TX-Kw: An Effective Temporal XML Keyword Search

Rasha Bin-Thalab
Department of Information System
Faculty of Computers and Information
Cairo University, Egypt

Neamat El-Tazi
Department of Information System
Faculty of Computers and Information
Cairo University, Egypt

Mohamed E.El-Sharkawi
Department of Information System
Faculty of Computers and Information
Cairo University, Egypt

***Abstract*—Inspired by the great success of information retrieval (IR) style keyword search on the web, keyword search on XML has emerged recently. Existing methods cannot resolve challenges addressed by using keyword search in Temporal XML documents. We propose a way to evaluate temporal keyword search queries over Temporal XML documents. Moreover, we propose a new ranking method based on the time-aware IR ranking methods to rank temporal keyword search queries results. Extensive experiments have been conducted to show the effectiveness of our approach.**

Keywords—temporal XML; Keyword Search; ranking

I. INTRODUCTION

The success of keyword search in IR has encouraged its emergence in XML [1-3] and databases [4, 5]. Although, temporal data are used commonly in historical applications (web logs, financial, scientific, and georeferencing applications), existing XML keyword search methods are not aware of temporal expressions in keywords. Temporal keyword refers to exploiting time dimension that is embedded inside the XML documents to provide alternative search methods and user experience.

A study made by Zhang et al. [6] showed that about 13.8% of queries have explicit time predicate and 17.1% of queries implicitly contain temporal intent. An example of a query with explicit time provided is "U.S. Presidential election 2008". An implicit time query example can be "Germany FIFA World Cup", here the time is not declared but the user is referring to the World Cup event in 2006. Furthermore, database applications contain information for long time periods, like, DBLP which keeps all publications that cover the years 1954 up till now. When searching in these documents archives, a temporal dimension plays an important role.

Keyword search in XML model is used to find nodes that contain keywords and checks the interconnections among them based on their lowest common ancestors (LCA) [1]. For example, query Q1 "Michael, Adams" in Fig. 1 returns node; 0.2.0.2.

However, LCA has a lot of drawbacks since it does not give a meaningful answer in all cases, e.g., as in query Q1, node 0.2.0.2 might not be the user intention. Another drawback is that it does not consider ID/IDREF relationship between nodes, which may result in missing some relevant results. Recent approaches [4, 7, 8] preferred to model XML document as a set of interrelated objects rather than nodes. Each object is represented as a sub tree rooted by a representative node with its set of attributes. Also, ID/IDREF connections are considered in such approaches to increase relevant results.

In this paper, we integrate temporal constraints into keyword search approaches for temporal XML databases (TX-Kw). We perform semantic matching at object level rather than using traditional LCA techniques. There are three basic reasons that motivated us to use object-level in temporal keyword search over temporal XML documents. First, XML can be recognized as a set of real world objects, each of which has attributes and interacts with other objects through relationships in certain temporal intervals, for example player, team and coach entities in NBA DB as shown in Fig. 1 are considered objects in real world. Second, users aim to find a specific object information by typing a set of words and a specific time about such object. They do not aim to find if the information exists or not by retrieving the node that contains this information. Finally, temporal nature of nodes in temporal XML documents can be captured well in objects as long as their attribute values and relationships output change over time. Thus, object-level may be very helpful to give more relevant answers especially if we adapt ranking objects rather than ranking nodes by taking time dimension into account. In this case, keyword search results is either a single object which contains all keywords in the time specified or a set of interconnected objects that contain the keywords in that specified time.

Our objective in this paper is to effectively retrieve a single temporal object (STO) or a set of related temporal objects (RTO) that are the closest to user intention while considering ID/IDRef links. A brute force approach, which considers all result objects before the top-k scores, requires expensive processing time. We build several index structures for keywords and temporal objects to provide better performance. We propose an efficient algorithm based on these structures to get top-k results of temporal objects.

We summarize the contribution of this paper as follows:

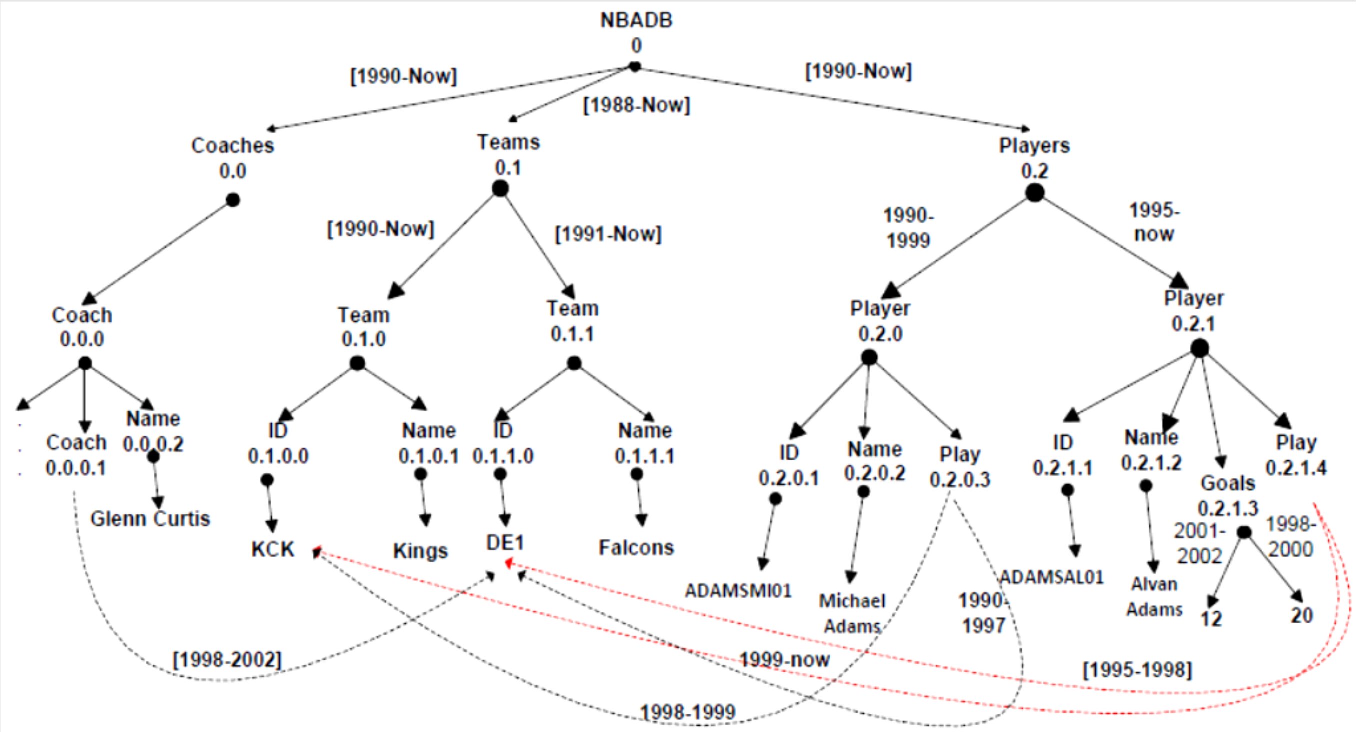


Fig.1. NBA DB portion (with Dewey IDs)

