STRUCTURE-BASED INTERROGATION AND AUTOMATIC QUERY REFORMULATION

Mohamed Ben Aouicha, Ines Kamoun, Mohamed Tmar and Abdelmajid Ben Hamadou
MIRACL, Sfax University, Sfax, Tunisia

Keywords: XML retrieval, XML query optimization, Virtual link, INEX.

Abstract: This paper presents an information retrieval model on XML documents based on tree matching. Queries and documents are represented by extended trees. An extended tree is built starting from the original tree, with additional weighted virtual links between each node and its indirect descendants allowing to directly reach each descendant. Therefore only one level separates between each node and its indirect descendants. This allows to compare the user query and the document with flexibility and with respect to the query structural hints. The content of each node is however very important to decide whether a document element is relevant or not, thus the content should be taken into account in the retrieval process. We separate between the structure-based and the content-based retrieval processes. The structure notion should be taken into account during the retrieval process as well as during automatic query reformulation. We propose an approach to automatic query reformulation starting from the original query on one hand and the fragments judged relevant by the user on the other. Structure hints analysis allows us to identify nodes that match the user query and to rebuild it during the automatic query reformulation step. The main goal of this paper is to show the impact of structural hints in XML retrieval and XML query optimization. Some experiments have been undertaken into a dataset provided by INEX[a] to show the effectiveness of our proposals.

[a]INitiative for the Evaluation of XML retrieval, an evaluation forum that aims at promoting retrieval capabilities on XML documents.

1 INTRODUCTION

The standardization of the Web to XML schemas presents new problems and hence new needs for customized access to information. Being a very powerful and often unavoidable tool to customized access to information of all kinds, information retrieval systems arise at the forefront of this issue.

Many traditional information retrieval systems (IRS) have been adapted to handle XML retrieval. While both text and structure are important, IRS usually give higher priority to text when ranking XML elements. IRS adapt unstructured retrieval methods to handle additional structural constraints. Such approaches are called text centric XML retrieval.

The traditional information retrieval systems handle documents with different formats but do not exploit the structure of documents, including the reformulation phase of the query. However, a structured document is characterized by its content and structure. This structure, as defined by elements, possibly completes semantics expressed by the content and becomes a constraint with which information retrieval systems must comply in order to satisfy the user information needs.

Indeed, the user can express his need by a set of keywords, as in the traditional information retrieval systems, and can add structural constraints to better target the sought semantics.

Thus, taking into account the structure of the documents and that of the query by the information retrieval systems handling structured documents is necessary not only in the retrieval process but also in the query reformulation one.

We propose in this paper to evaluate the impact of structure handling in both the interrogation and the query reformulation process. The structure hints in the user query are taken into account at first (before any content treatment) in interrogation process, and in the query reformulation process, the query structure could be devoted to some modification based on the structure of the relevant judged document fragments. thus, we put the emphasis on the structure by analyzing the structure features and relation that are

Ben Aouicha M., Kamoun I., Tmar M. and Ben Hamadou A..
STRUCTURE-BASED INTERROGATION AND AUTOMATIC QUERY REFORMULATION.
DOI: 10.5220/0003624601230128
In *Proceedings of the International Conference on Knowledge Management and Information Sharing* (KMIS-2011), pages 123-128
ISBN: 978-989-8425-81-2

the most significant to the interrogation and the query reformulation process.

This paper is organized into six sections. The second section gives a survey of related work in XML retrieval. The third section presents our XML retrieval model. In the fourth section, we present an automatic query reformulation approach. The fifth section presents the experiments and the obtained results. The sixth section concludes.

2 RELATED WORK IN XML RETRIEVAL AND AUTOMATIC XML QUERY REFORMULATION

In traditional information retrieval (IR), the documentary unit from which the terms are extracted and which will be presented to the user is the entire document. But in the context of XML retrieval the documentary unit can have different forms. It can be the entire document (any element of the document) or an item or subtree of an XML document. Thus XML retrieval systems must take into account this new dimension during the interrogation of any XML corpus and during the query reformulation.

Several teams have adapted classical IR methods to XML retrieval. The vector-space based XML retrieval method proposed by defines each dimension by sub-trees that contain at least one indexing term. Queries and documents are then represented by vectors in this space and the system computes matches between them using well-known vector-space similarity measures (Cosine, Dice, Overlap ...). Schlieder and Meuss (Schlieder and Meuss, 2002) describe similar approaches. They proposed the ApproXQL model, which integrates the document structure in the vector space model similarity measure. The query model is based on tree matching: it rewrites the queries and the documents independently and then performs XML retrieval based on the traditional vector space model properties.

