

An XML-based Query Language Used in Structural Search Activity on Web

Sabin Corneliu Buraga

Faculty of Computer Science, "A.I.Cuza" University
G-ral Berthelot Street, 16, Iasi 6600 Romania

Phone: +40 32 201529, Fax. +40 32 201490, E-mail: busaco@infoiasi.ro

Teodora Rusu

"P.Poni" Institute of Macromolecular Chemistry
Ghica Voda Street, 41A, Iasi 6600 Romania

Phone: +40 32 144909, Fax. +40 32 211299, E-mail: teia@uaic.ro

Abstract

We propose a markup language based on XML to formulate various queries to search hypermedia information, using the AI different approaches. The search activity is using a document structure based method and it can be divided into several parts. In the first part, we are searching the information using a traditional search engine and we are storing first significant found pages. The user's queries can be formulated by using an XML-based language, called *WQFL (Web Query Formulating Language)*, defined in this paper. The second part consists in the encoding of the Web pages structural information (the position, the occurrences and the context of some HTML elements), building a matrix. The elements of this matrix will be encoding to obtain an integer number. In the third phase, we can apply AI methods to obtain the most relevant Web page, such as a self-organizing feature maps neural network based on the competitive learning or genetic algorithms. A SAX library available on Linux can process the WQFL language.

Topic: **C6 (Artificial Intelligence)**

1. INTRODUCTION

The Web, the world's largest hypertext structure, has already in the last ten years a huge development. Recently, many worldwide consistent or semi-consistent collections of scientific data, in specific 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). Despite many theoretical and technical advances, it is harder to discovering the existence of relevant data needed for a particular problem and the most common approach in searching information, on Web mainly, is the keywords based method.

The growing of information available on Internet shows the weakness of this traditional Web search technique. Therefore, increased usage of online search by non-specialists has increased the need for a more effective and friendlier search experience.

The need of intelligent knowledge discovery and flexible query interfaces is decisive.

2. STRUCTURAL SEARCH

We observe the following practical situations:

- Complex queries are using Boolean connectors (such as *and*, *or*, *near* or *not* operators used by all actual search engines).
- The users want to retrieve particular Web documents that have different structures (e.g. without tables, only three pictures to be placed on bottom of the Web page etc.).
- The search activity can be more productive by using 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 "microcontroller" + "article" + without applets + without sounds + with <3 tables on top + <7 paragraphs. This query can be modeled by our proposed 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 will be generated a WQFL document.

The given keywords (e.g. "article") will be used effectively by the search engine to return first N significant pages and the remaining expressions (e.g. with <3 tables on top) will be processed in the activity of structural search.

The second part's goal is to encode the Web pages structural information. According to the given possibilities of WQFL language, some users want the graphical content to be placed on top of the Web pages and maximum 7 paragraphs etc. That information is stored into WQFL documents. We'll retain only the position (top, middle, and bottom) and the occurrences of some HTML elements and attributes (e.g. <p>, <table>, , <applet> and so on), building a matrix. The elements of this matrix will be encoded by means of a special operator to obtain an integer positive number. The choice of that operator will depend on AI method used in the next phase. If there is more than one page associated with the same number, we will keep only one. After this process, each page is denoted by its *structural information number* [2].

We can use the following HTML elements (tags) to perform the structural search activity: <p> (paragraph), (still image), <object> (multimedia or generic object), <table> (tabular data), <a> (anchor), <script> (script code, such as JavaScript or VBScript programs), <applet> (Java applet) etc. For each element, we can keep three values that represent the occurrences of that element on top, middle and bottom of the Web page.

2.1 Example

Let consider the query "microcontroller and document" + with <7 paragraphs on top + with <3 images on bottom + with <5 images on middle + without links + without multimedia content + without tables. We consider only the "<" relation operator that can appear in the query expressions.

After the keywords suppression, we can build the following matrix:

7	0	0	0	0
0	5	0	0	0
0	3	0	0	0

where on rows we wrote the position (top, middle, bottom) occurrences of the elements and each column correspond to a HTML tag in this order: <p>, , <table>, <embed> or

<object>, <a>. The computed structural information number can be encoded using this matrix by means a special operator.

In the next stage, we will choose an AI approach to determine the best-found page, according to user's request and the result will correspond (hopely) to desired document.

We can adopt a self-organizing feature maps neural networks based on the competitive learning [2, 3] or genetic algorithms approaches to select the most relevant page [5, 8].

The user will obtain the URL (Uniform Resource Locator) of this page to browse its content. As we seen, this URL was stored after the classical searching activity was performed by the traditional search engines (such as *AltaVista* [9] or recently award-winner *Google* [10]).

3. WEB QUERY FORMULATING LANGUAGE (WQFL)

3.1 XML

In this section, we will propose an XML-based language that can be used to formulate structured queries for Web search activity described below.

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 [7, 13]. The ambition of XML is to provide 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*, *EFDL (Extensible Forms Description Language)* or *RDF (Resource Description Framework)* [13].

In the present, there are some proposals for locating and querying data repositories using XML: *WebSemantics* (this language permits describing, publishing, discovery and access to sources containing typed date) [6], *QL* (a query language used to search data in distributed databases) or *XQL (Extensible Query Language)*, a declarative, path-oriented query language for XML) [1, 12].

