Processing Queries with Metrical Constraints in XML based IR Systems

Shmuel T. Klein

Department of Computer Science

Bar Ilan University, Ramat-Gan 52900, Israel

Tel: (972–3) 531 8865 Fax: (972–3) 736 0498

tomi@cs.biu.ac.il

Abstract: XML documents combine features from classical IR systems allowing free text, with explicit structures as in databases. Many query languages have been specially designed for IR applications on XML documents. This work concentrates on a special type of language for which the problem of processing queries including metrical constraints is investigated. The main question is how to define the distance between terms in different locations of the XML tree in an intuitively justifiable way, without jeopardizing the ability to get good retrieval results in terms of recall and precision. A new definition is given and its usefulness is shown on several examples from the INEX collection.

1. Introduction and Background

The eXtensible Markup Language (XML) [20] is increasingly gaining importance as a standard way of disseminating information that may on the one hand be highly structured, but on the other, include long stretches of free text. Classical Information Retrieval (IR) systems have been used to process free text efficiently, whereas Database (DB) systems and special query languages were built to deal with structured data. There is an obvious need for the development of advanced tools striving to unify these approaches, and research on the connection between XML and IR has been flourishing lately: the ACM Special Interest Group on Information Retrieval (SIGIR) held workshops on the topic in its annual conferences in 2000, 2002 and 2004, *JASIST* published a special issue [16] and

the INitiative for the Evaluation of XML retrieval (INEX) convenes annually since 2002, bringing together some forty international research groups [10, 11].

XML is a markup language in which text may be enclosed by start and end tags for markup, the tag name providing some information on the content enclosed between the corresponding tags. The tagging may be nested as, for example, in `<date> <day>` 20 `</day> <month>` March `</month> <year>` 2004 `</year> </date>`. One may also assign attributes to the start tags of elements, e.g., `<date style="YYMMDD">` 040320 `</date>`. The nesting induces a tree structure and we shall refer interchangeably to an XML file and its corresponding XML tree. As example, refer to the file below in Figure 1(a) and the corresponding tree in Figure 1(c). The structural information in Figure 1(b) is explained below in the section on data structures.

XML documents can be viewed as plausible extensions of entities dealt with by two quite different research communities: IR experts would refer to an XML file as if it were a regular textual document, onto which some hierarchical structure has been superimposed by means of special tags that can be nested; the DB approach would be to consider an XML collection as a database in which some fields are not limited to contain certain values, but may hold free text.

Accordingly, there are two schools regarding XML search engines and query languages: IR queries express some — often vague — information need and are formulated with the help of keywords and possibly some Boolean operators. Since IR is an intrinsically fuzzy process, not all the retrieved items are necessarily relevant, nor can one be assured that all the relevant items are indeed retrieved, hence the definition of performance measures *precision* and *recall*, respectively. In the case of XML, some of the information is conveyed by the structure of the data rather than by the data itself, which led the INEX participants to partition their test topics into two classes: content-only queries and content-and-structure queries. Researchers in databases, on the other hand, deal with much more focused and well defined queries, using query languages that are in fact sophisticated programming languages and are able to describe precisely what is being looked for.

There have been several attempts to adapt DB oriented XML query languages to an IR environment [9, 5, 22]. The range of proposed languages and corresponding processing

algorithms covers from the simplest that could possibly work [15], over models for ranking flat text queries in XML [14], to queries that refine the granularity of the retrieved items [13] and up to an extreme model in which every single character can be retrieved [12]. One of the main problems is that an IR user has generally little knowledge of the exact structure of the texts that are processed. The queries should therefore be able to refer to information regardless of whether it appears explicitly as text or is implied by tags, attributes or the structure of the XML tree.

Our main concern in this paper is the study of *metrical constraints* in an XML environment. Specific limitations on the distances between keywords can indeed be a powerful tool to enhance queries, which may improve both recall and precision. It is, however, not always straightforward to define the distance between two terms in a given XML text, because the linear flow may me disrupted by the presence of XML commands in brackets, as well as by hierarchical nesting.

A preliminary investigation has been carried out in [7], which dealt with information retrieval in the presence of *annotations*. These annotations may be short references to other texts, various types of footnotes, or commentaries written by the same or a different author, like annotations to *Alice in Wonderland* and the like. Similar applications could apply to *hypertext* environments (see, e.g., [8]). We now wish to extend this work to the more general XML style, which differs essentially from the above. The main divergences are:

- when considering a main text with its annotations, the latter are considered of somewhat inferior status, whereas in an XML file, all parts are judged *a priori* of similar importance, regardless of their locations within the hierarchical structure;
- an annotated text consists of only two layers: the main text and the subordinate annotations, even though these may be of different type and nature, and are even allowed to refer to the same point in the main text. Nested annotations were not considered. On the other hand, XML supports nesting in a natural way and induces a multi-layered tree structure.

One can, of course, circumvent the whole problem by restricting queries to look just

for the co-occurrence of certain keywords within the same document. This is probably the approach of most users and for most search engines. Nevertheless, more sophisticated users would try to improve the performance of their IR systems by submitting more involved queries, which is why metrical constraints have to be defined precisely.

The example in Figure 1(a) should clarify the above. Consider a part of an XML file (from `http://www.cs.wisc.edu/niagara/data/edmunds/`), and suppose we are interested in *Japanese cars* with *unlimited rust warranty*. Using the italicized terms as keywords with appropriate metrical constraints would retrieve the above passage if we consider it as plain text, ignoring all the brackets, but it would probably retrieve many more that are not relevant, thereby affecting precision. On the other hand, by considering each level independently and not allowing relationships between terms in different locations in the tree, even this passage could be skipped, lowering recall. The problem here is that XML files have a tendency to partition and label even text portions most of the users would accept as being closely related in a free text. In our example, the keywords appear on different levels, in different branches, and some even in the bracketed tags themselves. There is thus a need to define distances across the possible locations in an XML tree, as has been done in [21] in an application to ranking. Our approach concentrates on the Boolean retrieval process itself.

