XML Document Retrieval with PADRE

Nick Craswell[1] *David Hawking*[1] *Alexander Krumpholz*[2] *Ian Mathieson*[2]

James A. Thom[3] *Anne-Marie Vercoustre*[2] *Peter Wilkins*[2]

Mingfang Wu[2]

[1]CSIRO Mathematical and Information Sciences
GPO Box 664, Canberra, ACT 2601, Australia

[2]CSIRO Mathematical and Information Sciences
Private Bag 10, South Clayton MDC, VIC 3169, Australia

[3]School of Computer Science and Information Technology, RMIT University
GPO Box 2476V, Melbourne 3001, Australia

Email for correspondence: *Anne-Marie.Vercoustre@csiro.au*

Abstract

One paradigm of XML retrieval is database-style querying of semi-structured data. Another paradigm is based on information retrieval involving ranking of documents or document fragments. The INEX project attempts to integrate these paradigms and provide an environment for conducting retrieval experiments on semi-structured data. This paper discusses our participation in the INEX project and what we discovered about combining these paradigms.

Keywords Document Databases, Document Standards, Information Retrieval

1 Introduction

Managing semi-structured data, such as collections of XML documents, involves the use of both database and text retrieval technology. As the volume and complexity of semi-structured data available to users increases better retrieval mechanisms are required to support users' increasingly sophisticated information needs.

XML retrieval falls into two broad approaches: database-style querying of XML semi-structured data (supported by query languages such as XQuery) and text retrieval of marked-up documents and portions of documents using text-based search engines. In a recent survey of indexing and searching XML Documents [7] several approaches for combining structural information with ranking are described. The two communities have also had a very different approach to evaluation. The database approach is interested in exact matches to the query and concentrates more on the power of expression of the query language and efficient query interpretation. The information retrieval approach is focusing on effective ranking of the answers through precision/recall measures, in relation to user evaluation of answer quality. Some retrieval languages such as XIRQL [4] attempt to bridge the divide between the database and information retrieval approaches, however there has been no easy way to evaluate such languages until now. The experiment devised for INEX (Initiative for Evaluation of XML retrieval) [3] attempts (with only partial success) to integrate these two very different approaches. The aim of the INEX initiative is to provide the means, in the form of a large testbed (test collection and queries) and appropriate scoring methods for the evaluation of XML documents retrieval. The queries include two different sets of queries: content-based queries and content-and-structure-based queries. In both cases, the scoring method takes into account the granularity of the answer.

This paper reports our participation in the INEX project. In Section 2 we introduce the INEX project. In Section 3 we present our approach and techniques for addressing the INEX retrieval tasks. We analyse and discuss some current limitations of our approach in Section 4. Section 5 presents some related work.

2 INEX - XML document search

The INEX [3] *Initiative for the Evaluation of XML retrieval* is organized by the European DELOS Network of Excellence for Digital Libraries. INEX

Proceedings of the 7th Australasian Document Computing Symposium,
Sydney, Australia, December 16, 2002.

involves nearly 50 groups from around the world undertaking a series of retrieval tasks (topics) on a collection of XML documents (XML sources for about 12,000 articles in IEEE Computer Society publications since 1995).

```
<?xml version="1.0"
    encoding="ISO-8859-1"?>
<!DOCTYPE INEX-Topic SYSTEM
    "inex-topics.dtd">
<INEX-Topic topic-id="17"
    query-type="CAS"
    ct-no="088">
  <Title>
    <te>bb</te>
    <cw>W. Bruce Croft</cw>
    <ce>bb/au</ce>
    <cw>not(W. Bruce Croft)</cw>
    <ce>fm/au</ce>
  </Title>
  <Description>
    Retrieve bibliographic references
    for works by W. Bruce Croft where
    Croft is not the author of the
    paper containing the reference.
  </Description>
  <Narrative>
    Relevant items will consist of a
    set of article doi's and the bb
    bibliographic entries from that
    article where one of the authors
    is W. Bruce Croft. These should be
    extracted only from articles where
    Croft is NOT one of the authors of
    the article.
  </Narrative>
  <Keywords>
    W. Bruce Croft, W.B. Croft
  </Keywords>
</INEX-Topic>
```

