

# RMIT INEX experiments: XML Retrieval using Lucy/eXist

Jovan Pehcevski  
School of CS and IT  
RMIT University  
Melbourne, Australia 3000  
jovanp@cs.rmit.edu.au

James Thom  
School of CS and IT  
RMIT University  
Melbourne, Australia 3000  
jat@cs.rmit.edu.au

Anne-Marie Vercoustre  
CSIRO-ICT Centre  
Melbourne, Australia 3000  
Anne-Marie.Vercoustre@csiro.au

## ABSTRACT

This paper reports on the RMIT group's approach to XML retrieval while participating in INEX 2003. We indexed XML documents using Lucy, a compact and fast text search engine designed and written by the Search Engine Group at RMIT University. For each INEX topic, up to 1000 highly ranked documents were then loaded and indexed by eXist, an open source native XML database. A query translator converts the INEX topics into corresponding Lucy and eXist query expressions, respectively. These query expressions may represent traditional information retrieval tasks (unconstrained, CO topics), or may focus on retrieving and ranking specific document components (constrained, CAS topics). With respect to both these expression types, we used eXist to extract final answers (either full documents or document components) from those documents that were judged highly relevant by Lucy. Several extraction strategies were used that differently influenced the ranking order of the final answers. The final INEX results show that our choice for a translation method and an extraction strategy leads to a very effective XML retrieval for the CAS topics. We observed a system limitation for the CO topics resulting in the same or similar choice to have little or no impact on the retrieval performance.

## Keywords

XML Search & Retrieval, eXist, Lucy, INEX

## 1. INTRODUCTION

During INEX 2002, different participants used different approaches to XML retrieval. These approaches were classified into three categories [1]: extending well known full-text *information retrieval* (IR) models to handle XML retrieval; extending *database management systems* to deal with XML data; and *XML-specific*, which use native XML databases that usually incorporate existing XML standards (such as XPath, XSL or XQuery). Our modular system utilises a combined approach using traditional information retrieval features with well-known XML technologies found in most native XML databases.

Lucy<sup>1</sup> is RMIT's fast and scalable open source full-text search engine. Lucy follows the content-based information retrieval approach and supports Boolean, ranked and phrase queries. However, Lucy's smallest unit of retrieval is a whole document, thus ignoring the structure specified using the document schema as in the XML retrieval approach. Indeed,

<sup>1</sup><http://www.seg.rmit.edu.au/lucy/>

when dealing with information retrieval from a large XML document collection, sections that belong to a document, or even smaller document components such as paragraphs, may be regarded as appropriate units of retrieval. Accordingly, it is important to have an IR-oriented XML retrieval system that will be able to identify and rank these units of retrieval.

eXist<sup>2</sup>, an open source XML database, follows the XML-specific retrieval approach. It is the XML-specific approach that deals with both the content and the structure of underlying XML documents and incorporates keyword, Boolean and proximity search. Most of the retrieval systems that follow this approach use databases specifically built for XML. These databases are often called *native XML databases*. However, most of these systems do not support any kind of ranking of the final answers, which suggests a need of applying an appropriate retrieval strategy to determine the relevance of the answers to a given retrieval topic.

The XML retrieval approach we consider at INEX 2003 is that for many retrieval topics, one way of obtaining satisfactory answers is to use either *proximity* or *phrase search* support in XML retrieval systems. That is, a final answer is likely to be relevant if it contains (almost) all of the query terms, preferably in a desired order. The native XML databases, as explained above, provide all the required support to enable this functionality. However, when a native XML database needs to load and index a large XML collection, the time required to extract the most relevant answers for a given query is likely to increase significantly. Moreover, the XML database needs to determine a way to somehow assign relevance values to the final answers. Accordingly, it would be more efficient if the XML database has to index and search a smaller set of XML documents that may have previously been determined *relevant* for a particular retrieval topic. The database would then need to decide upon the most effective strategy for extracting and ranking the final answers. We have therefore decided to build a system that uses a combined IR/XML-specific retrieval approach. Our modular system effectively utilises Lucy's integrated ranking mechanism with eXist's powerful keyword search extensions. The INEX results show that our system produces effective XML retrieval for the content-and-structure (CAS) INEX topics.

