

The impact of leaf nodes relevance values evaluation in a propagation method for XML retrieval

Karen Sauvagnat
IRIT

118 route de Narbonne
31062 Toulouse Cedex 4
+33 5 61 55 68 99
sauvagna@irit.fr

Mohand Boughanem
IRIT

118 route de Narbonne
31062 Toulouse Cedex 4
+33 5 61 55 68 99
bougha@irit.fr

ABSTRACT

In this paper, we describe an IR approach to XML retrieval using content and structure conditions based queries. The XFIRM model we propose is based on a complete query language, derived from Xpath, and on a relevance values propagation method. We propose here to evaluate the importance of the weighting formula used to assign relevance values to leaf nodes prior to propagation. Our experiments are evaluated thanks to the INEX evaluation initiative. Results show a relatively high precision, which, depending on the query context, can be increased thanks to a combination of IR models.

Categories and Subject Descriptors

H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval models – *retrieval models*.

General Terms

Algorithms, Experimentation

Keywords

XML retrieval, XFIRM, relevance propagation

1. INTRODUCTION

One of the main advantages of the XML format is its ability to combine structured and un-structured (i.e. text) data. As a consequence, the granularity of the information processed by Information Retrieval Systems (IRS) dealing with XML documents is not necessarily the whole documents : indeed, the structure of XML documents allows IRS to retrieve information units (i.e. nodes of XML documents). The main challenge in XML retrieval is to thus retrieve the most exhaustive¹ and specific² information units [7] answering a given query. Approaches dealing with this challenge can be divided into two main sub-groups [3]. On the one hand, the data-oriented approaches use XML documents to exchange structured data (as for example whole databases). The database community was the first to propose solutions for the XML retrieval issue, using data-oriented approaches. Unfortunately, the suggested approaches typically expect binary answers to very specific queries. On the other hand, the document-oriented approaches consider that tags, inserted by documents creators, describe the documents logical structure. The IR community has adapted

¹ An information unit is exhaustive to a query if it contains all the required information

² An information unit is specific to a query if all its content concerns the query

traditional IR approaches to address the user information needs in XML collection. The increased interest of the community has become apparent within the 2 past INEX evaluation initiatives [4] [5].

Fuhr et al. [6] proposed an augmentation method for processing XML documents. In this approach, standard term weighting formulas are used for indexing so called “index nodes” of the document. Index nodes are not necessarily leaf nodes, because this structure is considered to be too fine-grained. For computing inner nodes indexing weights, the weights from the most specific index nodes are propagated towards the inner nodes. During the propagation, the weights are down-weighted by multiplying them with a so-called augmentation factor. This way, more specific elements are preferred during retrieval.

The approach we describe in this paper is also based on an augmentation method. However, in our approach, all leaf nodes are indexed. Indeed, we think that even the smallest leaf node can contain relevant information (it can be a title or sub-title node for example). Moreover, the distance separating nodes in the document tree is a parameter used when propagating the relevance values in the tree. The goal of this paper is to evaluate the importance of leaf nodes relevance value calculation in the overall performance of the system. In section 2, we quickly present the XFIRM (*XML Flexible Information Retrieval Model*) model and the associated query language. Section 3 evaluates our approach via experiments carried out on the INEX collection.

2. THE XFIRM MODEL

2.1 The XFIRM Query language

The XFIRM model is based on a complete query language, that allows queries to be expressed with simple keyword terms and/or with structural conditions [9].

In its more complex form, the language allows hierarchical conditions on document structure to be expressed and the user can specify which type of element needs to be returned (thanks to the *te*: (*target element*) operator):

Some examples of XFIRM queries can be found below:

(i) // *te*: p [*weather forecasting systems*]

(ii) // *article*[*security*] // *te*: *sec* [“*facial recognition*”]

(iii) // *te*: *article* [*Petri net*] // *sec* [*formal definition*] AND *sec* [*algorithm efficiency*]

(iv) // *te*: *article* [] // *sec* [*search engines*]

This respectively means that (i) the user wants a paragraph about *weather forecasting systems*, (ii) a section about *facial recognition* in an article about *security*, (iii) an article about *Petri net* containing a section giving a *formal definition* and

another section talking about *algorithm efficiency*, and (iv) an article containing a section about *search engines*.

