SEARCH SEMI-STRUCTURED DATA ON WEB

Sabin-Corneliu Buraga[1], Teodora Rusu[2]

[1]Faculty of Computer Science,
"Al.I.Cuza" University of Iaşi, Romania
Berthelot Str., 16 – 6600 Iaşi, Romania,
tel: +40 (32) 201529, fax: +40 (32) 201490,
e-mail: busaco@infoiasi.ro

[2]"P.Poni" Institute of Macromolecular Chemistry
Ghica Voda Str., 41 A – 6600 Iaşi, Romania,
tel: +40 (32) 144909, fax: +40 (32) 211299,
e-mail: teia@ichpp.tuiasi.ro

Abstract: The paper present a Web robot designed to search semi-structured documents. These documents consist in a set of HTML mark-ups (tags). To properly store the found documents it can be used our defined XML-based language called *WQFL* (*Web Query Formulating Language*). Also, this language can be used to formulate structured queries for Web search activity. The Web robot is implemented in Java language that assures the platform-independence of the application. To process WQFL documents the robot uses *JDOM* library.

Keywords: Web, search, semi-structured data, XML, robot

1. INTRODUCTION

The World-Wide Web, the world's largest hypertext structure, has already in the last ten years a vast expansion. In recent times, many worldwide consistent or semi-consistent collections of scientific data, in particular disciplines, have become accessible on Internet. These sources of information comply with a standard for interoperability. The data may also conform to a common semantics (each item of particular data has a precise formal definition).

Regardless of many theoretical and technical advances, it is harder to discover the existence of appropriate data needed for a particular problem and the most frequent approach in searching information, on Web mainly, is the keywords based method. Different Web robots perform this search activity. The growing of information available on Internet shows the weakness of the traditional Web search techniques. As a result, increased usage of online search by non-specialists has increased the need for a more effective and friendlier search experience.

Search services generally can be distinguished according to how they accumulate and organize their meta-information: *automatic acquisition and indexing* (performed by Web robots) and *manual acquisition and categorization* (accomplished by trained information specialists). Each method is not enough in the present.

The paper presents a Web robot designed to search semi-structured documents (particularly used in the Chemistry field of science). These documents consist in a set of HTML mark-ups (tags).

To properly store the found documents, it can be used our defined XML-based language called *WQFL* (*Web Query Formulating Language*). Also, this language can be used to formulate structured queries for Web search activity.

The Web robot is implemented in Java language that assures the platform-independence of the application. Also, the robot can use different modules written in *Jess* – a script language and a development environment for expert systems. To process WQFL documents the robot uses *JDOM* library – an open-source implementation of World-Wide Web Consortium's DOM – level 1 recommendation.

2. SEMI-STRUCTURAL SEARCH

The following practical situations are observed (Buraga and Rusu, 2000):

- Compound queries are using Boolean connectors (such as *and*, *or*, *near* or *not* operators used by all actual search Web engines).
- The users would like to retrieve particular Web documents that have diverse data structures (e.g. without tables, only seven pictures to be placed on bottom of the desired Web page and so on).
- The search activity can be more productive by using different AI methods.

In the first phase of search process, the user is going to formulate a query. By using different (Web) interfaces, we can design textual or graphical complex queries. The users need to be able to formulate complex and flexible queries such as *"macromolecular" + "entities" + without applets + without sounds + with <3 tables on top + <7 paragraphs.* This query can be formulated by an XML-based language called **WQFL** (**Web Query Formulating Language**). This markup language will be presented later in this paper (see section 3). For each found page, the robot will generate a WQFL document.

The given keywords (e.g. "macromolecular") will be used effectively by the search engine (for example, *Google* or *Altavista*) to return first *N* significant pages and the remaining expressions (e.g. with <3 tables on top) will be processed in the activity of semi-structural search by the Web robot. In fact, our Web robot will operate like a proxy.

