INEX – a Broadly Accepted Data Set for XML Database Processing?*

Pavel Loupal and Michal Valenta

Dept. of Computer Science and Engineering
FEE, Czech Technical University
Karlovo náměstí 13, 121 35 Praha 2
Czech Republic
P.Loupal@sh.cvut.cz, valenta@fel.cvut.cz

Abstract. The aim of the article is to inform about the INEX initiative, its testing data set, actual results, and future plans. We discuss and demonstrate possible utilization of the INEX data set for our own research and testing purposes. Our example – adaptation of approximate tree embedding algorithm - provides a basis for discussion about INEX data set suitability and about eventual consecutive experiments.

1 Introduction

Until now, there is no broadly accepted database nor data set for testing new search or index algorithms or query language specifics in branch of XML processing research. Such referential data set would be useful mainly for more accurately comparing of individual algorithms and approaches.

INEX data set can be discussed as a hot candidate for such purposes, although the INEX initiative focuses itself rather to information retrieval research than to XML query languages aspects. But its data set seems suitable, because it is large enough and its structure is also appropriately complex.

Hence the aim of the article is to inform about INEX initiative, its background, participants, plans, and results in order to initiate relevant discussion of accepting or rejecting INEX data set as a referential database for comparing research results.

We have developed a simple web based application which provides access to INEX data set and enables easy algorithm testing and results evaluation. We have prepared an example – adaptation of approximate tree embedding algorithm – in order to provide couple of concrete arguments for discussion about INEX data set relevance.

The paper is organized as follows: The second section brings basic information about INEX initiative. Subsection 2.1 informs about background, plans, founders, and participants of INEX initiative. Subsection 2.2 discusses INEX

* The research was partially supported by the grant GAČR 201/03/0912

data set structure, organization of consecutive tasks, and several results. The third section is dedicated to our utilization of INEX data set. Subsection 3.1 introduces our approach for accessing the INEX data set. Subsection 3.2 demonstrates the application on concrete example – adaptation of approximate tree embedding algorithm.

2 INEX initiative

2.1 History, participants, purposes

INitiative for the Evaluation of XML retrieval (INEX) was founded three years ago. The motivation and the main aim of the project is to provide a referential database for purposes of data retrieval research community.

Actually 69 participants mainly from universities have taken their active part in INEX project. Concrete list of participant's organizations and responsible persons could be found in INEX home page [5]. In the head of initiative stand Norbert Fuhr, Saadia Malik (Duisburg-Essen University) and Maunia Lalmas (Queen Mary University London).

Project is organized as a set of consecutive steps. Each step consists of the set of tasks which are spred among all participants. When all individual tasks are solved by their responsible participants, the result is considered together and evaluated by participants forum discussion. Then step is closed and project moves to the next stage.

The first step of the project consisted only from acquiring appropriate data set. It was supplied by IEEE – several volumes of IEEE journals. In the second step set of data retrieval queries were developed and evaluated by participants. The third step consisted of hand made relevance assessment process of individual queries. The next step is actually object of participant's discussion. It should be focused to the efectivity of relevance assessment process and to the study of searchers behaviors and also to the topics of distributed data sources. Follow open discussion of the third INEX workshop in [5].

2.2 Data structure, queries, relevance assessments

INEX data set (actual version 1.4) has 536MB of XML data. It is exactly 12,107 articles from 6 IEEE transactions and 12 journals from years 1995 to 2002. Pictures are not included – data set consists only of XML formated text.

Data set is organized in file structure. Root directory consists of two subdirectories – *dtd* (holds structure information - DTD specification article element) and *xml*. Each journal/transaction has its own two-letter named subdirectory inside xml directory. Journal/transaction is further divided into the directories by the year of publication. Finally each article is stored in individual xml file, which name consists of a letter following by four-digit number and xml suffix. Structure is schematically shown in figure 1.

In average each article contains 1,532 XML nodes, where the average depth of node is 6.9. See [3] for detail characteristics of data set.

```
/inex-1.4
        /dtd
             ...
             xmlarticle.dtd
        /xml
             /an
                    /1995
                             ...
                             a1019.xml
                             a1032.xml
                             a1034.xml
                             ...
                    /...
                    /2002
             /...
             /ts
```

Fig. 1. INEX data set file structure

DTD specification or article element is too complex to be clearly presented here. Instead of this more illustrative fragment of typical article is shown in figure 2. Picture is taken from [3].