Figure 1: INEX topic 17

Each group submitted suggestions for topics, and altogether 60 topics were selected from those submitted by the participating groups. Figure 1 is an example of such a topic. Of these 30 topics only specified *content words* using the <cw> elements within the <Title>; these correspond to the query terms in a conventional information retrieval query. More constrained were 30 topics with structural aspects as defined by either or both *target element* (as specified by the <te> tag) and *content element* (<ce>) constraints within the <Title>.

The <te> element specifies a target element (or elements) for the topic; in the example shown in Figure 1 the target is the element <bb> — each element contains one bibliographic entry from the references listed at the end of an article. For topics where no target was specified the retrieval system should return an element or elements from matching documents that most closely meet the information need expressed in the query.

The <ce> elements specify that the content words (<cw>) are contained in particular elements; in the example there are two pairs of such constraints — the first requires that Bruce Croft is one of the authors in the bibliographic element, the second requires that Bruce Croft is not an author of the article containing the reference. The <cw> elements can appear without a <ce> constraint, in which case the content words might appear anywhere within the answer element.

For each topic, the participating groups returned (up to) the top 100 answer elements as ranked highest by their system. The answers from all groups were then combined into a pool of between one or two thousand answer elements for each topic that needed to be assessed. Relevance assessments in INEX are undertaken by the participating group who had (usually) formulated the original topic. Our group assessed three of the 60 topics. These assessments involved four-scales of relevance (from *irrelevant* to *highly relevant*) and four degrees of document coverage (*no coverage* to *exact coverage*) for assessing both answer elements as well as surrounding and component elements.

3 System used for INEX experiments

Our aim in participating in the INEX XML-search experiments was to gain experience with a large collection of XML documents and to identify the requirements for a system that can answer realistic queries from people as opposed to queries sent by applications over XML data repositories.

Our approach was to extend our in-house enterprise search engine, PADRE [6], to deal with XML documents and then analyse the results and potential limitations.

3.1 PADRE

PADRE [6] is the core of the Panoptic Enterprise Search Engine [2]. It combines full-text and metadata indexing and retrieval. After all query terms have been processed the matched documents are sorted, first on the basis of how many terms in the query match and then on the basis of the relevance score. The particular relevance formula used [6] is a slightly modified form of the Okapi BM25 function developed by Robertson et al. [8].

3.2 Architecture

As shown in Figure 2 the system includes two main components:

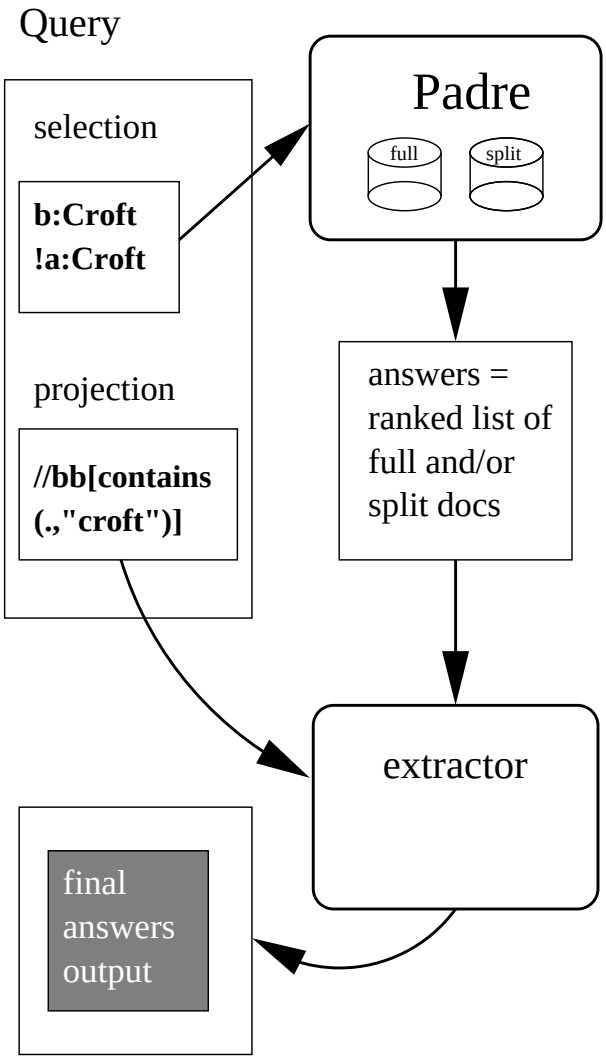

Figure 2: Architecture

- The PADRE search engine selects and ranks the more relevant documents or document fragments that fulfil the constraints (selection).
- The Extractor identifies the type of expected answers and extracts them from the previously returned fragments or documents (projection).

An interface takes the input query and sends the selection query to PADRE and the projection type to the extractor. It also generates the final results in the INEX format.

A separate translation program takes as input the INEX topics and generates a list of queries in a format that is recognised by our system.

3.3 Indexing

We used PADRE's capability for field-based indexing of metadata. For example the index terms for `<p>XML Search</p>` may be represented as `c:XML` and `c:Search`, where the prefix `c` specifies the element in which the information appears.

The fields that will be indexed are explicitly declared in a configuration file of mappings between a path and a given letter. Examples of mappings:

1. `//article/fm/au` → `a`
2. `//bb` → `b`
3. `//p` → `c`
4. `//p1` → `c`

Mapping (1) specifies that a word w occurring in an `<au>` element occurring within an `<fm>` element in an `<article>` will be indexed as `a:`w, while mapping (2) specifies that words occurring a `<bb>` element (for example authors in `//bb/au` in the bibliographic part) will be indexed as `b:`w. Mappings (3) and (4) specify that a word w occurring in a `<p>` or `<p1>` element will be indexed as `c:`w.

A query such as "give me the documents authored by Bruce Croft" can be expressed as: `a:Bruce a:Croft` or alternatively as `a:"Bruce Croft"`. The first query will return initially the documents that contain Bruce and Croft in one article's author element (4 fully matching for the IEEE collection), then documents that contain either word (106 partial matching). The second query will return only documents that contain both words (4 matching), in that order, and may then miss other occurrences such as Bruce W. Croft (not applicable here). The query could also be expressed `a:Bruce +a:Croft` to make the word Croft mandatory then returning only the 4 fully matching.

The above example shows that PADRE mappings will ignore unmapped sub-tags. For example:

```
<au sequence="first">
<fnm>W. Bruce</fnm>
<snm>Croft</snm>
</au>
```

The previous mappings will result in "Bruce" and "Croft" indexed as `a:Bruce, a:Croft`, ignoring the tags `<fnm>` and `<snm>`.

We limited the mappings to what we intuitively thought would be necessary for actual queries, given the DTD and the type of INEX topics.

3.4 Splitting the documents

As described in Figure 2, the system relies on PADRE to select the documents and rank them. Wilkinson [9] demonstrates that extracting elements, such as sections or paragraphs, from the ranked documents is the worst strategy for ranking those elements. So we decided to split the collection in documents fragments to study different strategies that could make better use of the initial PADRE ranking for the ranking of the

final answers. It was expected that it would result in good ranking for the answers to content-only queries, as well as improving the ranking for structured queries, although further extracting may be necessary to return the expected result elements (see next section).

After short analysis of the collection, we split the documents into XML elements corresponding to a small number of tags that take in account two aspects:

- a reasonable granularity of fragments; we did not split into elements as small as titles or bibliographic references; and
- the expected types of result elements, such as sections, paragraphs, front head, figures, table, etc.