The second part's objective is to encode the Web pages structural information. According to the given potential of WQFL language, some users would like the graphical content to be found on top of the Web pages and maximum 7 paragraphs etc. That information is stored into WQFL documents. The robot will process each found Web document and it will retain only the position (top, middle, and bottom) and the occurrences of some HTML elements and attributes (e.g. `<p>`, `<table>`, `<img>`, `<applet>` and so on).

For example, the robot can use the following HTML elements (tags) to perform the semi-structural search activity (Gogan and Buraga, 2000): `<p>` (paragraph), `<img>` (still image; JPEG, GIF, or PNG file image formats), `<object>` (multimedia – sound, movie, animation, 3D virtual world – or generic object, such as ActiveX component), `<table>` (tabular data), `<a>` (anchor), `<script>` (script code, such as JavaScript or VBScript programs), `<applet>` (Java applet) etc. For each element, the robot will keep three values that represent the occurrences of that element on top, middle and bottom of the Web page. The generated WQFL documents will be stored locally.

Later, another software tool can perform a semi-structured search by using a query language such as *QL* or *XQL* (*Extensible Query Language*, a declarative, path-oriented query language for XML) to discover the desired information (Deutsch *et al.*, 1998). Another approach is to consider the genetic programming techniques for searching the best page or best group of pages, according to user's request and the result will correspond to most wanted document.

3. WEB QUERY FORMULATING LANGUAGE (WQFL)

3.1 Extensible Markup Language (XML)

Derived from SGML (Standard Generalized Markup Language), the *XML* (*Extensible Markup Language*) language is a recommendation of the World Wide Web Consortium for a meta-language to define markups (annotations) for content publishing on the Web and other areas (Bray *et al.*, 2000). The aspiration of XML is to make available some benefits not available in

HTML, such as arbitrary extensions of a document elements (tags) and their attributes, support for documents with complex structure, and validation of document structure with respect to an optional document-structure grammar, called a *DTD* (*Document Type Definition*). A DTD specifies what elements may occur and their order of occurrence and how the elements may nest in an XML document that conforms to this DTD.

Since 1998, XML has grown into a large family of standards integrating key technologies from three previously independent domains: documents, databases, and the Internet. Several examples are *SMIL* (*Synchronized Multimedia Integration Language*), *MathML*, or *RDF* (*Resource Description Framework*), all World-Wide Web Consortium's recommendations (W3C, 2001).

3.2 DTD for WQFL

This subsection describes the WQFL elements and attributes by using the DTD formal rules. The following DTD defines some of the constituents of the WQFL language:

```
<!-- WQFL elements -->
<!ELEMENT webquery
       (engine+,query,structure,page*)>
<!-- web query information -->

<!ELEMENT engine      (#PCDATA)>
<!-- search engine -->
<!ELEMENT query       (#PCDATA)>
<!-- query expression -->

<!ELEMENT structure   (element*)>
<!-- structural data -->

<!ELEMENT element     (occur*)>
<!-- elements data -->

<!ELEMENT occur       EMPTY>
<!-- position occurrences -->

<!ELEMENT page        (#PCDATA)>
<!-- found page(s) information -->

<!-- WQFL attributes -->

<!ATTLIST webquery
   timestamp   PCDATA #IMPLIED
   <!-- query timestamp -->
   maxpages    NUMBER #IMPLIED
   <!-- maximum number of found pages -->
   language    PCDATA #IMPLIED
   <!-- Web pages language(s) -->
>

<!ATTLIST engine
   url         PCDATA #REQUIRED
   <!-- search engine URL -->
   info        PCDATA #IMPLIED
   <!-- additional information
       (e.g. use a specific language) -->
>

<!ATTLIST element
   name        PCDATA #REQUIRED
   <!-- stored element name (e.g. <a>) -->


  order        NUMBER #IMPLIED
  <!-- order of relevance -->
  appear       yes|no  yes
  <!-- the element must appear
       (position don't matters) -->
  context      PCDATA #IMPLIED
  <!-- context of element occurrence
       (e.g. parent tag) -->
>

<!ATTLIST occur
   top         NUMBER #IMPLIED
   <!-- position occurrences -->
   middle      NUMBER #IMPLIED
   bottom      NUMBER #IMPLIED
>

<!ATTLIST page
   url         PCDATA #REQUIRED
>
```