The second stage of INEX project was focused to construction of suitable data retrieval queries (*topics*). Topics were constructed by individual participants and then were accepted or rejected by discussion forum of all participants. Each participant had to design 6 queries.

Topics were divided into two groups – Content Only (*CO*) and Content And Structure (*CAS*) topics. CAS topics have a structure condition inside their specification, for example they interests only in abstracts etc. CAS topics are then classified either Strict (*SCAS*) or Vague (*VCAS*). The final set of INEX'03 topics consists of 36 CO and 30 CAS queries.

Then each participant did 3 runs of each topic and select the first 1000 most relevant documents. Individual runs were averaged so there was a set of 1000 most relevant documents for each topic. Statistics of individual runs were computed and they are available for participants purposes.

Evaluation stage covers hand made relevance assessments of 1000 documents selected in the previous stage with respect to the given topic. The evaluation was done through web based assessment application, see figure 4. Relevance assessment was expressed by two independent scales – *specificity* and *exhaustivity*.

```
|<article>                                |  <sec>
|  <fm>                                   |    <st>...</st>
|    ...                                  |    ...
|    <ti>IEEE Transactions on ...</ti>    |    <ss1>...</ss1>
|    <atl>Construction of ...</atl>       |    <ss1>...</ss1>
|    <au>                                 |    ...
|      <fnm>John</fnm>                    |    </sec>
|      <snm>Smith</snm>                   |   ...
|      <aff>University of ...</aff>       |  </bdy>
|      </au>                              |  <bm>
|    </au>...</au>                        |    <bib>
|    ...                                  |     <bb>
|  </fm>                                  |       <au>...</au><ti>...</ti>
|  <bdy>                                  |     ...
|     <sec>                               |     </bb>
|    <st>Introduction</st>                |    ...
|    <p>...</p>                           |    </bib>
|    ...                                  |  </bm>
|    </sec>                               |</article>
```

Fig. 2. INEX typical article structure

Each scale has three values – *marginal, fairly, high* specific/exhaustive, so element can be maked by one of nine assessment values. The tenth assessment value is *not relevant*. This is the way to express more relevant parts inside the document. Icons were used to express given relevance mark for chosen XML node in INEX assessment interface.

Moreover there are several parent-child dependencies in assigning relevance mark to the element. For example exhaustivity level of a parent element is always equal to or greater than the exhaustivity level of its children elements. These dependencies are fixed and they are automatically checked by assessment interface.

The result of assessment process is published as XML document according to DTD specification is shown in figure 3.

Nowadays, all INEX topics have been processed and there is XML file with assessment results for each topic. This stage of project has been discussed in the second INEX workshop, see [4] for details.

The next stage of INEX project will focus on detail study of searchers behaviors (this research starts from analysis of handy made assessments from previous stage) and also to data retrieval from heterogeneous sources and distributed systems. Actually, details of the further stage of INEX project are discussed in forum of project participants. See discussion in [5] for details.

```
<!ELEMENT assessments (file+)>
<!ATTLIST assessments
          topic           CDATA              #REQUIRED
>
<!ELEMENT file            (path*)>
<!ATTLIST file
          file            CDATA              #REQUIRED
>
<!ELEMENT path            EMPTY>
<!ATTLIST path
          path            CDATA              #REQUIRED
          exhaustiveness  ( 0 | 1 | 2 | 3 ) #REQUIRED
          specificity     ( 0 | 1 | 2 | 3 ) #REQUIRED
>
```

