Information Retrieval of Sequential Data in Heterogeneous XML Databases

Eugen Popovici, Pierre-François Marteau, Gildas Ménier

VALORIA Laboratory, University of South-Brittany
BP 573, 56017 Vannes Cedex, France
{Eugen.Popovici, Pierre-Francois.Marteau, Gildas.Menier}@univ-ubs.fr

Abstract. The XML language is a W3C standard sustained by both the industry and the scientific community. Therefore, the available information annotated in XML keeps and will keep increasing in size. Nonetheless, not only the volume of the XML information is increasing but also its complexity. The XML documents evolved from plain structured text representations, to documents having complex and heterogeneous structures and contents: video descriptions, mathematical formulas, time series or sequences like musical pieces or biological data. In this article we introduce a retrieval scheme designed to manage sequential data in an XML context based on two levels of approximation: on the structural localization/organization of the sequential data and on its content. To this end we merge methods developed in two different research areas: XML information retrieval and sequence similarity search. We also provide adapted index structures and operators for approximate query sequential data in a heterogeneous collection of XML documents.

1. Introduction

The XML language is a W3C standard that has rapidly been adopted and sustained by both the industry and the research community. In the recent years, we witness at an increasing volume of XML digital information produced by day-by-day or by specialized scientific activities. Nonetheless, not only the volume of the XML information is increasing but also its complexity. The XML documents evolved from plain structured text representations to documents having complex and heterogeneous structures and contents: multimedia description (MPEG7-DDL) and synchronization (SMIL), mathematical formulas (MathML), time series or sequences. We are mostly interested by the last two mentioned categories that are a ubiquitous form of data in financial, medical, scientific, musical or biological applications.

Flexible querying of scientific experimental results, patient's medical records, financial summaries, musical pieces or biological sequences published as XML documents are only a few examples of applications that involve managing sequential data in an XML context. We can thus state that there is a real need for high-performance systems and methods able to extract, index, and query heterogeneous types of sequential information from heterogeneous collections of XML documents.

In musical and biological fields special DTDs have been designed for midi files – MidiXML [1], musical scores – MusicXML [2] and biological sequential data [3] representation. In these cases the applications handle normalized sequential data and *data-centric* oriented XML documents. Data-centric documents have fairly regular structure, fine-grained data and little or no mixed content.

A heterogeneous collection of XML documents contains many un-normalized or various kinds of sequential data. This is more suited to a *document-centric* view of the database. Document-centric documents have less regular or irregular structure, larger grained data and lots of mixed content. Furthermore, the order in which sibling elements and PCDATA occurs is almost always significant.

Approximate matching in XML is closely related to the *document-centric* view and provides the possibility of querying the information acquired by a system having an incomplete or imprecise knowledge about both the structure and the content of the XML documents [4], [5], [6], [7].

One basic requirement of an XML query engine based on information retrieval concepts is to dispose of "vague predicates"/specialized similarity operators to adequately manage different data types [4] and to improve the precision of the IR system [8].

The approaches proposed in [4], [5], [6], [7] study the flexible querying on both XML structure and content (usually text), but do not specifically take into consideration sequential/time series data, nor its organization within the XML documents, which is the focus of our approach.

In our work we merge methods developed in two different research areas: XML information retrieval and sequence similarity search in order to provide adequate approximate operators for managing sequential/time series data in a heterogeneous XML environment.

In section 2 we introduce and formalize the underlying data model of our application and we identify relations between the structure of the XML documents and several common types of sequential data. In section 3 we present a hybrid indexing scheme allowing the implementation of semistructured and sequential searching operators. In section 4 we devise an approximate searching scheme for ranking the results by taking into account similarities between both the structural location and the content of the sequential data with the user requests. In Section 5 we conduct preliminary experiments dedicated to midi files retrieval embedded in heterogeneous XML databases. Finally, in Section 6 we summarize our conclusions and show some futures work perspectives.

2. Data Model

An XML document can be represented by an ordered tree whose nodes contain heterogeneous pieces of information (TEXT or PCDATA such as (parts of) sequences or time series). Each XML element may be related to attributes "name-value" fields, and each attribute value may contain (a part of a) sequential data.

Consider for example a phone number (a whole sequence) or a musical note (a sequence symbol) either as the content of an XML element node or as an XML attribute value.

To describe a sequence embedded in a heterogeneous XML environment we concurrently use its structural location in the collection – i.e. the set of XML contexts associated with the sequence symbols – and its content – i.e. the sequence symbols values.

2.1 XML Context

A Document Object Model (DOM) is an algorithmically structure that echoes the document organization in a graph, or tree. For an XML document, the DOM scheme is the tree of XML elements (as nodes) (Fig. 1 shows an example).