2. Formal definition of the problem

XML documents are generally processed using specially designed languages, like XPath 1.0 [17], XPath 2.0 and XQuery 1.0 [18], which have been suggested by the W3C, presenting sophisticated rules to select specific nodes in the tree. We shall, however, adopt a more IR based approach, and henceforth deal with queries of the following type: a query consists of m keywords and $m-1$ binary distance constraints, as in

$$A_1 \;\; [l_1 : u_1] \;\; A_2 \;\; [l_2 : u_2] \;\; \cdots \;\; A_{m-1} \;\; [l_{m-1} : u_{m-1}] \;\; A_m. \tag{1}$$

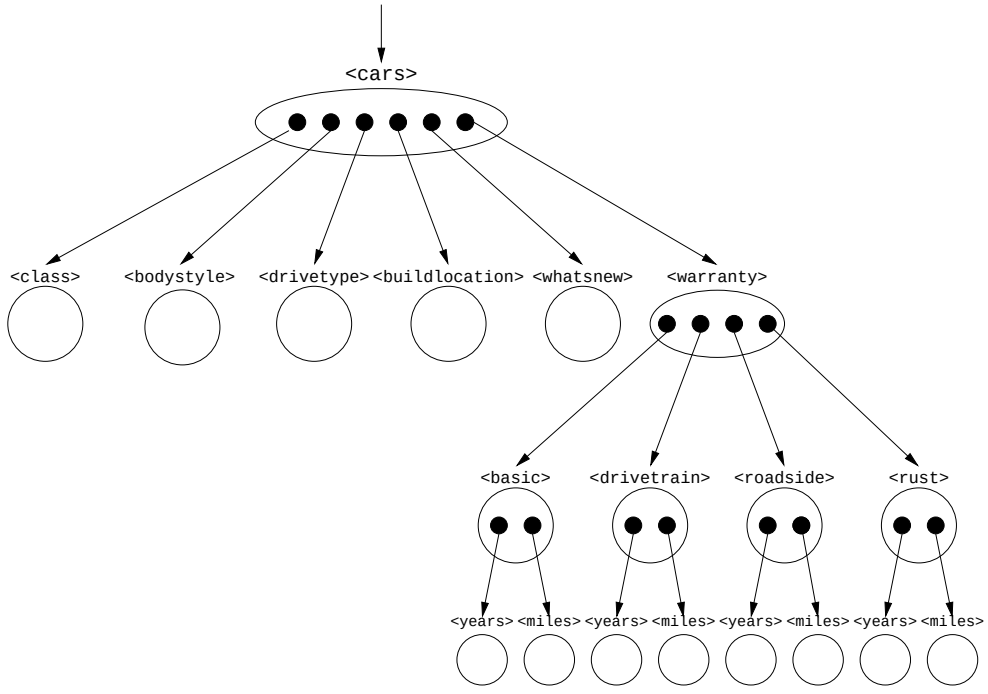
This is a conjunctive query, requiring all the keywords A_i to occur within the given metrical constraints specified by l_i, u_i, which are (positive or negative) integers satisfying $l_i \leq u_i$ for $1 \leq i < m$, with the couple $[l_i : u_i]$ imposing a lower and upper limit on the distance

```
<cars> ...
<class>Sport Utility</class>
<bodystyle>Wagon</bodystyle>
<drivetype>RWD</drivetype>
<buildlocation>Tahara, Japan</buildlocation>
<whatsnew>Base models have been killed, leaving Limited
and SR5 trim levels equipped with a standard automatic
transmission, Vehicle Skid Control (VSC), traction control
and ABS with electronic brake force distribution and brake
assist. All 4Runners have power door locks this year, as well
as ... </whatsnew> ...
<warranty>
<basic><years>3</years><miles>36,000</miles></basic>
<drivetrain><years>5</years>
<miles>60,000</miles></drivetrain>
<roadside><years></years><miles></miles></roadside>
<rust><years>5</years><miles>Unlimited</miles></rust>
</warranty> ...
</cars>
```

(a) *The text itself*

```
0  <cars>
   1  <class>
   1  <bodystyle>
   1  <drivetype>
   1  <buildlocation>
   1  <whatsnew>
   1  <warranty>
      2  <basic>
         3  <years>
         3  <miles>
      2  <drivetrain>
         3  <years>
         3  <miles>
      2  <roadside>
         3  <years>
         3  <miles>
      2  <rust>
         3  <years>
         3  <miles>
```

(b) *Tree structure*



(c) *XML tree*

FIGURE 1: *Example of a part of an XML file*

from A_i to A_{i+1}. Negative distance means that A_{i+1} may appear before A_i in the text. The distance is measured in words.

Such a query language is used for over thirty years at the *Responsa Retrieval Project* [6]. Note that similarly to DB queries, the metrical constraints allow a precise description of the required expressions. The fuzziness of the IR approach is deferred here to user feedback: if the number of retrieved items is too large or too small or the items themselves are not satisfactory, the user can broaden or restrict the query iteratively by changing the constraints and/or the keywords.

In a more general setting, one could also consider extended queries, consisting of several disjuncts, each having a form similar to (1). The requested set of locations to be retrieved is then simply the union of the sets of locations to be retrieved for each of the disjuncts. We may therefore restrict our attention to queries of the form (1).

In its simplest form, the keyword A_i is a single word or a (usually very small) set of words given explicitly by the user. In more complex cases a keyword A_i will represent a set of words $A_i = \bigcup_{j=1}^{n_i} A_{ij}$, all of which are considered synonymous in the context of the given query. For example, a variable-length-don't-care-character $*$ can be used, allowing the use of prefix, suffix and infix truncation, or one could use a thesaurus (`month` representing `January`, `February`, etc.), or some morphological processing (`do` representing `does`, `did`, `done`, etc.).

For every word W, let $\mathcal{C}(W)$ be the ordered list of the coordinates of all its occurrences in the text. The problem of processing a query of the form (1) consists then, in its most general form, of finding all the m-tuples $(a_1, \ldots, a_m)$ of coordinates satisfying

$$\forall i \in \{1, \ldots, m\} \quad \exists j \in \{1, \ldots, n_i\} \qquad \text{with} \quad a_i \in \mathcal{C}(A_{ij})$$

and

$$l_i \leq d(a_i, a_{i+1}) \leq u_i \qquad \text{for } 1 \leq i < m,$$

where $d(x, y)$ denotes the distance in words from x to y, i.e., if $I(x)$ denotes the index of the word x in the sentence, then

$$d(x, y) = \begin{cases} I(y) - I(x) & \text{if } x \text{ and } y \text{ are in the same sentence} \\ \infty & \text{otherwise.} \end{cases}$$

Every m-tuple satisfying these constraints will be retrieved and the corresponding locations in the text are presented to the user [3]. Our problem is to extend these definitions to an XML environment. For simplicity of exposition, we shall ignore attributes in XML nodes and refer to simple XML trees in which each node corresponds to one bracketed tag and may contain a sequence of items, each of which being either a word or a pointer to a child node in the tree. For example, the root of the tree in Figure 1(c) corresponds to the sub-part of the file in Figure 1(a) enclosed between the tags `<cars>` and `</cars>`. For an example in which a node may include both words and pointers, refer to Figure 3(b) below.

The following example shows how users can utilize the metrical constraints to help them retrieve relevant data, in spite of having no information on the distance between their chosen keywords in the various occurrences in the text. Consider a query about solving some differential equations, formulated as:

```
solv* [-5:9] differential [1:1] equation*
```

The constraint `[1:1]` between the last two term means that they ought to appear consecutively, according to our belief that `differential equation` is a widespread phrase that rarely contains additional words. On the other hand, the distance between `solv*` (standing for `solve`, `solving`, `solved`, etc.) and `differential` could be larger, as a text may contain a phrase like `solving these differential equations`, or even `solving this well known set of difficult non linear differential equations`. One can obviously come up with even longer and still relevant expressions, but with increasing distance, the occurrence of such phrases becomes very unlikely. Choosing the distance 9 as limit reflects a certain tradeoff depending on our knowledge (or guess) about what might appear in the text. The negative distance $l_i = -5$ means that the term `solv*` may in fact occur *after* `differential`, but that in this case it should be closer. This should capture occurrences of phrases like `differential equations solved by`... or `differential equations that can be solved by`..., etc. Again, larger distances are certainly possible, and the query expresses our assertion that phrases with larger distances should not be retrieved. If one wishes to improve recall, the distances should be increased, but this will generally come at the price of reduced precision, and vice versa.

