SIRIUS XML IR System at INEX 2006: Approximate Matching of Structure and Textual Content

Eugen Popovici, Gildas Ménier, Pierre-François Marteau

SIRIUS XML IR System at INEX 2006: Approximate Matching of Structure and Textual Content

Eugen Popovici, Gildas Ménier, and Pierre-François Marteau

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

Abstract. In this paper we report on the retrieval approach taken by the VALORIA laboratory of the University of South-Brittany while participating at INEX 2006 ad-hoc track with the SIRIUS XML IR system. SIRIUS retrieves relevant XML elements by approximate matching both the content and the structure of the XML documents. A weighted editing distance on XML paths is used to approximately match the documents structure while the IDF of the researched terms are used to rank the textual content of the retrieved elements. We briefly describe the approach and the extensions made to the SIRIUS XML IR system to address each of the four subtasks of the INEX 2006 ad-hoc track. Finally we present and analyze the SIRIUS retrieval evaluation results. SIRIUS runs were ranked on the 1st position out of 77 submitted runs for the Best In Context task and obtained several top ten results for both the Focused and All In Context tasks.

1 Introduction

This study reports on the second year of experiments conducted by the VALORIA laboratory at the University of South-Brittany with the SIRIUS XML IR system [1] within the framework of the INEX evaluation campaigns.

The main contributions brought relatively to our last year participation are: i) the evaluation of the retrieval approach against a new collection, a new set of topics and new tasks, ii) the implementation of the approximate search and indexing process using a distributed inverted file architecture; and iii) the use of selective indexing profiles defining how the structure and the content of XML tags should be indexed. As for the last year we continue to investigate if and how the approximate match of the structural constraints in the queries may help retrieval and to experiment with different methods for removing overlapping elements.

The paper is organized as follows. In Section 2 we present the main functionalities and characteristics of the SIRIUS XML IR system. In Section 3 we introduce our retrieval approach for the INEX 2006 ad-hoc task. In Section 4 we present and analyze the SIRIUS retrieval evaluation results for the Thorough, Focused, All In Context and Best In Context tasks. Finally, in Section 5 we conclude the paper.

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 185–199, 2007.

2 SIRIUS XML IR System

SIRIUS [2, 3] is a lightweight indexing and search engine for XML documents developed at the VALORIA laboratory of the University of South-Brittany. The retrieval approach implemented in SIRIUS is document oriented. It involves an approximate matching scheme of the structure and textual content. Instead of managing the matching of whole XML trees, SIRIUS splits the documents object model in a set of paths. This set is indexed using optimized data structures. In this view, the request is a path-like expression with conditions on the attribute values. For instance */document(> date "1994")/chapter(= number 3)/John* is a request aiming to extract the documents (written after *94*) with the word John in the chapter number *3*. The matching process takes into account mismatched errors both on the attributes and on the XML elements. It uses a weighted editing distance on XML paths: this provides an approximate matching scheme able to manage jointly the request on textual content and on document structure. The search scheme is extended by a set of IR retrieval operators, and features a set of thesaurus rewriting rules.

2.1 Indexing Scheme

Each element in an XML document may be composed of a set of possible nested XML elements, textual pieces of information (TEXT or CDATA), unordered <attribute, value> pairs, or a mixture of such items. XML documents are generally represented as rooted, ordered, and labeled trees in which each node corresponds to an element and each edge represent a parent-child relationship.

XML Context. According to the tree structure, every node n inherits a path $p(n)$ composed with the nodes that link the root to node n. This path is an ordered sequence of XML elements potentially associated to unordered <attribute, value> pairs $A(n_i)$, that determines the XML context in which the node is occurring. A tree node n, containing textual/mixed information can be decomposed into textual sub-elements. Each string s (or word, lemma, …) of a leaf node is also linked to $p(n)$. This XML context characterizes the occurrence of s within the document and can be represented as follows:

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

Index Model. The indexing process involves the creation of an enriched inverted list designed for the management of these XML contexts. For this model, the entries of the inverted lists are the textual sub-elements s of a tree node. For a sub-element s of a node n, four pieces of information are attached:

- a link to the URI of the document <*fileId*>,
- the <*preorder*> and <*postorder*> positions of the node n in the XML tree,
- an index specifying the positions of s within the document <*wordOffset*>,
- a link toward its XML context $p(n)$ <*ctxtId*>.

2.2 Searching Scheme

Most of the time, for large heterogeneous databases, one cannot assume that the user knows all of the structures – even in the very optimistic case, when all of the structural properties are known. Some straightforward approaches (such as the XPath search scheme [4]) may not be efficient in these cases. As the user cannot be aware of the complete XML structure of the data base due to its heterogeneity, efficient searching should involved exact and approximate search mechanisms.

The main structure used in XML is a tree: It seems acceptable to express a search in term of tree-like requests and approximate matching. We proposed [6], to focus on path matching rather than on tree matching – in a similar way with the XML fragment approach [5]. The request should be expressed as a set of path *p(r)* that is matched with the set of sub-path *p(n)* in the document tree. This breaks the algorithmic complexity of tree matching techniques while still providing high precision results [6]. This 'low-level' matching only manage subpath similarity search with conditions on the elements and attributes matching. This process is used to design a more higher-level request language: a full request is a tree of low-level matching goals (as leafs) with set operators as nodes. These operators are used to merge leaf results. The whole tree is evaluated to provide a set of ranked answers. The operators are classical set operators (intersection, union, difference) or dedicated fuzzy merging processors. The system analyzes a request and produces a set of weighted results. Let $\{ (e_i, v_i) \}$ the set of weighted results produced by the system, where e_i is a an element of the result and $v_i \in [0..1]$ a weight showing the relevance of the returned element to the request.

