Dynamic Element Retrieval in a Semi-structured Collection

Carolyn J. Crouch, Donald B. Crouch, Murthy Ganapathibhotla, and Vishal Bakshi

Department of Computer Science
University of Minnesota Duluth
Duluth, MN 55812
(218) 726-7607
ccrouch@d.umn.edu

Abstract. This paper describes our methodology for the dynamic retrieval of XML elements, an overview of its implementation in a structured environment, and the challenges introduced by applying it to the INEX Wikipedia [4] collection, which can more aptly be described as semi-structured. Our system is based on the vector space model [9] and its basic functions are performed using the Smart experimental retrieval system [8]. A major change in the system this year is the incorporation of a method for the dynamic computation of query term weights [6] to be correlated with the dynamically generated and weighted element vectors. Dynamic element retrieval requires only a single indexing of the document collection at the level of the basic indexing node (in this case, the paragraph). It returns a rank-ordered list of elements equivalent to that produced by the same query against an all-element index of the collection. (A detailed description of this method appears in [1].) As we move from a well structured collection, such as the INEX IEEE documents, to Wikipedia, changes in the structure of the articles must be accommodated.

1 Introduction

When we began our work with INEX in 2002, our goal was to assess the utility of Salton's vector space model [9] for XML retrieval. Familiarity with Smart [8] and faith in its capabilities led us to believe that this approach was promising if problems such as flexible retrieval (i.e., retrieval of elements at the desired degree of granularity) and ranking issues could be resolved. For the past several years, our research has focused on an approach for the dynamic retrieval of elements which provides a solution to both these problems.

The evolution of this approach is described in our earlier workshop papers, in particular [2] and [3]. In dealing with the INEX IEEE collections, we utilized Fox's extended vector space model [5], which allows for the incorporation of objective identifiers (such as date of publication) along with the normal content identifiers in the representation of a document. The body portion of the document (i.e., its text) is represented by one of the subjective subvectors in the extended vector representation.

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 82–88, 2007.

The INEX Wikipedia collection [4] does not carry with it the corresponding information. Wikipedia articles are easily represented within the traditional vector space model, as seen below.

In INEX 2006 we use a system which generates and retrieves elements dynamically and returns a rank-ordered list of elements to the user. Results published elsewhere have demonstrated the successful utilization of this approach for structured retrieval [1]. Our current investigations center on how best to employ this approach in dealing with semi-structured data.

2 Background

This section presents a brief overview of the model and term weighting method upon which our system is based—i.e., the vector space model and *Lnu-ltu* term weighting. Details of this weighting scheme may be found in [10,11]. It is of particular interest in element retrieval where the elements often vary considerably in length, depending on type (e.g., paragraph versus section and body). *Lnu-ltu* weighting attempts to deal with the ranking issues resulting from disparity in document (element) length. (See [1] for a more detailed discussion of this issue.)

A basic model in information retrieval is the vector space model [9], wherein documents and queries are represented as weighted term vectors. The weight assigned to a term is indicative of the contribution of that term to the meaning of the document. The similarity between vectors (e.g., document and query) is represented by the mathematical similarity of the corresponding term vectors.

Of particular interest in this work is the issue of term weighting. We found in earlier experiments that best results were achieved when *Lnu-ltu* term weighting [11] was used with inner product as the similarity measure. *Lnu* term weights, used for the element vectors, are defined below:

$$\frac{(1+\log(term\_frequency)) \div (1+\log(average\_term\_frequency))}{(1-slope)+slope\times((number\_unique\_terms) \div pivot)}$$

where *tf* represents term frequency, slope is an empirically determined constant, and pivot is the average number of unique terms per document, calculated across the entire collection. Query terms are weighted using the *ltu* formula, as follows.

$$\frac{(1+\log(term\_frequency))\times\log(N \div n_k)}{(1-slope)+slope\times((number\_unique\_terms) \div pivot)}$$

Note that this formula depends both on N (the collection size) and n_k (the number of elements that contain the term).

3 System Description

This section describes first the current operation of the system and then a particular problem of interest that was solved during the past year by Ganapathibhotla—a method for the dynamic *ltu*-weighting of the query [6].

3.1 Dynamic Element Retrieval

XML text is processed in this system as follows. The documents are parsed using a simple XML parser which we wrote. We selected the paragraph—in our view, the smallest meaningful unit of text—as our basic indexing unit in the early stages of investigation. Thus a parsing of the documents into paragraphs is produced: paragraphs and queries are translated into Smart format and indexed by Smart. *Lnu-ltu* term weighting is applied. Retrieval takes place by running the *ltu*-weighted topics against the *Lnu*-weighted paragraph indexing of the collection using Smart. The result is a list of *elements* (paragraphs) ordered by decreasing similarity to the query.