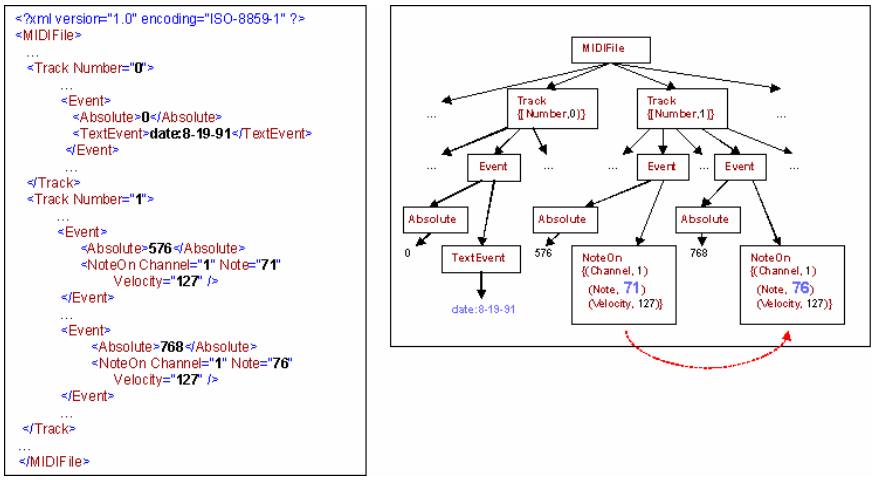

Fig. 1. An excerpt of an XML MIDI file [1] with its associated DOM tree.

According to the tree structure, every node n of the DOM tree inherits a path $p(n)$ composed with the nodes that link the root to the node n. More precisely, $p(n)$ is an ordered sequence of nodes $p(n)=n_0n_1..n_i..n_dn$, where n_0 is the root node and $d+2$ the length of the sequence. An unordered set of <*attribute, value*> pairs $A(n_i)=\{(a_j, v_j)\}$ may be attached to each node n_i of the ordered sequence, so that $p(n)$ can be represented as follows:

$$p(n)=<n_0, A(n_0)> <n_1, A(n_1)> \ldots <n_n, A(n_n)> \quad (1)$$

A node n in the DOM tree can be decomposed into structured and unstructured sub-elements. Moreover, an unstructured sub-element *(USE)* or an attribute value v_j may be decomposed into tokens t_i (or words, symbols, etc…). Each token t_i is related to an XML context $p(n)$ that characterizes its occurrence within the document.

In the example of Fig. 1:

p("date: 8-19-91") = <MIDIFile, {}> <Track, {(Number, 0)}> <Event, {}> <TextEvent, {}> <PCDATA, {value="date: 8-19-91"}>

2.2 Sequential Data

The XPath 1.0 Recommendation [9] defines the term *document order* as the order in which the first character of the XML representation of each node occurs in the XML representation of the document after the expansion of general entities, except for namespace and attribute nodes whose document order is application-dependent.

The XML document structure and the *document order* encode useful and potentially (semantically) rich information about the sequential organization of the data. We describe and formalize hereinafter our approach in exploiting this kind of information for XML sequence extraction and representation.

Sequence Definition. Formally, a sequence $S=s_0s_1..s_i..s_m$, is defined relatively to a collection of XML documents as an ordered and finite non-empty set of symbols $\{s_i\}$ selected from an alphabet Ω. An alphabet symbol s_i may be represented by:

- $v_j \in A(n_i)$ an attribute value,
- $t_i \in v_j \in A(n_i)$ a token composing an attribute value,
- USE_i an unstructured sub-element[1] of one of the XML nodes n,
- $t_i \in USE_i$ a token of the unstructured sub-elements USE_i of the XML nodes.

In the XML context from the previous example, a symbol value may indicate the *"0"* value of the *Number* attribute – for an attribute value type – or the *TextEvent* content "*date: 8-19-91"* for an *USE* type.

A sequence symbol s_i is linked with one of the unique indexed reference locators $rl_0rl_1..rl_i..rl_n$ of the XML collection set. A reference locator rl_i (defined in Section 3.1) points to a unique position in the collection of XML documents and includes a reference to the symbol's XML context $p(n)$.

Two symbols associated to the same XML context $p(n)$ could be associated: one being considered to be an order key o_i (e.g. a timestamp), the other a sequence symbol s_i.

Sequence Structural Types. The above sequence definition allows representing sequences of symbols s_i associated with any arbitrary XML contexts from the collection. From a more practical point of view, several particular structural types of sequences frequently occur and prove to be of interest:

- *node level sequence*: the whole sequence is contained in a node of the DOM tree – as an *USE* or an attribute value –, usually a leaf (content-based information retrieval). This sequence representation is widely used in bioinformatics [3],
- *document level sequence*: composed by the symbols associated to the approximate matched XML contexts of a single XML document – i.e. this includes perfect paths matches (e.g. see the link between the two nodes of the DOM tree from Fig. 1.), sibling nodes and nodes having a k-level common ancestor. This representation is imposed by the DTD's used in the musical field [1], [2],

[1] In the case of a node having a mixed content, only the unstructured content is considered.