<sup>2</sup><http://exist-db.org/>

---

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd">
<inex_topic topic_id="117" query_type="CO" ct_no="98">

<title>
Patricia Tries
</title>

<description>
Find documents/elements that describe
Patricia tries and their use.
</description>

<narrative>
To be relevant, a document/element
must deal with the use of Patricia Tries
for text search. Description of the standard
algorithm, optimised implementation and use
in Information retrieval applications are all
relevant.
</narrative>

<keywords>
Patricia tries, tries, text search,
string search algorithm,
string pattern matching
</keywords>

</inex_topic>

```

---

Figure 1: INEX Topic 117

## 2. INEX TOPICS

As in the previous year, INEX 2003 has used the same set of XML documents that comprises 12107 IEEE Computer Society articles published within the period 1997-2002 and stored in XML format. INEX 2003 also introduced a new set of ad-hoc retrieval topics which in contrast to the previous year were differently formulated. Revised relevance dimensions, *exhaustivity* and *specificity*, for assessing the relevance of the retrieval topics were also introduced.

Two types of XML retrieval topics are explored in INEX: content-only (CO) topics and content-and-structure (CAS) topics. A CO topic does not refer to the existing document structure. When dealing with CO topics, an XML retrieval system should follow certain rules that will influence the size and the granularity of a resulting document component. Not every document component can be regarded as a meaningful answer for a given query. Some of them are too short to act as meaningful answers while some of them are too broad. Thus, if an XML retrieval system shows poor performance (in terms of its effectiveness), the rules that decide upon the answer size and granularity should be changed accordingly.

A CAS topic, unlike a CO topic, enforces restrictions with respect to the existing document structure by explicitly specifying the type of the unit of retrieval (section, paragraph, or other). When dealing with CAS topics, an XML retrieval system should (in most cases) follow the structural

constraints described in the topic, which will result in answers having the desired (or similar) structure. In this case, the size and the granularity of a final answer are determined in advance.

The rest of this section describes INEX topics 117 and 86, which are respectively the CO and CAS topics proposed and assessed by our group. Some issues were observed during our relevance assessments for these topics. Our final results at INEX 2003 show that these issues, when addressed correctly, significantly improve the performance of an XML retrieval system. We also discuss the implications of these INEX topics for using the combined Lucy/eXist retrieval system and report other comments and suggestions.

### 2.1 INEX Topic 117

Figure 1 shows the INEX CO topic 117. This topic searches for documents or document components focusing on algorithms that use Patricia tries for text search. A document or document component is considered relevant if it provides description of the standard/optimised algorithm implementation or discusses its usage in information retrieval applications.

Our first observation is that this topic (unintentionally) turned out to be a difficult one, since:

- *Patricia* (usually) represents a person's first name, rather than a data structure;
- *tries* is a verbal form, and
- keywords like *text*, *string*, and *search* appear almost everywhere in the INEX IEEE XML document collection.

The relevance assessments were long and difficult, mainly because there were too many answers (due to *Patricia* and *tries*), there were not many highly relevant answers, and the few somewhat relevant answers were hard to evaluate consistently both for exhaustivity and specificity.

For this and similar topics, it appears that the only way to obtain satisfactory results is to use either *proximity* operators or *phrase search* support in full text retrieval systems. In the context of XML, an interesting question is whether the granularity of XML document components can be used as the proximity constraint. For example, it is more likely that paragraphs containing few of the query keywords will be regarded more relevant than a document that contains all keywords in different sections. On the other side, since users expect meaningful answers for their queries, the answers are expected to be rather broad, so retrieved document components should at least constitute a section, possibly a whole document. Accordingly, an XML retrieval system should follow an effective *extraction strategy* capable of producing more relevant answers.

### 2.2 INEX Topic 86

Figure 2 shows the INEX CAS topic 86. This topic searches for document components (sections) focusing on electronic

---

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd">
<inex_topic topic_id="86" query_type="CAS" ct_no="107">

<title>
//sec[about(.,'mobile electronic payment system')]
</title>

<description>
Find sections that describe technologies
for wireless mobile electronic payment systems
at consumer level.
</description>

<narrative>
To be relevant, a section must describe
security-related technologies that exist
in electronic payment systems that can be
implemented in hardware devices.
The main interests are systems that can be
used by mobile or handheld devices.
A section should be considered irrelevant
if it describes systems that are designed
to be used in a PC or laptop.
</narrative>

<keywords>
mobile, electronic payment system,
electronic wallets, e-payment, e-cash,
wireless, m-commerce, security
</keywords>

</inex_topic>

```

---

**Figure 2: INEX Topic 86**

payment technologies implemented in mobile computing devices, such as mobile phones or handheld devices. A section will be considered highly relevant if it describes technologies that can be used to securely process electronic payments in the mobile computing devices.

In order to consistently assess the relevance of the resulting document components (for this topic, most of these components were sections), two assessment rules were applied: document components focusing *only* on mobile computing devices were considered irrelevant, and document components focusing on security issues *in general* were also considered irrelevant.