Fuhr (Fuhr and Grossjohann, 2001) uses a query language XIRQL that combines the structural and the content based approach. It integrates features related to data-centric by using ideas from logic-based probabilistic Information retrieval models, in combination with concepts from the database area. Balog et al. propose a general probabilistic framework for entity search to evaluate and provide insight in the many ways of using these types of input for query modelling (Balog et al., 2010). They focus on the use of category information and demonstrates the effectiveness of category-based expansion using example entities.

3 OUR XML RETRIEVAL MODEL

We present in this section an XML retrieval model based on tree flexible matching. Instead of flexible matching on the original representations of the query and the document, we apply an exact match method on flexible representations of them.

Formally, an XML tree is a set of node paths $A \longrightarrow B$ where node A is the parent of node B. The XML tree root is the only one that has no parent. So an XML tree T should have the following property:

$$\{N, \forall N', N' \longrightarrow N \notin T\} = \{root\} \quad (1)$$

where N and N' represent XML nodes in tree T.

A fragment can be defined by the result of the inner join of two paths. For example, if an XML tree contains paths $A \longrightarrow B$ and $B \longrightarrow C$, we can define another path $A \longrightarrow B \longrightarrow C$. A path is then an ordered list of nodes $N_1 \longrightarrow N_2 \longrightarrow \ldots \longrightarrow N_n$. No cycles have to occur in a tree path, thus if $N_1 \longrightarrow N_2 \longrightarrow \ldots \longrightarrow N_n$ is a path resulted from the tree T, all N_i should be different. This brings to another property which requires that each node but the root has only one parent:

$$\forall N, |\{N' \longrightarrow N \in T\}| = \begin{cases} 0 & if\ N = root \\ 1 & otherwise \end{cases} \quad (2)$$

An XML tree is then a set of paths that fulfills the properties 1 and 2.

The structure retrieval can be viewed as comparing between the structures of two XML trees. Comparing between trees was initially introduced by the tree to tree correction theory (Selkow, 1977). To compare between two trees, we have a tendency to build a tree starting from the second tree and then compute correction costs. The correction process considers hidden relations between nodes. For example, if node A is the parent of node B which is the parent of node C in the tree T, and node A is the parent of node C in the tree T', the T to T' correction can be held by removing node B and linking node A with node C. Each update operation has a cost and the tree to tree correction cost (the edit distance) is the total cumulated cost of the necessary correction operations. The total cost c of tree to tree correction can be viewed as the inverse of the similarity between both of them. Typically, if $c = 0$, this means that no correction operation is needed to build a tree starting from the other, so they are quite similar.

This process can be applied to estimate the structure similarity between two XML documents, or an

XML document and an XML query. The main motivation of its application is that it allows to perform structure comparison, which is necessary in XML retrieval and second, it provides a ranked list of document trees (or document sub-trees), where the score is the inverse of the total cumulated cost of the document tree to the query tree correction (or the query tree to the document tree correction). However, due to some practice reasons, this cannot be applied to XML retrieval. The tree to tree correction algorithms are not scalable: they cannot be applied on XML retrieval on very large corpuses.

The structural retrieval process should look for the deepest and widest sub-tree shared by both representations (Selkow, 1977). Instead of applying a tree to tree flexible correction algorithm in the original trees, we use a dual process: an exact match algorithm in flexible representations of each tree. A flexible representation of an XML tree explores all hidden relations that could exist between nodes. We call such representation the extended tree. This operation is done since indexing the XML corpus.

To do so, we add to each path $A \longrightarrow B$ a weight reflecting the importance of the relation between nodes A and B. According to the parent-child relation, this weight is equal to 1. The more A is distant from B in the original path, the less its weight is. The weight depends on the distance between two nodes that occur in a path. We use the weighting function f defined by $f(A \longrightarrow B) = exp(1 - d(A,B))$ where $d(A,B)$ is the distance that separates node A from node B. We denote this path by $A \xrightarrow{w} B$ where $w = exp(1-d(A,B))$ is the weight of the path $A \longrightarrow B$.