- *collection level sequence*: composed by the symbols associated to the approximate matched XML contexts by crossing the physical boundaries of the XML documents. This sequence type may be of interest when searching sequences of information spread among several documents and that are not entirely dependent of the *document order* – e.g. time stamped information selected by the social security ID number from medical records databases.

One of the main advantages of the first two sequence types is the possibility of maintaining the *document order* by default when two ordering keys o_i and o_j, with $i \neq j$, have identical values or the sequence order relation $\leq$ is either:

- unspecified (no valid ordering key o_i has been extracted and associated with the sequence symbols s_i), or
- a partial order (indeterminate for certain cases, like the relationship between the temporal information: durations, dateTime, etc. as defined in XML Schema Part 2: Datatypes Recomandation [10].

For sequences constructed with symbols extracted from different XML documents, the *document order* could be locally applied within each XML document, but no global order of the symbols in the sequences can be inferred without the use of external information (i.e. user provided order keys) and/or heuristics. A simple example will be the use of the document creation date for ordering the documents. We may also consider using the timestamp information associated with the symbols closest ancestors in the XML trees. In our scheme we make no assumptions about the global order of the symbols extracted from different XML documents.

Thus, we propose a model in which the symbols are organized in sequences by taking into account: (1) similarities between their structural positions in the XML document trees (i.e. using *p(n)*), (2) type compatibility between their values (numbers, dates or strings) and (3) an order relation $\leq$. A symbol may occur in more than one sequence and a sequence may contain symbols with identical values.

Sequence Extraction. The users' interests in a heterogeneous XML environment can be highly diversified. Some could search the chorus of a musical piece or, in another case, similarities between the blood pressures curves of several patients' medical records. In these conditions we will probably fail to index all the different sequences that could match the users' subjective and time evolving interests.

We assume that the users or the system administrators detain at least an imprecise, incomplete or fuzzy knowledge of the particular underlying organization of the sequential data in which they are interested in. This assumption makes credible the fact that they will be able to supervise the sequential data extraction process conforming to their specific needs.

The Sequence Extraction Process. The sequence extraction process is based on a construction operator *makeSeq* that receives three arguments:

- an XML context *p(n)* with the type and position of the requested symbol *s* and (optionally) of the order key *o*,

- the minimum accepted threshold for the match of the symbols location with the provided XML context (see Section 4.1) in order to be included in the current sequence, and
- the sequence expected type: node, document (default value), or collection.

During the extraction process, the compatibility constraints expressed on the symbols values are mandatory while the structural location of the symbols and their order relation are treated as approximate. Therefore applying the `makeSeq` operator to the input XML data – i.e. to the collection of XML contexts – result in a set of symbols *s* that are associated to XML contexts similar with the one provided by the user.

In this phase of the process, the sequences are built as indicated by the sequence expected type parameter and are eventually ordered by using the symbols order keys o_i . In the case of a mixture of symbols associated or not to an order key within the same sequence, the symbols without an order key are discarded.

3. Indexing Scheme

We propose a hybrid index model (Fig. 2) designed to merge both types of data: semistructured and sequential data.

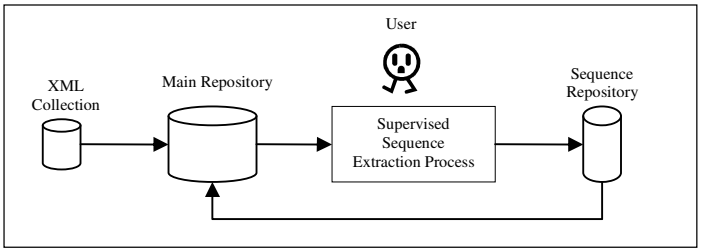

Fig. 2. The index model and the sequence extraction process

3.1 Main Repository

The main repository uses the inverted lists as basic structures. For this model, the entries of the inverted lists are of three kinds:

- structural entries, i.e. nodes *n* of the DOM tree,
- tokens of the unstructured sub-elements of the DOM tree nodes $t_i \in USE_i \in n$,
- sequential entries, i.e. unique sequence IDs. – *USIDs*.

Each entry has associated a list of reference locators rl_i pointing to a unique position in the collection of XML documents. A reference locator rl_i has attached three pieces of information:

- a link to the URI of the document,
- a link toward the XML context *p(n)*, and
- an index specifying the location of the token within the DOM node *n*.

For the structural entries (i.e a node n of the DOM tree), only the link to the URI of the document and a link to the XML context $p(n)$ are required.

