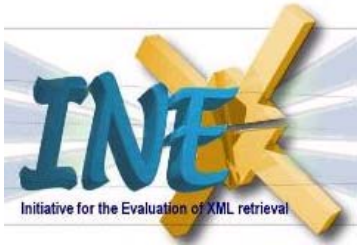

INEX'03 Guidelines for Topic Development

The aim of the INEX initiative is to provide means, in the form of a large test collection and appropriate scoring methods, for the evaluation of content-oriented XML retrieval. Within the INEX initiative it is the task of the participating organisations to provide the topics and relevance assessments that will contribute to the test collection. Each participating organisation therefore plays a vital role in this collaborative effort.

1. Introduction

Test collections, as traditionally used in information retrieval (IR), consist of three parts: a set of documents, a set of information needs called topics, and a set of relevance assessments listing for each topic the set of relevant documents.

A test collection for XML retrieval differs from traditional IR test collections in many respects. Although it still consists of the same three parts, the nature of these parts is fundamentally different. In IR test collections, documents are considered as units of unstructured text, topic statements are generally treated as collections of terms and/or phrases, and relevance assessments provide judgements whether a document as a whole is relevant to a query or not. XML documents, on the other hand, organise their content into smaller, nested structural elements. Each of these elements in the document's hierarchy, along with the document itself, is a retrievable unit. Regarding the topics, with the use of XML query languages, users of an XML retrieval system are able to combine both content and structural conditions within their information need and restrict their search to specific structural elements within an XML collection. Finally the relevance assessments for an XML collection must also consider the structural nature of the documents and provide assessments at different structural levels.

This guide deals only with the topics of the test collection and provides detailed guidelines for their creation for INEX 2003.

2. Topic creation criteria

Creating a set of topics for a test collection requires a balance between competing interests. It is a well-known fact that the performance of retrieval systems varies largely for different topics. This variation is usually greater than the performance variation of different retrieval methods on the same topic. Thus, to judge whether one retrieval strategy is in general more effective than another strategy, the retrieval performance must be averaged over a large, diverse set of topics. In addition, to be a useful diagnostic tool, the average performance of the retrieval systems on the topics can be neither too good nor too bad as little can be learned about retrieval strategies if systems retrieve no or only relevant documents.

When creating topics, a number of factors should be taken into account.

1. **The author of a topic should be either an expert or the very least be familiar with the subject area covered by the collection!** (Note that the author of a topic should also be the assessor of relevance!)
2. Topics should reflect what real users of operational systems might ask.
3. Topics should be representative of the type of service that operational systems might provide.
4. Topics should be diverse.
5. Topics may also differ in their coverage, e.g. broad or narrow topic queries.

3. Query types

As last year, in INEX 2003 we distinguish two types of query:

- *Content-only (CO) queries*: are requests that ignore the document structure and contain only content related conditions, e.g. only specify what a document/component should be about (without specifying what that component is). The need for this type of query for the evaluation of XML retrieval stems from the fact that users either do not care about the structure of the result components or are not familiar with the exact structure of the XML documents.

- *Content-and-structure (CAS) queries*: are topic statements, which contain explicit references to the XML structure, and restrict the context of interest and/or the context of certain search concepts.

4. Topic format

Both CO and CAS topics are made up of four parts:

- *Topic title*: a short version of the topic statement. It serves as a summary of both the content and structural requirements of the user's information need. The exact format of the topic title is discussed in more detail later in this section.
- *Topic description*: a one or two sentence natural language definition of an information need.
- *Narrative*: a detailed explanation of the topic statement and the description of what makes a document/component relevant or not.
- *Keywords*: a set of comma-separated scan terms that are used in the collection exploration phase of the topic development process (see Section 5.2) to retrieve relevant documents/components. Scan terms may be single words or phrases and may include synonyms, broader or narrower terms from those listed in the topic description or topic title.

The format of the topic title in 2003 is different to that used in INEX 2002. This year, the format is based on XPath, the proposed language for addressing parts of XML documents. The XPath notation is adopted in INEX 2003 to refer to the logical structure and the attributes of the XML documents. However, since XPath is a very rich and powerful language, we restrict ourselves to a subset of XPath, which has been identified by the INEX 2002 Topic Format working group as providing an "IR minimum". This subset corresponds (mainly) to the use of path expressions as described in Section 2 of the document XML Path Language (XPath) Version 1.0, W3C Working Draft 16 November 1999 (available at http://www.w3.org/TR/xpath). More precisely, the topic format will make use of Axes (Section 2.2), Predicates (Section 2.4), and will use the abbreviated syntax described in Section 2.5 of the aforementioned document.