3. Possible solutions

For the special case of annotations, three approaches are possible:

1. they can be ignored, that is, marked as non-retrievable text;
2. they can be embedded as regular text;
3. they can be treated as kind of special text.

While the first possibility may be plausible for annotations due to their lower priority and the general assumption that only a small part of the documents appear in annotations, this is not an alternative in an XML environment. For the example in Figure 1, it would mean that all the subtrees would be ignored, or at least all the subtrees of depth more than 1. Thus all the details about the warranties (which kind, for how long or for how many miles) would be lost. This could badly affect recall.

The second option is equivalent to a linearization of the XML file, which implies that the information conveyed by the hierarchical structure is lost. All the words in the file are taken into account, so no information should be skipped, but many irrelevant passages will now be retrieved, lowering precision. Examples for this second option will be given below.

We are thus left with the third alternative, which, in the XML case, would mean that we need data structures and processing tools to permit standard full-text retrieval in spite of the presence of XML markers. In particular, proximity searches should be processed correctly, i.e., the numbering of the words in each layer should not be altered by the appearance of a sub-branch of the XML tree.

Consider, for example, the following sentence and its corresponding XML tree in Figure 2:

$$a_1 \quad a_2 \quad a_3 \quad \langle \texttt{xxx} \rangle \quad b_1 \quad b_2 \quad \langle \texttt{yyy} \rangle \quad c_1 \quad c_2 \quad \langle \texttt{/yyy} \rangle \quad b_3 \quad \langle \texttt{/xxx} \rangle \quad a_4 \quad a_5. \qquad (2)$$

the distance between a_i and a_j should be $j-i$, without taking the subtree $\langle \texttt{xxx} \rangle \ \ldots \ \langle \texttt{/xxx} \rangle$ into account. However, in a certain sense, b_1 also follows immediately the term a_3, and

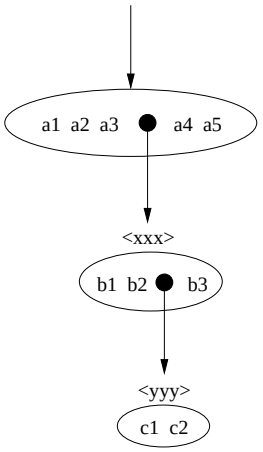


FIGURE 2: *XML tree of the example sentence labeled* (2)

similarly c_1 should be considered as following b_2 at distance 1, and also a_3 at distance 3. This implies several ambiguities in the required internal relative numbering of the terms, which can only be solved by means of some additional tags that have to be adjoined to each coordinate in the postings list.

It should therefore be decided how to define the distance between terms belonging to different layers of the XML tree. The problem is, that our intuitive grasp of the relationships between such terms does not always suggest symmetrical decisions. While most users would accept in the example in (2) defining b_1 to be an immediate successor of a_3, would it be the same when trying to define the distance from b_3 to a_4? More generally, should we define a distance between words x and y, when y appears *after* x but in a higher layer than x? Should this distance then depend on the length of the path from their lowest common ancestor, where the length should be considered both in terms of number of words and in terms of number of layers? We shall advocate positive answers to some of these questions and suggest a generalized definition of the distance function, which is both intuitively plausible and mathematically consistent in a sense to be specified below.

In particular, we may wish to define a distance from a term x to any other belonging to a subtree rooted at the node to which x belongs, but restrict *upward* pointers, as in the example, from c_2 to a_5. Obviously, such definitions require additional information to be stored in each of the coordinates in the inverted files build upon the XML data.

This information must in effect encode the exact position of each term in the hierarchical structure, which may create a serious storage problem. One should thus also deal with practical implementation issues, like compressing such extended concordances efficiently [2].

3.1 Extending the data structures

There are several possibilities for choosing the appropriate data structures to allow efficient processing of the queries of an IR system, depending on the size of the underlying database. For small systems, brute force pattern matching is often fast enough. One would then first locate the different keywords in the query, and then check if the metrical constraints are satisfied. For an XML environment, there is an additional need to verify if special tags in brackets appear between the located keywords.

In the most general case, *inverted files* are used, generally coupled with a ranking mechanism based on the vector space model. First, the *dictionary* of all the different terms in the text is formed, then the *concordance* is built, which stores, for each term, the lexicographically sorted coordinates of all its occurrences. In a typical large full-text IR system, a coordinate is usually given by some k-tuple, such as (d, p, s, w), where d could be the index of the document of the given occurrence, p the index of the paragraph within the documents, s the number of sentence within the paragraph and w the number of word in the sentence. Other hierarchies are also possible, in particular, an XML file has already an underlying tree structure. A coordinate for a term in an XML file would then consist of a sequence of integers giving the index of the given node in the tree, followed by the index of the word within the text stored in the given node. For more details on the system architecture, as well as possible weighting schemes, see, e.g., [1].

Consider for instance again the part of the XML file given in Figure 1. Like any tree, the hierarchical structure can be described by listing the nodes in DFS order, giving for each node its level and the corresponding tag. Figure 1(b) depicts this tree structure for the XML file of Figure 1(a). Similarly to the Dewey classification scheme used in Library systems, any node x can then be defined by a sequence of integers of the form $(k\,; n_1, \ldots, n_k)$, where k is the depth of x in the tree, $(0\,;\,)$ represents the root of the tree,

and n_i is the running index (starting to count from 1) of x as child of its parent, which is the node defined by the sequence $(i-1\,;n_1,\ldots,n_{i-1})$. Thus the node labeled `buildlocation` would be identified by $(1\,;4)$, and the sequence corresponding to the milage of the rust warranty is $(3\,;6,4,2)$.