The inverted list resulting from the indexing process is encoded using binary randomized search trees, namely TREAPS as defined in [11], associated to hashtable structures. TREAP structures can balance the search tree according to the asked frequency of items. This provides an access speed-up over the use of regular hashtables [12]. The inverted lists are implemented as disk-resident index structures.

3.2 Sequence Repository

A supplementary index structure is used to optimize the management of sequential data. A well known and efficient data structure for indexing and searching sequential data in text processing [13] and bioinformatics applications [14] [15] is the generalized suffix tree (GST). In a nutshell, the suffix tree is an indexing structure for all the suffixes of a string and it can be constructed in linear time and linear space [16]. A generalized suffix tree is a suffix tree for representing all the suffixes of a set of strings [17]. In our implementation we use the Ukkonen's on-line construction algorithm [18] for building a GST.

For each extracted sequence S_i, all its symbols values s_i are indexed in the generalized suffix tree structure with respect to the sequence order relation $\leq$.

The same sequence of indexed values S_i can have associated multiple sets of reference locators $\{rl_i\}_0\{rl_i\}_1...\{rl_i\}_n$ as it may occurs in different locations in the collection of XML documents. We consider a sequence to occur in two different locations in the XML collection if: (1) all its symbols values s_i are equal with the ones of an already indexed sequence and (2) at least one of their symbols associated reference locators rl_i are not matching. Otherwise, the sequence is considered to be a true duplicate and thus, discarded.

Consequently, for each indexed sequence we receive a unique sequence id *USID*, from the generalized suffix tree. The *USID* is further used as an entry key in the inverted lists to link the multiple sets of reference locators to the referred sequence. The sets of reference locators have all the same cardinality $|S_i|$ and are stored contiguously in the inverted list. The order of the reference locators from each set respects the sequence order relation $\leq$.

4. Searching Scheme

We introduce a searching scheme designed to manage unstructured sequential/time series data in an XML context based on two levels of approximation: on the structural localization/organization of the sequential data and on its content.

4.1 Structure Approximate Matching

As the user cannot be aware of the complete DOM structure of the database due to its heterogeneity, efficient searching should involved exact and approximate search

mechanisms. The format of the indexed documents being XML, it is natural to consider that the structural query itself complies, at least partially, with the XML standard. If so, the structural searching mechanism can be handled through approximate tree matching algorithms that try to match the query DOM tree to the DOM trees for the corresponding indexed documents.

Tree matching algorithms exist based on editing distance and mappings – see [19] and [20] for instance. The complexity of the matching of two trees T_1 and T_2 is at least for [19] $O(|T_1|.|T_2|)$, where $|T_i|$ is the number of nodes of tree T_i. The complexity is much higher for common subtree search [20].

This kind of tree matching is not suited to the task we intend to perform. First of all, the matching complexity is too high considering the size of documents and data bases we wand to handle. Secondly a sequence may involve paths of more than one DOM tree of the collection. Therefore we have chosen to perform an approximate search based on the matching of *p(n)* sub-structures of indexed DOM trees [12]. In other words, we are rather dealing with approximate root-leaf or root-node path alignment rather than complete tree matching algorithm.

In particular, a sequence S is defined as an ordered non-empty set of finite symbols each of them having attached a *p(n)* sub-structure of the XML collection T_i^D DOM trees. The ordered set of *p(n)* sub-structures of the DOM trees represents the structural part of the sequence, while the symbol values its content. Thus, approximate matching the structural part of an indexed sequence is based on T_i^D approximate *p(n)* path alignments.

Approximate Matching of *p(n)* Sub-structures. For that purpose, we evaluate the similarity between a root-node path p^R expressing an elementary structural query and T_i^D the DOM tree(s) corresponding to an indexed document/or sequence S as follows:

$$\delta(p^R, T_i^D) = \min_i \ \delta_L(p^R, p_i^D) \tag{2}$$

where δ_L is a Levenshtein type distance [26] and $\{p_i^D\}$ the set of root-node paths for document DOM tree or sequence S.

The complexity of such an algorithm – see [22] for justification – is :

$$O(length(p^R).depth(T^D).|\{p_i^D\}|) \tag{3}$$

where $|\{p_i^D\}|$, stands for the cardinality of the set $\{p_i^D\}$. For documents or sequences associated with DOM tree(s) having a small depth this complexity is perfectly tractable, even when the number of documents is high as far as the number of leaves is reasonable. Nevertheless, for most of day-to-day documents, equivalent DOM trees will exhibit a very low depth compared to their rather huge width.

Path Similarity Computation $\delta(p^R, T^D)$. Let p^R be the path for the structural request R and $\{p_i^D\}$ the set of root-leave paths of the DOM tree(s) associated to an index document or sequence S.