Textual Content Ranking Scheme. We compute the relevance value $v_i \in [0..1]$ for all the XML elements e_i containing at least one researched term τ_k of a content only request *CO*. The ranking scheme takes into account the number and the discriminating power of the retrieved terms in the collection. We used a dedicated TFIDF [7] function for this purpose:

$$v_i(e_i, CO) = \xi \cdot \sum_k \lambda_k . \tau_k \ . \tag{2}$$

where k is the number of terms τ_k in the *CO* request, λ_k is an IDF weighting factor specifying the discriminating power of the term τ_k in the collection :
$\lambda_k = 1 - log((1 + |D\ \tau_k|) / (1 + |D|))$; where $|D\ \tau_k|$ is the number of documents in which τ_k is occurring ; $|D|$ the total number of documents in the collection ; and ξ a normalization constant $\xi = 1 / \Sigma_k (\lambda_k)$;

Approximate Path Search. Let p^R be a structural constraint, expressed as a path goal with conditions or constraints to be fulfilled on the attributes. We investigate the similarity between a p^R (coding a path with constraints) and p_i^D (a root/../terminal(r) path of the tree T^D associated to an index document D) as follow:

$$\sigma(p^R, p_i^D) = 1/(1 + \delta_L (p^R, p_i^D)) \ . \tag{3}$$

where δ_L is a dedicated editing distance (see [8]).

The search complexity is $O(l(p^R).deep(T^D).|\{ p_i^D \}|)$ with $|\{ p_i^D \}|$ the size of the set $\{ p_i^D \}$ (i.e. the number of different paths in D, starting at the root and leading to the last element of the p^R request – terminal(r)), $l(p)$ the length of the path p and $deep(T)$ the deepest level of T. This complexity remains acceptable for this application as 99% of the XML documents have fewer than 8 levels and their average depth is 4 [9]. We designed [6] an editing pseudo-distance 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 Levenshtein distance, computes a minimal sequence of elementary transformations 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 $C_{subst}(n, n')$.
- **Deletion:** a node n in p_i^D is deleted for a cost $C_{del}(n)$,
- **Insertion:** a node n is inserted in p_i^D for a cost $C_{ins}(n)$.

Weighting Scheme for INEX. The NEXI language [10] allows only the descendant relationship between the nodes in a path. Therefore the XML path expressed in the request is interpreted as a *subsequence* of an indexed path, where a subsequence need not consist of contiguous nodes. To model this, we relaxed in **[1]** the weights of the path editing distance in order to allow node deletions in the indexed paths without any penalty: $C_{del}(n) = 0$, $C_{ins}(n) = \xi$, and $C_{subst}(n, n') = \xi$. 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: $C_{subst}(n, n')=\{ \xi \text{ if } (n \neq n'); \frac{1}{2}\cdot\xi \text{ if } (n = n') \ \& (\neg attCond(n')); 0 \text{ if } (n = n') \ \& (attCond(n')) \}$, where *attCond* stands for a condition stated in the request that should apply to the attributes.

For a sequence $Seq(p_i^D, p^R)$ of elementary 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 [11] computes the best $Seq(p_i^D, p^R)$ (i.e. minimizes $GC()$ *cost)* with a complexity of $O(length(p_i^D) * length(p^R))$ as stated earlier. Let

$$\delta_L(p^R , p_i^D,) = Min_k\, GC(Seq_k(p^R, p_i^D)) . \quad (4)$$

Given p^R and p_i^D, the value for $\sigma(p^R , p_i^D) \rightarrow 0$ when the number of mismatching nodes and attribute conditions between p^R and p_i^D increases. For a perfect match $\sigma(p^R , p_i^D) = 1$, i.e. all the elements and the conditions on attributes from the request p^R match correspondent XML elements in p_i^D .

The weights used to compute the structural similarity relate to an end user having precise but incomplete information about the XML tags of the indexed collection and about their ancestor-descendant relationships. The structural similarity takes into account the order of occurrence of the matched nodes and the number of nodes with no matching in the request. It heavily penalizes any mismatch relatively to the information provided by the user but it is independent to mismatches/extra information extracted from the indexed paths.

Merging Structure and Content Matching Scores. We add structural matching information to the set of solutions returned by the system using a weighted linear aggregation between the conditions on structure $\sigma (p^R , p_i^D)$ and the initial/textual ranking score v_i as follows:

$$v'_i = \beta \cdot \sigma(p^R, p_i^D) + (1 - \beta) \cdot v_i . \quad (5)$$

The value of the $\beta \in [0..1]$ parameter may be used to emphasize the importance of the structural versus textual content matching scores.

