World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:2, No:2, 2008

Experiments on Element and Document Statistics for XML Retrieval

Mohamed Ben Aouicha, Mohamed Tmar, Mohand Boughanem, and Mohamed Abid

***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 structural constraints of the query. The content of each node is very important to decide weither 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 content-based score of each node is commonly based on the well-known $Tf \times Idf$ criteria. In this paper, we compare between this criteria and another one we call $Tf \times Ief$. The comparison is based on some experiments into a dataset provided by INEX[1] to show the effectiveness of our approach on one hand and those of both weighting functions on the other.

***Keywords*—**XML retrieval, INEX, $Tf \times Idf$, $Tf \times Ief$

I. Introduction

EXTENSIBLE Markup Language (XML) [1] is becoming widely used as a standard document format in many application domains. We believe since few years that a great volume of static and dynamic data were produced in XML.

Therefore, XML retrieval becomes more and more essential [4]. XML documents covers a big part not only on the web, but also on modern digital libraries, business to business and business to consumer software and essentially on Web services oriented software. This is due to the great importance of structured information.

While both text and structure are important, we usually give higher priority to text when ranking XML elements. We adapt unstructured retrieval methods to handle additional structural constraints. Such approach are called text centric XML retrieval. The vector-space based XML retrieval method proposed by [20] 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 similarity measures (Cosine, Dice, Overlap . . .). Schlieder and Meuss [16] 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 vector space model basics.

Several teams have used a language modeling approach to XML retrieval. Ogilvie and Callan [21] use a tree-based generative language model for ranking documents and components. They build a language model for nodes and another for leaf nodes depending on their components. Inner nodes are estimated using a linear interpolation among the children nodes. The probabilistic model has been applied to XML documents by [19] and [11].

Contrarily to text-centric XML, data-centric XML mainly encodes non-text data. When querying data-centric XML, the user imposes exact match conditions in most cases. This approach is commonly used for data collections with complex structures and non-text data. There are powerful query languages for XML that can efficiently handle structure.

The most known of such languages is XQuery [22]. However, it is challenging to implement an XQuery-based typically to provide ranked lists of elements. Amer Yahia [23] uses a pettern matching model based on Xquery to handle XML retrieval.

Fuhr [5] 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 IR models, in combination with concepts from the database area.

In this paper, we separate between content and structure since indexing queries and documents [12]. We handle the document structure by retrieving candidate document fragments that almost follow the query structure and then complete their scores by content retrieval. By means of structure, scores are assigned to document fragments, highest scores are assigned to those that have exactly the same structure as the query.

Besides, we assume that content retrieval is the main decisive criteria of relevance. Although XML retrieval is based on structure constraints, the content is still the

M. B. Aouicha is with the Institut de Recherche en Informatique de Toulouse, 118 Route de Narbonne, 31062, email: mohamed.benaouicha@irit.fr

M. Tmar is with the Institut Supérieur d'Informatique et du Multimédia de Sfax, Route de Tunis, B.P.: 1030, 3018, email: mohamedtmar@isimf.rnu.tn

M. Boughanem is with the Institut de Recherche en Informatique de Toulouse, 118 Route de Narbonne, 31062, email:boughane@irit.fr

M. Abid Ecole Nationale d'Ingénieurs de Sfax, Route de Soukra, 3038, mohamed.abid@enis.rnu.tn

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

most important element to retrieve by the XML retrieval system and to exploit by the user. The main goal of this paper is to compare between two content-based weighting functions: $Tf \times Idf$ and $Tf \times Ief$. $Tf \times Idf$ weighting formula is commonly used in traditional information retrieval, it shows that a term weight depends on its frequency on the document and the total number of documents containing it.

This paper is organized into five sections. The second section exposes tree extension used to build a flexible representation of documents and queries and the exact match algorithm we use starting from these representations to perform structure retrieval. In the third section, we present content retrieval and how content and structure retrieval scores are combined to meet a definitive score to a candidate fragment. We also explore the weighting functions used in our experiments.