We designed an editing pseudo-distance [26] using a customised cost matrix to compute the match between a path p_i^D and the request path p^R. This scheme, also

known as modified Levenstein distance, computes a minimal sequence of elementary transformation to get from p_i^D to p^R. The elementary transformations are:

- **Substitution**: a node *n* in p_i^D is replaced by a node *n'* for a cost $C_{subst}(n, n')$. Since a node n not only stands for an XML element, but also for attributes or attributes relations, we compute $C_{subst}(n, n')$ as follows:

```
1. n.element <> n'.element : Csubst(n, n') = 2
          (full substitution)
2. n.element == n'.element :
          if n'.attributesCond(n.attributes) is true
               Csubst(n, n') = 0 (no substitution)
          else Csubst(n, n') = 1 (only attribute
substitution)
```

  where `attributesCond` stands for a condition (stated in the request) that should apply to the attributes (for example the value of attribute *Channel* for the *NoteOn* element should be equal to "*1*"),
- **Deletion**: a node *n* in p_i^D is deleted for a cost $C_{del}(n)(=2)$,
- **Insertion**: a node *n* is inserted in p_i^D for a cost $C_{ins}(n)(=2)$.

For a sequence $Seq(p_i^D, p^R)$ of operations, the global cost $GC(Seq(p_i^D, p^R))$ is computed as the sum of the costs of elementary operations.

The Wagner&Fisher Algorithm [22] computes the best $Seq(p_i^D, p^R)$ (i.e. minimizes *GC() cost)* with a complexity of :

$$O(length(p_i^D) * length(p^R)) \text{ as stated earlier.} \quad (4)$$

We postulate this scheme adequate for this application because of the limited depth of the XML DOM (mostly < 20) [23]. Let

$$GC_{min}(p_i^D, p^R) = Min_k\, GC(Seq_k(p_i^D, p^R)) \quad (5)$$

Given p_i^D and p^R, the highest possible value for *GC* can be evaluated to:

$$GC_{max}(p_i^D, p^R) = C_{subst} * (min(length(p_i^D), length(p^R)) + C_{ins} * (| length(p_i^D) - length(p^R) |) , \quad (6)$$

which can be interpreted as: all the XML elements are different, leading the *GC* to $min(length(p_i^D), length(p^R))$ substitutions and $|length(p_i^D) - length(p^R)|$ insertions: this value of *GC* can be interpreted as no possible match (or no acceptable match). Here we have $C_{subst}=2$ and $C_{ins}=2$.

We postulate a 'yet' acceptable match: a perfect match for the XML elements, and no match for the attributes:

$$GC_{mid}(p_i^D, p^R) = C_{subst} * (min(length(p_i^D), length(p^R) . \quad (7)$$

That is: the paths p_i^D and p^R have the same number of nodes, the same XML elements, but the attributes in p_i^D do not match the conditions in p^R. Here, we have $C_{subst}=1$ (attribute substitution only).

And, of course, $GC_{perfect}(p_i^D, p^R) = 0$ for a perfect match (same XML Elements, true conditions for the attributes).

The distance $d(p_i^D, p^R)$ is then computed mapping the GC_{min} value on the *[0,1]* interval (Fig. 3).

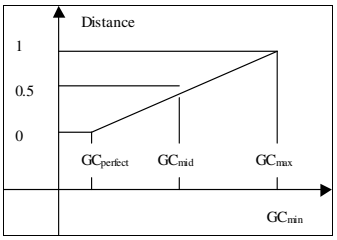

Fig. 3. Distance normalization

A structural similarity value is computed as follows:

$$Struct_{Sim}(p_i^D, p^R) = 1 - d(p_i^D, p^R) . \quad (8)$$

For a whole document *D* or sequence *S*, and a structural requests *R*:

$$Struct_{Matching}(D|S, R) = \max_i (Struct_{Sim}(p_i^D, p^R)) \quad (9)$$

Thus, the documents/sequences which lead to the higher structural matching value are the best document/sequences matching candidate for a request p^R. Note that the matching value belongs to [0,1], ranging from 0 (not acceptable – no matching), 0.5 (yet acceptable) to 1 (perfect matching).

4.2 Sequence Approximate Matching

We presume that a user has an imprecise, incomplete, or approximate knowledge of the sequential content extracted from the collection of XML documents. As a direct consequence, a sequential query S_q will represent only a short, probably inaccurate, fragment of an indexed sequence. These types of queries are usually met in the query by humming systems [24] or in matching biological data [14],[15], [17]. Under these conditions, an exact searching scheme for sequence retrieval will fail to responds to the user information needs. Thus, we choose to use a retrieval scheme that allows to approximate match a sequential query S_q with any subsequence S_i^j of the indexed data.

