XML Query Support for Web Information Extraction: A Study on HTML Element Depth Distribution

Michal Krátký[1], Marek Andrt[1], Vojtěch Svátek[2]

[1]Department of Computer Science, VŠB – Technical University of Ostrava
17. listopadu 15, 708 33 Ostrava–Poruba, Czech Republic
{michal.kratky,marek.andrt}@vsb.cz

[2]Department of Information and Knowledge Engineering, University of Economics, Prague, W. Churchill Sq. 4, 130 67 Praha 3, Czech Republic
svatek@vse.cz

Abstract. Knowledge-based web information extraction methods can achieve very high precision in restricted domains; they are however slow and suffer from performance degradation beyond their specific domain. We thus plan to adapt an existing XML storage and query engine to act as efficient pre-processor for such methods. The critical point of the approach is the amount of information provided as XML environment of the start-up terms/elements. For this purpose, we carried out a statistical analysis of depth distribution in the WebTREC collection.

1 Introduction

Information Retrieval (*IR*) [2] and *Information Extraction* (*IE*) are two complementary research areas. While IR addresses the discovery of whole (or, at least, sections of) documents, IE breaks them down to the level of arbitrarily small content fragments. Simple form of IR support was already considered in traditional IE models such as those examined in MUC[1] contests. However, the rapid evolution of XML query technology [18] together with partial reconciliation between HTML and XML (esp. larger support for XHTML) recently paved the way for more sophisticated IR–IE coupling. Although XHTML only imposes a very loose structure over the text of web pages, some aspects of its tree structure may be useful for IE purposes. This is particularly true for wrapper-like IE approaches, which explicitly model the structure of HTML formatting such as tables, lists or systematic font alterations. However, traditional wrappers [11, 14] typically assume extraction from complete pages structured in a uniform database-like manner. Obviously, manipulating small portions of XHTML trees only may only bring benefits if *sparse fragments* of useful information are to be extracted from large amounts of HTML data – we might call it *selective wrapping*. Moreover, the overhead of initial XML indexing only pays off when the

[1] http://www.itl.nist.gov/iaui/894.02/related_projects/muc/

same source information is used several times, e.g. for extraction of different entities, for their classification and the like.

There are rather few approaches to automated web analysis that systematically address different types of information available within websites and would thus benefit from XML indexing support. One of them is the *Rainbow*[2] project. On the other hand, the required XML indexing and querying functionality is currently being provided in the form of web services, in the *AmphorA*[3] tool. *AmphorA* can potentially provide the *Rainbow* tools with input data of adequate type and size, i.e. it can act as sophisticated pre-processor for information extraction methods. The starting point for such data supply can be keyword-based searching of XML data [12]. The critical point of the approach is however the amount of information provided as XML environment of the start-up terms/elements. The first, very rough approximation of adequacy for XML environment may merely be inferred from the typical depth of interesting HTML element types (which are likely to contain the text to be extracted).

Due to the fact that we depict an usage of XML indexing engine for purpose of IE, in Section 2 recent XML indexing approaches and query languages are described. Section 3 presents the tools Rainbow and AmphorA. Section 4 describes the WebTREC collection which is used for analysis of depth distribution of HTML element types, next the result of that analysis is presented. In the conclusion we summarize the paper content and outline possibilities of a future work.

2 XML Storage and Querying – State of the Art

2.1 Query languages over XML

A number of languages have been developed for querying over XML data e.g., XML-QL [7], XPath [4], and XQuery [19].The common feature of such languages is the usage of regular path expressions for formulation of the path in the graph modelling an XML document. Such a path is a sequence of element or attribute names from the root element to a leaf. XML query languages such as XQuery contains considerably complex techniques for querying XML data which lead to rather complex implementation. XPath is a frequently used subset of XQuery. This query language uses expressions to select a set of elements or attributes. Expressions comprise location paths, patterns and predicate filters. It is possible to specify the path location either absolutely or relatively. Moreover it is possible to use the axis which corresponds to the direction from the current to the specified element. Location paths select element type and its location. In order to remove undesired elements, it is possible to use predicate filters. XPath supports filters to position, elements, attributes, boolean tests, strings and numbers.