3 SIRIUS Approach for the INEX 2006 Ad-Hoc Task

The retrieval task we are addressing at INEX 2006 is the ad-hoc retrieval of XML documents. This involves the searching of a document collection of 4.6 GB made of 659,388 English articles from Wikipedia using a set of 125 topics. The structural part of the collection corresponds to the Wikipedia templates (about 5000 different tags). The topics may contain both content and structural conditions and, in response to a query, arbitrary XML elements may be retrieved by the system. An example of an INEX 2006 topic with the *title* and *castitle* expressed in NEXI language [10] is given in Fig. 1.

```
<inex_topic topic_id="406" ct_no="198">
  <title>book architecture</title>
  <castitle>//template[about(.//@name,book reference)]//*[about(.,architecture)]</castitle>
  <description>Show me books about architecture</description>
  <narrative>After coming home from a trip to venice...</narrative>
  <ontopic_keywords>+house</ontopic_keywords>
</inex_topic>
```

Fig. 1. An excerpt of the INEX 2006 topic 406

Content only (CO) queries contain just search terms (see the *title* part in Fig. 1) while the content and structure (CAS) queries (see the *castitle* part in Fig. 1) are topic statements that contain explicit references to the XML structure, and explicitly specify the contexts of the user's interest (e.g. target elements) and/or the context of certain search concepts (e.g. support elements).

3.1 Indexing the Wikipedia Collection

SIRIUS has the capability of using indexing profiles for a specific collection. The *indexing profiles* are composed of rules defining how the structure and the content of each specified XML tag should be indexed. By default, all the non empty XML tags are fully indexed. Using these profiles we may decide or not to index the attributes associated to a given tag, to index only the content of the *presentation tags* or *jump tags* [12], or to completely ignore some *logical tags* for a specific collection. The use of indexing profiles may reduce significantly the volume of the requested disk space for the index and improves the system performances both in indexing and retrieval time.

We use the rules shown in Table 1. to index the Wikipedia collection. This indexing profile was manually defined as we assumed that the jump and presentation tags contained information that should not be retrieved out of their context. The logical tags <*name*>, <*title*> and <*caption*> are of a particular importance for the Wikipedia collection, as this will ensure that the <*title*> of a <*section*> will always be

retrieved with the <*section*> itself, that the <*name*> of an <*article*> will be retrieved with the whole <*article*>, and that the <*caption*> of a <*figure*> or <*table*> will be retrieved only associated to the element to which they are referring to.

Table 1. Indexing rules for the Wikipedia collection

	Ignore tags	Ignore tag attributes
Presentation tags	`emph2, emph3, emph4, sup`	`table, tr, td, font`
Jump tags	`collectionlink, unknownlink, outsidelink, languagelink`	
Logical tags	`title, name, image, caption`	

The Wikipedia collection is processed using an XML SAX parser and standard methods for stop words removal and stemming. At indexing time, the most frequent words are eliminated using a stop list. The XML elements containing no valid textual content after stop words removal are not indexed. The index terms are stemmed using the Porter algorithm [13]. The index model (Section 2.1) is implemented on top of the Berkeley DB[1] library using a combination of BTrees and Hashtables structures. The inverted file index is constructed in parallel by using a *Physical Document Partitioning* approach [14]. The total size of the index is about 86% of the initial database size – i.e. 4GB.

3.2 Processing NEXI Requests

Processing CO requests. CO queries are INEX topics containing only textual search terms (i.e. see the *title* part in Fig. 1). We compute the relevance score for all the leaves elements of the XML tree containing at least one of the researched terms using a variant of the TF-IDF ranking scheme (see eq. 2). In our approach we consider the XML element containing a researched term as the basic and implicitly valid unit of retrieval regardless of its size.

Processing CAS requests. For CAS topics, we have two cases: simple queries of the form *//A[B]* – i.e. the request specifies only the target elements, and complex queries of the form *//A[B]//C[D]* – i.e. the request specifies both target (i.e. *//C[D]*) and support (i.e. *//A[B]*) elements.

Processing the Support and Target Elements. For simple type queries of the form *//A[B]* like *//template//*[about(.,architecture)]* (see topic in Fig. 1), we rank the textual content of the nodes using the same ranking scheme as for the CO requests. The structural constraints from the requests are interpreted as structural hints [10]. We compute the similarity between the structural constraints expressed in the request – i.e. *//template//** – and the XML paths of the candidate fragments using a modified editing distance (see eq. 3) involving specific heuristics for attributes and attributes values [1]. Finally we merge the content and structural match scores using a weighted linear aggregation method (see eq. 5).

[1] http://www.sleepycat.com/

Processing the Containment Conditions. To process complex queries of the form *//A[B]//C[D]* (see the *castitle* part in Fig. 1) we compute the relevance for both the support elements *//A[B]* and target elements *//A//C[D]*. Next, we select only the target elements that have at least a relevant support element occurring in the same document. The logic behind this is that if a relevant support element exists in a document, its weight should be propagated using a *max* function to the root node of the XML tree that is an ancestor – i.e. support element – for all the elements of the tree. This applies inclusively to target elements.

The similarity computation for a complex request involves modifications of the relevance associated with a result element. The relevance of a result element is computed as the arithmetic average between the relevance of the target element and the maximum relevance of its support elements.

Formally, let $\{(e_i, v_i)\}$ the set of target results, $\{(e_j, v_j)\}$ the set of support elements, where e_i is a an element of the result and $v_i \in [0..1]$ its relevance weight. Let e^D a descendant of document D. The set of weighted results produced by the system is $\{ (e_i^D, v'_i) \}$ with $v'_i = (v_i + Max_j (v_j)) / 2$ where $\exists e_j^D \in \{ (e_j, v_j) \}$.

Using this approach, the target elements without support elements are discarded from the final answers, while the ones supported by highly relevant elements are boosted in the final ranking. The final results are sorted by relevance values and the top N results returned.