It is evident from the above rules that for a document component to be considered marginally, fairly or highly relevant, it should *at least* contain a combination of some important words or phrases, such as *mobile*, *security*, *electronic payment system*, *e-payment*, and so on. In this sense, the issues encountered while assessing INEX CAS topic 86 were very similar with the ones discussed earlier for INEX CO topic 117. The only difference is that for this topic, the unit of retrieval is known in advance (<sec> identifies the type of document component to be retrieved), although by no

means this should be regarded as a mandatory constraint, since the INEX DTD specifies different types of document components that may be regarded as sections (such as *sec*, *ss1*, or *ss2*). It is therefore reasonable to expect that the extraction strategy previously applied to the CO topics would lead to more effective results for the CAS topics. The final INEX results for the CAS topics shown later in Figure 5 confirm this expectation.

### 2.3 Implications of INEX topics

It is evident from the previous observations that using either Lucy or eXist will partially satisfy the information need expressed with both the CO and the CAS topics. Lucy supports phrase search and ranking, however proximity support is limited, and the unit of retrieval is a whole document. eXist supports proximity operators and phrase search, and additionally allows final answers containing any of the query terms. However, it does not rank the final answers, and unless explicitly specified in the query, it does not impose additional constraints on the granularities of the returned answers. We identify later that this missing feature represents a serious system limitation for the CO topics. Accordingly, we decided to take into account the positive aspects of both systems and build a modular system that incorporates a combined approach to XML retrieval. Section 3 describes our approach in detail.

### 2.4 Other comments and suggestions

As a result of our active INEX participation this year, particularly while creating the INEX topics 86 and 117 and assessing the relevance of corresponding documents and document components, we observed some additional issues.

- In proposing a retrieval topic, should a participant make a statement about what XML retrieval feature he/she is trying to evaluate?
- Should the INEX initiative start making a classification of these various features? The features that we refer here might include, for example, usefulness of existing links and references in XML documents, proximity search, selection criteria, granularity of answers, and so on.

Although the INEX 2003 assessment tool was much better than the one used in 2002, the assessment task is still very time consuming. We suggest whether less answers could be pooled for assessment and whether the assessment tool could be furthermore improved to reduce some interaction required by users. The last suggestion might for example include less required “clicks” and the ability to select a group of answers as irrelevant (regardless whether they represent documents or document components).

## 3. MODULAR SYSTEM ARCHITECTURE

For INEX 2003, we decided to build a modular system that uses a combined approach to XML retrieval, comprising two modules: the Lucy full-text search engine and the eXist native XML database. Before we explain our approach in detail, we briefly summarise the most important features of both modules.

### 3.1 Lucy search engine

Lucy is a compact and fast text search engine designed and written by the Search Engine Group at RMIT University. Although Lucy primarily allows users to index and search HTML<sup>3</sup> (or TREC<sup>4</sup>) collections, we have successfully managed to index and search the entire INEX IEEE collection of XML documents. However, Lucy's primary unit of retrieval is a whole document and currently it is not capable of indexing particular document components, such as `<author>`, `<sec>`, and `<p>`. Lucy has been designed for simplicity as well as speed and flexibility, and its primary feature, which is also evident in our case, is the ability to handle a large amount of text. It implements an inverted index structure, a search structure well researched and implemented in many existing information retrieval systems. Witten et al. [8] provide a detailed explanation for efficient construction of an inverted index structure such as implemented in Lucy.

Lucy is a fast and scalable search engine, and incorporates some important features such as support for Boolean, ranked and phrase querying, a modular C language API for inclusion in other projects and native support for TREC experiments. It has been developed and tested under the Linux operating system on an Intel-based platform, and is licensed under the GNU Public License.

### 3.2 eXist: a native XML database

Since January 2001, when eXist [3] started as an open source project, developers are actively using this software for various purposes and in different application scenarios. We use eXist as a central part of our modular XML retrieval system. eXist incorporates most of the basic and advanced native XML database features, such as full and partial keyword text searches, search patterns based on regular expressions, query terms proximity functions and similar features. Two of eXist's unique features are efficient index-based query processing and XPath extensions for full-text search.

*Index-based query processing.* For the purpose of evaluating XPath expressions in user queries, conventional native XML database systems generally implement top-down or bottom-up traversals of the XML document tree. However, these approaches are memory-intensive, resulting in slow query processing. In order to decrease the time needed for processing the queries, eXist uses an inverted index structure that incorporates numerical indexing scheme for identifying the XML nodes in the index. This feature enables eXist's query engine to use fast path join algorithms for evaluating XPath expressions. Meier [3] provides detailed technical explanation of this efficient index-based query processing implementation in eXist.