For purpose of IE it is suitable to extend XML query language in the IR manner. For example, the operator $\sim=$ is applied for keyword-based searching. Let us

[2] http://rainbow.vse.cz
[3] http://arg.vsb.cz

take queries `//doc[//h1∼='Information']` or `//table[th∼='Prize']`, where $\sim=$ is an operator containment of a term in the element content or attribute value.

2.2 Recent XML indexing approaches

Currently there are several approaches to indexing XML or, more general, semi-structured data. We may divide them into two categories. One kind of approaches use traditional relational technology, while other approaches use special data structures for representation of XML data.

The first category comes from the idea that semi-structured data can be stored as a ternary relation. However that straightforward mapping is not efficient. In these approaches proprietary repositories are used to store the document schema and data. Such solutions provide a large flexibility but sometimes they incur space and time costs because of replicating the schema and processing replicated schema. The classical research based on relational decomposition of XML document is STORED [8]. It attempts to generate a good relational schema automatically for the existing XML data instance. The generated STORED mapping may not capture the whole data instance. Portions of data which do not fit into the relational schema are stored in an overflow graph which can be implemented by non-relational storage or by the simple ternary relation. Mapping algorithms rely on the notions of storage patterns and pattern support. Storage pattern consists of a prefix path and a set of attributes and sub-elements which is called the body. Pattern support is the number of XML elements reachable in the document by the pattern prefix path and containing nodes described by the pattern body. The algorithm calculates the patterns with the highest support and then it groups them to satisfy the restrictions on the number of tables, attributes and total disk space. Finally patterns are translated into STORED mappings and the accompanying overflow mappings.

An XML indexing and retrieval model called Hybrid storage model [16] applies different methods to indexing text according to element content and attribute value. In principle, the first one is realized as a full-text index while the second one is implemented using a conventional relational DBMS. The resultant hybrid system architecture called XRS-II brings to the user possibility of issuing complex queries which combine similarity queries used in classical full-text search engines as well as attribute matching used in relational databases.

As was mentioned above the rest of approaches use special data structures as a trie (e.g. Index Fabric [6] and DataGuide [15]), multi-dimensional data structures (e.g. XPath Accelerator [9] or [13]), or signature data structures [20]. The XPath Accelerator may employ multi-dimensional structures to store 5-dimensional points which represents XML nodes. As such, the index is capable to support all XPath axes. The index can be implemented and queried using purely relational techniques, but it performs especially well if the underlying RDBMS provides support for R-trees. XPath accelerator use Dietz's numbering scheme. Each node v is represented by its 5-dimensional descriptor: $desc(v) = < pre(v), post(v), par(v), att(v), tag(v) >$, where $pre(v)$ and $post(v)$ are members of Dietz's numbering scheme, $par(v)$ represents $pre(v)$ of parent applicable node, att(v) indicate attribute or element and $tag(v)$ contains a unique number of name element or attribute. Preorder and postorder of a node h represents the plane which is divided by certain point corresponds with node h to four partitions. The lower-right partition contains all *descendants* of h, in the upper-left partition we find the *ancestors* of h, the lower-left region hosts the node *preceding* h in document order, and finally the upper-right partition represents the nodes *following* h in document order. That means then only these two coordinates of 5-dimensional points allow effective querying to 7 axis of XPath (descendant, descendant-or-self, parent, ancestor, ancestor-or-self, following, preceding) and with $par(h)$ and $attr(h)$ allow efficient querying of all XPath axis.

Some approaches like [13] employ data structures like R-tree [10] and UB-tree [3]. These approaches are based on the idea that the root to a leaf path may be represented as a point of a vector space. Such an approach is more efficient in that case of long path queries than XPath Accelerator. We see it is important to know the element depth distribution for purpose of efficiency of query performance.

3 Overview of Tools: Rainbow and AmphorA

3.1 The Rainbow project

The *Rainbow*[4] project represents a family of more-or-less independent web-mining projects undertaken by our research group[5]. Their unifying principles are commitment to *web-service* (WSDL/SOAP) front-end and agreement on a shared *upper-level ontology*. Furthermore, for each *application*, the developers involved also agree on a *domain* and share the source of training/testing *data*. Otherwise, the *formal principles* of analysis methods vary (from linguistic through statistical to e.g. graph theory), and so does the *representation of data*, also nicknamed as 'web view' (such as free text, HTML trees or link topology). In this way, the natural complementarity and/or supplementarity of information inferable from different types of web data can be exploited. Three application areas have been attacked so far: recognition of web *pornography*, extraction of information about *companies* and extraction of product offers from *bicycle* catalogues. More information can be found at the project homepage http://rainbow.vse.cz.