4 Experimental Results

We submitted a total of 20 runs to all of the four tasks of the ad-hoc retrieval track: Thorough, Focused, All In Context and Best In Context [15]. In all the submitted runs we used the same basic retrieval approach:

- To answer INEX 06 topics, we use automatic transformation of the *title* and *castitle* part of the topics expressed in NEXI [10] to SIRIUS recursive query language as described in [1].

CO runs

- The XML elements directly containing the research terms are considered as independent and the only valid units of retrieval;
- IDF weighting for textual content of the leaf nodes containing the researched terms (i.e. **IDF**, see eq. 2.);
- Strict and vague search for phrase matching. In the strict sequence matching runs the researched terms must occur in sequence and belong to the same XML element. This is not required for the vague phrase matching runs (i.e. **noSEQ**) that rank as best results the XML elements containing all the researched terms without taking into account their order of occurrence.

*CAS runs (*cas*)*

- The structural constraints on both the support elements (where to look) and on the target elements (what to return) are interpreted vaguely, as structural hints. The vague interpretation of the structural constraints is implemented using a modified

editing distance (**EDs**) on the XML paths with conditions on attributes and attributes values (see Section 2.2, eq. 2 and 3) .

- We use weighted linear aggregation for content and structure matching scores. (see eq. 5) The runs (**W0_1**, **W0_5**) use different values for the β parameter to emphasize the importance of the structural versus textual content matching (i.e. β=0.1 biases the ranking towards the textual content while β=0.5 uses equal weights for merging the structural and content matching relevance scores).
- We use boolean (**BOOL**) merging operators at document level.

4.1 Thorough Task

At the Thorough task, the system estimates the relevance of elements in the collection. We submitted five runs identified by runId's using combinations of the abbreviations introduced above. We report in Fig. 2 and Table 2 the evaluation curves for the ep/gr evaluation metric and the ranks obtained by all the submitted runs. The results may contain overlapping elements (i.e. Overlap=off). Details of the evaluation metrics can be found in [16].

Table 2. Task: Thorough, Metric:ep-gr, Quantization: gen, Overlap=off, R: rank/106 runs

			filtered assessments	
RunId	*MAep*	*R*	*MAep*	*R*
IDF_BOOL_noSEQ	**0.0158**	**42**	**0.0296**	**37**
IDF_BOOL	0.0151	45	0.0287	42
casEDsW0_1_IDF_BOOL_noSEQ	0.0146	48	0.0274	43
casEDsW0_5_IDF_BOOL_noSEQ	0.0134	50	0.0253	45
casEDsW0_5_IDF_BOOL	0.0130	51	0.0242	48

We obtained average rankings for the Thorough task. This is not surprising as the implementation of our approach is biased towards focused retrieval. A rather surprising result is the fact that using the structural hints does not improve the quality of the retrieved results. Rather the opposite. The best overall performance is obtained by the run using only the textual content and no phrase constraints (IDF_BOOL_noSEQ) with a MAep value of 0.0158 and respectively 0.0296 when evaluated against the *filtered assessments*[2].

4.2 Focused Task

The aim of the Focused retrieval strategy is to find the most exhaustive and specific element in a path. In other words, the result list should not contain any overlapping elements. For the Thorough task we considered the XML element containing a researched term as the basic and implicitly valid unit of retrieval regardless of its size. This approach "naturally" implements a focused strategy as it returns the most focused elements containing the research terms. However, cases where nested/overlapping XML elements could be returned as valid results may occur.

[2] "element links" (i.e. collectionlink, wikipedialink, redirectlink, unknownlink, outsidelink and weblink) in the assessments have been given an exhaustive value of ? (corresponding to "too small" in INEX 2005 relevance definition).

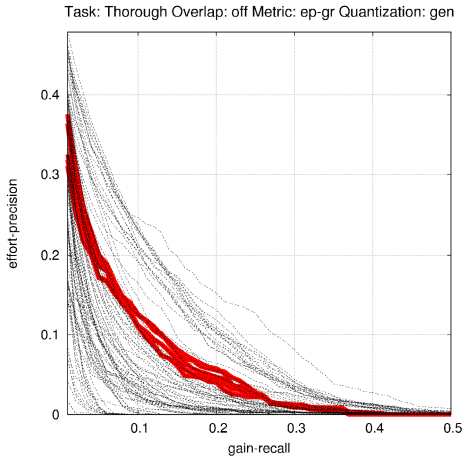

Fig. 2. Task: Thorough, Metric:ep/gr , Quantization: gen, Overlap: off, Filtered assessments

We implemented a two steps post filtering process to remove the overlapping elements from the results list [1]: i) we recalculate the relevance of the elements in the answer list in order to reflect the relevance of their descendants elements (if any); and ii) we select non overlapping elements from the list.

The weights are calculated in a bottom-up manner from the leafs to the highest non overlapping nodes composing the answer by using two strategies:

- *MAX* - the max relevance value is propagated recursively to the highest non overlapping elements; and
- *AVG* - the relevance of a node is computed as the arithmetic average of all its descendant relevant nodes including its own relevance.

To select the non overlapping elements we compared the following strategies:

- *HA* - the highest ancestor from the answer list is selected;
- *MR* - the most relevant answer is selected recursively from the answer list as long as it not overlaps with an already selected element – i.e. for equally relevant overlapping elements we choose either the descendant (*MRD*) or the ancestor (*MRA*).