*XPath extensions for full-text searching.* Standard XPath implementations do not provide very good support for querying document-centric XML documents. Document-centric documents, as oppose to data-centric ones that usually contain machine-readable data, typically include mixed content and longer sections of text. eXist implements a number of XPath extensions to efficiently support document-centric queries, which overcome the inability of standard XPath

functions (such as `contains()`) to produce satisfactory results. For example, the `&=` operator selects document components containing *all* of the space-separated terms on the right-hand side of the argument. `|=` operator is similar, except it selects document components containing *any* of the query terms. In the next section we provide examples of the way we used these operators in the INEX topic translation phase.

eXist is a lightweight database, completely written in Java and may be easily deployed in several ways. It may run either as a stand-alone server process, or inside a servlet-engine, or may be directly embedded into an existing application.

### 3.3 A combined approach to XML retrieval

Section 2 observes the implications of the INEX topics that influenced our choice for a combined approach to XML retrieval. However, due to the advanced retrieval features described previously it becomes evident that using eXist alone should suffice in satisfying the XML retrieval needs. Indeed, some applications have shown that eXist is already able to address real industrial needs [3]. Despite all these advantages, we were not able to use eXist as the *only* XML retrieval system for two main reasons: first, we were using eXist version 0.9.1, which did not manage to load and index the entire IEEE XML document collection needed for INEX, and second, although we could retrieve relevant pieces of information from parts of the IEEE document collection, eXist does not assign relevance values to the retrieved answers. Accordingly, since *ranking* of the retrieved answers is not supported, we decided to undertake a combined XML retrieval approach that utilises different *extraction strategies* to rank the answers. With respect to a specific extraction strategy, a document component may represent a highly ranked answer if it belongs to a document that has previously been determined relevant for a particular retrieval topic.

Figure 3 shows our combined approach to XML retrieval. The system has a modular architecture, comprising two modules: Lucy and eXist. We use INEX topic 86, as shown in Figure 2, to explain the flow of events.

First, the INEX topic is translated into corresponding queries understandable by Lucy and eXist, respectively. Depending on the type of the retrieval topic (CO or CAS), the topic translation utility follows different rules. For the INEX CO topics, such as topic 117 shown in Figure 1, queries that are sent to both Lucy and eXist include only terms that appear in the `<Keywords>` part of the INEX topics. For the INEX CAS topics, as shown in Figure 3, query terms that appear in both `<Title>` and `<Keywords>` parts of the INEX topics were used.

For example, we use the query terms from the `<Keywords>` part of the INEX topic 86 to formulate the Lucy query:

```
.listdoc
'mobile "electronic payment system"
"electronic wallets" e-payment e-cash wireless
m-commerce security'
```

<sup>3</sup><http://www.w3.org/MarkUp/>

<sup>4</sup><http://trec.nist.gov/>

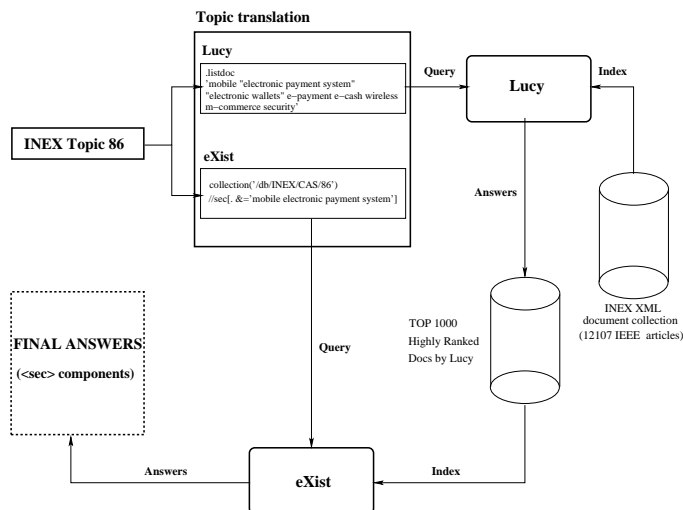


Figure 3: A modular system architecture.

However, before submitting a query to the system, the INEX document collection needs to be indexed. We use Lucy to create an inverted index from all the documents in the large IEEE XML collection. We then search this indexed data by entering the queries derived from the translation rules, as explained above. For the purpose of ranking its answers for a given query, Lucy uses a variant of the Okapi BM25 [5] probabilistic ranking formula. Okapi BM25 is one of the most widely used ranking formula in information retrieval systems. It is thus expected that, for a given INEX topic, Lucy will be able to retrieve highly relevant XML documents early in the ranking. Therefore, for each INEX topic, we retrieve (up to) 1000 highest ranked XML documents by Lucy. It is our belief that the information contained in these documents is sufficient to satisfy the information need expressed in the corresponding INEX topic. However, at this phase of development, Lucy's only unit of retrieval is a whole document. Accordingly, for a particular INEX topic, we still have to extract the relevant parts of these highly ranked documents. Wilkinson [7] shows that simply extracting components from highly relevant documents leads to poor system performance. Indeed, there may be cases when a section belonging to highly ranked document is irrelevant as opposed to a relevant section belonging to lowly ranked document. However, we believe that the retrieval performance of a given system may be improved using a suitable *extraction strategy*. We implemented several extraction strategies using eXist's XPath extensions. We provide examples how we use these XPath extensions while translating INEX topic 86 as follows.