The fourth section presents the experiments and the obtained results. The fifth section concludes.

II. Structure indexing and retrieval

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:

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

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 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 $N_.$ should be different. This brings to another property which requires that each node but the root has only one parent:

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

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 [17]. To compare between two trees, we trend to build a tree starting from the second tree and compute correction costs while correcting. 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. This operation has a cost and the tree to tree correction cost is the total cumulated cost of correction operations.

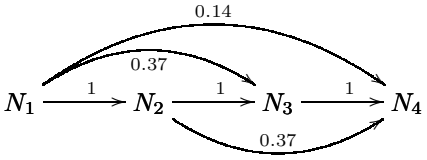

Fig. 1. Original and additional paths, the original paths are weighted by 1.

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 second, so they are totally 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 structure retrieval [19]. First, we assume that the similarity between tree T and tree T' should be equal to the similarity between tree T' and tree T, whereas tree T to tree T' correction has not the same cost as tree T' to tree T correction. In fact, removing and appending nodes have not the same cost, in addition, depending on the tree to tree correction algorithm, when we start from a tree T to meet another tree T', it is possible that the operations are not the inverse of those that we use to meet the tree T starting from the tree T'. Second, the tree to tree correction algorithms are very much 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 [17]. 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. Naturally, 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 $A \longrightarrow B$.

Figure 1 shows how a path $N_1 \longrightarrow N_2 \longrightarrow N_3 \longrightarrow N_4$ is extended to a set of weighted paths.

We apply an exact match algorithm on the extended trees. 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 [7].

The search strategy we adopt is iterative. We start from extended document and query trees. We build a returned fragment incrementally, starting from common potential 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. Algorithm 1 shows this search strategy. Figure 2 illustrates the structure score computation. Each query node is matched with a document node and the structure score is the sum of each path weight multiplied by its matched one: $0.37 \times 1 + 0.14 \times 0.14 + 0.37 \times 0.14 + 1 \times 0.37 = 0.81$.

Algorithm 1 Structure retrieval

E_q {The extended query tree}
E_d {The extended document tree}
$F \leftarrow \emptyset$ {the set of returned fragments, initially empty}
for all $n_q \xrightarrow{w_q} n'_q \in E_q \{n'_q$ is an ancestor of $n_q\}$ **do**
  for all $n_d \xrightarrow{w_d} n'_d \in E_d, n_q \equiv n_d$ and $n'_q \equiv n'_d \{n'_d$ is an ancestor of n_d, $\alpha \equiv \beta$ shows that nodes α and β should have the same tag name, so they can be matched$\}$ **do**
    if $\exists (f_q, f_d, C, r) \in F, (n_q \xrightarrow{\cdot} ., n_d \xrightarrow{\cdot} .) \in C$ or $(. \xrightarrow{\cdot} n_q, . \xrightarrow{\cdot} n_d) \in C$ { Fragments f_q and f_d can be enriched by all matchable paths $n'_q \xrightarrow{\cdot} .$ and $n'_d \xrightarrow{\cdot} .$, C is a set of pairs where each pair $(p, p') \in C$ shows that path p in the query tree is matched with path p' in the document tree, r is the structure score computed incrementally while building potential structure relevant fragments$\}$ **then**
      for all $n'_q \xrightarrow{w'_q} n''_q \in E_q, n'_d \xrightarrow{w'_d} n''_d \in E_d, n''_q \equiv n''_d$ $\{n'_q \xrightarrow{w'_q} n''_q$ and $n'_d \xrightarrow{w'_d} n''_d$ are used to enrich the retrieved fragments f_q and $f_d\}$ **do**
        $(f^*_q, f^*_d, C^*, r^*) \longleftarrow (f_q, f_d, C, r)$ {we clone (f_q, f_d, C, r) towards a new fragment (f^*_q, f^*_d, C^*, r^*) and then enrich it by paths $n'_q \xrightarrow{w'_q} n''_q$ and $n'_d \xrightarrow{w'_d} n''_d\}$
        $f^*_q \longleftarrow f^*_q \cup \{n'_q \xrightarrow{w'_q} n''_q\}$
        $f^*_d \longleftarrow f^*_d \cup \{n'_d \xrightarrow{w'_d} n''_d\}$
        $C^* \longleftarrow C^* \cup \{(\mathrm{n'}_q \xrightarrow{w'_q} n''_q, \mathrm{n'}_d \xrightarrow{w'_d} n''_d)\}$
        $r^* \longleftarrow r^* + w'_q \times w'_d \{$update the score$\}$
        $F \longleftarrow F \cup \{(f^*_q, f^*_d, C^*, r^*)\} \{$update the set $F\}$
      end for
    else
      {the path does not yet exist among the already built fragments, it can be added as a new relevant fragment each with only one path$\}$ $F \longleftarrow F \cup \{(\{n_q \xrightarrow{w_q} n'_q\}, \{n_d \xrightarrow{w_d} n'_d\}, \{(n_q \xrightarrow{w_q} n'_q, n_d \xrightarrow{w_d} n'_d)\}, w_q \times w_d)\}$
    end if
  end for
end for