The sequence similarity is based on a distance δ obtained by applying a dynamic programming technique: an editing Levenshtein distance [26] or Dynamic Time Warping (DTW) distance[2] [25], [26]. The both distances have the same computational complexities $O(|S_i||S_q|)$ and allow the approximate matching of two sequences or time series of different lengths.

We want to retrieve all the similar subsequences S_i^j from the database with a user query S_q, having the distance δ less than a specified threshold ε - *the P-against-all problem* [17]. The sequential scan complexity for achieving this goal is expressed as:

[2] DTW is a pseudo-distance as it is not respecting the triangular inequality, see [28] for demonstration.

$$O\left(m\overline{|S_i|}^2 \cdot |S_q| \right), \tag{10}$$

where m is the number of data sequences whose average length is $\overline{|S_i|}$. The use of a hybrid method based on a suffix tree as an index structure and a dynamic programming method can reduce the problem complexity and efficiently retrieve similar subsequences [14],[17], [26]. The performance gain of the method comes from (1) the branch-pruning method that reduces the research space using the threshold value ε and (2) the suffixes with common prefixes that share the cumulative distance tables during the index traversal. The time complexity of this approach is:

$$O\left(\frac{m\overline{|S_i|}^2 |S_q|}{R_d R_p} \right), \tag{11}$$

where R_d *(≥ 1)* is the reduction factor saved by sharing the cumulative distance tables, and the R_p *(≥ 1)* is the reduction factor gained from the branch-pruning. In the worst case where there is no common subsequence and the branch-pruning cannot help, both values of R_d and R_p are 1, and therefore the complexity becomes the same as that of the sequential scan [26].

The similarity between a sequential query S_q and an indexed subsequence $S_i{}^j$ is given by their normalized distance δ and is computed as follows:

$$Seq_{Sim}(S_q, S_i^j) = b^{-\delta(S_q, S_i^j)}, \tag{12}$$

where $b>1$, usually e and $Seq_{Sim}(S_q, S_i{}^j) \in [0..1]$.

The value of the b parameter sets the sensitivity of the sequence similarity indicator. It specify the distribution of the possible distance values δ in the *[0..1]* interval and boost the best matches. The sequence similarity takes values between 0 - for no correspondence between sequences, and 1 - for a perfect matching.

The match of a sequential query S_q with an index sequence S_i is defined as the best match between the query and any subsequence $S_i{}^j$ of S_i :

$$Seq_{Matching}(S_q, S_i) = \underset{j}{Max}(Seq_{Sim}(S_q, S_i^j)), \tag{13}$$

It takes values between 0 – for no match, and 1– for perfect match.

4.3 The Fusion of Structure and Sequence Approximate Matching Scores

As the fusion of complementary information provided by different sources results in an information gain due to the utilization of multiple sources of information vs. a single source, the *information fusion* [27] is a thoroughly studied research area.

In our scheme we have chosen the weighted geometric mean to fusion the two levels of approximation: on the structural localization/organization of the sequential

data and on its content. The geometric mean is a way to construct an aggregate measure between different indicators that is sensitive to small values. This is appropriate for our purpose of retrieving sequences with highly similar content and being related to highly relevant structures to the user query. The weighted geometric mean is defined as:

$$\Phi(Seq_{Matching}, Struct_{Matching}) = \sqrt[\alpha_1+\alpha_2]{Seq_{Matching}^{\alpha_2} \cdot Struct_{Matching}^{\alpha_1}}, \quad (14)$$

where $\alpha 1/\alpha 2 = \lambda$ is a parameter allowing to specify the importance of the indicators to the final score.

We rewrite the above formula in order to transform the logarithmic scale of the λ parameter to a linear scale that is more suited to the user common-sense understanding:

$$\Phi(Seq_{Matching}, Struct_{Matching}) = \sqrt[1+\lambda]{Seq_{Matching} \cdot Struct_{Matching}^{\lambda}}, \quad (15)$$

where $\lambda = -\log_2(1-\gamma)$ and $\gamma \in [0..1)$.

The γ parameter is application dependent and it is used for specifying the degree of penalty applied to the final score with respect to the structural matching indicator. A $\gamma=0$ value will discard the sequence structural factor from the calculus of the overall score, while a $\gamma \to 1$ value will boost its importance at maximum. At $\gamma=0.5$ the fusion process will equally take in consideration the two indicators.

5. Experimental Results

We present some preliminary experimental results dedicated to midi files retrieval in a heterogeneous XML MIDI library.

We have implemented the approximate sequential matching operators and the fusion method based on the presented index model in the SIRIUS XML Information Retrieval Engine [12]. The prototype is entirely developed in Java and uses the Dynamic Time Warping [25], [26] to compute the similarity between sequences. The implementation of the similarity search for sequence retrieval follows the algorithms introduced in [26].