For INEX CAS topics in general, and INEX topic 86 in particular, the terms that appear in the <Title> part are used to formulate eXist queries. However, since a document component is likely to be relevant if it contains *all* or *most* of the query terms that appear in the <Title>, we undertake several extraction strategies while implementing our INEX runs. The extraction strategies are described in detail in Section 4, where we also explain how we constructed our INEX runs. In general, these strategies depend on the combined usage of Boolean AND and OR operators, identified by the &=

and |= operators in eXist, respectively. In that sense, the INEX topic 86 may be translated either as:

```
collection('/db/INEX/CAS/86')
//sec[. &='mobile electronic payment system']
```

if one wants *all* query terms to appear in the resulting section, or:

```
collection('/db/INEX/CAS/86')
//sec[. |= 'mobile electronic payment system']
```

if one wants *any* of the query term to appear in the resulting section.

We follow the first translation rule for our example in Figure 3. Final answers will thus constitute <sec> document components (if any) that contain all the query terms. By following this rule, we reasonably expect these document components to represent relevant answers for the INEX topic 86. On the other hand, it is clear that if the second translation rule is applied for the same topic, it may produce very many irrelevant answers as well as some further relevant answers. Accordingly, it is very important to decide upon the extraction strategy that will yield in highly relevant answers for a given INEX topic. We discuss the results for different extraction strategies in the following section.

#### 4. INEX RUNS AND RESULTS

The retrieval task performed by the participating groups in INEX 2003 was defined as ad-hoc retrieval of XML documents. In information retrieval literature this type of retrieval involves searching a static set of documents using a new set of topics, which represents an activity very commonly used in library systems.

Within the ad-hoc retrieval task, INEX 2003 defines additional sub-tasks. These represent a CO sub-task, which involves content-only (CO) topics and a CAS sub-task, which involves content-and-structure (CAS) topics. The CAS sub-task comprises a SCAS sub-task and a VCAS sub-task. The SCAS sub-task requests that the structural constraints in a query must be *strictly* matched, while VCAS allows the structural constraints in a query to be treated as *vague* conditions.

For each topic belonging to a particular sub-task up to 1500 answers (full documents or document components) were required to be retrieved by the participating groups. In order to assess the relevance of the retrieved answers, the revised relevance dimensions (exhaustivity and specificity) need to be quantized in a single relevance value. INEX uses two quantization functions: *strict* and *generalised*. The strict function can be used to evaluate whether a given retrieval method is capable of retrieving highly relevant and highly focused document components, while the generalised function credits document components according to their *degree of relevance* (by combining the two relevance dimensions, exhaustivity and specificity).

Our group submitted 6 official runs to INEX 2003, 3 for each CO and SCAS sub-task, respectively. Figures 4 and 5 show