Consider all the elements in this list having a non-zero correlation with the query. Each such element represents a terminal node (or paragraph) in the body of a document with some relationship to the query. Please note that although we have used the term *paragraph* here as a designator for smallest meaningful unit, in this context it means all the leaf nodes of a document tree. Thus the term *paragraph* can be used to refer to figure captions, lists, section titles, tables, abstracts—all the content-bearing elements that partition the document into mutually exclusive parts. Although some of these elements may not be leaf nodes according to their DTDs, they are treated as leaf nodes in this context because their child nodes are too small to be meaningful units in themselves.

For a particular query, Q, a search by Q against the paragraph index identified above produces a rank-ordered list of elements. Those elements having a positive correlation with Q identify the set of all documents of possible interest to it. We consider the n top-ranked elements in this list. Our method of dynamic element retrieval builds a tree representation for each document having an element in this list. Each tree is built based on a schema of the document (produced as a by-product of parsing). Given its set of terminal nodes in the form of term-frequency vectors, a document tree is built, bottom-up, according to its schema [3,7]. The content of each internal node is based solely on the content of its children. As each element vector is produced, it is *Lnu*-weighted and correlated with Q, which is itself *ltu*-weighted. After all element vectors, including the body element, have been generated, weighted and correlated with Q, the process continues with the next document. The resulting set of element vectors (i.e., all the elements from each document with a terminal node in the set of n top-ranked elements retrieved by Q) are then sorted and the top-ranked elements are reported.

3.2 Dynamic Query Weighting

Consider the formulas for term weighting given in Section 2. The *Lnu* term weighting of the element vectors at execution time is a relatively simple process. *Lnu* weights do not require information on global frequency that is not available in the dynamic

environment. The values of slope and pivot having been previously determined for the collection, *Lnu* weights are quickly computed.

The situation with the dynamic computation of the query term weights is quite different. Consider the *ltu* formula. At each level in the document tree, the values of N and n_k are element-dependent. Dynamic element retrieval is based on an initial retrieval against a paragraph indexing of the collection. The values of N (the number of paragraphs in the collection) and n_k (the number of paragraphs containing the term) are readily available as a by-product of the indexing process. In order for Q to be correctly weighted and correlated against each element vector in the document tree, the values of N and n_k associated with each query term must be the corresponding global values (i.e., the number of *elements* in the collection and the number of *elements* containing the query term).

The value of N is easily supplied by keeping track of the various types of elements encountered during the parsing process. Obtaining the value of n_k associated with a specific query term is more challenging. We have its (local) value at the level of the terminal node (i.e., paragraph). To determine its global value (i.e., the number of elements containing the term), we need information about the structure of each document tree in which the term is contained. In particular, for each occurrence of the term as a word type in a terminal node, we need to determine the number of parent elements in which it occurs. For example, suppose query term t_1 occurs in two different paragraphs of the same document. We need know whether both paragraphs occur as children of the same parent node (say subsection) or as children of different parents (two different subsections), and so on up the tree. And this process must be repeated for every document tree which contains t_1.

A very clever way to determine the number of containing elements for a particular term was devised by Ganapathibhotla [6], using the inverted file entry associated with the term in the paragraph indexing and a mapping between paragraph identifiers and their xpaths (required in our system for interaction between Smart and INEX formats). (See [1], [6] for details.) The calculation of n_k at execution time, clearly not feasible if it were required in the weighting of element vectors, is quite feasible in the weighting of query vectors, which are by their nature very short in comparison.

3.3 What About *n*?

There are very few parameters of interest associated with our method of dynamic element retrieval. Slope and pivot, used in the *Lnu-ltu* term weighting scheme, are collection dependent; determining slope requires some investigation and tuning. But the only truly interesting parameter (in the sense that it determines the number of trees generated and hence largely the time required for dynamic element retrieval) is n—the number of top-ranked paragraphs fed to our dynamic retrieval routine. It determines the upper bound on the number of trees built for each query. The actual number is determined by the number of paragraphs in this set belonging to the same document or set of documents.

Although not reported here in detail, our experiments with the 2004 and 2005 INEX IEEE collections reveal some interesting results. In these experiments, n varied

from 1 to 1000 (specifically, n = 1, 5, 10, 25, 50, 100, 250, 500, 1000). For the 2004 collection, under both generalized and strict quantizations and considering values of P@n for 10, 20, 50, 100, 500, and 1500 and average precision, dynamic element retrieval never required a value of n greater than 100 to produce a result equivalent to retrieval against the all-element index. For the 2005 collection, the results were very similar. The average number of trees built per query at n = 100 was 64 (for 2004) and 66 (for 2005). Looking at the average number of trees generated per query over all specified values of n greater than 50 indicates that, on average, fewer than 2/3 n trees are actually built.

For reasons discussed in the following section, corresponding experiments for the INEX Wikipedia collection are still in progress.