We apply an exact match algorithm on the extended trees (Ben Aouicha et al., 2006) (Ben Aouicha et al., 2009). As the complexity of the exact match algorithms is less than the approximative matching algorithm one, we use this kind of algorithms to compare between two flexible representations. All the complexity is then moved at the representation phase, which corresponds to the indexing process. This phase is only performed once, and the interrogation process is in contrast performed by the application of an exact match algorithm on the extended trees representations which are stored in the index. Starting from tree T and tree T' this algorithm looks for the deepest and widest sub-tree shared by T and T' and computes the similarity depending on the weights of paths appearing in each one. Relevant fragments are those having similar structure than the user query. This can be done by looking in the extended document for fragments having exactly the same structure as the extended query or a part of it. This allows flexible XML retrieval (Bordogna and Pasi, 2000). The search strategy we adopt is iterative. We start from the extended document and query trees. We build a returned fragment incrementally, starting from potentially common roots, we build for each one its common child nodes, and then for each child node, we build its common child nodes and so on until leaf nodes. When building relevant candidate fragments, we compute the structure-based score by cumulating the product of the paths weights in the document and their matched ones in the query tree.

The final score is a linear combination of the structure-based score r_s and the content-based score r_c:

$$r(f_q, f_d) = \alpha \times r_c + (1-\alpha) \times r_s \qquad (3)$$

where α is a parameter that emphasizes the retrieval process on the structure constraints or the keywords. For our experiments, we use $\alpha = 0.6$.

4 AUTOMATIC QUERY REFORMULATION

In our approach we focus on the structure of the original query and that of document fragments deemed relevant to the user structure hints. Indeed, this study allows us to reinforce the importance of these structures in the reformulated query to better identify the most relevant fragments to the user's needs. The analysis of structures allows us to identify the most relevant nodes and the involved relationships.

4.1 Line of Descent Matrix

According to most approaches to automatic query reformulation, the query construction is done by building a representative structure of relevant objects and another structure for irrelevant ones, and then build a representation close to the first and farther from the second.

For example, the Rocchio's method (Rocchio, 1971) considers a representative structure of a document set by their centroid. A linear combination of the original query and the centroids of the relevant documents and irrelevant ones can be assumed as a potentially suitable user need.

We propose to traduce the documents and the query in a matrix format instead of a wighted term vector like in Rocchio's method which is more suitable for flat document. We build for each document a matrix called *line of descent matrix* (LDM), which must show all existing ties of kindship between different nodes. This representation should also reflect the

positions of the various nodes in the fragments as they are also important in the query reformulation. For an XML tree (or subtree) A, we associate the matrix defined by M_A:

$$M_A[n,n'] = \begin{cases} P & \text{if } n \to n' \in A (n \text{ is the parent of n'}) \\ 0 & \text{otherwise} \end{cases}$$

Where P is a constant value which represents the weight of the descent relationship.

As for us, we represent each of the relevant fragments and the initial query in the LDM form. The value of the constant P for the query LDM construction is greater than that used for the construction of other LDMs (which represent the relevant fragments) to strengthen the weight of the initial query edges following the principle used in the Rocchio's method which uses reformulation parameters having different effects (1 for the initial query, α for the relevant documents centroid and β for the non relevant documents centroid where $0 \leq \alpha \leq 1$ and $-1 \leq \beta \leq 0$). Note that no complexity analysis is here needed because of the low number of relevant judged documents comparing to the corpus size. In our experiments, we undertake the query reformulation in a pseudo-feedback way on the top 20 ranked documents resulting from the first round retrieval. In the other hand, the total number of tags is over 160 in all the collection and about 5 in a single fragment, so the matrix size can not exceed 25.

In a fragment can be returned even if the structural conditions of the query are not entirely fulfilled. This means that if a fragment of an XML document is similar but not identical to the query, it can be returned. The information retrieval systems now has to query with tolerated differences (a few missing elements or more additional ones) between the query structure and the document. Consequently, we believe that the most effective way to bring this tolerance is to assure that one element is not only connected to its child nodes, but to all of its direct and indirect descendants. A relationship between nodes in the same line of descent is weighted by their distance in the XML tree.

We propose the TC function which is a transitive closure on the weights of the nodes edges with a common ancestor. The resulted value will be added to the weight of the edge itself in the LDM as follows:

$$\forall (n,n',n'') \in N^3, M_A[n,n''] \quad \leftarrow M_A[n,n''] + TC(M_A[n,n'], M_A[n',n''])$$

With:

$$TC(x,y) = \frac{x \times y}{\sqrt{x^2+y^2}}$$

As for us, this transitive closure will be applied to each LDM of each fragment judjed as relevant and also to the LDM of the query.