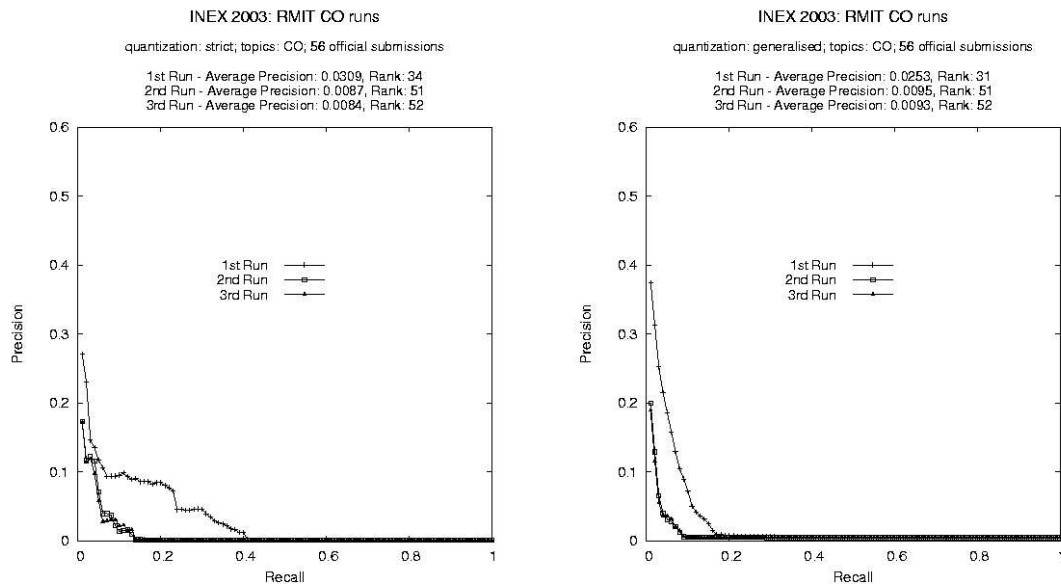


Figure 4: Results for the RMIT CO runs using both strict and generalised quantization functions

the results for both the CO and SCAS runs when both strict and generalised quantization functions are used. The rankings of the runs are determined according to the average precision over 100 recall points considering each corresponding INEX topic. Two of our three runs for each sub-task were automatically constructed while one was manually. The automatic runs were constructed using the translation rules explained in the previous section. We manually constructed the other runs in order to produce more meaningful queries for each INEX topic. Each run was constructed by using elements in the following answer lists: [A] that uses eXist’s  $\&=$  (logical AND) operator and enforces strict satisfaction of logical query conditions (the elements that belong to the answer list [A] will therefore represent document components containing *all* the query terms or phrases); [B] that uses the  $|=$  (logical OR) operator, “relaxes” the query conditions and allows for document components containing *any* of the query terms or phrases; and a combined answer list that contains the elements in the answer list [A] followed by the elements in the answer list [B–A].

Three retrieval runs were submitted for the CO sub-task. We constructed the first CO run by retrieving the 1500 highest ranked documents for each INEX topic. As described in the previous section, the `<Keywords>` part of each INEX topic was automatically translated as an input query to the Lucy search engine. The final rank of a document was then determined by its similarity with the given query as calculated by Lucy using a variant of Okapi BM25. As shown in Figure 4 this run performed better than the other two CO runs in both cases when strict and generalised quantization functions are used, which suggests that a whole document is often likely to be considered a preferable answer for an INEX CO topic.

For the other two runs, for each INEX CO topic we first used Lucy to extract (up to) the 1000 highest ranked documents. Then we used eXist to index and retrieve the fi-

nal answers from these documents. We reasonably expected that the most relevant document components required to be retrieved for each INEX topic were very likely to appear within the 1000 highest ranked documents. Since the CO topics do not impose constraints over the structure of resulting documents or document components, we used the `//**` eXist construct in our queries. The “\*\*” operator in eXist uses a heuristic that retrieves answers with different sizes and granularities. For our second CO run, the `<Keywords>` part of each topic was automatically translated as an input query to the eXist database, and its final answer list includes only elements from the answer list [B]. We used the manual translation process for our third run, where the final answer list includes the elements in the answer list [A] followed by the elements in the answer list [B–A]. Although we expected the third run to perform better than the second, Figure 4 shows that both these runs performed poorly in both cases when strict and generalised quantization functions are used, regardless of choices for the translation method and the extraction strategy. At this phase of development, the heuristic implemented in the “\*\*” operator in eXist is not able to determine the most meaningful units of retrieval nor influence the desired answer granularity for a particular CO topic. Next we show that this is not the case for the CAS topics, where the type of the unit of retrieval is determined in advance and the choices for the translation method and the extraction strategy have a significant impact on the system’s performance.

Three runs were submitted for the SCAS sub-task. As discussed previously, both `<Keywords>` and `<Title>` parts from INEX CAS topics were used to generate the input queries for Lucy and eXist, respectively. Our first SCAS run was automatic and its final answer list includes the elements in the answer list [A] followed by the elements in the answer list [B–A]. The queries for the second SCAS run were manually constructed and its final answer list includes the elements from the same answer lists as for the first run.