4 Problems Posed by Semi-structured Data

We encountered some interesting problems in adapting our method for dynamic element retrieval to the INEX Wikipedia collection. The IEEE collections are well structured. We found these traditional documents could be represented quite naturally using Fox's extended vector space model. Wikipedia documents, on the other hand, are easily represented using the traditional vector space model. The really significant difference between these two collections from our point of view, however, lies in how they are structured. Dynamic element retrieval depends on having all the terminal nodes of a document represented in the paragraph index. The initial paragraph retrieval gives us a good indication of which documents are of interest to the query in this case because all paragraphs that correlate highly with it are identified (thereby identifying their parent documents). The Wikipedia collection, on the other hand, contains untagged text which is distributed throughout the documents at the body and section levels. This untagged text cannot be retrieved except as a component of its parent element.

This is not a problem with respect to retrieval from an all-element index. The elements are parsed, collected, and indexed. Retrieval takes place in the normal manner. With dynamic element retrieval, untagged text impacts the method at two points: (1) during parsing, when untagged text must be identified (to be subsequently used in generating the document schemas so that the bottom-up generation of the document tree can take place properly with untagged text included at its parent level); and (2) during the initial retrieval against the paragraph (or terminal node) index, when documents potentially important to the query are identified. The interesting question here, which we have yet to answer, is whether the untagged text is important from a retrieval viewpoint.

Our current methodology deals with this problem by gathering all untagged text within an element and treating it as a separate child element (equivalent to a paragraph element) of its parent. Thus untagged text within a section becomes a separate element, (specially tagged as an <mt> element), which is attached to its parent section, and untagged text at the body level is treated similarly and attached as a child of the body element. Using this approach, dynamic element retrieval can proceed in the same manner used for structured text. The issue of interest here is

whether the inclusion of <mt> elements with the paragraph elements in the initial retrieval materially affects the elements retrieved dynamically and if so, to what extent.

Experiments are currently underway to determine the answers to this and other, related questions. The system requires tuning against the relevance assessments to determine an appropriate value of slope in the *Lnu-ltu* term weighting formula; the process, while straight-forward, is time-consuming. The size of this collection is also a factor. We have faced a number of space-related issues and hardware failures which have deterred progress on this work.

5 Conclusions

Our current system has achieved its major goal—it retrieves elements dynamically and returns a rank-ordered list of elements equivalent to that retrieved by a search of the corresponding all-element index. Exact *Lnu-ltu* term weights are utilized in this process. It requires only a single indexing of the collection at the paragraph level rather than either an all-element or multiple indexings, which are expensive to produce and maintain. As [1] shows, this method works well for structured retrieval. As we adapt our methods for utilization in the semi-structured environment of the INEX Wikipedia collection, we aim to determine the impact of this structural change on the retrieval process.

References

[1] Crouch, C.: Dynamic element retrieval in a structured environment. ACM Transactions on Information Systems 24(4), 437–454 (2006)

[2] Crouch, C., Mahajan, A., Bellamkonda, A.: Flexible retrieval based on the vector space model. In: Fuhr, N., Lalmas, M., Malik, S., Szlávik, Z. (eds.) INEX 2004. LNCS, vol. 3493, pp. 292–302. Springer, Heidelberg (2005)

[3] Crouch, C., Khanna, S., Potnis, P., Daddapaneni, N.: The dynamic retrieval of XML elements. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, pp. 268–281. Springer, Heidelberg (2006)

[4] Denoyer, L., Gallineri, P.: The Wikipedia XML corpus. In: INEX Workshop Pre-Proceedings, pp. 367–372. (2006) http://inex.is.informatik.uni-duisberg.de/2006

[5] Fox, E.A.: Extending the Boolean and vector space models of information retrieval with p-norm queries and multiple concept types. Ph.D. Dissertation, Department of Computer Science, Cornell University (1983)

[6] Ganapathibhotla, M.: Query processing in a flexible retrieval environment. M.S. Thesis, Department of Computer Science, University of Minnesota Duluth, Duluth, MN (2006) http://www.d.umn.edu/cs/thesis/Ganapathibhotla.pdf

[7] Khanna, S.: Design and implementation of a flexible retrieval system. M.S. Thesis, Department of Computer Science, University of Minnesota Duluth, Duluth, MN (2005) http://www.d.umn.edu/cs/thesis/khanna.pdf

[8] Salton, G. (ed.): The Smart Rretrieval System—Experiments in Automatic Document Processing. Prentice-Hall, Englewood Cliffs (1971)

[9] Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Comm. ACM 18(11), 613–620 (1975)
[10] Singhal, A.: AT&T at TREC-6. In: The Sixth Text REtrieval Conf (TREC-6), NIST SP 500-240, pp. 215–225 (1998)
[11] Singhal, A., Buckley, C., Mitra, M.: Pivoted document length normalization. In: Proc. of the 19th Annual International ACM SIGIR Conference, pp. 21–29 (1996)