III. Content retrieval

The content-based score is computed for each document node according to a given query. This score is computed independently from the structure-based retrieval as follows:

$$rsv_c(n) = \sum_{p \in n \cap q} w(p, n)$$

where $w(p, n)$ is the weight of term p in node n and $n \cap q$ is the set of terms apprearing in both the query and the element node. The term weight in a document node is computed as follows:

$$w(p, n) = idf_p \times \sum_{n \rightarrow c_1 \rightarrow c_2 \ldots c_k} \frac{tf(p, c_k)}{d(n, c_k) + 1} \quad (3)$$

where $tf(t, c_k)$ is the frequency of term p in node c_k and idf_p is the inverse of document frequency of term p and is done by the following:

$$idf_p = \frac{N}{n_p}$$

where N is the total number of documents and n_p is the number of documents containing term p. Equation 3 shows that each node content is propagated to its ancestor nodes. For example, if node A is the parent of node B which contains term p, we assume that node A contains term p and we downweight it in node A by dividing it by $2 = d(A, B) + 1$ [24]. If node C is the parent of node A, term p is propagated from node B to node C and we divide its weight by $3 = d(C, B) + 1 \ldots$

According to the $Tf \times Ief$ weighting formula, N is replaced by the total number of elements and n_p by the number of elements containing term p. Both of these weighting functions have been tested in our experiments.

IV. Experiments and results

Experiments have been undertaken into a dataset provided by INEX. It contains 16819 articles taken from IEEE publications in 24 journals covering the period of 1995-2004 and totaling about 750 megabytes, and 87 queries (40 for CO+S and CO tasks and 47 for CAS task on which we place the emphasis in this paper).

The INEX metric for evaluation is based on the exhaustivity and specificity measures which are analogous to the traditional recall and precision measures. 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

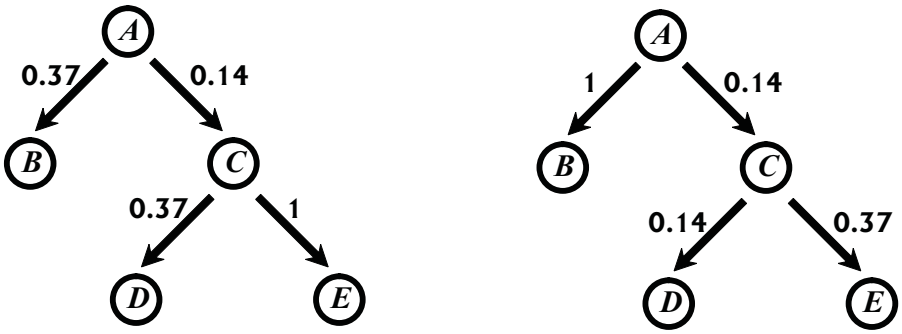

Fig. 2. Structure retrieval

component satisfies the information need.