We experimented with different settings for computing the elements relevance and selecting the non overlapping answers for the Focused tasks within the framework of the INEX 2005 campaign [1]. This year we selected only the MAX_MRD and MAX_HA strategies for the focused task as they obtained the best results during the INEX 2005 evaluation.

We report here the nxCG values @5, @10, @25 and @50 (see [16] for metric descriptions) for all the submitted focused runs, along with their official ranks in the INEX06 campaign. The runs are evaluated on both the original (see Tables 3 and 5) and filtered assessments (see Tables 4, 6 and Fig. 3).

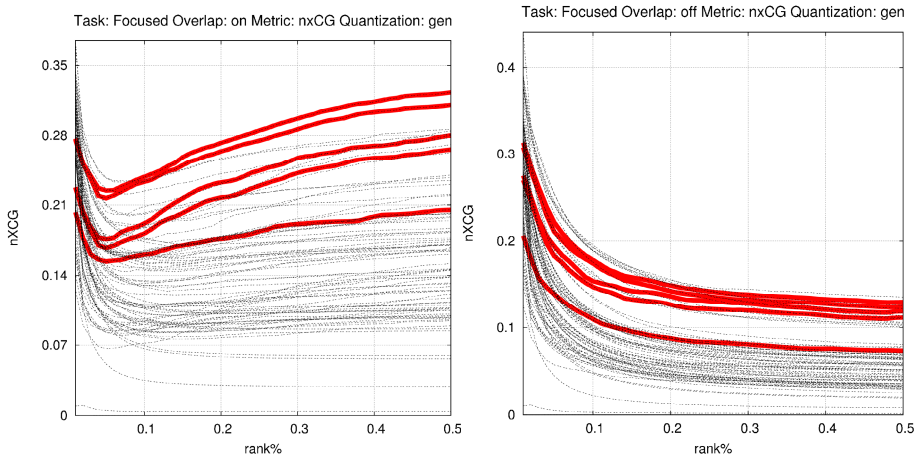

Fig. 3. Task: Focused Metric:nxCG, Quantization: generalised, Filtered assessments, Overlap= on (left) ; and Overlap=off (right)

Table 3. Task: Focused, Metric: nxCG, Quantization: generalised, Overlap=on, R: rank/85 runs

RunId	*nxCG@5*	*R*	*nxCG@10*	*R*	*nxCG@25*	*R*	*nxCG@50*	*R*
IDF_BOOL_noSEQ_MAX_MRD	0.2882	47	**0.2759**	**24**	**0.2393**	**13**	**0.2095**	**6**
IDF_BOOL_MAX_MRD	**0.2889**	**45**	0.2695	28	0.2391	14	0.2022	9
casEDsW0_5_IDF_BOOL_noSEQ_MAX_MRD	0.2335	66	0.2215	60	0.1965	35	0.1638	29
casEDsW0_5_IDF_BOOL_MAX_MRD	0.2338	65	0.2202	61	0.1933	40	0.1572	35
casEDsW0_1_IDF_BOOL_noSEQ_Foc_MAX_HA	0.2055	73	0.1996	65	0.1693	54	0.1436	46

Table 4. Task: Focused, Metric: nxCG, Quantization: generalised, Overlap=on, Filtered assessments, R: rank/85 runs

RunId	*nxCG@5*	*R*	*nxCG@10*	*R*	*nxCG@25*	*R*	*nxCG@50*	*R*
IDF_BOOL_noSEQ_MAX_MRD	0.2832	48	**0.2752**	**23**	**0.2475**	**9**	**0.2289**	**4**
IDF_BOOL_MAX_MRD	**0.2840**	**46**	0.2679	28	0.2469	10	0.2211	7
casEDsW0_5_IDF_BOOL_noSEQ_MAX_MRD	0.2297	66	0.2218	59	0.2039	31	0.1782	23
casEDsW0_5_IDF_BOOL_MAX_MRD	0.2301	64	0.2196	61	0.2004	33	0.1709	32
casEDsW0_1_IDF_BOOL_noSEQ_Foc_MAX_HA	0.2043	73	0.2012	63	0.1757	47	0.1562	42

Table 5. Task: Focused, Metric: nxCG, Quantization: generalised, Overlap=off, R: rank/85 runs

RunId	*nxCG@5*	*R*	*nxCG@10*	*R*	*nxCG@25*	*R*	*nxCG@50*	*R*
IDF_BOOL_noSEQ_MAX_MRD	**0.3227**	**38**	**0.3238**	**16**	**0.2807**	**12**	**0.2424**	**9**
IDF_BOOL_MAX_MRD	0.3180	40	0.3093	21	0.2768	14	0.2339	12
casEDsW0_5_IDF_BOOL_noSEQ_MAX_MRD	0.2770	60	0.2829	36	0.2475	21	0.2071	20
casEDsW0_5_IDF_BOOL_MAX_MRD	0.2719	62	0.2735	41	0.2418	27	0.2002	21
casEDsW0_1_IDF_BOOL_noSEQ_Foc_MAX_HA	0.2073	73	0.2063	66	0.1779	59	0.1477	46

Table 6. Task: Focused, Metric: nxCG, Quantization: generalised, Overlap=off, Filtered assessments, R: rank/85 runs