- Modeling XML into temporal objects based on their ID/IDRef attributes
- Build an efficient temporal keyword search algorithm for processing temporal keyword search queries over temporal XML documents.
- Design adaptive temporal ranking method for XML keyword search

The rest of this paper is structured as follows. Section II describes related work. Section III presents background and definition used through the paper. Section IV introduces semantic matching of objects to keywords. Section V addresses ranking functions used to rank result objects. Section VI presents structure indexes for enhancing performance. Keywords search algorithms are presented in section VII. We implement experiments to compare our algorithm to the state-of-the-art methods in Section VIII. Finally, Section IX provides concluding remarks and future works.

II. Related Work

In literature, keyword search has been studied well in XML environment [9]. We categorize these approaches into two categories; tree and graph models. Approaches for XML graph model are more related to our work. On XML tree model, XSearch [10] and SLCA [2] provided an efficient way to calculate the smallest LCA (SLCA) XML node that contains all keywords. However, successive works were proposed to improve effectiveness like VLCA [11], and efficiency ELCA [3, 12]. Next, Bao [13] proposed an IR-style approach (called XReal) which basically utilizes the statistics of underlying XML data to present a novel XML TF*IDF ranking strategy to rank the individual matches of all possible search intentions. On XML graph model, ObjectRank [4] is one of the earliest studies which designed a semantically meaningful ranking method using the authority transfer paradigm. However, ObjectRank results in only single objects and does not take a group or a cluster of objects as alternative results. On the other hand, several XML approaches were presented to implement graph model in XML keyword search result, in more effectiveness in trade of efficiency. XKeyword [8] provided an efficient keyword proximity queries for large XML graph databases. The authors adopt the concept that a keyword proximity query is a set of keywords and the results are trees of XML fragments (called Target Objects) that contain all the keywords. However, ranking of target objects is restricted to the distance between elements which leads to missing objects that are more related to the keywords, although they do not contain them. Recently, Bao [7] presented an object-level to retrieve effective results which are based on schema document. The result is either a single object or a set of interconnecting objects. In fact, the authors only considered object class but ignored object ID. Thus, they cannot discover duplicate objects and suffer the same problems as LCA-based approaches.

All the mentioned approaches did not take benefit from the temporal nature of temporal XML documents in retrieving results of temporal keyword search queries.

In information retrieval, several proposals addressed time-aware ranking of pages in www environment. They are divided into two categories: link based [14, 15], and content-based [16, 17]. In link-analysis approach, Yu [18] modified the PageRank [19] algorithm by accumulating the weights of its citations, where each citation receives a weight that exponentially decreases by its age. Berberich [14] also extended PageRank to rank documents with respect to freshness. The difference is that this work defines freshness as a linear function that will give a maximum score when the date of document or link occur within the user specified period and decrease a score linearly if it occurs outside the interval. Second type of ranking methods is based on an analysis of document content [17]. Jatowt [20] presented an approach to rank a document by its freshness and relevance. A document is ranked high if it is modified significantly and recently. Diaz and Jones [21] used timestamp from document metadata to measure the distribution of retrieved documents and create the temporal profile of a query.

To the best of our knowledge, the first study of temporal keyword search in XML has been addressed by Manica [22]. They identified temporal constraints in a keyword query and intercepted the query processing, executed by a conventional XML search engine, in order to evaluate those constraints. However, temporal ranking results were not handled.

III. Background, Notions And Definitions

In this section, we describe the concept of temporal objects (TOs), which we use in this paper. To define temporal objects in XML document, we combine the definitions of temporal object in [23, 24] and an object tree in [7] as follows.

Definition 1. *A temporal object O_t represents a real-world entity or concept, each object has an object ID, attributes and lifespan. We define a temporal object in an XML document as a subtree (object tree) annotated with lifespan. Each object is represented by <OID, att_list, lifespan, OList>*

where "*OID*" is the object identifier. "*att_list*" is the list of attributes the object has. "*lifespan*" is the life time of the object in the system. "*Olist*" contains a list of <OID, Time> pairs denoting the objects that are connected to the object with the timestamp for that connection.

A set of objects with similar characteristics (attributes) are grouped into what is called a class. An object class (called class for brevity) consists of a signature that defines the object in reality.

How to identify the temporal objects is orthogonal to this work; here, we adopt the inference rules in XSeek [26] to help identify the object trees from XML Schema. In the case of no obvious schema, any other program for extraction schema is used. As we can see from Fig. 1, there are five temporal object instances for three classes; 2 objects for Player class, 2 objects for Team class, and 1 object for Coach class. A dashed line represents the ID/IDREF edges connecting objects. Note that nodes Players, Teams and Coaches are connection nodes which connect the node Players with the player objects, (the same for Team and Coach). In XML model, a real object class is distinguished in form of a subtree due to its hierarchal inheritance.

Another important concept is introduced, connections, which is used to define relationships between temporal objects. We distinguish between two types of connections; containment and references. Two temporal objects have a containment connection if there is a containment edge annotated by timestamp connecting them (parent-child edge) denoting when this connection is active. On the other hand, reference connection is used if there is an ID/IDREF edge annotated by a certain timestamp between two temporal objects.

In this work, we apply a discrete notion of time and assuming the integers Z. The temporal expression T can refer to any time interval [b, e] where b M e. Year is used as a time granularity for simplicity.

IV. Temporal Object Matching Semantics

In context of XML keyword search, a temporal keyword query is a set of keywords attached with time (e.g., "Michael, Adams, 1995"). Usually, when a user issues his keyword search, he intends to get almost a single object that contains all the keywords he issued in the specific time domain or even a set of interrelated objects that contain all keywords and intersect at the time interval provided. These objects are more likely to have a well known relationship among them.