3.2 DTD for WQFL

First, we'll describe the WQFL elements and attributes. The following DTD defines them:

```
<!-- 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 returned pages -->
  language       PCDATA #IMPLIED
  <!-- desired 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 user query and for each found Web page, a WQFL document is generated. 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 send to the search engine and the `<structure>` element will contain the structural information given by the user, for each desired HTML element. 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 Example

For the query "microcontroller and documents" + with <7 paragraphs on top + with <3 images on bottom + <5 images on middle + without links in paragraphs + with tables, the generated WQFL document is:

```

<?xml version="1.0">
<!DOCTYPE webquery PUBLIC "-//WQFL 1.0//EN">
<webquery timestamp="18.09.2000 10:33" maxpages="15">
  <engine url="http://www.google.com">Google</engine>
  <engine url="http://www.altavista.com">Altavista</engine>
  <query>microcontroller and documents</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" context="p">
    </element>
    <element name="table" appear="yes">
    </element>
  </structure>
</webquery>

```

The WQFL document for the user's query is generated in the first phase of search activity. For each found Web page, a WQFL document is also created. 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 entire search activity development is described in section 2.

3.4 Benefits

The WQFL documents can be generated also by a graphical query Web interface to search data on Internet or different databases, using certain input methods (textual, graphical or gesture techniques). The WQFL documents can store metadata information about Web pages.

The described structural search approach can be applied to any XML documents. Instead of HTML element names, the WQFL documents can contain position occurrences information of any XML tags (such as SMIL, XHTML or MathML elements) and WQFL will function as a query language for XML data.

The WQFL documents can be used as a standard format for interchange HTML information between Web robots and agents, in indexing and classification activities performed by the search engines.

4. IMPLEMENTATION PROPOSAL

For XML documents processing, the Web Consortium was proposed an object-oriented model: *DOM (Document Object Model)* [13]. A flexible and easy to use DOM implementation is *SAX (Simple API for XML)* [4], written in C or Java languages. For the parsing process of user's query, the Web pages and the WQFL documents, we'll intend to use a free implementation of SAX, the *libxml* library [11], available on Linux platforms, part of *GNOME (GNU Network Object Model Environment)* project.

Using *libxml*, the WQFL documents can be parsed by a *CGI (Common Gateway Interface)* program, which is executed on Web server from a HTML document.

To determine the best-found page, we suggest two methods. One is using a neural network based on the competitive learning. The maximum *N* different structural information

numbers will be input for a self-organizing feature maps neural network and the winner neuron will give the best-found page according to user's request [2].

The second approach is to consider the genetic programming techniques for searching the best page. The genetic population will consist in N individuals and each individual will store (in 2 base) the structural information numbers of the found Web pages. The genetic operators are the classical mutation and crossover operators [5], but we intend to consider other operators for improving searching experiments.

5. CONCLUSIONS AND FURTHER WORK

In this paper we describe a document structure based method to search information on Internet and we propose for modeling the user's queries an XML-based language called WQFL (Web Query Formulating Language). This language can formulate complex queries and can retain some structural information (type and occurrences) about HTML (or other XML) documents' elements. The process of searching the most relevant page can be based on AI approaches, such as neural networks or genetic/evolutionary programming.

To parse user's queries and generated WQFL documents, we intend to utilize a free implementation of SAX, the *libxml* library available on Linux platforms. The WQFL language can be used also to formulate queries to locate different information in distributed databases or specific data repositories.

The next goal of our study is to develop a Web interface for processing graphical queries and generating WQFL documents.

References

- [1] A.Deutsch et al. - "*XML-QL: A Query Language for XML*", W3C Note, Boston, August 1998: <http://www.w3.org/TR/NOTE-xml-ql>
- [2] O.Gogan, S.C.Buraga - "*The Use of Neural Networks for Structural Search on Web*", SINTES10 Conference Proceedings, Craiova, May 2000
- [3] S.Haykin - "*Neural Networks: A Comprehensive Foundation*", IEEE Press, New York, 1994
- [4] J.Henstridge - "*Using the SAX Interface of LibXML*", XMLDev, Oct.1999
- [5] Z.Michalewicz - "*Genetic Algorithms + Data Structures = Evolution Programs*", Springer Verlag, Berlin, 1992
- [6] G.Mihaila et al. - "*Equal Time for Data on the Internet with WebSemantics*", EDBT'98 Conference Proceedings, Valencia, March 1998
- [7] N.Welsh - "*A Technical Introduction to XML*", ArborText Inc., 1998
- [8] M.Wulkekuhler, W.Punch - "*Finding Salient Features for Personal Web Page Categories*", GARAGE Technical Report, Michigan State University, 1995
- [9] * * * - *Altavista*: <http://www.altavista.com>
- [10] * * * - *Google*: <http://www.google.com>
- [11] * * * - *libxml*: <ftp://ftp.gnome.org/pub/GNOME/sources/libxml/>
- [12] * * * - "*QL'98 - The Query Languages Workshop Proceedings*", Boston, December 1998
- [13] * * * - "*World Wide Web Consortium's Technical Reports*", Boston, August 2000: <http://www.w3.org/TR>