Information extraction plays an important role in *Rainbow*. Beside statistical (Hidden Markov Models) and shallow linguistic approaches to IE, we recently started to examine systematic use of higher-level *visual patterns* corresponding to different 'symbol-level' structures in HTML code. For example, the assignment

[4] Stands for 'Reusable Architecture for INtelligent Brokering Of Web information access'. Beyond the acronym (shared with a host of other research projects), the name is motivated by the idea that multiple independent tools for analysis of web data should synergistically 'shed light' on the web content, in a similar way as the different colours of the rainbow join together to form the visible light.

[5] Knowledge engineering group at the University of Economics, Prague.

of 'value' to 'variable' (or 'object' to 'predicate') with respect to a subject entity can be expressed by means of a row/column in a table, heading + list item, heading + short paragraph or the like [17]. The task of collecting candidate statements (say, RDF triples) of different kind from a complex website may be significantly eased by the existence of optimised XML index queriable via keywords and returning a 'reasonable' XML environment of the keywords rather than the whole HTML document.

3.2 Description of the AmphorA Web Service

The AmphorA applies the multi-dimensional approach to indexing XML data [13] for efficient querying such data. We support a subset of XPath for querying XML (XHTML as well) documents which are stored in XML index. A subset contains regular path expressions, predicate filters to elements and keyword-based searching (the operator $\sim=$ is applied).

The Amphora WS is implemented in C# language under .NET platform. The multidimensional approach for indexing XML is implemented in C++ and based on ATOM (Amphora Tree Object Model) framework which allows implementation of persistent data structures. The web service provides several methods which allow storage and querying web documents. Supported are *Index, DatabaseList, ResourceList* and *Query* methods. The `Index` method allows indexing of web sites. Parameter of that method is the URL of the root web page and result is unique number of site in database. The HTML pages are transformed to XML and indexed by the multi-dimensional approach. The `DatabaseList` method returns a string with unique numbers of web sites indexed in database. The `ResourceList` method with one parameter *dbId* which specifies unique number of web site returns the list of web page URLs within the scope of the web site. The `Query` method allows to query the specified database. The first parameter is a *dbId* of database and the second one is a string representing a query in a subset of the XPath language.

3.3 Coupling *Rainbow* with *AmphorA*

At the moment, only a small fragment of the functionality of *AmphorA* is available via web services and exploited by *Rainbow*, namely, the provision of whole documents and of lists of hyperlinks. However, we plan to couple both tools in a more efficient fashion. Knowledge-based methods such as those developed in *Rainbow* can achieve very high precision in restricted domains such as extraction of product information from company web pages. They are however slow and suffer from performance degradation beyond their specific domain. An XML tool such as *AmphorA* can potentially provide input data of adequate type and size, i.e. it can act as sophisticated pre-processor for information extraction methods. The starting point for such data supply can be keyword-based and/or XML-structure-based search. For example, if the goal is to extract product information from company sites, AmphorA could return tables where a heading contains keywords like 'price' or a column contains currency symbols, but also e.g. bulleted lists following a paragraph containing keywords like 'products' or 'offer'.

The critical point of the approach is however the amount of information provided as XML environment of the start-up terms/elements. Too narrowly defined environments may lead to missed context data that would subsequently be needed by the IE tool. On the other hand, too broadly defined environments may easily encompass the whole or most of the document, thus negating the benefits of pre-processing.

The first, very rough approximation of adequacy for XML environment may merely be inferred from the typical depth of interesting HTML elements (which are likely to contain the text to be extracted). For this purpose, we carried out a statistical analysis of depth distribution in the WebTREC dataset, described in the following section.