Each document component has one of the following values of specificity (resp. exhaustivity):

- 0: non specific (resp. non exhaustive)
- 1: slightly specific (resp. slightly exhaustive)
- 2: specific (resp. exhaustive)
- 3: very specific (resp. very exhaustive)

These measures are quantized onto a single relevance value. Quantization functions for three user standpoints are used:

$$\begin{aligned} f_{strict}(s,e) &= \begin{cases} 1 & if\,(e,s) = (2,1) \\ 0 & otherwise \end{cases} \\ f_{generalized}(s,e) &= e \times s \\ f_{geneLifted}(s,e) &= (e+1) \times s \end{aligned}$$

The INEX CAS (Content and structure) measures are described as follows:

- VVCAS: the information retrieval assessments of the whole topic (done against the narrative).
- SVCAS: the subset of VVCAS assessments that strictly satisfy the target element constraint.
- VSCAS: those VVCAS assessments that satisfy all support element constraints, regardless of the target element constraint. Boolean operators between support elements are followed strictly.
- SSCAS: those assessments that strictly satisfy all support element constraints as well as the target element constraint.

Where for XYCAS, X is the target element and Y is the support element, and either can be S for strict or V for vague. We emphasize our experiments on the VVCAS task which is the most appropriate to our model. The INEX metric we use for evaluation is based on the mean average effort precision ($MAep$).

First experiments were undertaken to show the effectiveness of our approach on XML retrieval. Figures 3, 4 and 5 show our results (bold curves) compared to official INEX obtained results by all participants on the MAep quantization measure.

Figures 6, 7 and 8 show the obtained results of using $Tf \times Idf$ and $Tf \times Ief$ (bold curve) weighting functions.

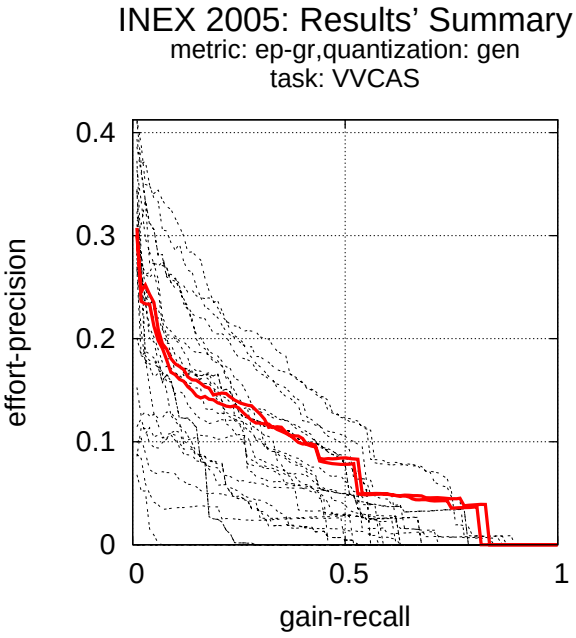

Fig. 3. Comparative results with INEX official participants, Generalized quantization function

Globally, the $Tf \times Ief$ weighting function provides better results. In high gain-recall values, the $Tf \times Idf$ provides slightly better results. We can conclude from our experiments the importance of content-based retrieval on XML corpuses and the importance of the term weighting function to choose on the XML retrieval process.