A. Single Temporal Object Matching Semantics

Definition 2. *Given a temporal keyword query Q_t, a temporal object O_t is defined as a Single Temporal Object (STO) found in the document if it contains all the keyword(s) as part of its attribute's value or structure tag names, and its lifespan interval intersect with Q_t time.*

One can conclude that STO plays LCA role in the temporal object oriented model. For example in Fig. 1, if we issue the query "Alvan, Adams, 2000-2005", STO returns the Player object rooted at node 0.2.1 rather than Name attribute at node 0.2.1.2 returned by traditional LCA.

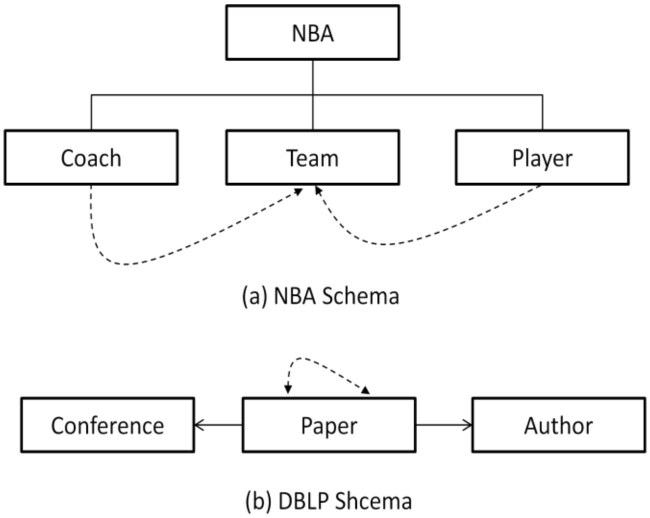
B. Related Temporal Objects Matching Semantics

Keywords in Q_t keywords can be found in different objects rather than a single one. Furthermore, the time interval in the selected objects has to intersect with Q_t interval. For example, given a query Q4:"team, Curtis,[2000-2002]" in Fig. 1, here the user wants to know the team which is coached by Curtis in the interval [2000- 2002], the "team" keyword is contained in two objects rooted at nodes 0.1.0 and 0.1.1, and "Curtis" found in object Coach rooted at node 0.0.0. If we take the LCA based on their Dewey Id, the root node NBA 0 is returned. When considering the ID/IDREF into account, we can see that there is an ID/IDREF edge between objects team (0.1.1) and coach (0.0.0) during the interval [1998-2002]. Thus the result in this case will be both objects; team (0.1.1) and coach 0.0.0 since they are connected by a reference edge.

There are three types of connections that can exist between any two objects *a* and *b*. First, *a* and *b* can be connected via a lowest common object ancestor LCOA (e.g., in DBLP, one paper is an ancestor of multiple papers by cites relationship). Second, *a* and *b* can have a common object descendant COD (e.g., in Fig. 1, team 0.1.1 is COD for both objects player (0.2.1) and coach (0.0.0)). Moreover, *a* and *b* can be connected via an n-hop connections meaning, if there are n-1 intermediate distinct objects o_1, ..,o_{n-1} such that there is a connection between each pair of adjacent objects and no objects of them are connected via ancestor-descendant relationship. For example, in DBLP, one paper may be connected with another paper through a set of intermediate paper citations although no direct connection between them.

Definition 3. *Related temporal objects result (RTO) is a tree structure composed of temporal objects having the query keywords and intersect with query temporal interval while having either ancestor/descendant or ID/IDREF relationships connection between them.*

In order to construct RTOs, we combine the use of schema structure of XML document and ID/IDRef edges. Fig. 2(a) shows schema of DBLP and NBA data sets. We can find that the set of possible objects could be connected to form RTOs.



(a) NBA Schema

(b) DBLP Shcema

Fig.2. RTOs for NBA and DBLP Schemas

V. Ranking

The challenge of capturing effective results of temporal keywords is the selection of temporal objects that have the highest ranking scores (top-k). This problem is studied very well in IR [6], [20]. Incorporating time dimension into ranking models can significantly improve result effectiveness of user intention

In this section we introduce a ranking model to increase effectiveness for temporal queries. Note that pages in IR environment are mapped into objects in XML environment.

Thus we have an object granularity rather than a page. The object contains values of attributes each of which may have temporal data (like birth-date of employees) which corresponds to content time in IR, and the object has a lifespan interval against publication time of pages.

Lifespan interval starts when the object inserted into the system and its end time is determined by the time deletion of the object or left up to now. Object lifespan also is sensitive to the application semantics, such as in DBLP, publishing year of an article is considered as the start of the article's object lifespan.

A. Ranking Model

Temporal objects as well as temporal queries contain two integral information keywords and time. Here, we design a ranking model which is based on a mixture model used for IR [25] that linearly combines keywords similarity and temporal similarity and then we map it into XML objects. Given a single temporal object O_t and a temporal query Q_t, we compute the similarity degree ρ_s of O_t to Q_t using the following formula:

$$\rho_s(Q_t, O_t) = \alpha S_k(Q_t, O_t) + (1-\alpha) S_t(Q_t, O_t) \quad (1)$$

where the mixture parameter α indicates the importance of keywords similarity $S_k(Q_t, O_{t,})$compared to temporal similarity $S_t(Q_t,O_t)$. The higher the score the more relevant the object is. Next we define each similarity computation.

B. Keyword Similarity S_k

$S_k(O_t,Q_t)$ ranks the keywords of Q_t in object O_t. We compute this ranking by also taking timestamp of keyword into account. We extend the CT-rank of Jin [20] which addressed the ranking of keywords and its validity time in web pages. However, our granularity is an object not a document. An object also has attributes which taken into account in our ranking function. Let Ktime denotes the frequency of keyword k with its timestamp t, < k, t >, in Ot. Ktotal is the total number of all keywords in object O_t. N_O is the total number of objects in the XML document. N_k is the total number of objects that contain keyword k. *score* (k,O_t) is used to compute rank of k in O_t. It is defined as follows:

$$score \quad (k, O_t) = \frac{K_{time(<k,t>,O_t)}}{K_{total}} \times \log\left(\frac{N_O}{N_k}\right) \quad (2)$$

Note that the score is based on TF/IDF [26] computation used in IR where mapping pages to objects. Since the object contains attributes, a keyword might occur more than once in the same attribute. K_{time} function is calculated in Equation 3.

$$K_{time(<k,t>,O_t)} = \sum\nolimits_{\forall a \in attr(O_{t,<k,t>})} \big(tf(a, <k, t>)\big) \quad (3)$$

tf (a, k) is the number of < k, t > pairs in the specified attribute. Finally, a contribution value cb is another factor to be considered in local score of an object, i.e. how many keywords there are in an object. For the whole keywords in the query Q_t, object O_t is ranked according to query keywords as:

$$S_k(Q_t, O_t) = \sum\nolimits_{k \in Q_t \wedge k \in O_t} (score(k, O_t)) \times cb \quad (4)$$

We add the contribution factor cb to measure the whole keywords in Q_t that are contained in O_t. We compute cb as total number of query keywords in object O_t divided by the total number of Q_t.