Fig. 3. INEX DTD topic assessments

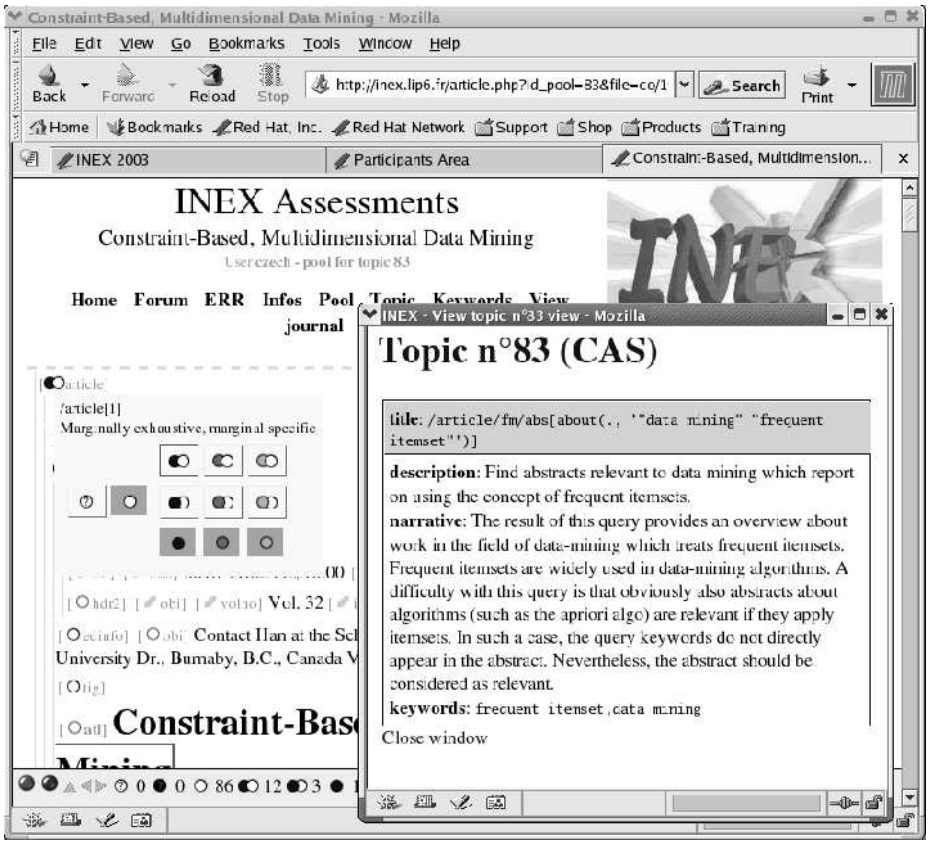

Fig. 4. INEX XML assessment interface

3 Data set utilization

The first step for utilization of the INEX data set for our research purposes is preparation of common interface for data access. As shown in figure 1 data is stored in directories relevant to journals and volumes. Our approach should keep this structure to avoid any disorders. Also for that reason we decided to use a solution based on a native XML database.

3.1 Native XML storage

Native XML databases have some advantages in comparison with ordinary method of storing XML files in a file system. The concept of native XML databases is based on filesystem structure i.e. data can be stored in collections which mean the same as directories in filesystems but additionally this kind of databases has many enhanced features useful for processing XML data.

For our purposes and with respect to INEX structure we decided to build our approach on the Apache Xindice native XML database [6]. This product is an open source project developed by the Apache Software Foundation with several contributors involved into its advancement.

Starting from version 1.1, Xindice is not a standalone server anymore. The server functions are now based on any Servlet 2.2 (or higher) compliant application server - in our case we decided to use Apache Tomcat 5.0.16 application server. Bundling database core functionality into a servlet container allows users not to create only standalone console applications but easily develop also web based applications using e.g. Java Server Pages (we use this technology in our tree-embedding example).

Regarding to user documentation Xindice was not designed for handling huge documents, rather, it was designed for collections of small to medium sized documents. The INEX structure complains optimally this requirement (see section 2.2).

Following built-in key features are important for data processing:

- **Standard API for accessing data** - In this case it is the XML:DB interface [7]. This simple interface is developer-friendly and supports common data processing methods.
- **XPath expresions** - In many applications XPath is only applied at the document level but in Xindice XPath queries are executed at the collection level. This means that a query can be run against multiple documents and the result set will contain all matching nodes from all documents in the collection.
- **Usage of metadata** - Xindice allows an user to store metadata that is associated with any collection or document. Metadata is data that is associated with a document or collection but is not part of that document or collection. This metadata could be used for storing temporar information

e.g. algorithm's subresults or more permanent data such as a list of indexed keywords contained by that collection or document.

Actually there are two ways for developers how to access Xindice database:

- **XML:DB XML Database API** - is used for developing applications in Java language. An example of usage of this Application Programming Interface (API) is shown in figure 5. This example shows an basic approach for retrieving a document from specified collection.
- **Xindice XML-RPC API** - is used when accessing Xindice from language other than Java.