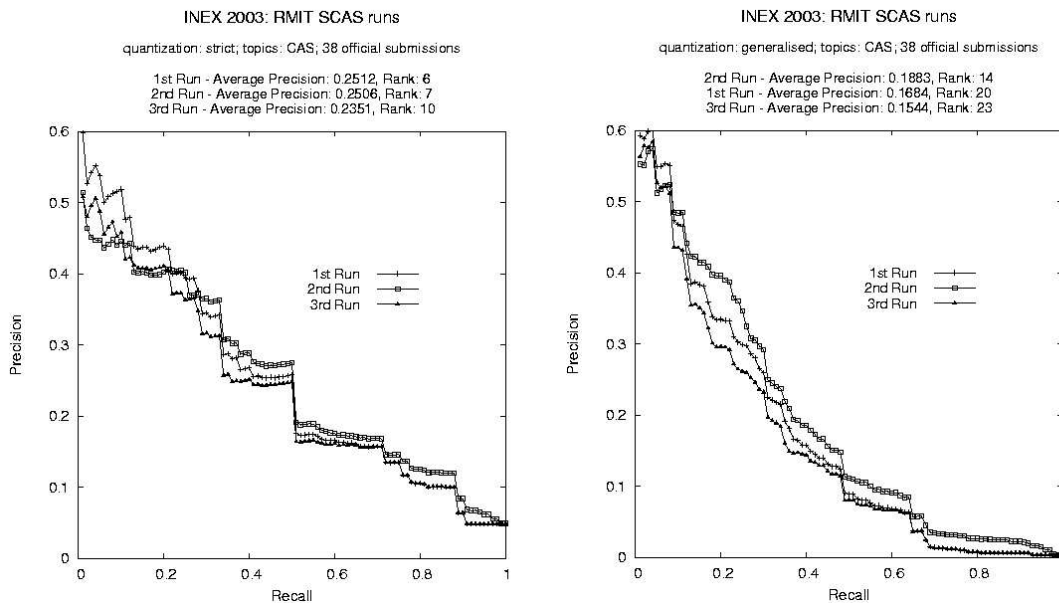


Figure 5: Results for the RMIT SCAS runs using both strict and generalised quantization functions

Figure 5 shows that these runs performed relatively better when using a strict quantization function compared with the runs from other participating groups at INEX 2003. Since the type of the unit of retrieval is determined in advance for the SCAS runs, the choice of the extraction strategy implemented in both runs appears to be very effective for retrieving highly exhaustive and highly specific document components. It can be observed that our system performs slightly more effective for the first than for the second run (6th compared to 7th out of 38 systems), and the first run performs better for recall values lower than 0.2. However, the choice of the translation method has an effect on the system's performance for recall values greater than 0.3, where the second run performs better than the first run. Figure 5 also shows that the choice of the extraction strategy is not as effective when using a generalised quantization function, where marginally/fairly exhaustive or marginally/fairly specific document components are regarded as partly relevant answers. Indeed, the ranks for both runs when evaluated using the generalised quantization function are not among the ten highest ranked INEX runs. In this case, the choice of the translation method results in second run performing better than the first run overall.

The third SCAS run was automatic, however its final answer list includes only the elements from the answer list [B]. By choosing this strategy we reasonably expected some irrelevant answers in the final answer list, but we hoped to find more relevant components in highly ranked documents. Indeed, as Figure 5 shows, irrespective of whether a strict or a generalised quantization function is used, our retrieval system is ranked lower for the third SCAS run compared to the previous two runs.

## 5. LIMITATIONS OF OUR SYSTEM

Previous sections describe the XML retrieval approach that we implemented while participating in INEX 2003. How-

ever, during different phases of our INEX involvement, particularly while constructing the INEX runs and assessing the relevance of retrieved results, we observed several system limitations. Although they can and should be considered as a weakness of our approach, the fact that we are able to identify them influences our future research directions. Some of these limitations include the following.

*No IR ranking of the final answers.* The choice of implementing an extraction strategy that may influence the rank of a final answer suggests that our system does not consider an IR ranking score for a particular answer. Although for a given INEX topic Lucy ranks the XML documents in a descending order of their query similarity, the unit of retrieval represents a whole document, and there is no support for existing XML technologies. eXist, on the other hand, has a tight integration with existing XML development tools and technologies, but does not rank the final answers according to their query similarity. We have thus decided that a particular extraction strategy should influence the final ranking score for a resulting document or document component. We have decided upon different extraction strategies while we constructed our INEX runs, and have shown that for the CAS topics some of them have a significant impact on the retrieval performance of our modular system.

*Complex usage.* Since our system has a modular architecture that incorporates a combined IR/XML-specific oriented approach to XML retrieval, its usage is very complex. It comprises two different retrieval modules (Lucy and eXist), each having different internal architectures and rules of use. Instead, it would be preferable to have only one system that incorporates the best features from the above modules.

*Significant space overhead.* The size of the INEX IEEE XML document collection takes around 500MB disk space. The inverted index file maintained by Lucy additionally takes

20% of that space. For each topic, (up to) 1000 XML documents are indexed by eXist, which adds up to approximately 12% of the space for the INEX collection. Although both Lucy and eXist implement efficient retrieval approaches, it becomes evident that their combination leads to significant disk space overhead. As for the previous limitation, one system that can deal with the above issues would also be preferable.