C. Temporal Similarity S_t

Two temporal expressions can affect the ranking of an object; lifespan (e.g. publication time of an article) and links associated with an object (in and out edges). To compute temporal similarity of temporal object O_t for XML document we adapted Berberich' T-rank [27] approach by changing the granularity to be objects in an XML document. Given a single temporal expression q_t in query Q, D is the document to be ranked, Berberich [27] equation defined in T-rank, as follows:

$$P(q_t|D) = \frac{1}{|D|} \sum_{T \in D} P(q_t|T)) \qquad (5)$$

As shown in "(5)", the probability of generating the query temporal expression qt from document D is an average of the probability of generating of time $P(qt|T)$ divided by the number of intervals in document $|D|$. The probability $P(qt|T)$ of generating a time interval qt given a partition time T of a document can be defined in two ways; either by ignoring uncertainty or taking uncertainty into account. This will be illustrated later.

Now we adapt the equation above into temporal XML objects. Each web document D is mapped into a temporal object Ot in XML. Rather than considering temporal expressions of document D, we use temporal expressions which are labeled on edges of an XML graph. In turn, these edges are divided into content and reference edges as defined previously in Section III. Such temporal edges affect the whole similarity degree of the object O_t. We compute the temporal similarity S_t of O_t given query temporal expressions Q_t as shown below in (6).

$$S_t(Q_t, O_t) = \sum_{q_t \in Q_t} S_t'(q_t|O_t) \qquad (6)$$

Consequently, S_t is computed as follow:

$$S_t'(q_t|O_t) = \left(\frac{1}{2}\right) \times (\frac{1}{|ec_t|} \sum_{o_t \in ec_t} \left(score_t(q_t|o_t)\right) + \frac{1}{|er_t|} \sum_{o_t \in er_t} \left(score_t(q_t|o_t)\right) \qquad (7)$$

Where $|e_{ct}|$ is the total number of temporal intervals on the containment edges of Ot, $|e_{rt}|$ is the total number of temporal intervals on the reference edges of O_t. To normalize temporal similarity into range [0-1], the score is divided by 2. $score_t$ is used to denote the probability of generated query q_t and object temporal T intervals which will be defined later. There are two ways to compute $score_t(q_t|o_t)$: uncertainty-ignore and uncertainty-aware as defined by Berberich [27].

Uncertainty-ignore mean that the time similarity is one if T and qt are exactly the same. As shown below.

$$score_t(q_t|o_t) = \begin{cases} 1, if\ q_t = o_t \\ 0, otherwise \end{cases} \qquad (8)$$

However, temporal expressions can refer to the same time interval even they are not exactly equal, i.e. the relevance of a temporal object may change over time. For this purpose, uncertainty-aware may give approximate similarity. An object with its time partition is closer to qt will receive a higher probability than an object with time far from qt. On the other hand, when uncertainty is considered, $score_t(q_t, o_t)$ is defined in "9" as follows:

$$score_t(q_t|o_t) = \begin{cases} \frac{|q_t \cap o_t|}{|q_t| \times |o_t|}, if\ q_t \cap o_t \neq \phi \\ \epsilon, \qquad otherwise \end{cases} \qquad (9)$$

We use ϵ as a very small value to overcome zero issue denoting no common time between the query and the object. The distance of a given interval t, denoted as $|t|$, is computed based on the values of begin and end interval as: $|t| = t.e - t.b$+1 where $t.b$ and $t.e$ represent the start and end of interval respectively. Intuitively, this function gives a similarity that decreases proportional to the difference between q_t interval and time of object o_t. An object o_t with its time closer to q_t will receive a higher similarity than an object with its time interval far from qt.

In summary, $S_k(k,O_t)$ gives the contents similarity of keyword with its time in object O_t and $S_t(O_t,Q_t)$ measures the active intervals of object O_t during the specified times in query Q_t.

D. Ranking multiple Objects

A set of RTOs are cooperated to contain all temporal query keywords with specified time while no single object contains all the keywords. Thus we need to compute the whole ranking of the participating objects. Given a query "w_1, ..,w_m, t" where w represents the keywords and t represent the time predicate, and its corresponding set of interconnected temporal objects RTO (O_1, ..,O_m). The related objects can be calculated as follows:

$$RTO(o_1, .., o_m) = \sum_{O_t}^{m} (\rho_s(Q_t, O_t)) \qquad (10)$$

Where $\rho_s(Q_t,O_t)$ is defined in "(1)" and m is the number of objects.

E. Optimizing temporal ranking into XML environment

The extension of time-aware ranking approach in previous section maps only the flat structure in web pages to XML objects. One advantage of XML documents is its hierarchical nature. In temporal XML model, one object may contain other objects. As a result the object rank is affected by the ranking of its sub objects. To capture transferring of ranks we propose a recursive formula ρ_n to compute XML similarity between a temporal XML object and a temporal keyword search query. Hierarchical structure between objects is captured in "(11)" by distinguishing between two cases. The first (base case) computes the similarity if the object is a single object O, otherwise (recursive case), it recursively computes the similarities for its nested objects, based on the similarity (ρ_n) value of each child chd of O_t.

$$\rho_n(Q_t, O_t) = \begin{cases} \rho_s(Q_t, O_t) & , O_t is\ a\ single\ object \\ \sum_{c \in chd(O_t)} \rho_n(Q_t, c) & , O_t is\ nested\ object \end{cases} \qquad (11)$$

VI. INDEX STRUCTURE

We pre-process the temporal XML document D by building a separate structure for each distinct keyword w. This is similar to a regular inverted index.

We precompute single keyword rank in object and combine them during run time.

A. Motivation

Many algorithms were proposed to evaluate XML keyword queries efficiently; SLCA [2] ELCA [3]. The basic idea is to utilize the document order of the nodes in the inverted lists to optimize the semantic pruning. Specifically, nodes in the XML tree are identified by Dewey id and an efficient eager stack algorithm is built. A graph model is used since the tree model does not effectively answer keywords queries. A graph model evaluates queries by finding the minimum connection tree MCT [28].

