is being managed and exchanged. The World Wide Web can be thought of as an enormous database containing highly heterogeneous and freely correlated data. Recently the extensible Markup Language (XML) [21] is being used as a language for describing semistructured data. XML allows describing documents which can be later visualized using different formats, thus realizing a complete separation among structure, content and style of documents.

The increasing amount of XML data will lead to the origin of new issues regarding efficiency in the representation of documents. An emerging problem with XML data is the synthesis and the (eventually lossy) compression of documents. A recent proposal for the compression of XML documents is XMill [12] that is a lossless (i.e., the original data are eventually restored) compressor/decompressor for XML data, to be used in data exchange and archiving. XMill applies classical entropy-based compression techniques after the execution of ad-hoc compression rules that are driven by the XML structure of the document and by the semantics of data. Another proposal is XMLZip [23], that essentially cuts the XML tree at a certain depth and compresses the two parts separately.

In classical compressors, both in general purpose ones (gzip) and in special-purpose ones (XMill), the compression is only used for archiving or exchanging but not for deriving a “synthetic” yet meaningful view of a document as it happens in the compression of images (JPEG) or video sequences (MPEG).

Our belief is that data synthesis and lossy compression will become crucial in next applications on Internet. Due to the manifold contents of XML documents (sentences, database-extracted records, and completely unstructured texts), we think that a hybrid approach combining summarization techniques, for the unstructured parts, and aggregation techniques for the structured parts of an XML document, can lead to a more effective way to synthesize XML documents.

Our proposal consists of a synthesizer/compressor for XML data in which a lossy compression ratio is first negotiated and then a synthetic/compressed view of the original document is delivered with this compression ratio. The synthesizer/compressor allows, in an orthogonal way (i) to restructure the schema of an XML document, allowing data aggregation and summarization in the new schema (Synthesis) and (ii) to further compress values contained in an XML document, by using a set of general- and special-purposes lossless/lossy compressors (Compression).

This approach introduces the concept of document synthesis, which goes beyond classical concepts of value approximation. To increase the compression ratio we can use lossy compressors that introduce an error in the reconstruction of the original document. To fit the user’s perception band, i.e. the amount of data and information he/she is able to face at a certain time, due to technological constraints (e.g. the communication bandwidth and the resolution of the terminal) and functional expectatives (decision maker vs operative employee), a schema restructuring and data ag-
Aggregation and summarization have to be used. The rest of the paper is organized as follows: in Section 2 we describe our system for XML synthesis/compression; in Section 3 we describe a preliminary implementation; in Section 4 we show some preliminary experimental results; Section 5 outlines conclusions and future work.

2 SqueezeX: A System for the Semantic (Lossy) Synthesis/Compression of XML Documents

The proposed system is essentially intended to be used as author system for the design of adaptive Web sites (eventually integrating data from different Web sources) to be browsed at different detail levels. In a different scenario, it could be useful for a “navigation aid” where a multidimensional view over (XML or XHTML) data is built “on the fly”.

SqueezeX comprises four main modules (Fig. 1): (i) the XML Design Assistant, which assists the author/knowledge worker in building the XML documents and metadata about them which will be used for the synthesis; (ii) the XML Structure Extractor, which tries to extract information about the structure of documents, when it is not known a priori or it is subject to change over time; (iii) the Synthesizer that restructures documents and aggregates and summarizes their elements; (iv) the Value Compressor, that performs both lossless and lossy compression on values using semantic information about them.

Figure 1. Architecture of the system.

Original XML documents are directly produced in XML or built by means of XML Wrapper modules [2, 5] which extract in a semi-automatic way data from external documents, databases, and in general (eventually dynamic) Web sources, and yield XML documents as output.

The Loss Negotiation module allows users to choose at run-time the detail level of displayed documents. Such detail level is set up on the basis of desired compression ratios (e.g. 80% of the original document’s size) and of the user’s perception band. The output of the Loss Negotiation module is an XML representation of the chosen detail level.

2.1 The XML Structure Extractor

The XML Structure Extractor performs several different semi-automatic tasks:

1. The extraction of an high-level schema of the documents containing information about their structure, in order to identify elements and attributes semantically suitable to be interpreted as dimension and measures. In this task, Ontologies are used for capturing terminological properties of elements and attributes (see e.g. WordNet [15]). This schema will be used for the Multidimensional-Based Aggregation. For the parts of the documents where elements contain only a small set of common attributes and/or sub-elements, so it is difficult to extract a multidimensional schema, on the basis of statistical properties of documents’s structure (see e.g. the SDR-Network approach [16]) the module extracts another high-level schema of the document, containing unstructured parts and a set of semistructured elements. Elements contained in this schema have the same starting and ending tag, but in general different attributes and sub-elements. This schema will be used for the Classification-Based Summarization.

