Extracting semistructured data from the Web: An XQuery Based Approach

Gilles Nachouki
Université de Nantes - Faculté des Sciences, IRIN, 2, rue de la Houssinière, BP 92208 44322 Nantes cedex 03
nachouki@irin.univ-nantes.fr

Mohamed Quafafou
Université de Nantes - Faculté des Sciences, IRIN, 2, rue de la Houssinière, BP 92208 44322 Nantes cedex 03
quafafou@irin.univ-nantes.fr

Abstract

This paper describes work in progress concerning the extraction of information from the web. This work is a part of frameworks consisting to extract, interconnect and access heterogeneous data sources. In this paper, we present a new approach for information extraction from the web. In this approach the web is viewed as a large database containing XML documents. The XQuery language is used in order to extract information from this database. An experimental tool has been developed in order to validate this proposal.

1. Introduction

With the growth of the web, many data sources have been made accessible through the web. These go from search engines, online product catalogues, to bibliographic listings. The potential use of such information sources has lead to the research on how to encapsulate such sources to make them available to computer programs. This research has lead to the introduction of programs called wrappers. Faced to the constant evolution of the web these wrappers need to be maintained and renewed often and since each HTML page has its own structure then an infinite number of wrappers must be generated. To face this problem it become interesting to think about another approach more suitable for the web environment. In this paper, we present a new approach which don’t need wrappers generation for each useful HTML page but needs only to convert and filter non-useful information in this page. The information extraction is based in our proposal on the XQuery language. Section 2 presents some related work in this domain. Section 3 introduces our approach. Section 4 presents the XQuery language. Section 5 shows some results given by the experimental tool.

2. Related Work

Many researchers have tackled problems related to information extraction from the web. Those researches have lead to the introduction of programs called wrappers. A wrapper for the web can be seen as a procedure allowing to extract structured data contained in a set of partly but similarly structured web pages[6]. In the construction of wrappers, three main points of view can be find in the literature: Most wrapper construction methods are automatic or semi-automatic based on inductive learning [6,5]. Structure discovery or extraction can also be used to resolve the wrapper generation problem [3]. When considering the problem of relation extraction or mining, an approach is to consider that their exists a duality between patterns and relations [2,7]. Many other wrapper induction also exist such as supervised interactive wrapper generation [1], building knowledge based wrappers (JEDI [4]). The originality of our approach is that we doesn’t need to build any wrappers to extract information from the web.
3. Information extraction in our approach

The Extraction of information from the web follows two steps (figure 1): (1) data filter and conversion; (2) data extraction. The first step consists to convert an HTML page to an XHTML document after filter of some non-useful tags. The aim of the XHTML document is to permit to query an HTML document using XQuery language. The filter process permits to reduce the current document by regrouping together some related information over the same tag. For example consider the following fragment of an HTML page:

```html
<TD align=center><Font Size=+2 Color=red><B> Semistructured data and XML </B></Font></TD>
```

This fragment is reduced as follows:

```html
<TD>Semistructured data and XML</TD>
```

This step shows that the web can be considered as a large XML database containing XML documents and information extraction from this database can be performed using XML query language e.g. XQuery language.

![Diagram](image1)

**Figure 1.** XQuery approach for extracting semistructured data from HTML pages

In order to illustrate the first step we consider the following example.

**Example 1.** Consider the following HTML page given by the URL: [http://www.hotels-de-nantes.com](http://www.hotels-de-nantes.com). This page gives information about Hotels in Nantes city (name, address, telephone number, ...). This URL is obtained by a user interface (not shown in this paper). Users specify a key word through this interface (in our example the key word is “Hotels Nantes guide” and this interface returns, based on a search engine (Google engine searcher), a list of addresses that contains this key word. For each selected URL (by user) the system returns the corresponding XHTML document. In the literature a lot of software are developed in order to convert HTML pages to XHTML documents (for example, the Tidy software which can be found at the following address: [http://www.w3.org](http://www.w3.org)). In our example the XHTML document corresponding to the URL is given in figure 2 as a tree document:

![Diagram](image2)

**Figure 2.** A part of the XHTML document after conversion

After conversion of the HTML page to an XHTML document, we begin the phase consisting to filter some non-useful attributes and elements from the XHTML document. The following table (table 1) gives a part of the list of elements and attributes containing information supposed non-useful for users.