The XML document is parsed to build two structure indexes; keyword and object lists. The first is used to retrieve the objects whose attributes contain the keywords and the second to track the objects relationships. The indexes are explained in the next subsections.

B. Keyword List

Since a temporal XML database is modeled as a set of interrelated objects with timestamps associated on their containment edges and references (ID/IDREF), the inverted list is more complex than that in the traditional one. Keyword index is composed of tuple $< obj\_id, K_{list}, TF_k >$ where *obj_id* is the object identity (usually given in the data set or using dewey id generated automatically by the system) which contains the keyword. However, to efficiently join objects later, we map each *obj_id* signature with an ordered number. K_{list} is a list of attributes that contain the keyword which is composed of $< attr\_name, time >$ pairs by specifying the attribute name and the time validity interval of the keyword. TF_k is the term frequency of the keyword k in the object which is computed during the construction of index as shown in the first part of Equation 2. For example, the term frequency of keyword 'Adams' in the object rooted at node 0.2.0 is computed as: 1/7 = 0.14, where 7 is the total number of keywords in the object. It is more efficient to compute keyword scores during the preprocessing time.

A B+ tree is built based on these objects ids and their time intervals to efficiently retrieve all the objects that contain the query keywords during a specific time.

Complexity size of keyword list is $O(K \times p)$ where K is the total number of keywords and p is the size of a tuple.

C. Object List

Object list is composed of tuple $< o\_id, Olist >$, where *Olist* is a list of tuple $< I, L_{Obj} >$ where I is the interval validity for the connection and LObj is a list of connected objects ids. This list is built during the XML parsing when a reference attribute is encountered. Although this connection is placed in one hop, it is possible to connect objects via other objects. At end of traversing the XML document, a threshold τ is given to determine the number of hops to compute connected objects. _ Value depends on the application and user intention. For example, for NBA database, τ value will take value 2 to detect any relationships between coaches and players. Additionally, user may (may not) wish to see all the connected objects even these objects are far away.

Any required increase of τ value more than one is performed after finishing document traversal using breadth first search algorithm BFS. For example, player object at node 0.2.1 connects with team 0.1.0 at [1995-1998] by one hop, and with 0.0.0 at [1998, 1998] by two hops.

Temporal ranking of objects is computed on the run time during query processing stage.

Complexity of computing the object list is $O(N \times N)$ in the worst case, where N is the total number of objects in The document. Regarding the size of our indexes, we store only ids of objects which reduce storage size significantly. Also, storage entries are built to keep track the objects positions in the XML document.

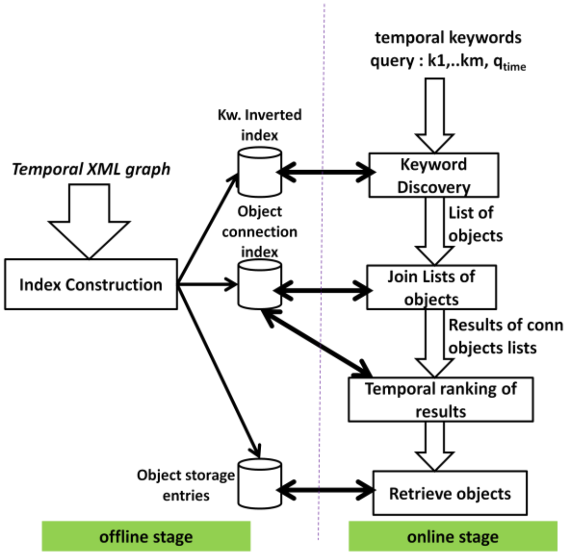


Fig.3. TX-Kw architecture

VII. Temporal Keyword Search Algorithm

Fig. 3 illustrates TX-Kw architecture which consists of two stages: offline and online stages. In the offline stage, indexes are built for keyword and object lists as explained in VI. Query processing is performed in the online stage. When the user poses a temporal query, the query processor parses the query to extract the keywords and time intervals included. For each keyword, the inverted index is checked to extract objects that contain this keyword at the temporal interval specified.

For each list objects, a connection is performed between these objects. Resulting objects are ranked according to the Equation 10.

Algorithm 1 presents keyword searching and result ranking algorithm. *qt* is the time interval presented in the query. LL[m] contains a list of objects that contain each keyword in the query. Recall that each object contains the object id and TF score of keyword within its object. When the query is issued, Algorithm 1 traverses lists in LL[m] to extract all possible connected objects. First, we initialize the *Cont_Table* used to store the contribution percentage of an object to the query keywords, *RSTO* and *RRTO* used to store single and related temporal objects respectively, and *HT* used to store the temporal ranking of objects. Then the Algorithm chooses the smallest list in LL as the starting list to join with other lists.

Two main steps the algorithm performs: *Compute_STO* and *Compute_RTO*. *Compute_STO* is called (line 4) which is defined in Algorithm 2 to compute single objects that contain all keywords within time query and stored in *RSTO* with its ranking score. Second, *Compute_RTO* is called (line 5) to extract related objects. Algorithm 3 returns *RRTO* which is a list of connected objects which cooperate to contain all query keywords. Finally we get top-k results by eliminating any repeated objects that exist in the lists and sorting R_{STO} and R_{RTO} in descending order. Objects are retrieved from storage entry file to output to the user. Function *Compute_CB* details is omitted due to its simplicity.