Below are examples of path expressions (taken from Section 2.5 of the XPath 1.0 standard):

- `para` selects the `para` element children of the context node
- `*` selects all element children of the context node
- `@attr` selects the `attr` attribute of the context node
- `@*` selects all the attributes of the context node
- `para[1]` selects the first `para` child of the context node
- `*/para` selects all `para` grandchildren of the context node
- `/doc/chapter[5]/section[2]` selects the second `section` of the fifth `chapter` of `doc`
- `chapter//para` selects the `para` descendants element of the `chapter` element children of the context node
- `//para` selects all the `para` descendants of the document root and thus selects all `para` elements in the same document as the context node
- `//olist/item` selects all the `item` elements in the same document as the context node that have an `olist` parent
- `.` selects the context node
- `.//para` selects the `para` element descendants of the context node
- `..` selects the parent of the context node
- `../@lang` selects the `lang` attribute of the parent of the context node
- `para[@type='warning']` selects all `para` children of the context node that have a `type` attribute with value `warning`
- `para[@type='warning'][5]` selects the fifth `para` child of the context node that has a `type` attribute with value `warning`
- `para[5][@type='warning']` selects the fifth `para` child of the context node if that child has a `type` attribute with value `warning`
- `chapter[title='Introduction']` selects the `chapter` children of the context node that have one or more `title` children with string-value equal to `Introduction`

- `chapter[title]` selects the `chapter` children of the context node that have one or more `title` children
- `employee[@secretary and @assistant]` selects all the `employee` children of the context node that have both a `secretary` attribute and an `assistant` attribute

4.1. The *about()* function

In INEX, an "aboutness" concept, in the form of an *about(path, string)* function, has been added to the standard XPath syntax to deal with the content aspect of a user query. This concept was necessary in order to introduce the uncertainty inherent in IR into the world of the more exact-match XPath principle. The *about()* function should be used as the basis to provide a ranking of the retrieved elements with respect to content. Note that the *about(path,string)* clause is different from the *contains(path,string)* function of the XPath standard (see XPath 1.0, http://www.w3.org/TR/xpath). The latter returns true if the text value of the element defined by the path contains the string argument, and otherwise returns false. On the other hand, the *about()* function returns true if the element defined by the path argument is "about" the concept(s) defined by the string argument without having to actually contain the exact string value.

The *about()* function is usually applied to a context element, `CE`. This is described by the following syntax: `CE[about(path, string)]`. A context element is described using a standard XPath path expression (see the examples of path expressions in Section 4). It defines a "base node" against which relative paths, using the "." notation, can be defined within the *path* argument of the *about()* function. For example, `//article[about(.//sec,'"XML retrieval"')]` represents the request to retrieve articles that contain within them a section about "XML retrieval". Another example is `//article[about(.//sec, '"XML retrieval"') and about(.//sec,'evaluation')]`, which is a representation of the request to retrieve articles, which contain a section about "XML retrieval" and also a section on evaluation (where the two sections may be different or may be the same). We will look at more complex structures when we discuss the format of the CAS topic titles. The *string* parameter may contain a number of space-separated terms, where a term may be a single word or a phrase encapsulated in double-quotes. Furthermore, the symbols + and − may be used to express additional preferences for certain terms, where + is used to emphasise a concept and − is used to denote an unwanted concept. In summary, a *string* parameter may incorporate the following components:

- Terms (single words or phrases)
- "" (double-quotes to encapsulate phrases)
- + (expressing "must be about")
- − (denoting "must not be about")

The syntax of a *string* argument is:

```
String   ::=   term ' '
             | '+'term ' '
             | '−'term ' '
Term     ::=   single word
             | '"'phrase'"'
```

A *string* must be enclosed between single quotes. For example, `//article[about(.//sec, '"XML retrieval" +XML -"information retrieval"')]` would correspond to the request to retrieve articles that contain a section which is about XML retrieval but not about information retrieval, and where XML is characterised as an important concept.