2. The construction of potentially feasible hierarchies over identified dimensions, both evaluating structural properties of the document (see e.g. the approach proposed in [3]) and making use of ontologies.

3. The evaluation of the achievable compression ratios and synthesis quality, in terms of errors. Original detailed data are in general estimated using interpolation; since different aggregations and structures for representing them yield different estimation errors, it is desirable to be able to compute in advance such errors in order to choose the best aggregations (see e.g. [18, 4] for some examples of error minimization techniques).

4. The identification of sets of elements/attributes containing data of the same type or ranging among few values, in order to suggest applicable value compressors. The module also evaluates the achievable compression quality: many lossy content compressors...
yield measurable errors to be evaluated by considering the differences between original and reconstructed data; in some cases, e.g. for wavelet transform (see Sec. 4), errors can be set up before compressing data.

5. The (visual) interaction with the author/knowledge worker in order to show and validate the knowledge extraction results and to choose the actual metadata to consider.

2.2 The Synthesizer

The Synthesizer performs both the aggregation of structured parts of documents and the summarization of poorly structured ones.

2.2.1 Multidimensional-Based Aggregation

In the multidimensional-based aggregation, XML documents (both data and structure) are processed in such a way elements can be regarded as tuples of a relation, to single out a number of dimensions and measures and provide multidimensional representations that will finally be structured as datacubes, with aggregate data on suitable dimension intervals. Documents are finally reorganized according to some aggregation functions.

The metadata on how to view and structure XML documents as datacubes (Multidimensional Description) identify (i) elements and attributes to be considered as dimensions and measures, (ii) feasible hierarchies over dimensions, and (iii) applicable aggregation functions.

On the basis of such description of documents and of the chosen level of detail, the Synthesizer builds aggregation queries (expressed in XQuery [22]), executes them by means of an XQuery engine and obtains the synthetic XML documents.

For instance, consider an XML document containing elements describing sales and having the following structure:

```
<sale ID="...">
  <item>...</item>
  <time>...</time>
  <location>...</location>
  <units>...</units>
</sale>
```

Possible dimension attributes could be the item, the time and the location, and a measure attribute the units sold. A suitable hierarchy over dimensions could be the following:

```
item : item → type
time : day → month → quarter → year
location : store → city → province → country
```

The XML multidimensional description could have the following structure:

```
<dimension
  path="/sale/item"
  name="item">
  <row>
    <item>...</item>
    <type>...</type>
  </row>
  ...
</dimension>

<dimension path="/sale/time"
  name="time">
  <row>
    <day>...</day>
    <month>...</month>
    <quarter>...</quarter>
    <year>...</year>
  </row>
  ...
</dimension>

<measure path="/sale/units"
  name="units"
  functions="sum avg min max"/>
```

Here, it is specified that (i) the dimension “item” is identified by the path /sale/item, (ii) the dimension “time” is identified by the path /sale/time, (iii) the measure “units” is identified by the path /sale/units and (iv) the applicable aggregation functions are the sum, the average, the minimum and the maximum. Furthermore, the dimension tables are given.

A detail level could be set up by specifying e.g. \{item, month, avg(units)\} meaning that the user requires the average value of the units measure attribute, grouping data by item and month. In this case, the synthetic version of the document would have the following structure:

```
<sales>
  <item>...</item>
  <month>...</month>
  <avg-units>...</avg-units>
</sales>
```

Or, the user could specify \{type, country, sum(units)\} meaning that he/she requires the sum of the units measure attribute, grouping data by type and country. So the synthetic version of the document would have the following structure:

```
<sales>
  <type>...</type>
  <country>...</country>
  <sum-units>...</sum-units>
```
2.2.2 Classification-Based Summarization

The classification-based summarization is used by the synthesizer when it is difficult to find a multidimensional interpretation of XML documents. In this case the goal is to identify groups of elements and associate them to classes.

The summarization works as follows:

- First, each element of the XML document is considered as a tuple, where each tuple attribute is an attribute of the element. The set of common attributes is used to build an Element Index on each element, and an Element Frequency Index giving the standard relative frequency for the elements appearing in the document.
- Then, using classification methods, the set of XML elements is partitioned into classes.
- After the XML elements have been classified, only the common properties of these classes, together with some statistics, are presented to the user. Or, if for the records contained in a given class it is possible a multidimensional interpretation, it is applied, and the corresponding datacube, with aggregation on the suitable dimensions, is presented to the user.