Any node in the tree can store free text, which could itself be indexed hierarchically as described above. We shall however prefer here a flat representation, identifying each word in the text by its running index w in the sequence of words, without taking sentences or paragraphs into account, for the following reasons:

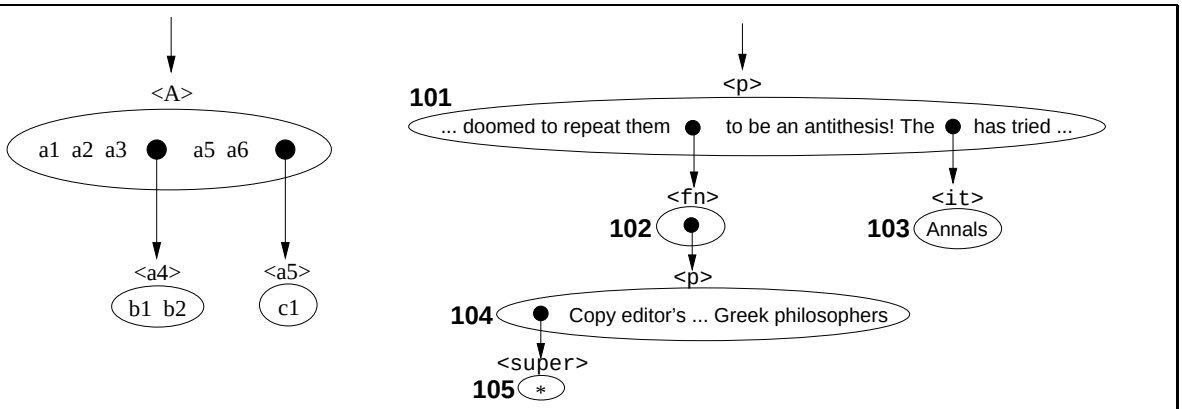
- Many textual entries in XML files are very short, and are in fact just a reformulation of some tabular data in a more verbose form. The definition of sentences for such data would be rather artificial and also wasteful from the storage point of view;
- Even for the larger textual passages, the partition into sentences is not always obvious, as relying only on punctuation signs may be misleading. To cite an example from [7], should the phrase "*Hello! I said to Mr. Jones.*" really be parsed as consisting of three sentences?
- A flat numbering will allow us below to define distances between elements at different levels of the XML tree, a task which would otherwise require a considerable amount of overhead storage.

The full coordinate of any word W in an XML file can thus be defined as having the form

$$(k\,;n_1,\ldots,n_k\,;w), \tag{3}$$

where w is the running index of the word W within the node $(k\,;n_1,\ldots,n_k)$. For example, the coordinate of the word `leaving` appearing in the `<whatsnew>` node is $(1\,;5\,;6)$, and the coordinate of the term `Unlimited` is $(3\,;6,4,2\,;1)$.

The Dewey numbering used above is not the only one possible for getting exact references of every node of the XML tree, see [4] for a recent survey. In one of the other possible schemes, each node is identified by a triplet $(start, end, level)$, where *start* and *end* are the indices of the given node in a preorder, respectively postorder, traversal of the XML tree,



(a) *XML tree of* (4) (b) *Example from INEX*

FIGURE 3: *Examples of text in internal nodes*

and *level* is the depth of the node in the tree. A recent alternative, called *BIRD* numbers, yields even better properties [19]. In both schemes, ancestor-descendant relations are not as obvious as for Dewey, but can be evaluated efficiently. On the other hand, their storage requirements are lower than for the full hierarchical Dewey layout, but it has already been noted in [2] that a list of hierarchical coordinates, though seeming wasteful at first sight, might at times be more compressible than an equivalent list of more compact items. Our proposal for defining the distance could in any case just as well be based on one of the alternative numbering schemes, and we shall use Dewey order only for the sake of simpler exposition.

Note that in the example in Figure 1, as well as in many common XML files, free text is stored only in the leaves of the tree, which means that text and XML tags are not freely intermixed. This implies that examples like that in Figure 3(a), corresponding to a text of the form

$$\langle \mathtt{A} \rangle \quad a_1 \quad a_2 \quad a_3 \quad \langle a_4 \rangle \quad b_1 \quad b_2 \quad \langle /a_4 \rangle \quad a_5 \quad a_6 \quad \langle a_7 \rangle \quad c_1 \quad \langle /a_7 \rangle \quad \langle /\mathtt{A} \rangle, \tag{4}$$

where the a_i, b_i and c_i are single terms, do not comply with this standard, making much of our discussion about the processing of metrical constraints across node boundaries obsolete. It has, however, already be noted [1] that this is not a general rule, and there are many counterexamples in the INEX collection, in particular in the presence of footnotes. Figure 3(b) displays such an example from `inex-1.4/xml/an/1995/a1003.xml`. We shall therefore stick to our more general settings, as they could be considered as a shorthand

way of writing in the case of empty leaves. For example, the text of the node labeled ⟨A⟩ of Figure 3(a) could be written in the more standard way as

$$\langle \mathtt{A} \rangle\ \langle a_1 \rangle \langle /a_1 \rangle\ \langle a_2 \rangle \langle /a_2 \rangle\ \langle a_3 \rangle \langle /a_3 \rangle\ \langle a_4 \rangle\ \ b_1\ b_2\ \langle /a_4 \rangle\ \langle a_5 \rangle \langle /a_5 \rangle\ \langle a_6 \rangle \langle /a_6 \rangle\ \langle a_7 \rangle\ c_1\ \langle /a_7 \rangle\ \langle /\mathtt{A} \rangle,$$

that is, each term a of free text within an internal node of the XML tree is considered as representing an empty sub-branch $\langle a \rangle \langle /a \rangle$.

3.2 Redefining the distance operator

To be consistent with the above discussion, the index of the term a_i in the example sentence labeled (4) should be i, that is, we count bracketed tags just as single words (and ignore in this study the possibility of parameters and attributes that are adjoined to the tags, or tags that consist of more than a single word). This is different of what we first suggested when discussing the example sentence labeled (2). For example, if the node containing the text of (4) is represented by $(k\,; n_1, \ldots, n_k)$, then the coordinates of a_i, for $i \notin \{4, 7\}$, would be $(k\,; n_1, \ldots, n_k\,; i)$, the coordinates of b_i would be $(k+1\,; n_1, \ldots, n_k, 4\,; i)$, and that of c_1 would be $(k+1\,; n_1, \ldots, n_k, 7\,; 1)$.

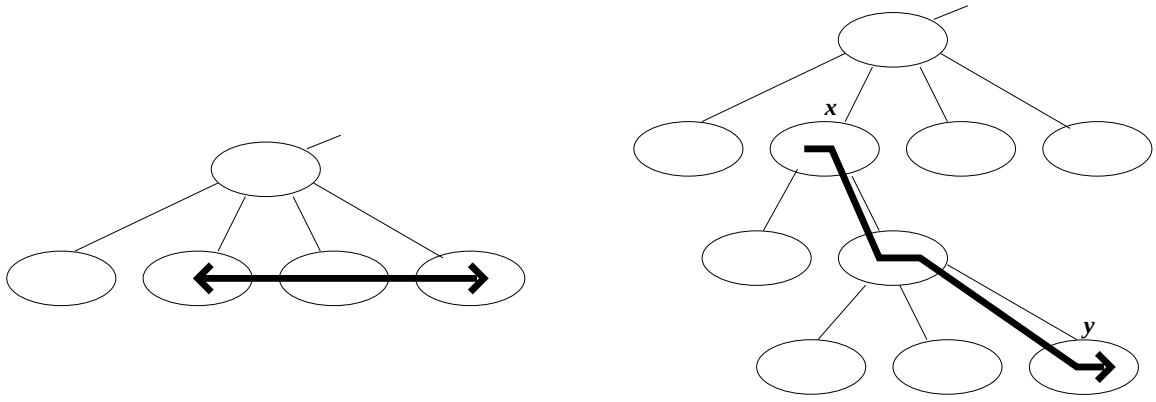
When defining the distance between two words C_1 and C_2, which has to be evaluated as some function of their coordinates, the easiest would be to require both words to belong to the same node of the XML tree, but that would be too restrictive. We wish to extend this definition also to more general cases. For example, we could define the distance from C_1 to C_2 also when:

1. the words belong to sibling nodes, i.e., the nodes they belong to have a common parent node; this reflects our intuition that many texts wouldn't have been disrupted by tags and thereby forcefully (and often possibly unnecessarily) layered, had they not appeared in an XML environment. In the example of Figure 3(a), we would like to define a distance between b_i and c_1. This extension should be optional, and controlled by a parameter L, to be defined below;

2. C_1 belongs to a node which is an ancestor of the node C_2 belongs to, unless the difference in levels between the two nodes is larger than some predetermined parameter

D. As additional constraint we require that C_1 should precede, within its node x, the tag B which is the root within x of the subtree to which C_2 belongs. The reason for this restriction is that if C_1 follows B, its distance to B would be negative, so it is not clear how to define the distance to an element in a lower level: partitioning the path from C_1 to C_2 into parts and adding the partial distances would not be consistent, in the sense described below, in the presence of negative distances.

In all other cases, the distance should not be defined, or equivalently, set to infinity. Note that this definition is not always symmetrical, and while $d(x_1, x_2) = -d(x_2, x_1)$ if x_1 and x_2 belong to the same level, it does not hold in the other cases.



(a) *for sibling nodes* (b) *for nodes on different levels*

FIGURE 4: *Schematic representation of coordinates for which a distance is defined*

Figure 4 is a schematic representation of the above definitions. The double-sided boldface arrow in Figure 4(a) represents the distance between words belonging to sibling nodes and corresponds to the first case above. For the second case, the one-sided arrow in Figure 4(b) represents the distance from a node x to a node y, which belongs to a lower level in the tree. The additional constraint implies that such arrows will always go top-down, left to right.

As example, refer to the trees in Figure 3. For the first case, we would like to define both distances $d(b_2, c_1)$ and $d(c_1, b_2)$ in the tree of Figure 3(a). For the second case, in Figure 3(b), the distance from the word antithesis in node **101** to the word Annals in node **103** should be defined, but not the distance from the same word antithesis to the word Greek

in node **104**; indeed, the black dot between the words them and to in node **101**, which is the tag acting as root of the subtree to which the term Greek belongs, precedes, rather than follows, the term antithesis and thus violates the constraint.

Distance function $d(C_1 = (k_1\,; n_1, \ldots, n_{k_1}\,; w_1), C_2 = (k_2\,; m_1, \ldots, m_{k_2}\,; w_2))$

$d(C_1, C_2) \longleftarrow \infty$ /* set default */

if $k_1 = k_2 = k$ then /* nodes on same level */

if $n_i = m_i$ for all $0 \le i \le k - L$ then

if $L = 0$ then

$d(C_1, C_2) \longleftarrow w_2 - w_1$

else if $L = 1$ /* siblings allowed */

if $n_k = m_k$ then

$d(C_1, C_2) \longleftarrow w_2 - w_1$

else if $n_k < m_k$ then

$$d(C_1, C_2) \longleftarrow \ell(n_1, \ldots, n_k) - w_1 + \sum_{j=n_k+1}^{m_k-1} \ell(n_1, \ldots, n_{k-1}, j) + w_2$$

else /* here $m_k < n_k$ */

$$d(C_1, C_2) \longleftarrow -\left(\ell(m_1, \ldots, m_k) - w_2 + \sum_{j=m_k+1}^{n_k-1} \ell(m_1, \ldots, m_{k-1}, j) + w_1\right)$$

else if $k_1 < k_2$ and $k_1 + D \ge k_2$ then

if $n_i = m_i$ for all $0 \le i \le k_1$ and $w_1 < m_{k_1+1}$ then

$$d(C_1, C_2) \longleftarrow m_{k_1+1} - w_1 + \sum_{t=k_1+2}^{k_2} m_t + w_2$$

FIGURE 5: *Distance definition function*

The parameter L indicates the number of levels allowed to the lowest common ancestor of the two coordinates, which means in our case that either $L = 0$ (same node required) or $L = 1$ (could be in sibling nodes), but L could also be chosen larger for certain applications, in particular when the underlying database is known to have a deep structure with many nesting levels. The parameter D controls the depth of what we still consider to be logically connected, with $D = 0$ prohibiting any level differences. The definition of a query should therefore be amended by allowing it to be preceded by an optional pair (L, D). If omitted,

some reasonable default could be assumed, e.g., $(1, 2)$.

Since the definition of the distance may also depend on the length of the text within a given node, we introduce the notation $\ell(n_1, \ldots, n_k)$ as representing the number of words (text and tags) stored in the node $(k \ ; n_1, \ldots, n_k)$. In particular, if in node $(k-1 \ ; n_1, \ldots, n_{k-1})$, the ith element is a word and not a bracketed tag, we define $\ell(n_1, \ldots, n_{k-1}, i) = 1$. The formal definition of the distance operator is then given in Figure 5. We assume two coordinates are given, $C_1 = (k_1 \ ; n_1, \ldots, n_{k_1} \ ; w_1)$ and $C_2 = (k_2 \ ; m_1, \ldots, m_{k_2} \ ; w_2)$. We deal here only with the cases $L \leq 1$, as suggested above, but the definition could be extended, for certain applications, also to $L > 1$, which would mean that a distance should not only be defined for siblings ($L = 1$), but also for cousins ($L = 2$), second degree cousins ($L = 3$), etc.

The definition, though, would get increasingly complex. For example, for $L = 2$, the distance from C_1 to C_2 in the case when C_1 precedes C_2, would be given by:

$$
\begin{aligned}
d(C_1, C_1) \ \longleftarrow \ & \underbrace{\ell(n_1, \ldots, n_k) \ - \ w_1}_{\mathsf{A}} \ + \ \underbrace{\sum_{j=n_k+1}^{\ell(n_1,\ldots,n_{k-1})} \ell(n_1, \ldots, n_{k-1}, j)}_{\mathsf{B}} \\
& + \ \underbrace{\sum_{r=n_{k-1}+1}^{m_{k-1}-1} \ \sum_{j=1}^{\ell(n_1,\ldots,n_{k-2},r)} \ell(n_1, \ldots, n_{k-2}, r, j)}_{\mathsf{C}} \\
& + \ \underbrace{\sum_{j=1}^{m_k-1} \ell(n_1, \ldots, n_{k-2}, m_{k-1}, j)}_{\mathsf{D}} \ + \ \underbrace{w_2}_{\mathsf{E}} \ .
\end{aligned}
$$

Figure 6 brings a typical example of a portion of an XML tree on which the distance between the coordinates C_1 and C_2 (symbolized by the black dots) is partitioned into the parts of the above formula which are labeled accordingly.