The the reformulated query structure is built starting from the obtained LDMs. Let us consider $F = \{f_1, f_2 \dots f_n\}$ where f_i are the relevant judged fragments and Q_0 is the initial query, the query structure is built starting from the cumulated LDM S:

$$\forall (n,n')^2 \in B^2, S[n,n'] = \sum_{f \in F} M_f[n,n'] + M_{Q_0}[n,n']$$

4.2 Building the New Query Structure

The query reformulation starts by identifying its root. The root is characterized by a high number of child nodes and a negligible number of parents. For example, to find the root we simply return the element R, which has the greatest weight in the rows of the matrix S and the lowest weight in its columns. The root R is then such that:

$$R = \arg\max_{n \in B} \sum_{n' \in B} S[n,n'] . \log \left(\frac{\sum_{n' \in B} S[n',n]}{\sum_{(n',n'') \in B^2} S[n',n'']} + 1 \right)$$

The argument to maximize reflects that the candidate nodes to represent the root should have as maximal low values as possible in the relative row ($\sum_{n \in B} S[n,n']$) and as minimal low values as possible in the column ($\sum_{n' \in B} S[n',n]$) relatively to the total sum of the matrix values ($\sum_{(n',n'') \in B^2} S[n',n'']$).

Once the root has been established, we proceed to the recursive development phase of the tree representing the structure of the new query. The development of the tree starts by the root R, and then by determining all the child nodes of R, the same operation is performed recursively for the child nodes of R until reaching the leaves elements.

5 EXPERIMENTS AND RESULTS

Experiments have been undertaken into a dataset provided by INEX. The INEX metric for evaluation is based on the exhaustivity and specificity measures which are analogous to the traditional recall and precision measures (Piwowarski and Dupret, 2006). The specificity is an extent to which a document component is focused on the information need, while being an informative unit. The exhaustiveness is an extent to

which the information contained in a document component satisfies the information need. Each document component has one of the following values of exhaustivity (non exhaustive, slightly exhaustive, exhaustive very exhaustive) and specificity (between 0 and 1). These measures are quantized onto a single relevance value.

First experiments were undertaken to show the effectiveness of our XML retrieval approach. Figures 1, 2 and 3 show our results (bold curves) compared to official INEX obtained results by all participants on the *MAep* (mean average effort precision) quantization measure.

We experimented different values of α used in equation 3, the value that provides the best results is 0.6 which shows that the strcture-based retrieval contribution is over 40% of the whole XML retrieval process.

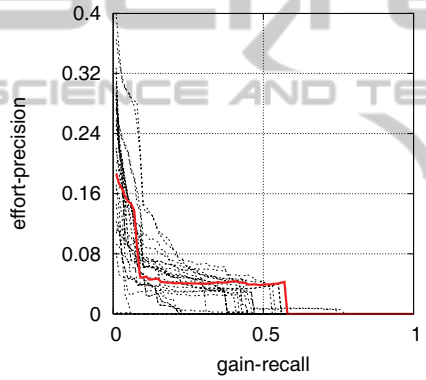

Figure 1: Comparative results with INEX official participants, generalized quantization function.

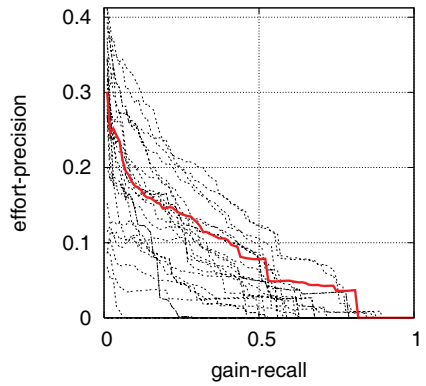

Figure 2: Comparative results with INEX official participants, generalized lifted quantization function.

It can be seen through figure 3 that we obtain better *MAep* results than all the INEX'2005 participants in the strict quantization functions. Figure 1 and 2 show that our system is very competitive to the

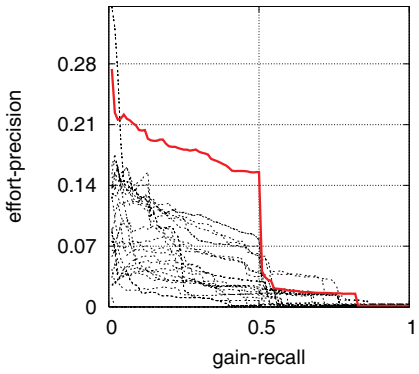

Figure 3: Comparative results with INEX official participants, strict quantization function.

INEX'2005 outsiders according to the other quantization functions.

At the other measures, the results are much closed to the best INEX'2005 official results.