4 Experimental Results

The WebTREC [1] collection is used to analyse the depth distribution of HTML elements. The documents in collection include the information returned by the http daemon (enclosed in `DOCHDR` tags) as well as the page content. The collection consists of websites or gathered from the *.gov* domain (early 2002). Download has been stopped after 1 million text/html pages. Collection also included text/plain and the extracted text of pdf, doc and ps. The documents were truncated to 100k (reducing size from 35G to 18G). The collection comprises 1,247,753 (1.25 million) documents, includes 1,053,372 documents with text/html mime type. The total size of the collection is 19,455,030,550 B = 18.1 G (without 100 k limit was 35.3 G). Because such web pages are HTML pages (not XHTML) we need to transform the pages in the XML (add missing tags and so on).

HTML pages are inserted into an XML document using tags `DOC`, `DOCNO`, and `DOCHDR`. Such tags are missing in the following tables. In Tables 1– 3 we can see element depth distribution in the Level 1, 2, and 3, respectively, of the XML tree. In Table 1 we see element distribution in the first level of the XML tree. A large amount of `html` tag was expected. Non-zero number of other tags gives evidence to anomalies in HTML pages.

Table 1. Element Distribution in the Level 1 of WebTrec Collection

Tag	%	**Tag**	%	**Tag**	%	**Tag**	%	**Tag**	%
html	95.03	body	0.81	center	0.27	table	0.13	ul	0.13
title	0.94	h1	0.4	link	0.27	a	0.13	base	0.13
head	0.81	br	0.4	p	0.13	hr	0.13	pre	0.13

Now in Tables 4– 6 we present the depth distribution of the element type `td`, `p`, and `li`, respectively.

Table 2. Element Distribution in the Level 2 of WebTrec Collection

Tag	%	**Tag**	%	**Tag**	%	**Tag**	%	**Tag**	%
head	43.14	html	0.99	a	0.6	center	0.4	li	0.2
body	38.17	div	0.8	br	0.6	tr	0.2	b	0.2
frameset	4.37	h1	0.8	hr	0.6	img	0.2	header	0.2
script	1.79	link	0.8	p	0.4	h2	0.2		
noframes	1.39	td	0.6	table	0.4	font	0.2		

Table 3. Element Distribution in the Level 3 of WebTrec Collection

Tag	%	**Tag**	%	**Tag**	%	**Tag**	%	**Tag**	%
title	14.8	a	4.87	font	2.88	style	1.19	tr	0.89
table	14.10	script	4.67	frame	2.09	form	1.09	h3	0.70
meta	14.00	div	4.57	hr	1.99	h1	1.09	head	0.60
p	9.33	map	3.48	body	1.39	noscript	1.09	layer	0.60
center	5.56	br	3.38	link	1.29	img	0.89		

Table 4. Depth Distribution of the element type `td`

Level	%	**Level**	%	**Level**	%	**Level**	%
0	0	3	4.22	6	15.64	9	9.42
1	0	4	7.05	7	20.03	10	7.02
2	0	5	8.39	8	16.18	11	5.70

Table 5. Depth Distribution of the element type `p`

Level	%	**Level**	%	**Level**	%	**Level**	%
0	0	3	7.61	6	13.94	9	5.76
1	2.05	4	8.58	7	16.99	10	5.04
2	4.10	5	10.65	8	11.89	11	3.89

Table 6. Depth Distribution of the element type `li`

Level	%	**Level**	%	**Level**	%	**Level**	%
0	0	3	5.16	6	11.28	9	15.42
1	0	4	6.49	7	13.75	10	9.86
2	2.16	5	7.44	8	17.03	11	7.4

The most typical sources for web information extraction are unstructured text (typically in `p` elements), tabular data (in `td` elements), and the content of lists (in `li` elements). We can see that these elements most often occur between levels 6–10 of the HTML structure. We can thus hypothesize that most adequate XPath queries should retrieve the XML environment of about 2-3 levels up from the indicative lexical item (and not reaching above level 3), to avoid returning the whole (or most) of the document. More detailed experiments are however obviously required to validate this hypothesis.

5 Conclusion

The long-term goal of our project is to use an XML-oriented IR tool for fine-grained pre-processing of source data for information extraction. The presented analysis of HTML element depth is a very humble first step in this direction. The next and more demanding step will be the search for frequent HTML sub-trees with respect to particular levels. We plan to develop the XML query functionality in parallel with research on visual patterns, which can be both 'relational' (e.g. the above mentioned RDF triples) or hierarchical as described by Burget [5].