Note that the defined distance is not a metric in the mathematical sense: it is not always positive, it is not symmetric and not even anti-symmetric, as $d(x, y)$ might be finite while $d(y, x) = \infty$. Moreover, the triangle inequality does not necessarily hold, as can be seen in the following example referring to Figure 2: using the parameter $D = 1$

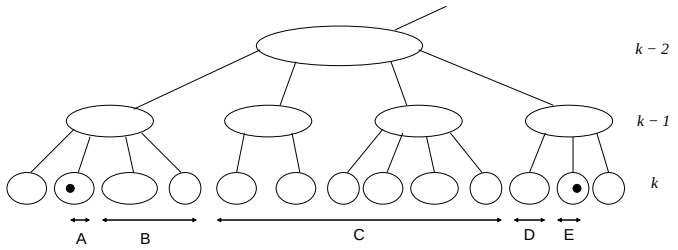


FIGURE 6: *Example of a part of an XML tree to visualize the distance between coordinates when* $L = 2$

(i.e., allowing only adjacent levels),

$$\infty \;=\; d(a_3, c_2) \;>\; d(a_3, b_1) \;+\; d(b_1, c_2) \;=\; 2 + 4.$$

There is, nevertheless, some consistency rule obeyed by the distance d, namely that if $d(x, y)$ is finite, then for any elements z and t on the path from x to y, $d(z, t)$ must be smaller. More formally, if $\mathcal{C}$ is the set of coordinates, then

$$\forall x, y, z, t \in \mathcal{C} \qquad x \preceq z \prec t \preceq y \qquad \longrightarrow \qquad d(z, t) \leq d(x, y),$$

with equality only if $x = z$ and $t = y$. Here $x \prec y$ denotes that coordinate x precedes coordinate y in the sense defined above, that is, either x and y belong to nodes on the same level of the XML tree and x has a (lexicographically) lower index, or y belongs to a node that is a descendant of the node of x, and x precedes in its node the pointer to the subbranch to which y belongs; $x \preceq y$ is a shortcut for $x \prec y \;\vee\; x = y$.

3.3 Implementation issues

A concordance consisting of coordinates of the form described in the displayed formula (3) creates a serious storage problem. This can be alleviated by indexing beforehand the nodes of the XML tree, and referring in the coordinate to the index rather than the hierarchical sequence of numbers. For even greater storage savings, the pointers to the nodes of the tree could be Huffman encoded, according to the frequencies of reference in the concordance of the various nodes of the tree.

There is, however, a problem with such a compression scheme, as one looses the information about the sibling and ancestor-descendent relationships that are needed in the

distance evaluation. This can be solved, by adding as additional data structure, some directed graphs $G_k = (V, E_k)$ to the system. The vertices V are the nodes of the XML tree, and $(x, y) \in E_k$ if and only if x and y are either siblings (have a common parent node), or there is a path of length $\leq k$ from x to y in the XML tree. Thus for $k = 1$, one gets that G_1 is the underlying tree structure of the XML file itself to which edges between sibling nodes have been added, and the graphs G_k can be useful to process queries with $D = k$. Figure 7 shows the graphs G_1, G_2 and G_3 corresponding to the INEX example of Figure 3(b). For example, the distance from the word doomed in node **101** to the word Greek in node **104** should be defined for $D \geq 2$, because there is an edge from node **101** to node **104** in G_2 and G_3, but not for $D = 1$, because this edge is lacking for G_1.

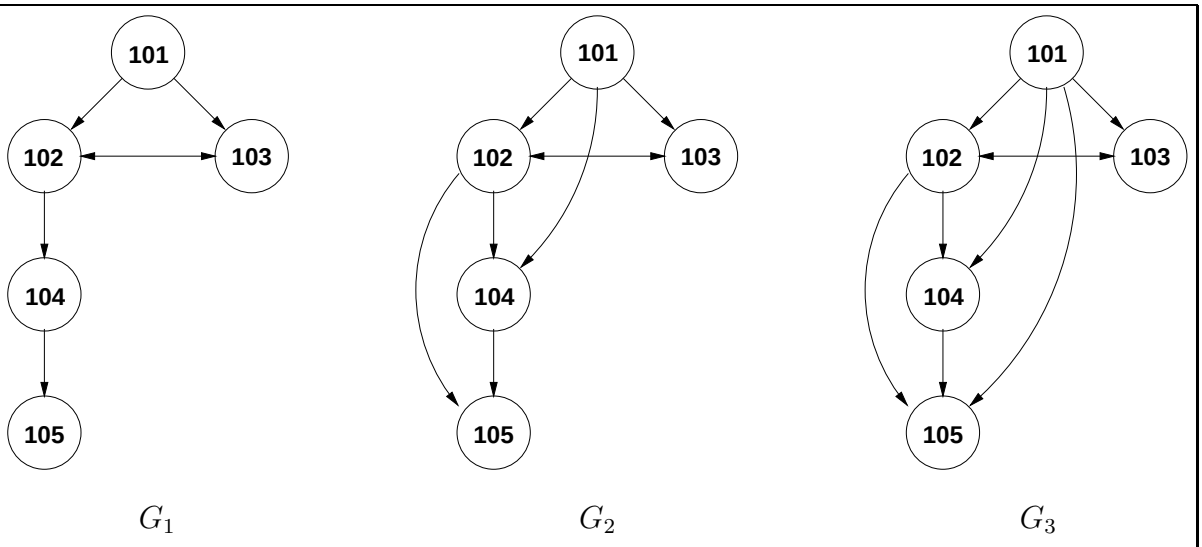


FIGURE 7: *The graphs G_k with $k \leq 3$ for the XML tree of Figure* 3(b)

Another important aspect is taking advantage of the information stored in the XML metadata. The bracketed commands not only induce the tree structure, but also label the nodes of this tree with *tags*. One should also consider defining distances between words in different nodes of the tree, if both are labeled with the same (or even related?) tags, and even if these nodes appear in different locations of the tree that are not connected in an ancestor-descendent relationship. Moreover, the tags themselves often convey relevant information and should therefore also be taken into consideration when the indices of the words are assigned. Thus it will not suffice to store, in each coordinate of the index, an encoding of its hierarchical position, but we need in addition also some pointer to the

corresponding tag.

Adding a special tag-field to each coordinate is probably too expensive and could be an overkill. One should rather define a set of preset "important" tags, or maybe select all those that appear more often than some chosen threshold, and build for each tag in that set a list of pointers to its occurrences in the various nodes of the tree. Conversely, each node corresponding to a tag of that selected list should have a pointer to the tag. Using these pointers and the graph G we are able to reconstruct, for each coordinate in the XML tree, the sequence of tags leading to it, which allows us to formulate sophisticated queries.

4. Examples

Our treatment has been mainly theoretical in nature. To get some examples on real data, we ran several sample queries on the INEX database (`inex-1.4/xml/`) of more than 600 MB of raw XML data, and compared the retrieval performance of various levels of sophistication of the query language.