Algorithm 3 shows the process to recursively join lists. The input is the *qt* as query time, and starting list *Ls*, *c* is the index of next list in *LL*[*m*] to be joined with *Ls*. The algorithm pushes each input object o_1 into a path stack which stores the connected objects, o_1 is joined with *LL*[*c*] using *match_obj* function, the returned list is a set of objects that are connected to o_1. The result is used in the next iteration to join with next list. Finally, when we finish traverse the lists for a given object, path stack computes their score by calling *CompRankL* and appends its score inside *RRTO*. *CompRankL* function computes the total rank of each object in a list based on two parts; keyword and time as explained in Equation 7, note that the keyword score is already computed while traversing the connected objects. To avoid re-computing temporal similarity, we use a hash table *HT* to store all computed scores of objects.

```
Algorithm 1 Temporal Kw Search (TX-Kw)
Input: qt:Interval time, LL[m]: object Lists, ObjIdx: object Index
Output: Ranked object(s) list: $R_{STO}$ and $R_{RTO}$
 1: Initialize HT,$R_{STO}$,$R_{RTO}$, Cont_T able
 2: Sort LL[m]
 3: L1 ← LL[1]
 4: $R_{STO}$ ← Compute_STO(L1,LL)
 5: $R_{RTO}$ ← Compute_RTO(L1,1, qt)
 6: Sort $R_{STO}$ in descending order
 7: Sort $R_{RTO}$ in descending order
 8: Output $R_{STO}$ and $R_{RTO}$
```

```
Algorithm 2 Compute_STO
Input: Ls: smallest list, LL[m]: object lists, qt: interval
Output: List of STO object $R_{STO}$
 1: for each object $o_1$ in Ls do
 2:     $cont_o$ ← Compute_CB($o_1$)
 3:     if $cont_o$ ==1 then /* $o_1$ is STO */
 4:         $score_{o1}$ ← $STO_{rank}$ (o1)
 5:         Add < $o_1$, $score_{o1}$ > to $R_{STO}$
 6:         Delete $o_1$ from LL[$m$]
 7:     end if
 8: end for
 9: function $STO_{rank}$($o_1$ :object)
10:     score ← φ
11:     for each list L1 in LL do
12:         $o_2$ ← find $o_1$.id in L1
13:         score = score + $CompRank_o$($o_2$)
14:     end for
15:     return score
16: end function
17: function $CompRank_o$ (o1object $o_1$)
18:     $S_k$ ← $o_1$.TF * log(N/$N_k$)
19:     if $o_1$ is not in HT then
20:         $S_t$ ← Compute $S_t$ using ObjIdx (Eq. 7)
21:         add $o_1$ to HT
22:     else $S_t$ ← HT[$o_1$]
23:     end if
24:     $Score_{o1}$ ← $S_t$ + $S_k$
25:     return $Score_{o1}$
26: end function
```

```
Algorithm 3 Compute_RTO
Input: L1: list, c: counter, LL[m]: object lists, qt: Interval
Output: $R_{RTO}$ :List of connected objects:
 1: for each object o1 in L1 do
 2:     path ← push o1
 3:     L2 ← match_obj(o1, LL[c], qt)
 4:     if L2 = φ then
 5:         path pop
 6:         continue
 7:     end if
 8:     increment c by 1
 9:     if $c \le m$ then
10:         Compute_RTO(L2, c)
11:         decrement c by 1
12:         path pop
13:     else
14:         for each object v in L2 do
15:             path ← push v
16:             sc ← $CompRank_L$ (path)
17:             Add <sc, path> to $R_{RTO}$
18:             path pop
19:         end for
20:         decrement c by 1
21:         path pop
22:     end if
23: end for
24: function match_obj(object: o, list : $L_{in}$, interval: qt)
25:     Retrieve Lo connected to o in qt from objIdx
26:     $L_{des}$ ← merge join Lo and $L_{in}$
27:     return $L_{des}$
28: end function
29: function $CompRank_L$ (L_obj)
30:     score ← φ
31:     for each object $o_1$ in L_obj do
32:         if o1 is not in HT then
33:             $S_t$ ← Compute $S_t$ using ObjIdx (Eq. 7)
34:             add $o_1$ to HT
35:         else $S_t$ ← HT[$o_1$]
36:         end if
```

```
            $cb \leftarrow$ Compute_CB($o_1$)
            $S_k \leftarrow o_1$.TF * log(N/N$_k$) × cb
            $s \leftarrow S_k + S_t$ (Eq. 1)
            score $\leftarrow$ score + $s$
        end for
        return score
end function
```

We use *match_obj* function to merge join lists to find the connected objects of an input object.

Complexity of Algorithm 1 is based on the complexity of Algorithms 2 and 3. Algorithm 2 traverse smallest list Ls to discover any possible STO objects and its complexity is $O(|Ls| \times \log(|LL|))$. In Algorithm 3, the complexity is $O(|Ls| \times m \log |Lx|)$ in the worst case where m is the number of keywords and L_x is the length of maximum list in LL. The overall complexity of Algorithm 1 is $O((|L_s|)\times(\log(|LL|)+ m \log |L_x|))$

VIII. Experimental Evaluation

In this section, we present different performed experiments to evaluate our approach. We compared our approach to two state-of- art approaches: SLCA [2] as an example of XML tree model and ISO_IRO [7] as an example of XML graph model.

We analyzed our results according to the three metrics that are used to analyze the results of any keyword search experiments; effectiveness, efficiency, and scalability.

A. Setup

We use two real data sets: DBLP [29] and NBA [30]. Table I shows the statistics of such data sets. DBLP contains the major conferences up to year 2002. We build the temporal XML indexes as follows: each conference is considered as a separate object and cite attribute is used to track the relationship between objects. NBA contains all information about players, coaches and teams in USA basketball starting from 1946 until 2008. It is a set of tables in a relational database converted into a single XML document where foreign keys are converted into id/idref attributes. Temporal intervals queries for conventional keyword search are executed for each instant in the interval. Here, we list sample queries for both data sets and its purposes:

User intention: List all coaches train player James Posey in [2005-2008]

TX-Kw: Coach James Posey [2005-2007]

Conventional Kw: Coach James Posey 2005, 2006, 2007

User intention: List all articles written by Elmasri in interval 1990-1993

TX-Kw: article Elmasri [1990-1993]

Conventional Kw: article Elmasri 1990, 1991, 1992, 1993

B. Efficiency

We use the query processing time as the main performance metric. For DBLP we test 7 queries, the first three are non-temporal queries and the rest are attached with one time instant.

The result of the log-scaled run time of tested approaches is shown in Fig. 4(a). We can see that our approach has better performance for temporal queries. However, ISO_IRO is slightly more efficient than TX-Kw for the first non temporal queries (Q1-Q3). The reason is that these queries retrieve the whole objects in the three algorithms but TX-Kw has an overhead computing for temporal ranking. Whereas SLCA is the more efficient keyword search index in XML tree models, its performance degrades in the XML graph model. This is because many keywords may be placed in one object which is easy for TX-Kw and ISO_IRO to distinguish them in the join process. On the other hand, SLCA has to search for each list of nodes that contain the keywords. Such number of nodes may be greater than the number of objects returned by the other algorithms. The worst case of SLCA is clear for temporal queries where more nodes are retrieved than that in TX-Kw.