5.1 Experimental Dataset

The experimental dataset is formed by a MIDI file collection (32 Disney themes) downloaded from the public domain[3]. The files are transformed[4] in the XML format conforming to the Standard MIDI File DTD [1] version 0.9.

In a MIDI file the sequences of notes are organized in tracks (maximum 16 channels) and events (see Fig. 1).

[3] http://themes.mididb.com/anthems/
[4] http://staff.dasdeck.de/valentin/midi/

To simulate the heterogeneity of the collection and to validate the approximate structural localisation of the sequential data, we randomly generate and append a meta-structure to each standard XML midi file.

5.2 Early evaluations

We present some early experiments of the XML data indexing and sequence extraction algorithms on datasets with sizes ranging from 1MB to 15MB. The system used for experimentations disposes of a 2.4 GHz processor and 512 RAM. The xml data indexing time represents the elapsed time for the creation of the inverted lists without taking into consideration the sequential data. The sequence extraction time stands for the time spent in the process of approximate matching the XML contexts and the time spent to index the extracted sequences.

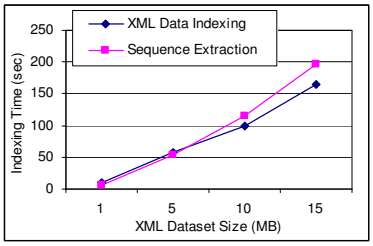

Fig. 4. XML Indexing / Sequence extraction time as the size of the index datasets.

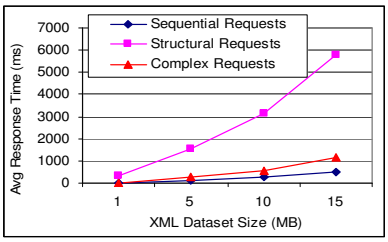

Fig. 5. Average response time for structural, sequential, and complex requests.

We can observe in Fig. 4 quasi linear indexing and extraction times with the size of the indexed datasets (i.e. the total length of the indexed sequences), which is quite encouraging. The average response time for 90 randomly generated requests are shown in Fig.5. The structural requests seems to have a polynomial behavior. The sequential queries are less sensitive to the size of the indexed data than the structural ones due to the organization of the GST index structure. A GST scales well to the dataset size as it uses the common prefixes of the index sequences to reduce the research space (see section 4.2).

This fact raises interesting perspectives for the optimization of the overall response time of the complex requests involving both the structure and the sequential content of the XML documents. We use the sequential queries as a first filter when answering to complex requests. This improves significantly the overall response time as shown in Fig. 5. Finally, the complexity of the alignment algorithms is maintained as low as possible to preserve the capability of indexing large data sets.

Considering the quality of the retrieved results, we could not evaluate it completely, as an evaluation framework for the retrieval of multimedia structured document fragments [29] is still under development at the moment of writing this article. A general opinion [8] and also our belief is that using similarity operators adapted to the document content types and to the XML structure in the retrieval process will improve the precision of the results.

6. Conclusion

We have described approximate searching mechanisms to retrieve and query sequential data from semi-structured information embedded in XML data sets. Such mechanisms are based on the alignment of root-node paths that are sub-structures of XML DOM trees. The proposed mechanisms allow to fusion structured data (<*attribute, value*> pairs) or structural organization of documents *(*<*MIDIFile*> <*Track*> <*Event*> <*NoteOn*>…) with unstructured data such as textual (free text) or sequential/time series data.

At the current authors' knowledge, there is no existent integrated method for approximate querying specific sequential data in a heterogeneous semi structured environment. Even if each part of the problem have been extensively studied and have beneficiated of strong research efforts of well established scientific communities, the fusion of the methods developed in this two research areas (sequential similarity search and XML information retrieval) was not yet deeply considered. The proposed scheme was designed in order to cover this gap and to highlight extended and useful querying capabilities for the final user.

Our main experimental contribution so far, shows that the fusion of structural and sequential search criteria could drastically improve the response time as well as the retrieval performances of the similarity search mechanisms when exploiting heterogeneous XML databases.

We intend to explore and enlarge the set of the sequential operators implemented in the system by making them aware of the temporal aspects of the data and by allowing a flexible ordering of the symbols in sequences. Both, the disk resident organization of the index structures and the parallelization of the research algorithms will be a straight forward research direction in order to validate our approach on important volumes of data.