To support further processing, some context information was added to the documents fragments. It includes:

- the name of the document this fragment was part of (for example, `/ex/1995/x6059.xml`);
- the actual path of the fragment in the full document (for example, `/article[1]/bdy[1]/sec[2]/p[3]`);
- the skeleton of the tree structure in which the element was embedded; and
- a DOCTYPE declaration corresponding to an extension of the initial DTD that made this fragment a valid document (this was achieved by making most elements optional in the initial DTD).

The two first types of information were the ones expected to be returned as INEX answer in case the fragment was selected. The last two were necessary to parse the fragment in case further processing was necessary (see extraction). Creating document fragments was an interesting experience in itself. Beside the problem of parsing fragments as mentioned above, there was the issue of dealing with entities and special characters. XML parsers interpret them correctly when reading a document but fail to output them back as entities when outputting a sub-tree.

After splitting, the resulting collection was made of the initial documents and all the fragments, making the number of documents increasing by a factor of 100, and the size (in byte) of the collection increasing by a factor of 10.

3.5 Extractor

The role of this module is to process each document (fragment) returned by PADRE and to check whether it is of the expected type(s) for the answer. If it is the case, then the fragment is the answer. If not, some processing is done to identify and extract the relevant elements within the fragment or its embedding document. We call this process the projection.

For example with topic 17, returned elements may be articles, sections or paragraphs, whilst the answers must be bibliographic references (bb). Since bibliographic references cannot be found in sections or paragraphs, the extractor must load the full article and extracts the bibliographic references to papers by Bruce Croft. The actual projection is defined as an XPath with possibly a predicate `contains`, such as `//bb[contains "Croft"]`.

More generally the algorithm works as follows (for a given query): If there is a projection defined, for each returned fragment f:

1. load the fragment, get the name of the embedding article, load the full article A.
2. apply the XPath projection to the article A; this return $e_1, e_2, \ldots e_n$ elements.
3. if $f = e_i$ for one e_i, return the XPath of f; end;
4. if not, calculate $f = father(f)$, if f is not nil go to step 3.
5. if there are e_i that are descendants of the original f, return all of those; end;
6. return the e_i (if any)

Figure 3 (a) and (b) give the XPaths that we use for defining the type of answer for topic 17. As you can see the constraints on `<bb>` elements are very coarse. First this is due to the query been translated automatically from the INEX topic. Second we were using XPath for a fast implementation and XPath does not support similarity measure. For some queries this resulted in returning many irrelevant fragments, and in any case unranked elements within the initial selected fragment. This will be discussed further in Section 4.

3.6 Query translator

Each topic was converted into a query containing a selection component and an optional projection component.

The selection component is translated from the `<ce>` and `<cw>` elements in the topic title and the projection component from the `<te>` element in the topic title. If there is more than one path specified in the `<ce>` element, a Cartesian product of the query terms and corresponding mappings is included in the query. For example, a topic with

```
<cw>Bruce Croft</cw>
<ce>fm/au|bb/au</ce>
```

would be translated as

```
a:Bruce a:Croft
b:Bruce b:Croft
```

We first convert target elements <te> to a standard XPath expression used in the projection component of the query. For example bb -> //bb. Then we use the mappings presented in Section 3.3 to translate this XPath into a Padre command (e.g. b:Bruce). However some elements such as /au may present problems, we could translate

```
au -> //au
```

but alternatively

```
au -> //article/fm/au
```

may be preferable so as to only get authors of the article. We tune this by ordering the mapping rules so the most useful rule matches first.

Examining the behaviour of manually constructed queries we also developed the following heuristics for the query translation process.

Where the selection involves dates, for example if the selection tag within the <ce> element is in the set {yr,pdt/yr,pdt} then we use metadata 'd' and convert <cw> to a numerical value. Also for dates we generate a strong (mandatory) selection condition. For example

```
<cw>1996 or 1997</cw>
<ce>yr</ce>
```

is translated to

```
+d>1995<1998
```

Compound phrases (especially in the keywords) should be searched using phrases. It is difficult to recognise all such phrases, however in some cases (for example in lists of keywords) where phrases are separated by a comma, we can easily identify them. We add the keywords (or phrases) as a list between brackets thus ranking documents with the keywords (or phrases) more highly. For example the phrase "information retrieval" should be translated to the additional restriction on the query ["Information Retrieval"].