TABLE I. Data sets statistics

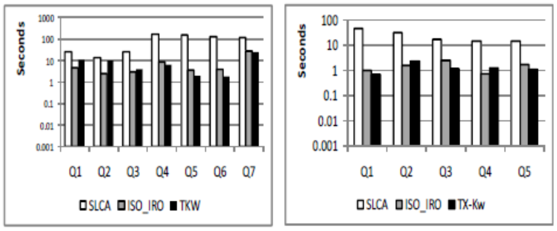
Data set	#nodes	#Kw	#object	Size	Idx size
DBLP	3329043	656387	328858	131MB	781MB
NBA	313251	3742	2517	6MB	23MB

TABLE II. Query keyword for efficiecny in DBLP

Qid	DBLP
Q1	Agrawal Databases
Q2	Ling tok wang
Q3	VLDB Agrawal Abbadi databases
Q4	Concurrency Control Algorithms Distributed Databases 1987
Q5	Ling Tok Wang 1993
Q6	XML Index 2002
Q7	book, java, 2001

TABLE III. Query keyword for efficiency in NBA

Qid	NBA
Q1	Player location Los Angeles 1990
Q2	pts position Robinson 1990
Q3	Coach Kareem Abdul-jabbar 1985
Q4	Colorado Dale Schlueter 1975
Q5	teams Robinson 1999



(a) DBLP Efficiency (b) NBA Efficiency

Fig. 4: Execution Time(log-scale)

For NBA, we test five temporal queries, the result of two of them (Q2 and Q4) are STO objects and the the result of the rest (Q1, Q3, Q5) are RTO objects. The performance of TX-Kw, as shown in Fig. 4(b), is better than ISO_IRO approach for RTOs queries since temporal constraints between objects are considered in the join process. ISO_IRO performance gets slightly better than TX-Kw for STO queries. The reason is that TX-Kw spends time in retrieving objects to filter them according to time before performing the join process. On the other hand, SLCA keeps its worst performance as in DBLP.

C. Scalability

Here, the scalability is measured in two ways: changing the number of keywords and increasing the size of time intervals.

Fig.s 5(a) and 5(b) show the log-scaled run time as keywords increase with a constant time (one year) for DBLP and NBA. TX-Kw has the best performance overall other approaches. The performance varies according to the number of retrieved objects since some keywords may exist in the same objects and this might lead to reduction in processing time. However, the execution time of SLCA increases as the number of keywords increase.

The second method of measuring scalability is using time interval variation with a constant number of keyword. The performance is shown in Fig.s 5(c) and 5(d).

We can see that the performance of SLCA and ISO_IRO decreases with the increase of time interval size while it remains approximately constant for TX-Kw. This is because the conventional keywords search require traversing keywords index with each year in the interval while in temporal approach TX-Kw involves only one index traversal. However, there is a slight increase in processing time for TX-Kw depending on the number of retrieved objects as the validity interval increases.

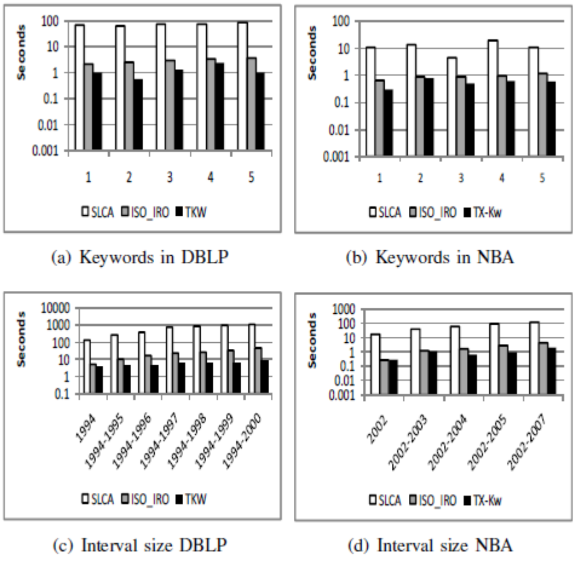
D. Effectiveness

We evaluate quality of results using precision, recall which is heavily used in IR. Precision is the number of relevant objects retrieved divided by the total number of retrieved objects. Recall is the number of relevant objects retrieved divided by the number of relevant objects.

To calculate the precision and recall we manually reformulated tested temporal queries on NBA to be executed into the XML language XQuery and used the results as a basis for evaluation. For DBLP, we tested queries generated by 5 users. For each query the user wrote his attention in natural language and keywords query.

The queries are executed using our approach and ISO_IRO, Table IV shows the average results for precisions and recall for both approaches.

Recall has a high value for both data sets of TX-Kw against ISO_IRO approach since time intervals are not considered in ISO_IRO which lead to empty results in some cases. On the other hand, precision of TX-Kw in Table IV is higher since it detects all possible connections and temporal constraints while ISO_IRO returns more irrelevant results.



(a) Keywords in DBLP (b) Keywords in NBA

(c) Interval size DBLP (d) Interval size NBA

Fig. 5: Scalability Evaluation (log-scale)

Fig.5. Scalability evaluation

TABLE IV. Effectiveness performance

Data set	TX-Kw		ISO_IRO	
	Recall	Precision	Recall	Precision
NBA	0.9575	0.88	0.79	0.304
DBLP	0.978	0.729	0.778	0.432

TABLE V. Ranking performance

Data set	TX-Kw		ISO_IRO	
	R-rank	MAP	R-rank	MAP
NBA	0.906	0.88	0.467	0.3225
DBLP	0.820	0.774	0.695	0.581

We evaluate our proposed ranking method using two popular IR measurements [22]: Mean Average Precision (MAP) and Reciprocal rank R-rank. We use MAP to measure the overall precisions. Precision is computed for each relevant object step. Then we take the average of computed precisions. While R-rank is the inverse of the first rank of correct object retrieved.

We set the mixture parameter to 0.5 to identify weight temporal ranking and keywords ranking. Furthermore, we

Use uncertainty-aware method to compute MAP and R-rank measurements. Ranking performance is shown in Table V. We note that TX-Kw ranking method works very well in NBA where its values metrics are above 0.90 for R-rank and 0.88 for MAP respectively. ISO_IRO ranking has less effectiveness in ranking temporal constraints since its values metrics are below 0.5 for both metrics.

IX. Conclusion

We proposed a new approach, TX-Kw, which supports temporal keyword search queries over temporal XML documents. We model temporal XML as interconnected objects by considering containment and ID/IDREF edges. We also utilized time-aware ranking in IR by mapping it to temporal XML as well as providing an algorithm for temporal ranking that captures the hierarchical structure of XML document. An efficient algorithm is proposed to improve the performance retrieval. Finally we conducted experiments to evaluate and compare our approach against existing keyword search methods by measuring their effectiveness and efficiency. The experiments showed better performance of our approach TX-Kw against other state-of-the-art methods. As future work, we consider integrating both keyword and object indexes to enhance efficiency of our approach.