RunId	*nxCG@5*	*R*	*nxCG@10*	*R*	*nxCG@25*	*R*	*nxCG@50*	*R*
IDF_BOOL_noSEQ_MAX_MRD	**0.3227**	**37**	**0.3238**	**14**	**0.2805**	**12**	**0.2440**	**9**
IDF_BOOL_MAX_MRD	0.3180	39	0.3084	20	0.2766	14	0.2353	12
casEDsW0_5_IDF_BOOL_noSEQ_MAX_MRD	0.2770	59	0.2829	33	0.2477	18	0.2082	17
casEDsW0_5_IDF_BOOL_MAX_MRD	0.2719	61	0.2726	39	0.2416	23	0.2011	18
casEDsW0_1_IDF_BOOL_noSEQ_Foc_MAX_HA	0.2073	73	0.2063	64	0.1780	54	0.1483	43

The MAX_MRD method for overlap removal retrieves more focused elements and seems to be more adequate for the focused task than its MAX_HA competitor. This may be considered with care as the results are influenced with different degrees by the structural matching process.

For the Focused task, the system is better ranked than on the Thorough task regardless if it is evaluated with the overlap 'on' or 'off'. SIRIUS has several results in the best top ten runs using the nxCG@25 and nxCG@50 metrics (see Tables 3, 4, 5 and 6; top ten results are highlighted, best results are in bold characters).

By analyzing the overall comportment of nxCG curves of Fig. 3 we observe that SIRIUS runs have a good recall. We also observe a slightly decrease in the system retrieval performance for the first ranked results. This may be determined by the indexing configuration settings (see Table 1). The indexing profile did not allowed for a large number of small/possibly relevant focused elements (i.e. jump tags & presentation tags) to be retrieved. This hypothesis is sustained by the slight increase in the SIRIUS performance when evaluating the runs against the filtered assessments. The differences are not major as the indexing profiles are not an exact match of the rules used to obtain the filtered assessments. The indexing profile eliminated only a part of the tags defined as "too small". When evaluating the runs against the filtered assessments the remaining element links (i.e. *redirectlink*, *wikipedialink* and *weblink*) as well as the eliminated tags but that were considered relevant by the assessors (*emph2*, *emph3*, *title*, *name*, and *caption*) [17] penalize the results. We observe that as for the Thorough task, the runs involving structural conditions performed worse than their content only pairs.

4.3 All in Context Task

For the INEX 2006 All In Context task, the systems have to find a set of elements that corresponds well to (all) relevant information in each article. The relevant elements must be clustered per article and ordered in their original document order when returned to the user. The assumption is that users consider the article as the most natural unit, and prefer an overview of relevance in their context.

For this task, we used as starting point the approach used for the Focused runs. We clustered the non overlapping results by file and ranked them according to their relevance inside each file. We set the article score equal to the most relevant element occurring inside each file. The files are ranked by their relevance. We returned the top N relevant results for each file, where N={5, 10} until reaching the INEX 2006 max results limit per topic (i.e. 1500 results).

The official and additional SIRIUS evaluation results for this task are given in Tables 7, 8, 9 and Fig. 4 (left) (top ten results are highlighted, best results are in bold characters).

Table 7. Task: All In Context (Article level), Metric: hixeval-article, R: rank/62 runs

RunId	*F[5]*	*R*	*F[10]*	*R*	*F[25]*	*R*	*F[50]*	*R*	*MAP*	*R*
IDF_BOOL_MAX_MRD_10	**0.1028**	**44**	**0.1150**	**40**	0.1181	39	0.1055	33	0.0752	39
IDF_BOOL_noSEQ_MAX_MRD_10	0.1027	45	0.1132	41	**0.1203**	**38**	**0.1079**	**32**	**0.0759**	**38**
casEDsW0.5_IDF_BOOL_noSEQ_MAX_HA_5	0.0881	50	0.0871	53	0.0966	48	0.0864	45	0.0623	48
casEDsW0.5_IDF_BOOL_MAX_MRD_5	0.0867	52	0.0910	52	0.0956	49	0.0864	46	0.0613	49
casEDsW0.1_IDF_BOOL_noSEQ_AVG_HA_5	0.0373	57	0.0457	57	0.0549	57	0.0533	57	0.0299	58

Table 8. Task: All In Context (Element level), Metric: hixeval-element, Overlap=off, R: rank/57 runs

RunId	*hixeval-element-intersection*		*hixeval-element-union*	
	F-avg	*R*	*F-avg*	*R*
casEDsW0.1_IDF_BOOL_noSEQ_AVG_HA_5	**0.4695**	**2**	0.2845	24
casEDsW0.5_IDF_BOOL_noSEQ_MAX_HA_5	0.4677	3	0.3306	15
IDF_BOOL_noSEQ_MAX_MRD_10	0.4658	5	**0.3492**	**8**
IDF_BOOL_MAX_MRD_10	0.4650	6	0.3464	9
casEDsW0.5_IDF_BOOL_MAX_MRD_5	0.3948	32	0.2752	31

Table 9. Task: All In Context (combining Article and Element levels scores), Metric: generalized Precision/Recall, R: rank/56 runs