When expressing the possible content conditions, the user can use simple keyword terms (or phrases), possibly preceded by + or - (which means that the term should or should not be in the results). Terms can also be connected by Boolean operators.

2.2 Query processing

The approach we propose to deal with queries containing content and structure conditions, is based on relevance values propagation. The query evaluation in XFIRM is carried out as explained below.

2.2.1 Query decomposition

Each XFIRM query can be decomposed in sub-queries SQ_i as follows:

$$Q = // SQ_1 // SQ_2 // \dots // te : SQ_j // \dots // SQ_n \quad (1)$$

Where the *te*: operator indicates which element is the target element. Each sub-query SQ_i can then be re-decomposed into elementary sub-queries $ESQ_{i,j}$, possibly linked by boolean operators. They are of the form:

$$ESQ_{i,j} = tg [q] \quad (2)$$

Where *tg* is a tag name and $q = \{t_1, \dots, t_n\}$ is a set of keywords, i.e. a content condition.

For example, the XFIRM query: *// te: article [search engines] // sec [Internet growth] AND sec [Google]*, can be decomposed into sub-queries and elementary sub-queries as follows:

$$SQ_1 = article [search engines]$$

$$SQ_2 = sec [Internet growth] AND sec [Google]$$

$$ESQ_{1,1} = article [search engines]$$

$$ESQ_{2,1} = sec [Internet growth]$$

$$ESQ_{3,1} = sec [Google]$$

2.2.2 Evaluating the relevance of leaf nodes

The first step in query processing is to calculate the relevance values of each leaf node *ln* according to the content conditions (if they exist). Let $q = \{t_1, \dots, t_n\}$ be a content condition. Relevance values are evaluated using $RSV_m(q, nf)$ (Retrieval Status Value), where *m* is an IR model.

$$RSV_m(q, ln) = f_{j=1, \dots, n} (q_j, w_j^{(ln)}) \quad (3)$$

where:

- q_j is the weight of the term t_j in the content condition *q*
- $w_j^{(ln)}$ is the weight of the term t_j in the leaf node *ln*
- *f* is a function that allows to aggregate the weights of the terms t_j for the leaf node *ln*.

2.2.3 Structure processing

2.2.3.1 Elementary sub-queries $ESQ_{i,j}$ processing

The result set $R_{i,j}$ of $ESQ_{i,j}$ is a set of pairs (*node, relevance*) defined as follows:

$$R_{i,j} = \{ (n, r_n) / n \in \{construct(tg)\} \text{ and } r_n = F_k (RSV_m(q, nf_k), dist(n, nf_k)) \} \quad (4)$$

Where : - r_n is the relevance value of the node *n*

- the *construct(tg)* function allows the set of all nodes having *tg* as tag name to be created

- the $F_k (RSV_m(q, nf_k), dist(n, nf_k))$ function allows relevance values of the leaf nodes nf_k being descendants of the node *n* to be propagated and aggregated, in order to calculate the relevance value of *n*. The distance $dist(n, nf_k)$ which separates the node *n* from the leaf nodes nf_k in the document tree (i.e. the number of arcs that are necessary to join *n* and nf_k) is used as a parameter during the propagation.

2.2.3.2 Subqueries SQ_i processing

Once each $ESQ_{i,j}$ has been processed, subqueries SQ_i are rebuilt using the commutative operators \oplus_{AND} et \oplus_{OR} defined below :

Definition 1 : Let $N = \{ (n, r_n) \}$ and $M = \{ (m, r_m) \}$ be two sets of pairs (node, relevance)

$$N \oplus_{AND} M = \{ (l, r_l) / l \text{ is the nearest common ancestor of } m \text{ and } n \text{ or } l=m \text{ (respectively } n) \text{ if } m \text{ (resp. } n) \text{ is ancestor of } n \text{ (resp. } m) \text{, } \forall m, n \text{ being in the same document and } r_l = aggreg_{AND}(r_n, r_m, dist(l, n), dist(l, m)) \} \quad (5)$$

$$N \oplus_{OR} M = \{ (l, r_l) / l=n \in N \text{ or } l=m \in M \text{ and } r_l = r_n \text{ or } r_m \} \quad (6)$$

Where $aggreg_{AND}(r_n, r_m, dist(l, n), dist(l, m)) = r_l$ defines the way relevance values r_n and r_m of nodes *n* and *m* are aggregated in order to form a new relevance r_l .