When a projection is part of a selection, the same constraints as for the selection should be added to the projection. For example in topic 17, since there are both a selection and a projection on article/bm/bib/bibl/bb we can add the constraints to projection.

There were other cases where manual queries could be carefully crafted but we were unable to discern general translation rules, for example +b:croft (instead of b:croft) as there are plenty of other Bruces as well as Bruce Croft.

3.7 The experiments

We provided three runs for INEX. Each run involved converting the INEX topics into queries sent to PADRE/extractor.

queries on full articles: In this run the queries were generated automatically using the <Title> and <Keywords> from the topic, and only the full articles were searched.

queries on split articles: In this run the queries were generated automatically using the same components of the topic, but fragments as well as full articles were searched.

manual queries: In this run the queries were improved by hand and sent against the split collection.

By comparing the scores of the two first runs we want to evaluate the benefice on indexing fragments. By comparing the last two runs we expect to evaluate the importance of generating good queries, although the manual queries could have been further improved.

As an example, Figure 3 shows each query used for topic 17 in the three runs. Figure 3(a) shows the query generated for both the run on the full articles and the run on the split articles. Figure 3(b) shows the manually constructed query.

4 Analysis of our approach

The results for these runs are not available until the INEX workshop in December. So we are unable to comment yet on whether querying on the splitted collection was more effective than querying the on the full collection. However we are able to make some initial observations of our approach.

Mappings

As we explain above, PADRE allows for field-based indexing and we define the fields to index by mapping paths to a given field. The way it is currently implemented, those fields are regarded as independent and disjoint, and do not account for field hierarchy. The drawback is that any specific mapping will invalidate more generic queries. For example if a mapping is defined for //bb/au, it would be possible to query about one bibliographic reference's author (for example "Bruce Croft"), but not about bibliographic references (<bb>) that contains "Bruce Croft", because "Bruce" and "Croft" would not have been indexed in the field corresponding to "b". Since, XML documents are intrinsically hierarchical, we think that the PADRE strategy for tag indexing should be modified to accomodate this.

Another current limitation in our mapping is that it is not possible to map to attributes. Given

```
<query topic-id="17">
  <selection>
    b:W b:Bruce b:Croft
    a:!W a:!Bruce a:!Croft
    ["W Bruce Croft" "W B Croft"]
  </selection>
  <projection>
    //bb[contains(.,"W") or
         contains(.,"w") or
         contains(.,"Bruce") or
         contains(.,"bruce") or
         contains(.,"Croft") or
         contains(.,"croft")]
  </projection>
</query>
```

(a) query on (i) full and (ii) split documents

```
<query topic-id="17">
  <selection>
    b:"W. Croft" b:Bruce +b:Croft
    -a:"W. Bruce" !a:Croft -a:"W.B."
    ["W. Bruce Croft" "W. B. Croft"]
  </selection>
  <projection>
    //bb[contains(.,"Croft")]
  </projection>
</query>
```

(b) manually constructed query

Figure 3: Topic 17 — full, split and manual queries

the often interchangeable nature of attributes and elements in XML DTDs, it is certainly a limitation.

When the DTD is very large and flexible, it may be difficult to define proper mappings. For example if we want to distinguish between authors of an article in the collection and authors of articles referenced in the bibliographic part, we need two different mappings. However, author of articles can be found with various paths such as `article/fm/au`, `index/index-entry/au`, but also `body/au` or worst in any paragraph with a free syntax (for example "by Bruce Croft"). The last case is obviously beyond XML-search capability, but could be still part of user requirements forsearching XML documents.

Since our mappings are defined at indexing time, they must be carefully designed in anticipating the kind of queries that will be issued. Each time the mapping are redefined the entire collection must be reindexed.