Although at this point we are not talking about relevance assessment we would like to make a note here to emphasise that for relevance assessments the symbols + and − should be interpreted with a fuzzy "flavour" and not simply as must contain or must not contain conditions. Following on from the definition of the *about()* function above, a component may be considered relevant even if it does not contain the query term(s), but is "about" the concept(s) expressed by the query term(s). Similarly a component may be relevant even if it contains, for example, only one half of a phrase.

4.2 CO Topics

The topic title of a CO topic is a short, usually a 2-5 terms representation of the topic statement. Since CO topics ignore the document structure, their topic title will only consist of one *about()* clause applied to any context elements denoted by the path `//*`. The *path* argument of the *about()* function must be set to "." (dot) to refer to the context element. The *string* argument is made up of terms that best describe what the user is looking for. Take as an example the topic title `//*[about(., '"XML retrieval"')`, which is the representation of the request to retrieve any elements that are about "XML retrieval".

In order to simplify this syntax, we remove all components of the topic title that are the same for all CO topics (e.g. the context element, the *path* argument, etc.). As a result, we end up with just the *string* argument of the *about()* function, e.g. replacing `//*[about(path, string)]` with `string`, where we also ignore the single quotes.

The topic title of a CO topic is therefore defined as a set of space separated terms, optionally associated with the symbols + and −, where a term may be a single word or a phrase encapsulated in double-quotes. The syntax of the CO topic titles hence matches the syntax of the *string* argument specified above (Section 4.1).

Examples of CO topic titles

1. Retrieve documents/components about computer science degrees that are not master degrees:

```
<title>"computer science" +degree −master</title>
```

2. Retrieve document/components about summer holidays in England:

```
<title>"summer holiday" +England</title>
```

Example of a CO topic

```
<inex_topic topic_id="1" query_type="CO">
       <title>
              "summer holiday" "winter holiday" +"England"
       </title>
       <description>
              Winter or summer holidays in England.
       </description>
       <narrative>
              To be relevant, a document or component must contain
              information about winter or summer holidays in England.
       </narrative>
       <keywords>
              summer, winter, holiday, England, skiing, beach
       </keywords>
</inex_topic>
```

4.3 CAS Topic

The general structure of a CAS topic title is as follows:

```
CE [ filter ] CE [ filter ] … CE [filter] CE [filter]
```

`CE` refers to the context element. The series of context elements, where the first `CE` acts as the root node, describes a branch of an XML tree. Each context element is relative to the context element that precedes it in the sequence. This branch forms the path of the target element that is to be returned to the user. A filter is defined as a set of about clauses (e.g. *about(path, string)*) and other predicate clauses (e.g. `@yr = '2001'`), which are joined by Boolean expressions. The *path* argument of the *about()* function can be expressed relative to the context element by using the "." notation. For example, `//article[.//@yr = '2001']//sec[about(.,'+"XML retrieval"')`, is the expression of a request to retrieve sections about "XML retrieval" of articles written in 2001. This query has two context elements, namely `//article` and `//sec`, which together define the target element. `//article//sec`.

A filter may contain a set of *about()* functions and/or a set of standard XPath string operators: =, !=, >, <, >= and <=. The conditions expressed by these functions and operators can be combined using the Boolean operators: AND and OR, together with the use of parenthesis to group such conditions

together. For example, `//article[about(.//p,'+"holiday"') AND .//@yr='2002']`, retrieves articles that contain paragraphs about "holiday" and have a published date of 2002. Note that while the series of context elements must describe a branch of the XML tree, the filter components allow for the definition of content conditions on different branches of a tree within the context element. Take the earlier mentioned example (Section 4.1) of `//article[about(.//sec,'"XML retrieval"') and about(.//sec,'evaluation')]` requesting article elements, which contain a section about "XML retrieval" and also a section on "evaluation" (where the two sections may be different or may be the same). Here two independent branches of the tree rooted in `//article` are described.
`NOTE THAT FOR AN INEX CAS TOPIC, IT IS A REQUIREMENT THAT A FILTER CONTAINING AN ABOUT() FUNCTION MUST BE SPECIFIED FOR THE LAST CONTEXT ELEMENT!` Multiple target elements are not allowed in INEX 2003. Also note that specifying one context element only, and setting it to `//*`, while setting the *path* argument of the *about()* functions to ".", we arrive back at a CO topic title.