<table>
<thead>
<tr>
<th>Elements</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>From</td>
<td>Bgcolor</td>
</tr>
<tr>
<td>img</td>
<td>Align</td>
</tr>
<tr>
<td>Br</td>
<td>Width</td>
</tr>
<tr>
<td>Hr</td>
<td>Name</td>
</tr>
<tr>
<td>A</td>
<td>Style</td>
</tr>
</tbody>
</table>

**Table 1.** A part of Non-useful elements and attributes in an XHTML document
The name of useful nodes in the XHTML document (after filter) is not important for users because no-semantic is assigned to them in the real world. So we have chosen to give them the same tag name. Figure 2 gives the final XHTML document after filtering and renaming nodes.

![Figure 2. The final XHTML document after filtering and renaming.](image)

The second step of our approach is based on XQuery language. We present in the following section, the XQuery modelling, parser and processing.

4. XQuery language

XQuery is designed by the W3C XML Query Working Group. XQuery is a query language for XML.

4.1. XQuery modelling

This section shows an UML schema modelling XQuery language. This schema is implemented using Java programming language and is composed of six packages (table2): Model, Object, XPath, Functions, Expressions and XUtil.

<table>
<thead>
<tr>
<th>Packages</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>This package describes the data model of XQuery and its relationships.</td>
</tr>
<tr>
<td>Object</td>
<td>Set of classes that represents the basic type of the language.</td>
</tr>
<tr>
<td>Xpath</td>
<td>Set of classes using XPath functionality included in XQuery.</td>
</tr>
<tr>
<td>Functions</td>
<td>Set of classes which implements XQuery functions.</td>
</tr>
<tr>
<td>Expressions</td>
<td>Contains classes that describe all the XQuery expressions.</td>
</tr>
<tr>
<td>Xutil</td>
<td>Set of utility classes.</td>
</tr>
</tbody>
</table>

![Table 2. XQuery modelling (UML packages)](image)

4.2. XQuery Parser

The XQuery Parser is developed for a lexical and syntactic analysis of XQuery queries. XQuery parser is generated using JavaCC [8] (Java Compiler Compiler developed by SUN) by projection of the XQuery Grammar on JavaCC utility.

4.3. Queries Processing

This section provides an algorithm for processing XQuery queries. The input of this algorithm is a query compliant with XQuery specification and the output of this algorithm is an XML document that represents the result of this query. The algorithm is composed of two principal steps: analysis and evaluation. The analysis step checks the correctness of a query and returns an XQuery expression. The next step consists of evaluating the returned expression by the parser in order to obtain the final result.

**Algorithm : XQuery analysis and evaluation**

**Input:** An XQuery query (Xq):

```xml
<Results>
  For (or Let) Expression1
  Where Expression2
  Return Expression3
</Results>
```

**Output:** An XML document corresponding to the result of the query

**Step1 (Analysis):** This step consists to analysis the query “Xq” given in input to the parser. It returns an XQuery expression “Expr” instance of the class XqueryExpression (defined in the package Expressions):

```
Expr = Parser.Start (Xq)
```

**Step2 (Evaluation):** This step consists to evaluate “Expr” returned by the XQuery parser. The evaluation of this expression is realised by the following instruction:
Result = Expr.Eval(context). We distinguish several sub-steps:

Step2,1: This step evaluates the For and Let clauses in Expression1 of the query. Users can formulate each of those two clauses as follows:

(M1): For (or Let) SX in (=) document("name")/path or
(M2): For (or Let) SX in (=) document("name")/path [Predicate]

For each (For or Let clause expression) do
Switch (Expr)
End For

Case 1 : ForClause
Creation of a list L1 where, L1 = \{(X,node1), (X,node2), ...\} containing all the nodes referred by the clause “For” (and satisfying Predicate in the case of M2). “X” is a variable containing, for each iteration, one node;

Case 2 : LetClause
Creation of a list L2 where, L2 contains the only node returned by the clause “LET” (and satisfying Predicate in the case M2).