Splitting the collection

Splitting the document collection into a large collection of fragments does not seem a long term option, even though it may prove to be successful. We have already mentioned the increased size of the resulting split collection and the processing involved. Another drawback is that words are indexed several times: for example, once in their original paragraph, once in their embedding section and once as part of the full paper. The consequence is that it introduces a bias in the frequency of words that is used in the PADRE relevance score. It would be more convenient and flexible to define indexes that can emulate fragmentation, if only for evaluating various fragmentation strategies.

Extractor

The main weakness in our current implementation is in the Extractor. As mentioned before it uses a Boolean function to extract the target answers (through an XPath predicate) while a similarity measurement would be adequate. This could be easily implemented using a similarity measurement (for example the cosine measure) between the constraint in the projection and the potential target elements within the current article. Frequency of words could be calculated on the fly related to the embedding article. Following Wilkinson [9], we can also limit the number of answers to a given number (for example 3 or 4), or to fragments that have a similarity measure above a given threshold.

5 Related Work

There has been interest for many years in retrieval of semi-structured information; originally for SGML but with the advent of XML there has been renewed interest in this area.

Indexing and splitting document fragments

A lot of previous work has focused on the following questions.

- What should the fragments be?
- How should relevance of fragments be used to determine the relevance of full documents?

In particular, Wilkinson and Zobel[10] compare various fragmentation schemas in Documents retrieval. They demonstrate that pages are better units of retrieval than either paragraphs or sentences. Another observation is that using fragments as a basis for retrieval, the longer relevant documents are more likely to be retrieved. That is, it helps eliminating bias in the cosine method against long documents. However, their experiments aim at evaluating the benefit of fragmentation for document retrieval, which, again, is not the XML-search objective. Giving advantage to long documents may well play against our objective of finding the most specific fragment.

Wilkinson [9] proposes several similarity measures for ranking documents from section retrieval,

or ranking sections from document retrieval. In the first case, combining the weight of the fragments (related to its type) and the rank of the fragment is improving the precision of retrieved documents, ordered based on their higher ranked sections. In the second case, the experiments determine that extracting the relevant sections from ranked documents results in very poor precision, while ranking sections within ranked documents, and discarding those sections below a threshold provides acceptable results. Since we have been using mostly the first strategy in our INEX experiments (by lack of time), there is room for improvement.

Fuller et al. [5] describe splitting and indexing an SGML-based hypertext collection. The markup language is used to create nodes and links between nodes. For example, a section node will contain its title and the introductory paragraph, while the paragraphs themselves will define other nodes linked back to their embedding section. In this model, a node section is not a viewable element, and the full section need to be rebuilt from its descendant paragraphs. In contrast, we split the XML documents into viewable fragments according to predefined elements such as front-head, sections, paragraphs, figures. A section fragment will contain the text of the full section. A consequence is that words are indexed several times: once in their original paragraph, once in their embedding section and once as part of the full paper.

Amann et al. [1] describe the use of mapping between a community ontology and XML sources. The ontology is understood by all members of the community and used as the common interface component for integrating, querying, structuring and exchanging information. The mapping is used for translating an ontology query into XML XPath queries that are sent to the XML sources for retrieval. The translation occurs at the time the query is issued. The advantage of this approach is that the underlying sources are kept independent and could be indexed by different systems. The only requirement is that they can interpret XPath queries. We use the mapping at indexing time. It makes it more efficient, although for a somehow simplified ontology (no relationship between the "concepts").

Displaying answers

Fuller et al. [5] propose two ways of displaying ranked fragments of documents through a table of content. In the first one the order of ranked fragments is kept across the display of a local table of contents for the corresponding document. In the second one, a ranking for the documents is calculated by propagating the ranks of the fragments to the document and presenting the documents in that order, while the initial rank of the fragments is still displayed in a local table of contents.

The INEX system for assessing results offers an interface for displaying fragments defined by their path within a document. The viewer opens the full document which displays the path of the fragment at the top of the document. The path can be clicked on to jump on the corresponding element, although this element would not be highlighted. Only the element at the end of the path is highlighted in red.