RunId	*gP[5]*	*R*	*gP[10]*	*R*	*gP[25]*	*R*	*gP[50]*	*R*	*MAgP*	*R*
IDF_BOOL_MAX_MRD_10	**0.2245**	**32**	**0.2164**	**24**	**0.1801**	**15**	0.1386	13	**0.1414**	**15**
IDF_BOOL_noSEQ_MAX_MRD_10	0.2231	33	0.2117	25	0.1779	16	**0.1417**	**11**	0.1408	16
casEDsW0.5_IDF_BOOL_noSEQ_MAX_HA_5	0.1881	43	0.1654	41	0.1414	33	0.1141	29	0.1133	29
casEDsW0.5_IDF_BOOL_MAX_MRD_5	0.1642	45	0.1553	44	0.1325	35	0.1059	33	0.1021	34
casEDsW0.1_IDF_BOOL_noSEQ_AVG_HA_5	0.1586	47	0.1448	46	0.1245	40	0.0988	34	0.0868	38

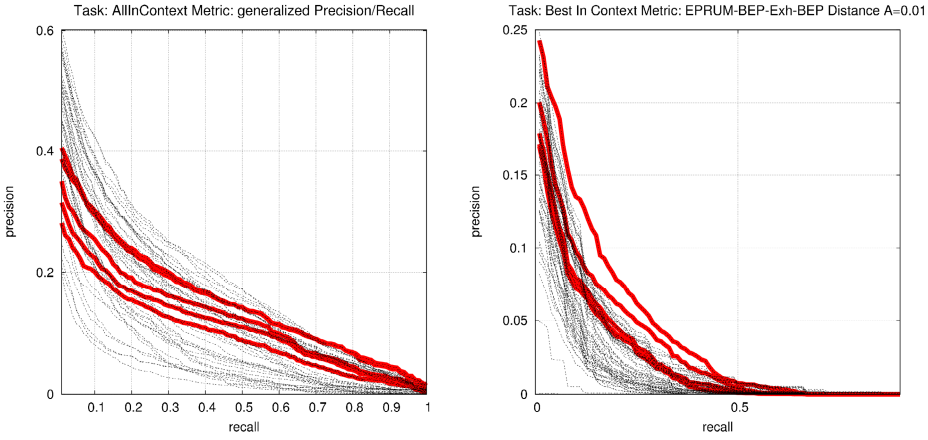

Fig. 4. Task: All In Context (combining Article and Element levels scores), Metric: generalized Precision/Recall (left) ; Task: Best In Context. Metric:EPRUM-BEP-Exh-BEPDistance, A=0.01 (right).

SIRIUS obtained relatively good rankings for the All In Context task (see Table 9). We may get an insight at the SIRIUS retrieval performance by analyzing All In Context Task additional scores for the article-level (Table 7) and element-level (Table 8). The element-level scores show that SIRIUS is able to detect and extract the amount of retrievable relevant information within an article with very good results. Unfortunately, SIRIUS retrieval performance highly degrades when evaluated at article-level. A possible way to improve the retrieval performances of the system is to rank the files using a global relevance value computed at article level.

4.4 Best in Context Task

For the Best In Context task we had to retrive a ranked list of articles. For each article, we must return a single element, representing the best entry point for the article with respect to the topic of request. For this task we used the same approach as for the All In Context Task with N set to 1. The official results evaluated with BEP-D (see Table 10) and EPRUM-BEP-Exh-BEPDistance [18] (see Table 11) were ranked several times in the top ten positions out of 77 submitted runs (see Fig. 4 – right). The top ten results are highlighted while the best obtained values are in bold characters.

Table 10. Task: Best In Context. Metric: BEPD, R: rank/77 runs.

RunId	*A=0.01*	*R*	*A=0.1*	*R*	*A=1*	*R*	*A=10*	*R*	*A=100*	*R*
IDF_BOOL_noSEQ_AVG_MRD	**0.1959**	**1**	0.2568	2	0.3642	6	0.5596	6	**0.7556**	**7**
IDF_BOOL_MAX_HA	0.1722	2	**0.2753**	**1**	**0.4095**	**1**	**0.5847**	**3**	0.7542	8
casEDsW0.1_IDF_BOOL_noSEQ_MAX_HA	0.1394	16	0.2303	8	0.3580	7	0.5239	18	0.6853	27
casEDsW0.5_IDF_BOOL_noSEQ_MAX_HA	0.1346	17	0.2222	12	0.3447	12	0.5048	24	0.6631	36
casEDsW0.5_IDF_BOOL_MAX_HA	0.1322	19	0.2114	17	0.3222	23	0.4691	36	0.6170	45

Table 11. Task: Best In Context. Metric:EPRUM-BEP-Exh-BEPDistance, R: rank/77 runs.

RunId	*A=0.01*	*R*	*A=0.1*	*R*	*A=1*	*R*	*A=10*	*R*	*A=100*	*R*
IDF_BOOL_noSEQ_AVG_MRD	**0.0407**	**1**	0.0579	8	0.0873	13	0.1489	16	0.2193	35
IDF_BOOL_MAX_HA	0.0304	4	**0.0607**	**6**	**0.1069**	**7**	**0.1770**	**8**	**0.2536**	**14**
casEDsW0.1_IDF_BOOL_noSEQ_MAX_HA	0.0233	24	0.0478	15	0.0881	12	0.1480	19	0.2180	36
casEDsW0.5_IDF_BOOL_noSEQ_MAX_HA	0.0218	31	0.0444	24	0.0812	20	0.1363	34	0.2031	42
casEDsW0.5_IDF_BOOL_MAX_HA	0.0214	34	0.0435	29	0.0785	23	0.1323	38	0.1969	44

The Best In Context task results confirmed that the runs using structural hints (**cas**) are ranked lower than the ones using only the textual content. We have a single content and structure run in the top ten results casEDsW0.1_IDF_BOOL_noSEQ_MAX_HA for A=0.1 when evaluated with the BEPD metric (see Table 10.).