For each found Web page, the robot generates a WQFL document. The document root element `<webquery>` will include a least one `<engine>` element, a `<query>` element, a `<structure>` element, and a zero or more `<page>` elements.

The `<query>` element will store the effective query expression to be sending to a search Web engine and the `<structure>` element will contain the semi-structural information of the found page. Some attributes (e.g. `name` or `url`) are mandatory and other can optionally appear (such as `maxpages`, `language` or `context`). The order of significance of an element is given by the value of `order` attribute of `<element>` tag.

3.3 Examples

An example of generated WQFL fragment of document follows:

```
<!DOCTYPE webquery
          PUBLIC "-//WQFL 1.0//EN">
<?xml version="1.0" ?>
<webquery
   timestamp="18.04.2001 10:33"
   location=
   ″http://ichpp.tuiasi.ro/~teia/m.html″>
  <engine
     url="http://www.google.com">
     Google
  </engine>
  <engine
     url="http://www.altavista.com">
     Altavista
  </engine>
  <query>macromolecular entities</query>
  <structure>
    <element name="p">
      <occur top="7" />
```

```
    </element>
    <element name="img" order="2">
      <occur middle="5" bottom="3" />
    </element>
    <element name="a" appear="no">
      <occur top="0"
             middle="0" bottom="0" />
    </element>
    <element name="table">
      <occur top="1" />
    </element>
  </structure>
</webquery>
```

To validate the document it can be used an XML processor such as Internet Explorer 5.0 or later (see figure 1).

The found page contains 7 paragraphs on top, 5 images on bottom, no other hyperlinks and only one table.

3.4 Benefits

The WQFL documents can store (within `<page>` element) the additional information about a certain Web page: location, size, metadata (creator, copyright, owner and so on), language etc.

The main benefits are the following: the WQFL documents can store metadata information about found Web pages and WQFL language can used as a standard format for interchange HTML and XML information between Web robots and agents.

Instead of HTML element names, the WQFL documents can include position occurrences information of any XML tags (such as SMIL, XHTML or MathML elements) and WQFL can be viewed as a query language for XML data.

Fig. 1 The validated WQFL document

4. IMPLEMENTATION OF THE WEB ROBOT

The Web robot will operate like a proxy. First, the user's query will be used to interrogate a main search engine (e.g. *Google*, *Altavista*, or *Excite*) and a number *N* of found pages will be returned. For each found page, the robot will generate a WQFL document. These documents can be stored locally.

Later, the robot can perform a semi-structured search by using a query language to discover the desired information.

The Web robot is implemented in Java language that assures the platform-independence of the application. For processing WQFL documents, the Web robot uses *JDOM* (*Java Document Object Model*) library (JDOM, 2001), an open-source and flexible implementation of World-Wide Web Consortium's DOM – level 1 recommendation (Wood, 1998).

To process the stored WQFL documents the robot uses *Jess* – a script language and a development environment for expert systems, written in Java and compatible to CLIPS (Watson, 1997). Jess can be used as a rule engine and can access Java native classes and interfaces. In the future, the Web robot will offer support for user-defined Jess modules.