2.2.3.3 Queries processing

The result set of sub-queries SQ_i are then used to process the whole queries. As defined above, in each query, a target element is specified.

$$Q = // SQ_1 // SQ_2 // \dots // te : SQ_j // \dots // SQ_n$$

The aim of processing the whole query is to propagate the relevance values of SQ_i sub-queries to the nodes belonging to the result set of sub-query SQ_j (which defines the target element). For this purpose, non-commutative operators ∇ and Δ are defined below :

Definition 2 : Let $R_i = \{ (n, r_n) \}$ and $R_{i+1} = \{ (m, r_m) \}$ be two sets of pairs (node, relevance)

$$R_i \nabla R_{i+1} = \{ (n, r_n) / n \in R_i \text{ is ancestor of } m \in R_{i+1} \text{ and } r_n = prop_agg(r_n, r_m, dist(m, n)) \} \quad (7)$$

$$R_i \Delta R_{i+1} = \{ (n, r_n) / n \in R_i \text{ is descendant of } m \in R_{i+1} \text{ and } r_n = prop_agg(r_m, r_n, dist(m, n)) \} \quad (8)$$

Where $prop_agg(r_n, r_m, dist(m, n)) \rightarrow r_n$ allows to aggregate relevance weights r_m of the node *m* and r_n of the node *n* according to the distance that separates the 2 nodes, in order to obtain the new relevance weight r_n of node *n*.

The result set *R* of a query *Q* is then defined as follows:

$$R = R_j \nabla (R_{j+1} \nabla (R_{j+2} \nabla \dots)) \quad (9)$$

$$R = R_j \Delta (R_{j-1} \Delta (R_{j-2} \Delta \dots)) \quad (10)$$

Indeed, this is equivalent to propagating relevance values of results set R_{j+1}, \dots, R_n and R_1, \dots, R_n respectively upwards and downwards in the document tree.

3. Experiments and results

Our experiments have been evaluated using the INEX evaluation initiative [4], on the 2003 SCAS (Strict Content and Structure) task. The INEX collection, 21 IEEE Computer Society journals from 1995-2002, consists of 12 135 documents with extensive XML-markup. Participants to INEX SCAS task have to perform CAS (Content and Structure) queries, which contain explicit references to the XML structure, and restrict

the context of interest and/or the context of certain search concepts.

The INEX metric for evaluation is based on the traditional recall and precision measures. To obtain recall/precision figures, the two dimensions of relevance (exhaustivity and specificity) need to be quantized onto a single relevance value. Quantization functions for two user standpoints were used: (i) a “strict” quantization to evaluate whether a given retrieval approach is capable of retrieving highly exhaustive and highly specific document components, (ii) a “generalised” quantization has been used in order to credit document components according to their degree of relevance.

3.1 Leaf nodes relevance evaluation

In order to evaluate the impact of the model used for leaf nodes relevance values evaluation, we tested the following functions :

$$- RSV(q, ln) = \sum_{i=1}^n w_i^q * w_i^{ln} \quad \text{with} \quad (11)$$

$$w_i^q = tf_i^q * ief_i \quad \text{And} \quad w_i^{ln} = tf_i^{ln} * ief_i$$

Where :

- tf_i is the term frequency in the query q or in the leaf node ln
- ief_i is the inverse element frequency of term i , i.e. $\log(N/n+1)+1$, where n is the number of leaf nodes containing i and N is the total number of leaf nodes.

$$- RSV(q, ln) = \sum_{i=1}^n tf_i^q * \frac{tf_i^{ln} * (h_1 + h_2 * ief_i)}{h_3 + h_4 * \frac{l}{\Delta l} + h_5 * tf_i^{ln}} \quad (12)$$

Where

- h_1, h_2, h_3, h_4 and h_5 are constant parameters, respectively set to 0.2, 0.8, 0.2, 0.7 and 1 for our experiments
- l is the length of the leaf node ln
- Δl is the average leaf node length

The second function is inspired by OKAPI [8] and SMART term weighting and is used in the full-text retrieval search engine Mercure [1]. The values of $h1$, $h2$, $h3$, $h4$ and $h5$ parameters have been set thanks to experiments on TREC collections [2].