References

1. MidiXML, Standard MIDI File DTD: MIDI XML, Version 1.0 - 13 January 2004, http://www.recordare.com/dtds/midixml.html, (2004)
2. MusicXML, MusicXML Definition, Version 1.0, January 2004, http://www.recordare.com/xml.html, (2004)
3. Robinson A., "XML's and DTD's for Biology", An XML Workshop for Biologists and Bioinformaticians, http://industry.ebi.ac.uk/~alan/XMLWorkshop/, (2000)
4. Fuhr N., Groβjohann K., XIRQL: An XML query language based on information retrieval concepts, ACM Transactions on Information Systems (TOIS), v.22 n.2, p.313-356, April 2004
5. Amer-Yahia S., Koudas N., Srivastava D., Approximate Matching in XML, Advanced Technology Seminar 5, ICDE 2003
6. Amer-Yahia S., Laks V. S. Lakshmanan, Shashank Pandit, FleXPath: Flexible Structure and Full-Text Querying for XML. SIGMOD Conference, Paris France, June 2004, pp. 83-94.
7. Carmel D., Maarek Y. S., Mandelbrod M., Mass Y. and Soffer A., Searching XML documents via XML fragments, SIGIR 2003, Toronto, Canada pp. 151-158.

8. Dorneles C. F. , Heuser C. A. , Lima A. E. N., Da Silva A., De Moura E., "Measuring similarity between collection of values", 6th ACM International Workshop on Web Information and Data Management, WIDM (2004)
9. Clark J., DeRose S., XML Path Language (XPath) Version 1.0, W3C Recommendation 16 November 1999, http://www.w3.org/TR/xpath.html, (1999)
10. Biron P., Malhotra A., XML Schema Part 2: Datatypes Second Edition, W3C Recommendation 28 October 2004, http://www.w3.org/TR/xmlschema-2/, (2004)
11. Seidel R., Aragon C. R., "Randomized Binary Search Trees", ALGORITHMICA, 16(4/5):464-497, (1996)
12. Ménier G., Marteau P.F., Information retrieval in heterogeneous XML knowledge bases, The 9th International Conference on Information Processing and Magement of Uncertainty in Knowledge-Based Systems , IEEE, 1-5 July, (2002), Annecy, France.
13. Navarro G., A Guided Tour to Approximate String Matching, ACM Computing Surveys, Vol. 33, No. 1, p. 31-88, March 2001
14. Meek C., Patel J.M., and Kasetty S., Oasis: An online and accurate technique for local-alignment searches on biological sequences. In Proc. 2003 Int. Conf. Very Large Data Bases (VLDB'03), pages 910-921, Berlin, Germany,Sept. 2003
15. Hunt E., Atkinson M.P., Irving R.W., Database indexing for large DNA and protein sequence collections, The VLDB Journal, Vol. 11 , n. 3, p.256 - 271, (November 2002)
16. McCreight E.M., "A Space-Economical Suffix Tree Construction Algorithm", Journal of the ACM, 23:262-272, (1976).
17. Gusfield D., Algorithms on strings, trees and sequences, Cambridge University Press, (1997)
18. Ukkonen E., "On-line construction of suffix-trees", ALGORITHMICA, Vol. 14, 249-260, (1995)
19. Tai, K.C, "The tree to tree correction problem", J.ACM, 26(3):422-433, (1979)
20. Wang T.L.J, Shapiro B., Shasha D., Zhang K., Currey K.M., "An algorithm for finding the largest approximately common substructures of two trees", In J. IEEE Pattern Analysis and Machine Intelligence, vol.20, N°8, August (1998)
21. Levenshtein A., "Binary Codes Capable of Correcting Deletions, Insertions and Reversals", Sov.Phy. Dohl. Vol.10, P.707-710, (1966)
22. Wagner R., Fisher M., "The String-to-String Correction Problem", Journal of the Association for Computing Machinery, Vol.12, No.1, p.168-173, (1974)
23. Mignet L.,Barbosa D.,Veltri P., The Web XML: A First Study (2003), http://citeseer.ist.psu.edu/mignet03web.html
24. Zhu Y., Shasha D., Warping indexes with envelope transforms for query by humming, Proceedings of the 2003 ACM SIGMOD, San Diego, California, p. 181 – 192.
25. Keogh E.J., "Exact Indexing of Dynamic Time Warping" , VLDB 2002: p. 406-417.
26. Park S., Chu W., Yoon J., Won J., "Similarity search of time-warped subsequences via a suffix tree", Information Systems, 28(7): 867-883, (2003)
27. Sander W.A., Information Fusion. In T.N.Dupuy, F.D.Margiotta, C.Johnson, J.Motley, and D.L.Bongard, editors, International Military and Defense Encyclopedia, Vol.3, G-L, pp.1259-1265. Brassey's, Inc., (1993).
28. Yi B., Jagadish H. V., Faloutsos C., "Efficient Retrieval of Similar Time Sequences Under Time Warping", ICDE, pp. 201-208, 1998.
29. R. van Zwol, G. Kazai, M. Lalmas, Multimedia track, INEX, April 2005 - December 2005, http://inex.is.informatik.uni-duisburg.de/2005/tracks/media/index.html