The obtained results on the strict quantization function over all measures show that our approach is able to return high specific elements to the INEX topics, which is structure oriented. Note that we obtain 0.502 for the most structured topic (for MAep and f_{strict}) over INEX 2005 topics, which confirms the effectiveness of our approach in highly structured queries context on one hand, and the impact of structure on XML retrieval on the other.

Additional experiments have been undertaken to show the effectiveness of our query reformulation approach. Since INEX does not provide a training dataset, we built it manually by dividing the INEX corpus into a training dataset containing 450 documents and the 16369 remaining documents are used as a test corpus. To evaluate the impact of the query reformulation, we use the same XML retrieval process as described below.

In Table 1 we present a comparison between the values obtained before reformulation (BR), after reformulation (AR) and the mean average of the official results of all INEX participants (AVR). The parameters δ and γ are respectively 0.70 and 8.50.

We can see through our experiments that our query reformulation approach significantly improves the results on the $f_{generalized}$ quantization function (until 0.38 %) which considers one element to be relevant provided that it is both exhaustive and specific. This function is an excellent illustration of structured information retrieval. We note that during these experiments we reformulate only the queries structures without changing their original content, and therefore we believe that this reformulation has brought an evolution that could be accentuated by the reformulation of the content.

Table 1: Comparative results before and after automatic query reformulation.

Quantization	Run	MAep
f_{strict}	BR	0.099
	AR	0.10076
	AVG	0.0286
f_{gen}	BR	0.0794
	AR	0.0832
	AVG	0.0526
$f_{genLifted}$	BR	0.0275
	AR	0.0279
	AVG	0.0197

6 CONCLUSIONS

We have presented in this paper an XML retrieval approach based on tree matching. The approach consists of comparing document and query representations, computing a structure and a content score to each document node and then combine them into a final score.

Each score is computed independently of the other, and the final score depends on both of them. Undertaking content and structure scores computation independently leads to the independence between content and structure scores distributions. This assumption is needed to estimate the score of each document node by its probability of relevance.

We also proposed a representation of the original query and relevant fragments under a form of a matrix. After some processing and calculations on the obtained matrix and after some analysis we have been able to identify the most relevant nodes and their relationships that connect them. The obtained results show that the automatic reformulation of the query structure contributes to the improvement of XML retrieval. The strategy of the reformulation is based on a matrix representation of the XML trees deemed relevant to the fragments and the original query.

We experimented our approach into a dataset provided by INEX. We have placed the emphasis on the CAS tasks, which represent an excellent illustration of flexible XML retrieval. This task is the most appropriate for our retrieval model. The system was designed especially for such tasks.

Additional experiments have been undertaken to show the effectiveness of using an automatic query reformulation model. Through our experiments, we show that the structure hints are very important not only for XML retrieval but also for XML query reformulation.

REFERENCES

Balog, K., Bron, M., and de Rijke, M. (2010). Category-based query modeling for entity search. *32nd European Conference on Information Retrieval, ECIR*, pages 319–331.

Ben Aouicha, M., Tmar, M., Boughanem, M., and Abid, M. (2006). Vers une stratgie de recherche d'information structure base sur la comparaison d'arbres. *CORIA*, pages 65–72.

Ben Aouicha, M., Tmar, M., Boughanem, M., and Abid, M. (2009). Experiments on element and document statics for xml retrieval based on tree matching. *International Journal of Computer and Information Science and Engineering, IJCISE*, 3(1):7–16.

Bordogna, G. and Pasi, G. (2000). Flexible querying of structured documents. *Proc. of the fourth International Conference on Flexible Query Answering Systems(FQAS)*.

Fuhr, N. and Grossjohann, K. (2001). Xirql: A query language for information retrieval in xml documents. *Proc. of the 24th annual ACM SIGIR conference on research and development in Information Retrieval, New Orleans, USA*, pages 172–180.

Piwowarski, B. and Dupret, G. (2006). Evaluation in (xml) information retrieval: Expected precision-recall with user modelling (eprum). *Proc. of the 29th annual ACM SIGIR conference on research and development in Information Retrieval*, pages 260–267.

Rocchio, J. (1971). *Relevance feedback in information retrieval*. Prentice Hall Inc., englewood cliffs, nj edition.

Schlieder, T. and Meuss, H. (2002). Querying and ranking xml documents. *Journal of the American Society for Information Science and Technology*, 6(53):489–503.

Selkow, S. M. (1977). The tree-to-tree edition problem. *Information processing letters*, pages 184–186.