In the first approach, the XML file is considered as flat text, the tags just being considered as additional terms. The hierarchical structure of the file, with its implicit information, is lost, but treating metrical constraints is easy, though probably not always reflecting the user's expectations. Indeed, if a phrase is disrupted by a sub-branch of the XML tree, then terms that usually appear together may find themselves at unforeseeable distance apart. For example, the distance between the terms `them` and `to` in Figure 3(b) would be at least 13! The second retrieval method restricts distances to terms belonging to a same node of the XML tree. This could be justified if the nodes contain large text portions, but for deep trees with little text within the nodes, it might be too restrictive. The third approach is the hybrid one described in this work and formally defined in Figure 5.

It should be emphasized that the following examples are by no means an attempt of a statistical validation of the suggested approach. For such a validation, the required sample ought to be large, and one would need a set of "random" or "typical" queries, which not only are hard to define, but would probably not be interesting from the retrieval point

of view. For the alternative of using some well-known large enough test set, like the INEX queries, one would need some automatic tool translating its queries into a set of corresponding queries including metrical constraints between some of their keywords as advocated in this paper. However, no such automatic tool could exist, as the proper formulation of the query — choosing the best keywords, the correct distances and how to combine all these together — is in itself the major parameter by which recall and precision can be controlled in the presence of metrical constraints. Typically, a user would start with some rough approximation of what she or he believes is a good query and then gradually refine it, based on the retrieval results.

The problem is that the performance of classical IR queries and that of queries with metrical constraints are not directly comparable, because the basic assumptions are different. In the former setting, the information sought by the user is often only vaguely described, and it is the duty of the retrieval system to appropriately extend the set of keywords, and to select, generally by means of some ranking scheme, the part of the set of text passages satisfying the request that will finally be presented. Usually, not all the retrieved items are relevant, nor are all relevant items retrieved, so precision and recall are smaller than 1, and sometimes even much smaller. When distance operators are added to the query language, one gets a much more precise description of the information needs of the user, and the system can be programmed to retrieve *all* the text locations satisfying the constraints, and only those. One can then define relevancy as obeying precisely the rules set by the query, so that one would always get a perfect score of 1 for both precision and recall.

Because of these different settings, we shall not try to give a quantitative estimate by how much the suggested approach would improve the performance on some given database and corresponding set of queries; the results would not be reliable even for the given set, and moreover, would say nothing about other queries on other XML files with different global structure. We rather bring a few real life examples which support the thesis about the usefulness of the suggested approach.

To compare the three approaches mentioned above, we used the following semi-automated process for each given query:

1. Create (manually) a corresponding Boolean query, choosing the keywords and combining them by AND and OR operators;

2. retrieve (automatically) all the paragraphs of the collection satisfying the Boolean query — denote the text formed by the union of these paragraphs by $\mathcal{S}$;

3. partition (manually) the set of retrieved items into relevant and non-relevant ones, and denote the corresponding subsets by $\mathcal{S}_R$ and $\mathcal{S}_{NR}$;

4. choose (manually and independently of the previous steps) the distance constraints;

5. run the new queries on the text $\mathcal{S}$ to obtain the set $\mathcal{R}$ of retrieved items (there is a different set $\mathcal{R}$ for each of the three approaches);

6. define precision as $\frac{|\mathcal{R} \cap \mathcal{S}_R|}{|\mathcal{R}|}$ and recall as $\frac{|\mathcal{R} \cap \mathcal{S}_R|}{|\mathcal{S}_R|}$.

The first two examples are INEX topics, known as topic number 31: *Computational Biology*, and topic number 35: *Medical association rule mining*. As additional topics we chose *Archival data at NASA*, and *Hidden Markov Models in speech processing*. Bear in mind that the title of a query seldom captures the exact definition of the relevant documents a user might wish to retrieve, and that with a query language as the one we deal with, it is the responsibility of the user to formulate a precise query, playing with the metrical constraints and using appropriate search terms, so as to get precision and recall as high as possible. The four topics above have been translated respectively into the following queries, using for all the default $(L, D) = (1, 2)$:

```
*ioinformatic* [-20:20] comput*
medical [-30:30] mining
(Archiv* ∨ archiv*) [-25:25] (nasa ∨ NASA)
(Speech ∨ speech ∨ *coustic*) [-15:15] (hidden ∨ Hidden ∨ HMM)
```

Note the use of the wild-card character `*` to get grammatical variants. For a long enough term like `Bioinformatics`, it could also be used to get both lower and upper case initials, but for the term `hidden`, these had to be given explicitly using the ∨ (OR) operator, since `*idden` could also have matched non-relevant words like `forbidden`. Note

also that for these queries, the order of appearance of the keywords in the text was not important, hence the use of symmetric distance constraints $[l_i : u_i]$ with $l_i = -u_i$; this is not always so, and in other queries, e.g., l_i could be 1, forcing the second keyword to appear after the first.

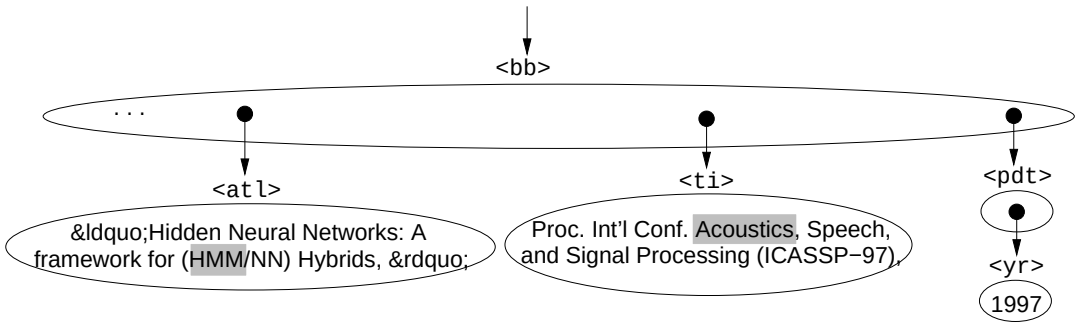
query	flat file		same node		new distance	
	recall	precision	recall	precision	recall	precision
comput. biology	0.83	0.950	0.78	0.947	0.91	0.956
medical mining	0.50	0.235	0.50	0.235	0.75	0.261
archive Nasa	0.73	0.733	0.60	0.692	0.80	0.750
speech HMM	0.86	0.980	0.71	0.987	0.89	0.990