2.3 The Value Compressor

In addition to the synthesis, we extend the approach of XMll to further compress the XML markup structure and data values with lossless compression techniques for the markup structure and lossless/lossy techniques for data. The compressor optimizes the representation of document's structure and identifies data contained in documents, grouping them in type- and semantic-dependent containers. Such containers are selectively compresses, eventually introducing a certain degree of loss. Finally, compressed containers are input to a gzip compressor; gzip combines Huffman coding and Lempel-Ziv algorithm [6] and therefore it is much more effective on homogeneous data.

Data contained in the document are distributed in the containers on the basis of the user’s choices, expressed in the negotiation phase. Such choices consist of expressions regarding data paths (Path Expressions) and associations of paths with compressors. So far our system comprises semantic compressors for integers, reals and strings (using the minimum number of bytes needed), IP addresses (encoding addresses with 4 bytes), value sequences (encoding exact values with a certain frequency, and representing intermediate data as differences, or using wavelets which allow a precise control of introduced errors), free text (transforming the original text in a more compact one whose meaning can be recognized by a human, applying some of the ideas proposed in [7, 14] and in particular well known classification techniques, to group sentences in clusters and to extract the most relevant sentences).

3 System Prototype

A prototype of the proposed system, regarding the multidimensional-based aggregation and value compression phases, has been implemented in Java [8]. At the time the implementation started we were not aware of any stable and reliable XQuery engine, so we built this preliminary prototype using the workaround of mapping data and dimension tables in a relational version and to exploit the SQL aggregation features.

![Figure 2. The implementation of the synthesizer.](image-url)

The Mapping module maps each of the processed XML documents in a relational database by associating elements and their attributes to tuples. The XML multidimensional description is mapped into two relational tables named dimension and measure; the former keeps track of the dimensions identified by specifying the path and name of the dimension in the star schema, while the latter keeps track of the measures, also taking into account the applicable aggregation functions. Then, for each dimension, the system builds a relational dimension table whose columns correspond to the hierarchies. The XML document to be synthesized is mapped by cutting down the content of the XML document; only elements and attributes to be interpreted as dimensions or measures are inserted as columns of the table.
Table 1. Experimental results of the multidimensional-based aggregation.

<table>
<thead>
<tr>
<th>Detail level</th>
<th>Number of elements</th>
<th>Compressed Size (Bytes)</th>
<th>Compression Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>{item, day, store, units}</td>
<td>5000</td>
<td>656770</td>
<td>0%</td>
</tr>
<tr>
<td>{item, quarter, store, sum(units)}</td>
<td>1152</td>
<td>140681</td>
<td>78.58%</td>
</tr>
<tr>
<td>{type, quarter, province, sum(units)}</td>
<td>96</td>
<td>12056</td>
<td>(\approx 98.1)%</td>
</tr>
<tr>
<td>{item, country, sum(units)}</td>
<td>12</td>
<td>1239</td>
<td>(\approx 99.7)%</td>
</tr>
<tr>
<td>{type, sum(units)}</td>
<td>2</td>
<td>187</td>
<td>(\approx 99.8)%</td>
</tr>
<tr>
<td>{sum(units)}</td>
<td>1</td>
<td>99</td>
<td>(\approx 99.9)%</td>
</tr>
</tbody>
</table>

The result of the negotiation phase is taken as input by the Query Construction module which builds and executes a SQL query on the basis of the detail level selected by the user. Finally, the Result Construction module builds the output XML document by simply creating for each tuple of the query result a node not shared with other tuples; since each tuple element represents an attribute value, it creates a node for each attribute. For the implementation of the content compressor we essentially built a Java object for each of the architecture modules. The parsing module uses the SAX parsing interface [13] thus avoiding the construction of an in-memory representation of the document to be compressed. At the output of the semantic compressors byte-wise objects are used which minimize the overheads due to internal Java object’s representation.

5 Conclusions and Future Work

In this paper we presented a lossy synthesis/compression system for XML documents. The main contribution of the paper is the attempt to produce a synthetic yet meaningful version of an XML document by restructuring its schema, and synthesizing it using a multidimensional approach and data mining techniques. The resulting reorganized XML document is further compressed using an approach similar to XMill, enhanced by the use of lossy compressors.

We implemented a first prototype of the system allowing the multidimensional interpretation of structured XML documents and then, in an orthogonal way, lossy/lossless compression, on the basis of user wishes. Preliminary results are very encouraging. Future work will concern the implementation of the XML structure extractor and of the classification-based summarization for poorly structured parts of documents.

References