3.2 Implementation issues

The transformation of INEX CAS queries to XFIRM queries was fairly easy.

During $ESQ_{i,j}$ processing, the most relevant leaf nodes are found, and for each of these leaf nodes, XFIRM looks for ancestors. In order to have a correct system response time, the propagation is stopped when 1500 “correct” ancestors have been found (i.e. ancestors having a correct tag name). When an INEX topic contains a condition on the article publication date, this condition is not translated in the XFIRM language, as propagation with a very common term (like a year) is too long. To solve this issue, queries are processed by XFIRM without this condition, and results are then filtered on the article publication date.

Moreover, a Dictionary index is used to find equivalent tags, according to INEX guidelines.

Finally, we use the following propagation functions while processing queries:

$$F_k(RSV(q, nf_k), dist(n, nf_k)) = \sum_{k=1..n} \alpha^{dist(n, nf_k)} * RSV(n, nf_k) \quad (13)$$

Where $\alpha \in]0..1]$ is a parameter allowing to adjust the importance of the distance between nodes in the different functions.

For our experiments, α is set to 0.9

$$agg_{AND}(r_n, r_m, dist(l, n), dist(l, m)) = \frac{r_n}{dist(l, n)} + \frac{r_m}{dist(l, m)} \quad (14)$$

$$prop_agg(r_n, r_m, dist(m, n)) = \frac{r_n + r_m}{dist(m, n)} \quad (15)$$

3.3 Runs

We tested 4 runs, described below:

1. **xfirm.T.tf-ief** : function (11) is used. Only the title field of INEX CAS queries is used for query indexing.
2. **xfirm.TK.tf-ief** : function (11) is used. Title and Keywords fields of INEX CAS queries are used for query indexing.
3. **xfirm.T.Mercure** : function (12) is used. Only the title field of INEX CAS queries is used for query indexing.
4. **xfirm.TK.Mercure** : function (12) is used. Title and Keywords fields of INEX CAS queries are used for query indexing.

In addition, these runs were compared to the best run we performed last year in the INEX SCAS task with a “fetch and browse” method: **Mercure2.pos_cas_ti** [10].

3.4 Analysis of the results

For all 5 runs, we issued 30 queries, expressed with the XFIRM language. The 5 runs resulted in 10 average precision (for strict and generalized quantization) that are shown in Table 1. The associated recall-precision curves for strict quantization are plotted in Figure 1.

Table 1: Average precision for the 5 runs

	Average precision (strict)	Average precision (generalized)
Xfirm.T.tf-ief	0,2675	0,2276
Xfirm.TK.tf-ief	0,2898	0,2300
Xfirm.T.Mercure	0,2448	0,2032
Xfirm.TK.Mercure	0,2467	0,2154
Mercure2. pos_cas_ti	0,1620	0,1637

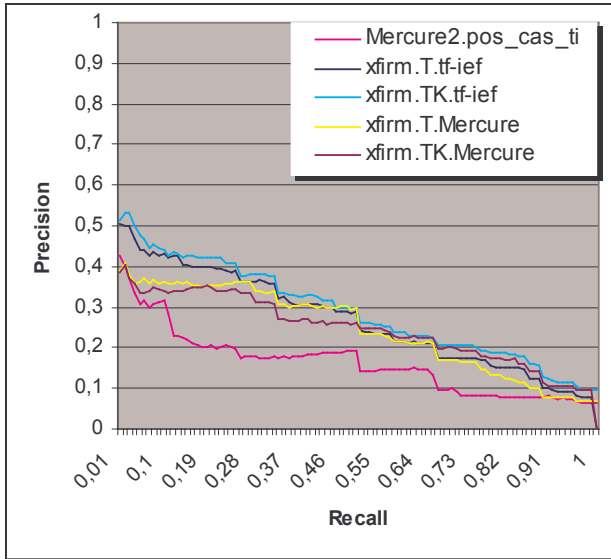


Figure 1: Recall/precision curves for strict quantization

The first point to notice is the relatively high precision for all runs. Almost all of them would have been ranked in the top ten of official INEX evaluation, with *Xfirm.T.tf-ief* and *Xfirm.TK.tf-ief* ranked third for strict quantization (Table2).