TABLE 1: *Comparative table of retrieval performance*

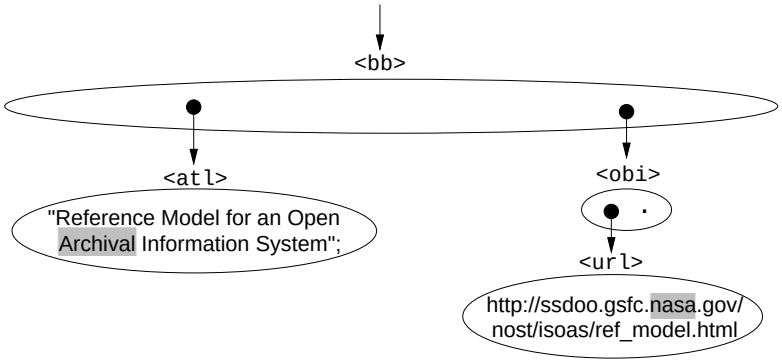
Table 1 summarizes the retrieval performance for each of the three models of distance definition in terms of recall and precision as defined in step 6 above. As this definition is not equivalent to that widely used for classical IR queries, it should not be surprising that almost all the values in the table are rather high, whereas published INEX experiments report quite low values. One can see that the new definition improves, on those examples, on both recall and precision over the other two simpler models, though only slightly. The reason for the small improvement may lie in the nature of the INEX database, which consists mainly of scientific papers and is therefore quite homogeneous. The hierarchical structures in INEX are generally not very deep, and since this exploits only a small part of the potential power of the XML layout, the advantages of the proposed distance definition over the previous ones may not appear to their full extent.

The reason for the low precision in the query on medical mining is that many retrieved passages spoke about researchers and their interest fields, and these often included both data mining and topics related to medicine, but were not about medical mining itself. The query on computational biology, on the other hand, had high precision, as the appearance of the keywords was almost always relevant. An example of a non-relevant occurrence can be found in `ex/2001/x6035.xml`: *...joined the Institute for Algorithms and Scientific Computing at the German National Research Center for Information Technology as a*

bioinformatics PhD student.



(a) *Example from* `tp/2002/i0467.xml` (retrieved)



(b) *Example from* `co/2001/r3024.xml` (not retrieved)

FIGURE 8: *Examples of relevant items with keywords in different nodes*

Figure 8 shows two examples in which a relevant passage contains keywords (highlighted in the figure) that appear in different nodes of the XML tree. In Figure 8(a) for the query on speech and HMM, the keywords appear on the same level, but not as siblings in the same node, and in Figure 8(b) for the query on NASA, the keywords not even appear on the same level. According to our rules, the second item would not be retrieved, since the nodes including the keywords, labeled `<atl>` and `<url>`, are neither siblings, nor is one a descendant of the other. In an approach restricting the search to within a given node, even the first item would clearly have been missed. The other extreme approach of ignoring all structure would possibly catch these two occurrences, but miss many others, as the distances between connected terms could be unreasonably increased by the appearance of sub-branches of the XML tree. Moreover, this other approach would probably also retrieve many more non-relevant items.

5. Conclusion

Files in XML format will become more widespread in the near future and our query languages should be adapted to deal with more complex queries. In this study, we have concentrated on metrical constraints, which are handled for long in classical full text IR systems, and have tried to generalize them to the special structures induced by the XML tree. The suggested method is more complicated, and thus consumes more time and space than the simpler alternatives of restricting oneself to search within a given node on the one hand, or ignoring the XML structure altogether on the other hand. The retrieval performance in both recall and precision may be improved, so that the investment in additional resources may be payed off.

References

[1] Anh V.N., Moffat A., Compression and an IR approach to XML retrieval, *Proc. first INEX Workshop* (2002) 99–104.

[2] Choueka Y., Fraenkel A.S., Klein S.T., Compression of Concordances in Full-Text Retrieval Systems, *Proc. 11-th ACM-SIGIR Conf.,* Grenoble (1988) 597–612.

[3] Choueka Y., Fraenkel A.S., Klein S.T., Segal E., Improved Techniques for Processing Queries in Full-Text Systems, *Proc. 10-th ACM-SIGIR Conf.,* New Orleans (1987) 306–315.

[4] Gou G., Chirkova R., XML query processing: a survey, Technical report TR–2005–22, North Carolina State University (2005).

[5] Cohen S., Kanza Y., Kogan Y., Nutt W., Sagiv Y., Serebrenik A., EquiX — A search and query language for XML, *Journal of the American Society for Information Science* **53** (2002) 454–466.

[6] Fraenkel A.S., All about the Responsa Retrieval Project you always wanted to know but were afraid to ask, Expanded Summary, *Jurimetrics J.* **16** (1976) 149–156.

[7] Fraenkel A.S., Klein S.T., Information retrieval from annotated texts, *Journal of the American Society for Information Science* **50** (1999) 845–854.

[8] Frei H.P., Stieger D., The use of semantic links in Hypertext Information Retrieval, *Information Processing & Management* **31** (1995) 1–13.

[9] Fuhr N., Grossjohann K., XIRQL: an XML query language based on Information Retrieval concepts, *ACM Trans. on Information Systems* **22** (2004) 313–356.

[10] Fuhr N., Gövert N., Kazai G., Lalmas M., ed., *Proc. first Workshop of the INitiative for the Evaluation of XML Retrieval (INEX)*, Dec. 9–11, 2002, Schloss Dagstuhl, http://qmir.dcs.qmul.ac.uk/inex/

[11] Fuhr N., Lalmas M., Malik S., ed., *Proc. second Workshop of the INitiative for the Evaluation of XML Retrieval (INEX)*, Dec. 15–17, 2003, Schloss Dagstuhl, http://inex.is.informatik.uni-duisburg.de:2003/

[12] Geva S., Extreme file inversion, *Proc. first INEX Workshop* (2002) 155–161.

[13] Mass Y., Mandelbrod M., Retrieving the most relevant XML components, *Proc. second INEX Workshop* (2003) 58–64.

[14] Ogilvie P., Callan J., Using language models for flat text queries in XML retrieval, *Proc. first INEX Workshop* (2002) 33–40.

[15] O'Keefe R.A., Trotman A., The simplest query language that could possibly work, *Proc. second INEX Workshop* (2003) 117–124.

[16] Special Issue on XML and Information Retrieval, *Journal of the American Society for Information Science and Technology* **53**(6) (2002).

[17] World Wide Web Consortium, XML Path Language (XPath), Version 1.0, W3C Recommendation 16 November 1999, http://www.w3.org/TR/xpath/.

[18] World Wide Web Consortium, XQuery 1.0 and XPath 2.0 Data Model, W3C Working Draft, 29 October 2004, http://www.w3.org/TR/xpath-datamodel/.

[19] Weigel F., Schulz K.U., Meuss H., The BIRD numbering scheme for XML and tree databases – Deciding and reconstructing tree relations using efficient arithmetic operations, *Proc. Third International XML Database Symposium, XSym–2005*, Trondheim, Norway, LNCS **3671**, Springer Verlag (2005) 49–67.

[20] The XML industry portal, http://www.xml.org.

[21] Yoo S., An XML retrieval model based on structural proximities, *Proc. first INEX Workshop* (2002) 125–132.

[22] Yu C., Qi H., Jagadish H.V., Integration of IR into an XML database, *Proc. first INEX Workshop* (2002) 162–169.