One example of usage of the XML:DB API is shown in figure 5.

```
Collection col = null;
try {
  String driver = "org.apache.xindice.client.xmldb.DatabaseImpl";
  Class c = Class.forName(driver);

  Database database = (Database) c.newInstance();
  DatabaseManager.registerDatabase(database);
  col =
    DatabaseManager.getCollection(
          "xmldb:xindice://nonstop.sh.cvut.cz:8080/db/inex/mu/2001");

  XMLResource document = (XMLResource) col.getResource("a1019.xml");
  if (document != null) {
    // Print out document's content
    System.out.println(document.getContent());
  }
  else {
    System.out.println("Document not found");
  }
}
```

Fig. 5. Example Java code for retrieving an XML document from database

3.2 Example – Approximate tree embedding algorithm

We have addopted an approximate tree embedding algorithm using the INEX data set. Our implementation uses core tree embedding algorithm written by Jan Váňa (see [2]) and is wrapped into a simple graphical user interface. This interface allows user to select a collection (with or without all subcollections) where to search and a query to search for. Details of this algorithm are described in following section.

The algorithm. Approximate tree embedding algorithms were studied in early 90's and Kilpelainen showed that the decision problem whether a tree can be embedd into another is NP-complete. Schlieder [1] created an algorithm which behaviour is polynomial in practical examples. Váňa [2] modified that algorithm and added few improvements in some special boundary cases and exceptional situations.

The core part of this algorithm could be described as *embeddTree*(T_q, T_d), where T_q is a query tree and T_d is data tree. One possible matching mapping between query and data tree is shown in figure 6. Algorithm searches for all embeddings of query tree in data tree and also returns a rating for each match - this "cost" basically means the number of elements which had to be skipped when embedding tree.

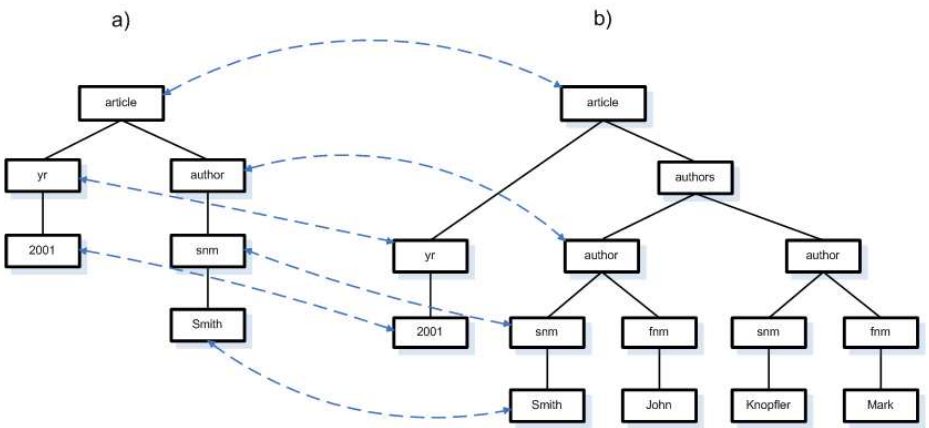

Fig. 6. Example matching between two trees. a) query tree T_q, b) data tree T_d

This paper is not primarily addressed to discuss this problem. This example was chosen for showing adaptation of such algorithm over the INEX data set. Detailed description of this algorithm can be found in [1], [2].

Implementation. We have created an application which integrates approximate tree embedding algorithm with access to documents stored in XML database. Web-based user interface allows user submit his query over specified collection and get results with their ranking. Our algorithm traverses (recursively) specified collection, fetches list of XML resources stored in and tries to embedd query tree into all documents.

Actually algorithm uses two metrics for rank matched result. The first one is based on number node which need to be skipped when embedding tree and the second one is the level of root node of the query tree T_q in data tree T_d.