V. 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

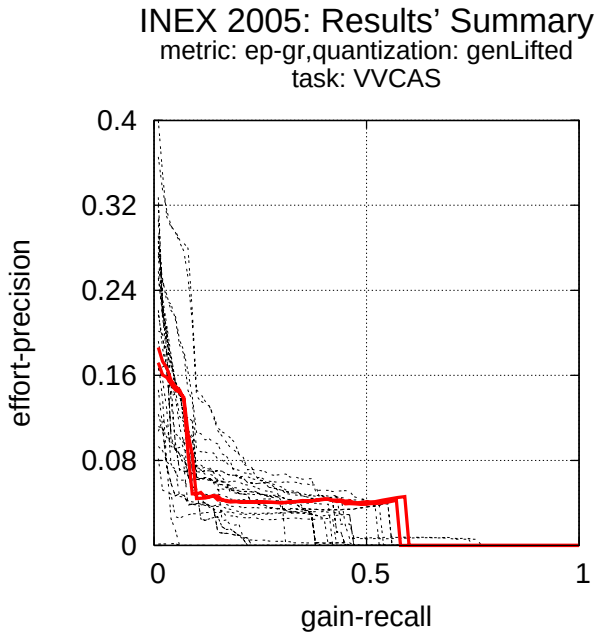

Fig. 4. Comparative results with INEX official participants, Generalized lifted quantization function

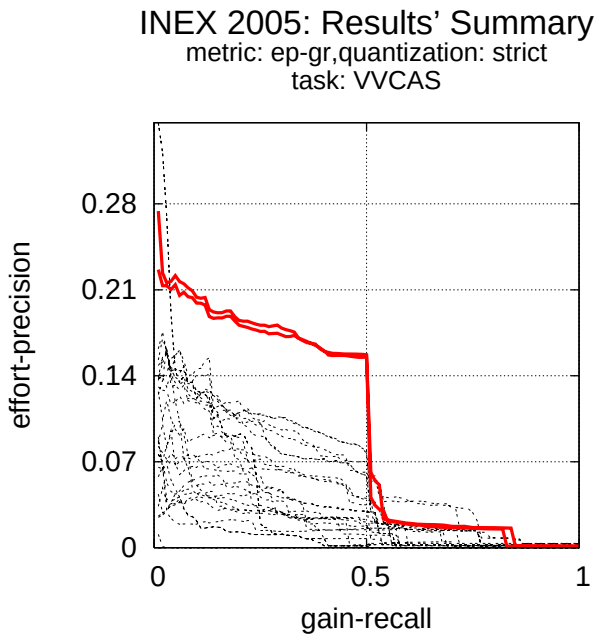

Fig. 5. Comparative results with INEX official participants, Strict quantization function

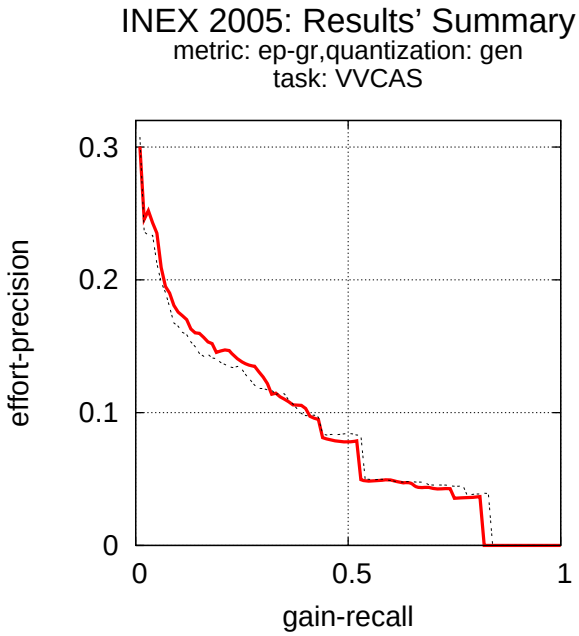

Fig. 6. $Tf \times Idf$ and $Tf \times Ief$ results, Generalized quantization function