References

1. TREC-2002 Web Track Guidelines, `http://es.csiro.au/trecweb/guidelines_2002.html`, 2002.
2. R. Baeza-Yates and B. Ribiero-Neto. *Modern Information Retrieval.* Addison Wesley, New York, 1999.
3. R. Bayer. The Universal B-Tree for multidimensional indexing: General Concepts. In *Proceedings of World-Wide Computing and Its Applications'97 (WWCA'97), Tsukuba, Japan*, Lecture Notes in Computer Science. Springer–Verlag, 1997.
4. A. Berglund, S. Boag, D. Chamberlin, M. F. Fernandez, M. Kay, J. Robie, and J. Siméon. XML Path Language (XPath) 2.0, W3C Working Draft, Version 2.0. Technical report, WWW Consortium, December, 2001, `http://www.w3.org/TR/xpath20/`.
5. R. Burget. Hierarchies in HTML Documents: Linking Text to Concepts. In *15th Int'l Workshop on Database and Expert Systems Applications*, pages 186–190, Zaragoza, 2004.
6. B. Cooper, N. Sample, M. J. Franklin, G. R. Hjaltason, and M. Shadmon. A Fast Index for Semistructured Data. In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB'01)*, pages 341–350. Morgan Kaufmann, 2001.
7. A. Deutsch, M. Fernandez, D. Florescu, A. Levy, and D. Suciu. XML-QL: A Query Language for XML. Technical report, WWW Consortium, August, 1998, `http://www.w3.org/TR/NOTE-xml-ql/`.
8. A. Deutsch, M. Fernandez, and D. Suciu. Storing semistructured data with STORED. In *Proceedings of 1999 ACM SIGMOD International Conference on Management of Data*, pages 431–442. ACM Press, 1999.
9. T. Grust. Accelerating XPath Location Steps. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, Madison, USA*. ACM Press, June 4-6, 2002.

10. A. Guttman. R-Trees: A Dynamic Index Structure for Spatial Searching. In *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data, Annual Meeting, Boston, USA*, pages 47–57. ACM Press, June 1984.
11. C. Knoblock, S. Minton, J. Ambite, N. Ashish, P. Modi, I. Muslea, A. Philpot, and S.Tejada. Modeling Web Sources for Information Integration. In *Proc. AAAI-98*, Madison, WI, 1998.
12. M. Krátký and M. Andrt. On Efficient Part-match Querying of XML Data. In *Proceedings of DATESO 2004*, 2004.
13. M. Krátký, J. Pokorný, and V. Snášel. Implementation of XPath Axes in the Multi-dimensional Approach to Indexing XML Data. In *Current Trends in Database Technology, International Workshop on DataX, EDBT 2004*, volume 3268 of *Lecture Notes in Computer Science*. Springer–Verlag, 2004.
14. N. Kushmerick, D. S. Weld, and R. Doorenbos. Wrapper Induction for Information Extraction. In *Intl. Joint Conference on Artificial Intelligence (IJCAI)*, 1997.
15. J. W. R. Goldman. DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases. In *Proceedings of 23rd International Conference on Very Large Data Bases (VLDB'97)*, pages 436–445. Morgan Kaufmann, 1997.
16. D. Shin. XML Indexing and Retrieval with a Hybrid Storage Model. *Knowledge and Information Systems*, 3(2), 2001.
17. V. Svátek, J. Bráza, and V. Sklenák. Towards Triple-Based Information Extraction from Visually-Structured HTML Pages. In *Poster Track of the Twelfth International World Wide Web Conference (WWW 2003)*, Budapest, 2003.
18. W3 Consortium. Extensible Markup Language (XML) 1.0, W3C Recommendation, 10 February 1998, `http://www.w3.org /TR/REC-xml`.
19. W3 Consortium. XQuery 1.0: An XML Query Language, W3C Working Draft, 12 November 2003, `http://www.w3.org/TR/xquery/`.
20. P. Zezula, G. Amato, F. Debole, and F. Rabitti. Tree Signatures for XML Querying and Navigation. In *Proceedings of XML Database Symposium, XSym 2003*, volume 2824 of *Lecture Notes in Computer Science*, pages 149–163. Springer-Verlag, 2003.