An example of Jess program used to perform Web searching follows:

```
; (A_leaf_url url) - just to check
; (A_node_url url) - to check and parse
; A_tmp - internal temp var
; A_this - stores the current Agent object

; C_string - stores the searched string
; C_list_model - the list model
; (watch all)

(set_list_model (fetch C_list_model))
(set-strategy breadth)

(defrule process_node
  (A_node_url ?node_url)
  (not (processed ?node_url))
  =>
  (assert (processed ?node_url))
  (bind ?url (check_url
    (string_to_url ?node_url)))
  (if (neq nil ?url)
    then
      (bind ?new_url (url_to_string ?url))
      (if (neq ?new_url ?node_url)
        then
          (assert (processed ?new_url))
      )
      (log status "Searching:" ?new_url)
      (bind ?http_client
```

```
        (get_http_client ?url))
      (parse_url ?http_client)
      (if (contains ?http_client
        (fetch C_string))
        then
          (if (add_address ?http_client)
            then
              (log info "Found:" ?new_url)
            else
              (dispose_http_client
                 ?http_client)
          )
        else
          (dispose_http_client
             ?http_client)
      )
  )
)

(run)
```

The interface of the Web robot is presented in figure 2 and uses *Swing*.

5. CONCLUSIONS

A Web robot was presented in this paper. The robot is designed to search semi-structured documents (particularly used in the Chemistry field of science). To properly store the found documents it can be used our defined XML-based language called *WQFL* (*Web Query Formulating Language*). Also, this language can be used to formulate structured queries for Web search activity. The Web robot is implemented in Java language and can use different modules written in *Jess* – a script language and a development environment for expert systems. To process WQFL documents the robot uses *JDOM* library – an open-source implementation of World-Wide Web Consortium's DOM – level 1 recommendation.

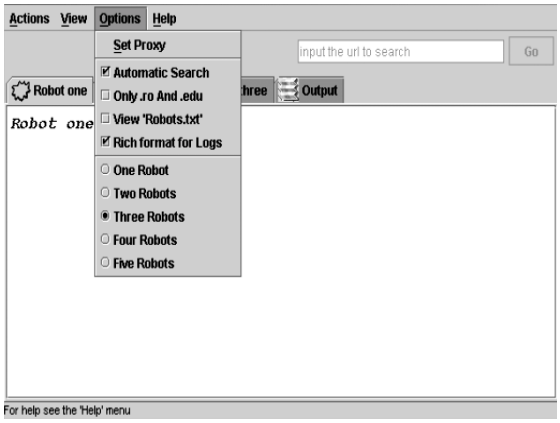

Fig. 2 Interface of the implemented Web robot

ACKNOWLEDGEMENTS

We express our gratitude to *Valentin Irimia* and *Valentin Păscăreanu* – graduates of Faculty of Computer Science, "Al. I. Cuza" University of Iaşi, Romania – for their help in implementation process of the presented Web robot.

REFERENCES

Bray, T. *et al.* (2000). *Extensible Markup Language (XML) – version 1.0 (updated)*. W3C Recommendation, Boston: http://www.w3.org/TR/REC-xml

Buraga, S.C., Rusu, T. (2000). An XML-based Query Language Used in Structural Search Activity on Web. In: *The 4th Conference on Technical Informatics - CONTI 2000 Proceedings. (Transactions on Automatic Control and Computer Science)*. Ştefan Holban (ed.), Timişoara, 107 – 112

Deutsch, A. *et al.* (1998). *XML-QL: A Query Language for XML*. W3C Note, Boston: http://www.w3.org/TR/NOTE-xml-ql

Gogan, O., Buraga, S.C. (2000). The Use of Neural Networks for Structural Search on Web. In: *The 10th International Symposium on Systems Theory – SINTES 10 Proceedings*. ROM TPT, Craiova, 53 – 56

JDOM (2001): http://www.jdom.org/

Watson, M. (1997). *Intelligent Java Applications for the Internet and Intranets*. Addison-Wesley, Reading MA.

Wood, L. (1998). *Document Object Model (DOM) Level 1 Specification*. W3C Recommendation, Boston: http://www.w3.org/TR/REC-DOM-Level-1

World-Wide Web Consortium (2001): http://www.w3.org/