## 6. RELATED WORK

Even before INEX, the need for information retrieval from XML document collections had been identified in the XML research community. As large XML document collections become available on the Web and elsewhere, there is a real need for having an XML retrieval system that will efficiently and effectively retrieve information residing in these collections. This retrieval system will need to utilise some form of an XML-search query language in order to meet the growing user demand for information retrieval. Thus, the needs and requirements for such a query language have to be carefully identified and appropriately addressed [4].

At INEX 2002 the CSIRO group proposed a similar approach to XML retrieval. Their XML retrieval system uses a combination of a selection and a post-processing module. Queries are sent to PADRE, the core of CSIRO's Panoptic Enterprise Search Engine<sup>5</sup>, which then ranks the documents and document components on the basis of their query similarity. In contrast to Lucy, whose primary unit of retrieval is a whole document, PADRE combines full-text and metadata indexing and retrieval and is capable of indexing particular document components, such as <author>, <sec> and <p>. Different "mapping rules" determine what metadata field is used to index the content of a particular document component. A post processing module was then used to extract and re-rank the final answers from documents and document components returned by PADRE [6].

In an effort to reduce the number of document components in an XML document that may represent possible answers for a given query, Hatano et al. [2] propose a method for determining the preferable units of retrieval from XML documents. We consider investigating these and similar methods for improving the effectiveness of our system for the CO topics.

## 7. CONCLUSION AND FUTURE WORK

We have described the combined approach to XML retrieval that we used during our participation in INEX 2003. Our retrieval system implements a modular architecture, comprising two modules: Lucy and eXist. For each INEX topic, we used Lucy, a full-text search engine designed by the Search Engine Group at RMIT, to index the IEEE XML document collection and retrieve the top 1000 highly ranked XML documents. We then indexed those documents with eXist, and implemented different topic translation methods and extraction strategies in our INEX runs. The INEX results show that these methods and strategies result in an effective XML retrieval for the CAS topics. Since our system is not yet able to identify the preferred granularities for the final answers, the methods and strategies are not as effective for the CO

topics. Further investigations need to be done in order to improve this functionality.

We have also observed several limitations of our modular system. In order to overcome these limitations, we intend to investigate more effective ways to use and combine the most advanced features of Lucy and eXist. It is our belief that they will result in more accurate and interactive XML retrieval.

## Acknowledgements

We would like to thank Wolfgang Meier for providing assistance with using eXist and to Falk Scholer, Nick Lester, Hugh Williams and other members of the Search Engine Group at RMIT for their useful suggestions and support with using Lucy.

## 8. REFERENCES

- [1] N. Govert and G. Kazai. Overview of the Initiative for the Evaluation of XML retrieval (INEX) 2002. In *Proceedings of the First Initiative for the Evaluation of XML Retrieval (INEX) Workshop*, Dagstuhl, Germany, December 2002.
- [2] K. Hatano, H. Kinutani, M. Watanabe, M. Yoshikawa, and S. Uemura. Determining the Unit of Retrieval Results for XML Documents. In *Proceedings of the First Initiative for the Evaluation of XML Retrieval (INEX) Workshop*, Dagstuhl, Germany, December 2002.
- [3] W. Meier. eXist: An Open Source Native XML Database. In *A. B. Chaudri, M. Jeckle, E. Rahm, R. Unland (editors): Web, Web-Services, and Database Systems. NODe 2002 Web- and Database-Related Workshops*, Erfurt, Germany, October 2002.
- [4] J. Pehcevski, J. Thom, and A.-M. Vercoustre. XML-Search Query Language: Needs and Requirements. In *Proceedings of the AUSWeb 2003 conference*, Gold Coast, Australia, July 2002. <http://ausweb.scu.edu.au/aw03/papers/thom/>.
- [5] S. Robertson and S. Walker. Okapi at TREC-8. *NIST Special Publication 500-246: The Eight Text RETrieval Conference (TREC-8)*, November 1999.
- [6] A.-M. Vercoustre, J. A. Thom, A. Krumpholz, I. Mathieson, P. Wilkins, M. Wu, N. Craswell, and D. Hawking. CSIRO INEX experiments: XML Search using PADRE. In *Proceedings of the First Initiative for the Evaluation of XML Retrieval (INEX) Workshop*, Dagstuhl, Germany, December 2002.
- [7] R. Wilkinson. Effective Retrieval of Structured Documents. In *W.B. Croft and C.J. van Rijsbergen (editors): Proceedings of the 17th Annual International Conference on Research and Development in Information Retrieval*, Dublin, Ireland, July 1994.
- [8] I. Witten, A. Moffat, and T.C.Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann Publishers, Los Altos, CA 94022, USA, 2nd edition, 1999.

<sup>5</sup><http://www.panopticsearch.com>