Table 2: Ranking of official INEX submissions and of our runs for strict quantization

rank	Avg precision	Organization	Run ID
1	0.3182	U. of Amsterdam	UamsI03-SCAS-MixedScore
2	0.2987	U. of Amsterdam	UamsI03-SCAS-ElementScore
	0.2898		<i>Xfirm.TK.tf-ief</i>
	0.2675		<i>Xfirm.T.tf-ief</i>
3	0.2601	Queensland University of Technology	CASQuery_1

The propagation method increases in a very significant way the results we obtained with a fetch and browse method (run *Mercure2.pos_cas_ti*).

The use of title and keywords fields of INEX topics increases the average precision of both runs *Xfirm.TK.tf-ief* and *xfirm.TK.Mercure* comparing to the runs *Xfirm.T.tf-ief* and *Xfirm.T.Mercure*, even if there is a loss of precision for some particular queries.

The IR model used for evaluating leaf nodes relevance values seems to have a high impact. The simplest formula (11) which does not take into account the leaf nodes length obtains the best results. However, if we examine the results for each query, formula (12) is more efficient when the target element of the CAS query is a whole article. If we combine the two runs *xfirm.T.tf-ief* and *Xfirm.T.Mercure*, using *Xfirm.T.Mercure* only when the target element is an article, we obtain almost 10,54 % of improvement for strict quantization, which seems to signify that the choice of weighting formula for leaf nodes relevance value calculation might depend on the query context.

Our methods need to be carried out on other topics/collections to confirm these performances. We view this work as a starting point for finding an appropriate formula for leaf nodes relevance value evaluation. Moreover, an evaluation of propagation functions has to be done.

4. REFERENCES

- [1] Boughanem, M., Chrisment, C., Soule-Dupuy, C. : *Query modification based on relevance back-propagation in ad-hoc environment*. Information Processing and Management, 35 (1999).
- [2] Boughanem, M. , Chrisment, C., Tmar, M.: *Mercure and MercureFiltre applied for Web and Filtering tasks at TREC-10*. In : TREC-10, Gaithersburg, Maryland (USA), Nov. 2001.
- [3] Fuhr, N., Grossjohann, K. “*XIRQL: A query Language for Information Retrieval in XML Documents*”. In Proc. ACM SIGIR, New Orleans, USA, 2001.
- [4] Fuhr, N., Malik, S., Lalmas, M : *Overview of the Initiative for the Evaluation of XML Retrieval (INEX) 2003*. In Proceedings of INEX 2003 Workshop, December 2003.
- [5] Gövert, N., Kazai, G. : *Overview of the Initiative for the Evaluation of XML Retrieval (INEX) 2002*. In Proceedings of INEX 2002 Workshop, December 2002.
- [6] Gövert Norbert, Abolhassani, M., Fuhr, N., Grossjohann, K. : *Content-oriented XML retrieval with HyREX*. In Proceedings of the first INEX Workshop, December 2002.
- [7] Lalmas, M., “*Dempster-Shafer theory of evidence applied to structured documents: modeling uncertainty*”. In Proc. ACM-SIGIR, Philadelphia, 1997.
- [8] Robertson, S.E., Walker, S., Hancock-Beaulieu, M.M. : *Okapi at TREC 3*. In Proceedings of the 3rd Text Retrieval conference (TREC 3), NIST, 1994, Virginia, USA.
- [9] Sauvagnat, K., Boughanem, M. : *Le langage de requête XFIRM pour les documents XML*. Actes du Congrè Inforsid 2004, Biarritz, France.
- [10] Sauvagnat, K., Hubert, G., Boughanem, M., Mothe, J. : *IRIT at INEX 03*. In Proceedings of INEX 2003 Workshop, December 2003