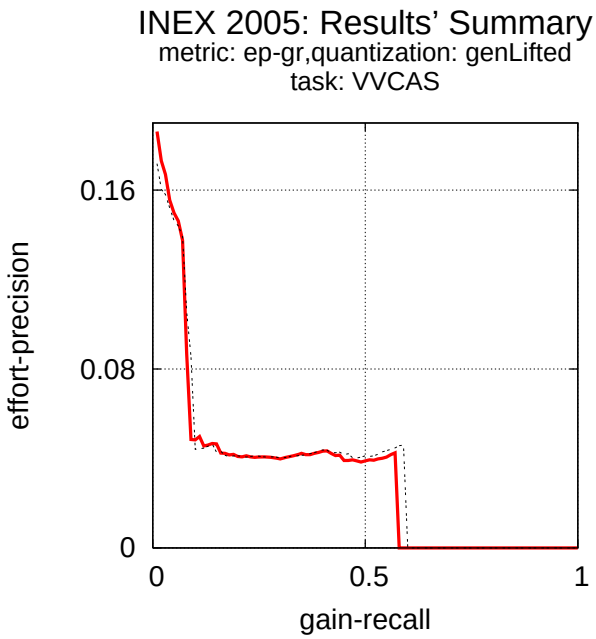

Fig. 7. $Tf \times Idf$ and $Tf \times Ief$ results, Generalized lifted quantization function

distributions. This assumption is needed to estimate the score of each document node by its probability of relevance.

We have experimented our approach into a dataset provided by INEX. We have placed the emphasis on the VVCAS task, which represents 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 $Tf \times Idf$ and $Tf \times Ief$ weighting functions. We have observed that the $Tf \times Ief$ weighting function provides globally better results. Currently, we are using an XML retrieval prototype based on flexible tree matching. Further experiments should be achieved on other topics/collections such as Wikipedia corpus provided by INEX. Besides, we plan to investigate an issue of combining $Tf \times Idf$ and $Tf \times Ief$ weighting functions first to consider

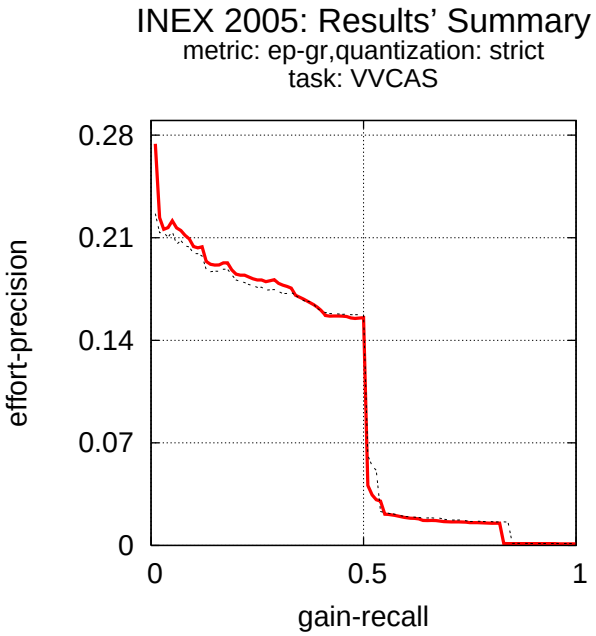

Fig. 8. $Tf \times Idf$ and $Tf \times Ief$ results, Strict quantization function

both of them in our XML retrieval system and second to take advantage from both of them to improve the system performance.