We have not worked on a displaying results yet. We believe that user studies should be carried out, taking in account the information task. The difficulty is to be able to distinguish between improving the precision of answers, and the bias introduced by the interface itself.

6 Conclusions and open questions

In this paper we have described our approach to XML-search based on PADRE, our full text and metadata search engine, with some extensions to support returning specific target fragments. This is still work in progress however preliminary results suggest splitting the documents is an effective strategy. Evaluation of our experimentations, especially with splitting the documents in the collection, will be available at the time of the conference.

INEX gave us the motivation and environment to carry this work, although in its first year it has mostly generated a lot of questions. Within the INEX community there is some disagreement between those approaching the problem from a database perspective (for example placing great importance to structural constraints) and those approaching the problem from an information retrieval perspective (placing more importance on the actual information needs). How to assess relevance of fragments and score answers is still a matter of discussion.

The collection used for the INEX experiment in 2002 contains a particular kind of XML documents, a significant portion of the markup relates to presentation rather than content of the documents. Many other kinds of semi-structured XML is used in a variety of applications ranging from data interchange to semi-structured documents. How well the types of information needs expressed in the INEX dataset apply to other classes of document could be explored.

Another issue is how hypertext links should be handled in querying structured documents. We may be interested in internal links within a document; for example, the connection between a figure and paragraph that contains a reference to the figure. Such queries cannot be expressed very easily in the current format for INEX topics, and answers

may not fit the restriction of just returning elements.

How results should be displayed to the user has not been addressed yet, and it is not clear whether returning fragments helps users to access more easily the information they need. A more open question may be: are fragments the right approach for more meaningful answers?

Nevertheless the INEX experiment has a provided a useful environment for investigating semi-structured retrieval and the collection will provide an important benchmark for future work in spite of its limitations.

Acknowledgements

This work was done while James Thom was a visitor to CSIRO.

Thanks to the INEX organisers.

References

[1] B. Amann, I. Fundulaki, M. Scholl, C. Beeri and A. Vercoustre. Mapping XML fragments to community web ontologies. In *Proceedings Fourth International Workshop on the Web and Databases (WebDB'2001)*, 2001.

[2] CSIRO and Australian National University. P@noptic enterprise search engine. `http://www.panopticsearch.com/`.

[3] DELOS Network of Excellence for Digital Libraries. Initiative for the Evaluation of XML retrieval. `http://qmir.dcs.qmw.ac.uk/inex/`.

[4] N. Fuhr and K. Grojohann. XIRQL: A query language for information retrieval in XML documents. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 172–180, New Orleans, USA, September 2001.

[5] M. Fuller, E. Mackie, R. Sacks-Davis and R. Wilkinson. Coherent answers for a large structured document collection. In R. Korfhage, E. Rasmussen and P. Willett (editors), *Proceedings of the 16th Annual International Conference on Research and Development in Information Retrieval*, pages 294–213, Pittsburg, USA, June 27-July 1 1993.

[6] David Hawking, Peter Bailey and Nick Craswell. Efficient and flexible search using text and metadata. Technical Report TR2000-83, CSIRO Mathematical and Information Sciences, 2000. `http://www.ted.cmis.csiro.au/~dave/TR2000-83.ps.gz`.

[7] R. W. P. Luk, H. V. Leong, T. S. Dillon, A. T. S. Chan, W. B. Croft and J. Allan. A survey in indexing and searching XML documents. *Journal of the American Society for Information Science and Technology*, Volume 53, Number 6, pages 415–437, 2002.

[8] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu and M. Gatford. Okapi at TREC-3. In D. K. Harman (editor), *Proceedings of TREC-3*, Gaithersburg MD, November 1994. NIST special publication 500-225.

[9] 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*, pages 311–317, Dublin, Ireland, July 3-6 1994. Springer-Verlag.

[10] Ross Wilkinson and Justin Zobel. Comparison of fragmentation schemes for document retrieval. In D. Harman (editor), *Proceedings of the Third Text REtrieval Conference*, pages 81–84, Gaithersburg, Maryland, 1994.