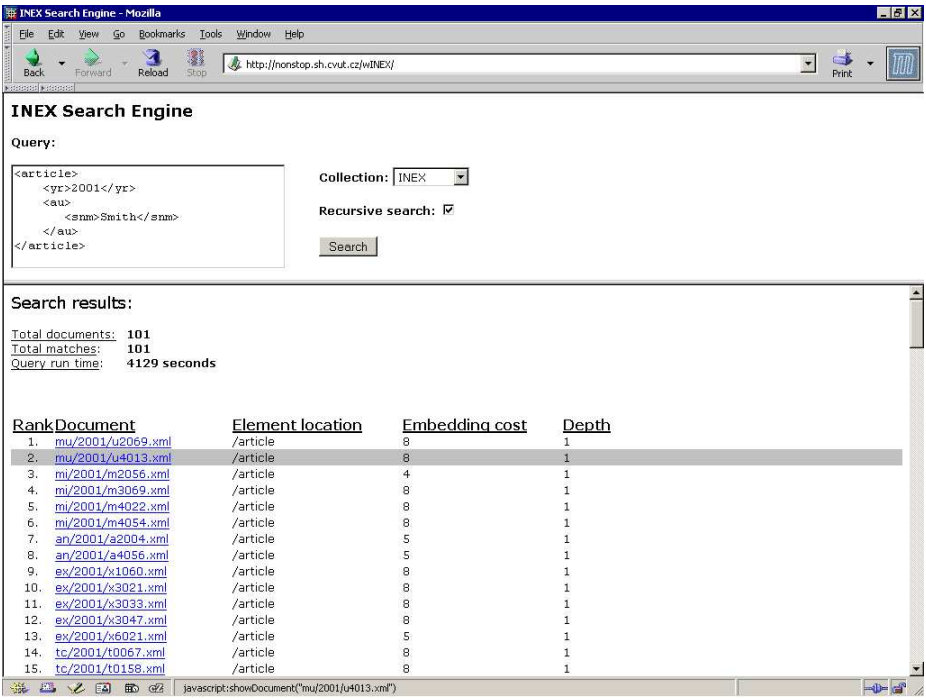

Fig. 7. Simple web interface for querying database

Results. Our adaptation of the approximate tree embedding algorithm allows user to get results as supposed by its description, i.e. it finds all embeddings in documents contained in specified collection(s). The correctness of the implementation was not proved exactly but our queries and their respective results were manually checked for match.

Performace. Our implementation uses platform independent Java XML:DB interface outlined above. This approach is probably slower than a similar native solution written in language like C++ but in this case speed is not the crucial requested property.

To get an estimation of time consumption when performing our algorithm we ran a simple query (see figure 8) on a computer with the Intel Pentium III processor (500MHz) with 256 MB of memory. This run over the complete INEX data set took about 68 minutes and returned 101 matches.

Usability. In spite of correctness of the algorithm, practical benefit is a moot question. The main drawback is strict comparison of data items used - the content of an element must exactly match data in given query. For practical purposes users would usually ask queries not about structual match but more on approx-

```
<article>
    <yr>2001</yr>
    <au>
        <snm>Smith</snm>
    </au>
</article>
```

Fig. 8. Query example - all articles published in the year 2001 written by author with surname "Smith"

imate content. Therefore, operators such as *contains()* or *starts-with()* known from XPath could extend practical algorithm applicability.

4 Conclusions

The INEX data set is a huge collection of "real" and meaningful documents. Storing such data in native XML database (in our case Apache Xindice) allows researchers to implement and test wide range of algorithms over these data. Technical environment supports developing both console and web-based Java applications using common standards for manipulating XML data.

Our example, approximate tree embedding algorithm, shows one example of possible utilization of the INEX data set. It is just one of possible implementations but the common access interface allows researches to "plug in" another solution of the embedding algorithm or even to use another algorithm which has a different goal.

Further work on this algorithm should be focused on experiments with order evaluating function. Some hypothesis about local and global rate of the order function had been stated in an unformal discussion in the last DATESO workshop. These hypothesis should be formalized, implemented, and proved on a "real" data set.

In addition the INEX data set has been adopted into our frame and can be used for arbitral experiments in the branch of XML database processing research. Although it is only a side-effect of the INEX project, we have shown in this article, it can be used also for our own research with a good benefit.

References

1. Schlieder, T., Naumann, F.: Approximate tree embedding for querying XML data. In ACM SIGIR Workshop On XML and Information Retrieval, Athens, Greece, 2000.
2. Váňa, J.: Integrity of XML data (*in Czech*). Master Thesis, Dept. of Software Engineering, Charles University, Prague. 2001.

3. Fuhr, N., Gvert, N., Kazai, G., Lalmas, M.: INitiative for the Evaluation of XML retrieval (INEX). Proceedings of the First INEX Workshop.ERCIM Workshop Proceedings. ERCIM, Sophia Antipolis, France, 2003.
4. Fuhr, N., Malik, S., Kazai, G., Lalmas M. (*Editors*): Proceedings of the 2nd Initiative on the Evaluation of XML Retrieval (INEX 2003). ERCIM Workshop Proceedings. 2003
5. INEX 2003 - home page. http://inex.is.informatik.uni-duisburg.de:2003/index.html.
6. Apache Xindice - Native XML database. http://xml.apache.org/xindice.
7. XML:DB initiative - Application Programming Interface for accessing native XML databases. http://www.xmldb.org.