References

[1] Guo, L., et al., XRANK: ranked keyword search over XML documents, in Proceedings of the 2003 ACM SIGMOD international conference on Management of data. 2003, ACM: San Diego, California. p. 16-27.

[2] Xu, Y. and Y. Papakonstantinou, Efficient keyword search for smallest LCAs in XML databases, in Proceedings of the 2005 ACM SIGMOD international conference on Management of data. 2005, ACM: Baltimore, Maryland. p. 527-538.

[3] Xu, Y. and Y. Papakonstantinou, Efficient LCA based keyword search in XML data, in Proceedings of the 11th international conference on Extending database technology: Advances in database technology. 2008, ACM: Nantes, France. p. 535-546.

[4] Balmin, A., V. Hristidis, and Y. Papakonstantinou, Objectrank: authority-based keyword search in databases, in Proceedings of the Thirtieth international conference on Very large data bases - Volume 30. 2004, VLDB Endowment: Toronto, Canada. p. 564-575.

[5] Ilyas, I.F., G. Beskales, and M.A. Soliman, A survey of top-k query processing techniques in relational database systems. ACM Comput. Surv., 2008. **40**(4): p. 1-58.

[6] Zhang, R., et al. Learning Recurrent Event Queries for Web Search. in Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP 2010, 9-11 October 2010, MIT Stata Center, Massachusetts, USA, A meeting of SIGDAT, a Special Interest Group of the ACL. 2010: ACL.

[7] Bao, Z., et al., An effective object-level XML keyword search, in Proceedings of the 15th international conference on Database Systems for Advanced Applications - Volume Part I. 2010, Springer-Verlag: Tsukuba, Japan. p. 93-109.

[8] Hristidis, V., Y. Papakonstantinou, and A. Balmin. Keyword Proximity Search on XML Graphs. in Proceedings of the 19th International Conference on Data Engineering. 2003.

[9] Tian, Z., J. Lu, and D. Li, A survey on XML keyword search, in Proceedings of the 13th Asia-Pacific web conference on Web technologies and applications. 2011, Springer-Verlag: Beijing, China. p. 460-471.

[10] Cohen, S., et al., XSEarch: a semantic search engine for XML, in Proceedings of the 29th international conference on Very large data bases - Volume 29. 2003, VLDB Endowment: Berlin, Germany. p. 45-56.

[11] Li, G., et al., Effective keyword search for valuable lcas over xml documents, in Proceedings of the sixteenth ACM conference on Conference on information and knowledge management. 2007, ACM: Lisbon, Portugal. p. 31-40.

[12] Lin, R.-R., Y.-H. Chang, and K.-M. Chao, Improving the performance of identifying contributors for XML keyword search. SIGMOD Rec., 2011. **40**(1): p. 5-10.

[13] Bao, Z., et al., Effective XML Keyword Search with Relevance Oriented Ranking, in Proceedings of the 2009 IEEE International Conference on Data Engineering. 2009, IEEE Computer Society. p. 517-528.

[14] Klaus, B., V. Michalis, and W. Gerhard. T-rank: Time-aware authority ranking. in Algorithms and Models for the Web-graph : Third International Workshop, WAW 2004. 2004. Berlin, ALLEMAGNE.

[15] Dai, N. and B.D. Davison, Freshness matters: in flowers, food, and web authority, in Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval. 2010, ACM: Geneva, Switzerland. p. 114-121.

[16] Li, X. and W.B. Croft, Time-based language models, in Proceedings of the twelfth international conference on Information and knowledge management. 2003, ACM: New Orleans, LA, USA. p. 469-475.

[17] Shaparenko, B., et al. Identifying Temporal Patterns and Key Players in Document Collections. in IEEE ICDM Workshop on Temporal Data Mining: Algorithms, Theory and Applications (TDM-05). 2005: Springer.

[18] Yu, P.S., X. Li, and B. Liu, On the temporal dimension of search, in Proceedings of the 13th international World Wide Web conference on Alternate track papers \&amp; posters. 2004, ACM: New York, NY, USA. p. 448-449.

[19] Brin, S. and L. Page, The anatomy of a large-scale hypertextual Web search engine, in Proceedings of the seventh international conference on World Wide Web 7. 1998, Elsevier Science Publishers B. V.: Brisbane, Australia. p. 107-117.

[20] 20. Jatowt, A., Y. Kawai, and K. Tanaka. Temporal Ranking of Search Engine Results. in WISE 2005. 2005. Berlin Heidelberg: Springer-Verlag.

[21] Diaz, F. and R. Jones, Using temporal profiles of queries for precision prediction, in Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval. 2004, ACM: Sheffield, United Kingdom. p. 18-24.

[22] Manica, E., C.F. Dorneles, and R. Galante, Supporting Temporal Queries on XML Keyword Search Engines. Journal of Information and Data Management, 2010. **1**(3): p. 471–486.

[23] Bertino, E., E. Ferrari, and G. Guerrini, A Formal Temporal Object-Oriented Data Model., in EDBT, P.M.G.B. Apers, Mokrane & Gardarin, Georges, Editor. 1996, Springer. p. 342-356.

[24] Bertino, E., et al., Extending the ODMG Object Model with Time, in Proceedings of the 12th European Conference on Object-Oriented Programming. 1998, Springer-Verlag. p. 41-66.

[25] Kanhabua, N., Time-aware Approaches to Information Retrieval, in Department of Computer and Information Science. 2012, Norwegian University of Science and Technology. p. 187.

[26] Salton, G., Automatic text processing: the transformation, analysis, and retrieval of information by computer. 1989: Addison-Wesley Longman Publishing Co., Inc. 530.

[27] Berberich, K., et al., A language modeling approach for temporal information needs, in Proceedings of the 32nd European conference on Advances in Information Retrieval. 2010, Springer-Verlag: Milton Keynes, UK. p. 13-25.

[28] Hristidis, V., et al., Keyword Proximity Search in XML Trees. IEEE Trans. on Knowl. and Data Eng., 2006. **18**(4): p. 525-539.

[29] http://www.cs.washington.edu/, Xml Data Repository. 2002.

[30] http://www.basketballreference.com/, Basketball database 2.1. 2008.