End Switch
End For

Step2,2: The clause “Where” is optional in XQuery language. Predicates in Expression2 of this clause consist of one of the following expressions (list not exhaustive):

XEqualExpr | XlessExpr | XgreaterExpr where
XEqualExpr : E1 = E2,
XlessExpr : E1 < E2,
XgreaterExpr : E1 > E2 and
E1 = $X/path1; E2 = $Y/path2

The evaluation of Expression2 follows the two following steps:
- Creation of a list L, L = L1 X L2 (X is a product Cartesian), L1 and L2 are two lists obtained from step2,1
- Restriction of L to the elements which respect the predicates in the clause “Where”

Step2,3: this step constructs new elements including values corresponding to the result of the query as specified in the Expression3 of the clause “Return”. Two cases are distinguished:

Case 1: Expression3 references to the nodes in the list L (L1 or L2). The evaluation consists to return the values of these nodes in the new building element;

Case 2: Expression3 is an expression of type Expression1 In this case step2,1 is called for the evaluation of this expression.

In order to illustrate the second step of our approach we consider the following example.

Example 2. Consider the XHTML document given in figure 2. The structure of this document is represented in figure 3 as DOM tree document.

The following query permits to select all the hotels given in this page.

(Query):
<Results>For $a in document("Hotels")/html/body/l/l/l RETURN
 <Hotel>$a</Hotel>
</Results>

Figure 3. DOM tree of the XHTML hotels Nantes guide

“Hotels” is the name of the XHTML document. The path html/body/l/l/l is the path to a node containing sub-nodes where each one describes one hotel.

5. Experimental results

We have developed an experimental tool in order to validate this proposal. The main features of this tool are to extract, interconnect and access heterogeneous data sources. We use this system in this paper in order to show only how to extract and query one (or several) HTML web page(s). Users non-familiar with XQuery language can use a generator interface in order to generate queries.
In this section we show two examples. The first one selects information from one HTML page and the second one joins between two HTML pages. Queries are generated using the generator interface.

The first example selects from the HTML page (URL: http://www.informatik.uni-trier.de/~ley/db/conf/index.html) all the conferences including a database domain. The XQuery query is given as follows.

(Query):
<Results>
FOR $a$ IN document("filtered.xml")/html/body/l/l
where contains($a,"Database")
RETURN
<Conferences>$a/text()</Conferences>
</Results>

Figure 4. Result of the query: Database Conference from DBLP conferences HTML page.

The second example selects from the HTML pages (URLs: http://www.bestofnantes.com/ files/hotel_resto/resultat.asp?fichier=resto&s=-2, http://www.hotels-de-nantes.com) the hotels and the Restaurants which are in the “Route de Vannes” street. The XQuery query is given as follows.

(Query):
<Results>
FOR $a$ in document("Hotels")/html/body/l/l/l
where contains($a,"route de Vannes")
RETURN
<Hotel>$b</Hotel>,
FOR $b$ in
where contains($b,"route de Vannes")
RETURN
<Restaurant>$b</Restaurant>
</Results>

Figure 5. Result of the query: Restaurant and Hotel in the “route de Vannes” street in Nantes City

6. Conclusion

In this paper, we have introduced a new approach to information extraction from the web. This method is based on XQuery language. In our proposal, the web is viewed as a database containing XML documents. Users can retrieve information from one or several HTML pages using this language. Many research works can follow this work, such as trying to extract multimedia information.

7. References