5 Conclusions

This year, at INEX 2006, we have pursuit the evaluation of the retrieval performances of the SIRIUS XML IR system [2, 3] started last year within the INEX 2005 campaign [1]. SIRIUS retrieves relevant XML elements by approximate matching both the content and the structure of the XML documents. A modified weighted editing distance on XML paths is used to approximately match the documents structure while the IDF of the researched terms are used to rank the textual content of the retrieved elements. A number of extensions were brought to the system in order to cope with the requirements of the Thorough, Focused, All In Context and Best In Context tasks.

We have submitted and evaluated 20 valid runs in all the INEX 2006 ad-hoc tasks, and showed the system ability to retrieve relevant non overlapping XML elements within the Focused, All In Context and Best In Context tasks. SIRIUS obtained average rankings for the Thorough task and top ten ranked results in the range of the 50 first retrieved answers for the Focused and All In Context task. For Best In

Context task the results were quite encouraging as the system was ranked on the 1st place out of 77 submissions for both BEPD and EPRUM metrics with A=0.01[3]. (see Tables 10, 11).

The runs using structural constraints were consequently outperformed by the runs using content only conditions, while the runs using strict constraints for phrase searching were outperformed by their relaxed variants.

Our experiments at INEX 2005 showed that taking into account the structural constraints improved the retrieval performances of the system and jointly showed the effectiveness of the proposed weighted editing distance on XML paths for this task. This observation was not confirmed by any of the tasks evaluated at INEX 2006. More experimental studies analyzing the use of structural hints within the XML IR requests are necessary to better understand the reasons for this behaviour.

References

1. Popovici, E., Ménier, G., Marteau, P.-F.: SIRIUS: A Lightweight XML Indexing and Approximate Search System at INEX 2005. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, pp. 321–335. Springer, Heidelberg (2006)
2. Ménier G., Marteau P.F.: Information retrieval in heterogeneous XML knowledge bases, IPMU, July 1-5, 2002, Annecy, France (2002)
3. Ménier, G., Marteau, P.F.: PARTAGE: Software prototype for dynamic management of documents and data. In: ICSSEA, 29 November-1 December, 2005, Paris, France (2005)
4. Clark, J., DeRose, S.: XML Path Language (XPath) Version 1.0, W3C Recommendation, November 16, 1999 (1999) http://www.w3.org/TR/xpath.html
5. Carmel, D., Maarek, Y.S., Mandelbrod, M., Mass, Y., Soffer, A.: Searching XML documents via XML fragments, SIGIR 2003, Toronto, Canada, pp. 151–158 (2003)
6. Amer-Yahia, S., Koudas, N., Marian, A., Srivastava, D., Toman, D.: Structure and Content Scoring for XML, VLDB, Trondheim, Norway, pp. 361–372 (2005)
7. Salton, G., Buckeley, C.: Term-weighting approaches in automatic text retrieval. Information Processing and Management 24, 513–523 (1988)
8. Levenshtein, A.: Binary Codes Capable of Correcting Deletions, Insertions and Reversals. Sov.Phy. Dohl. 10, 707–710 (1966)
9. Mignet, L., Barbosa, D., Veltri, P.: The XML Web: A First Study, WWW 2003, May 20–24, Budapest, Hungary (2003)
10. Trotman, A., Sigurbjörnsson, B.: Narrowed Extended XPath I (NEXI). In: Fuhr, N., Lalmas, M., Malik, S., Szlávik, Z. (eds.) INEX 2004. LNCS, vol. 3493, pp. 16–40. Springer, Heidelberg (2005)
11. Wagner, R., Fisher, M.: The String-to-String Correction Problem. Journal of the Association for Computing Machinery 12(1), 168–173 (1974)
12. Tannier, X., Girardot, J.-J., Mathieu, M.: Classifying XML Tags through Reading Contexts. In: DocEng, Bristol, United Kingdom, pp. 143–145 (2005)
13. Porter, M.F.: An algorithm for suffix stripping, Program. Program 14(3), 130–137 (1980)

[3] Note that high values of A (e.g. 10) does not discriminate whether the answer is near to or far from the BEP. Whereas, low values of A (e.g. 0,1) favour runs that return elements very close to a BEP.

14. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. ACM Press. Addison-Wesley, New York (1999)
15. Clarke C., Kamps J., Lalmas M.: INEX 2006 Retrieval Task and Result Submission Specification. In: INEX 2006 Workshop Pre-Proceedings, Dagstuhl, Germany, December 18–20, 2006, pp. 381–388 (2006)
16. Kazai, G., Lalmas, M.: INEX 2005 Evaluation Metrics. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, pp. 16–29. Springer, Heidelberg (2006)
17. Kamps, J., Koolen, M., Sigurbjörnsson, B.: The University of Amsterdam at INEX 2006. In: INEX 2006 Workshop Pre-Proceedings, Dagstuhl, Germany, December 18–20, 2006, pp. 88–99 (2006)
18. Piwowarski, B., Dupret, G.: Evaluation in (XML) information retrieval: expected precision-recall with user modelling (EPRUM). In: SIGIR 2006, pp. 260–267 (2006)