REFERENCES

[1] World wide web consortium (w3c). extensible markup language (xml) 1.0. *http://www.w3.org/TR/REC-xml*, 2000.
[2] Inex - initiative for the evaluation of xml retrieval. *http://inex.is.informatik.uniduisburg.de*, 2003.
[3] H. Blanken, R. Grabs, and G. Weikum. Intelligent search on xml. *Springer-Verlag*, 2003.
[4] D. Carmel, Y. Maarek, S. Mandelbrod, M. Mass, and A. Soffer. Searching xml documents via xml fragments. *Proc. of the 24th annual ACM SIGIR conference on research and development in Information Retrieval*, pages 151–158, 2003.
[5] N. Fuhr and K. Grossjohann. 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 Orlans, USA*, pages 172–180, 2001.
[6] M. Fuller, E. Mackie, R. Sacks-Davis, and R. Wilkinson. Structural answers for a large structured document collection. *Proc. of the 24th annual ACM SIGIR conference on research and development in Information Retrieval, Pittsburgh, USA*, pages 204–213, 1993.
[7] G. B. G. and Pasi. Flexible querying of structured documents. *Proc. of the fourth International Conference on Flexible Query Answering Systems(FQAS)*, 2000.
[8] T. Grust. Accelerating xpath location steps. *Proc. of the 2002 ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, USA*, pages 109–120, 2002.
[9] J. Kamps, M. Marx, M. D. Rijke, and B. Sigurbjornsson. Xml retrieval : What to retrieve ? *Proc. of the 24th annual ACM SIGIR conference on research and development in Information Retrieval*, pages 409–410, 2003.
[10] G. Kazai, M. Lalmas, and T. Roelleke. A model for the representation and focused retrieval of structured documents based on fuzzy aggregation. *Proc. of SPIRE2001, Chile*, pages 123–135, 2001.
[11] M. Lalmas. Dempster-shafers theory of evidence applied to structured documents: Modeling uncertainty. *Proc. of the 24th annual ACM SIGIR conference on research and development in Information Retrieval, Philadelphia, USA*, pages 110–118, 1997.
[12] R. Luk, H. Leong, T. Dillon, A. Chan, W. Croft, and J. Allan. A survey in indexing and searching xml documents. *Journal of the American Society for Information Science and Technology*, 6(53), 2000.
[13] M. Marx, J. Kamps, and M. D. Rijka. The university of amsterdam at inex 2002. *Proc. of the INEX 2002 Workshop, Germany*, pages 23–28, 2002.
[14] A. Moffat, R. Sacks-Davis, R. Wilkinson, and J. Zobel. Retrieval of partial documents. *Proc. of TREC-2*, 1993.
[15] F. N., G. N., K. G., and L. M. Inex : Evaluation initiative for xml retrieval. *Proc. of INEX 2002 Workshop, DELOS Workshop*, 2003.
[16] T. Schlieder and H. Meuss. Querying and ranking xml documents. *Journal of the American Society for Information Science and Technology*, 6(53):489–503, 2002.
[17] S. Selkow. The tree-to-tree edition problem. *Information processing letters*, pages 184–186, 1977.
[18] R. Wilkinson. Effective retrieval of structured documents. *Proc. of the 24th annual ACM SIGIR conference on research and development in Information Retrieval, Dublin, Ireland*, pages 311–317, 1994.
[19] J. Wolff, H. Flrke, and A. Cremers. Searching and browsing collections of structural information. *Proc. of IEEE advances in digital libraries, Washington, USA*, pages 141–150, 2000.
[20] Y. Mass, M. Mandelbrod, E. Amitay, D. Carmel, Y. S. Maarek and A. Soffer. JuruXML an XML retrieval system at INEX02. *http://inex.is.informatik.uni-duisburg.de:2003/proceedings.pdf*, pages 73-80, 2003.
[21] P. Ogilvie and J. Callan. Parameter estimation for a simple hierarchical generative model for XML retrieval. *http://inex.is.informatik.uni-duisburg.de:2005/proceedings.pdf*, pages 211-224, 2005.
[22] XQuery: A query language for XML. *http://www.w3.org/TR/xquery/*, 2001.
[23] S. Amer-Yahia, B. Chavdar, J. Dorre and J. Shanmugasundaram. XQuery full-text extensions explained. *IBM Systems Journal*, pages 335–352, 2006.
[24] K. Sauvagnat and M. Boughanem. The impact of leaf nodes relevance values evaluation in a propagation method for XML retrieval. *3rd XML and Information Retrieval Workshop, SIGIR 2004, Sheffield, England*, pages 19–22, 2004.