Examples of CAS topic titles[1]

1. Return section elements, which are about summer holidays, where the section element is a descendent of article element, and the article is from 2001 or 2002:

```
<title>
    //article[.//@yr = '2001' OR .//@yr = '2002']//sec[about(.,
    '"summer holidays"')]
</title>
```

   The above query has two context elements, `//article` and `//sec`, each with their own filters, one containing a standard Xpath predicate and the other containing an *about()* clause. The target element defined by the above query is `//article//sec`.

   Note that the following query is not a valid INEX query as it does not contain an *about()* function:

```
<title>
    //article[.//@yr = '2001' OR .//@yr = '2002']
</title>
```

   The following query is not valid because there is no filter applied to the last context element (e.g. //sec):

```
<title>
    //article[.//@yr = '2001' OR about(., '"summer holiday"']//sec
 </title>
```

In the remainder of the examples for simplicity we ignore the `<title> </title>` tags.

2. Retrieve all articles that were published in 2001 and are about summer holidays:

```
//article[.//@yr = '2001' AND about(./, '"summer holidays"')]
```

3. Return article elements published in 2001 that contain section elements about summer holidays:

```
//article[.//@yr = '2001' AND about(.//sec, '"summer holidays"')]
```

4. Return articles from 2001, which contain section elements about summer holidays or section elements about winter holidays:

```
//article[.//@yr = '2001'  AND (about(.//sec, '"summer holidays"')OR
about (.//sec, '"winter holidays"'))]
```

   A query requesting articles from 2001 containing section elements about summer and winter holidays would be as follows:

```
//article[.//@yr = '2001' AND (about(.//sec, '+"summer holidays"
+"winter holidays"')]
```

5. Return section elements, which are about summer holidays and that are the grandchildren of article elements, where the article is from 2001 or 2002:

[1] Note that these examples do not conform to the structure or content of the INEX document collection

```
//article[.//@yr = '2001' or .//@yr = '2002']/*/sec[about(., '"summer holidays"')]
```

6. Return articles on XML retrieval, where the article contains a section on evaluation:

```
//article[about(., '"XML retrieval"') AND about(.//sec, 'evaluation')]
```

7. Retrieve articles that were published in 2002 and contain a section about "XML retrieval":

```
//article[about(.//sec, '"XML retrieval"') AND .//@yr='2002']
```

8. Retrieve those sections of articles published in 2002 that are about "XML retrieval":

```
//article[.//@yr='2002']//sec[about(.//sec, '"XML retrieval"')]
```

9. Retrieve those sections of articles that contain both a figure about "CORBA" and a figure caption about "XML":

```
//article//sec[about(.//fig, 'CORBA') AND about(.//figc, 'XML')]
```

Example of a CAS topic

```
<inex_topic topic_id="2" query_type="CAS">
	<title>
		//article[.//@yr = '2001' OR .//@yr = '2002']//sec[about(., '"summer holidays"')]
	</title>
	<description>
		Summer holidays either of 2001 or of 2002.
	</description>
	<narrative>
		Return section elements, which are about summer holidays, where the sections is descendent of article element, and the article is from 2001 or 2002.
	</narrative>
	<keywords>
		summer, holiday, 2001,2002
	</keywords>
</inex_topic>
```

4.4. Equivalent tags

This section lists the defined set of "equivalent" tags (alias/role/metedata) in the INEX test collection. We are proposing aliases for the following classes of nodes (identified directly from the DTD):

Paragraph-like nodes: ilrj|ip1|ip2|ip3|ip4|ip5|item-none|p|p1|p2|p3
Section nodes: sec|ss1|ss2|ss3
List environments: dl|l1|l2|l3|l4|l5|l6|l7|l8|l9|la|lb|lc|ld|le|list|numeric-list|numeric-rbrace|bullet-list
Headings: h|h1|h1a|h2|h2a|h3|h4

4.5. Topics DTD

The overall structure of the INEX topics is given in the DTD below (Note that additional attributes may be added at a later stage).

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!ELEMENT inex_topic (title, description, narrative, keywords)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT narrative (#PCDATA)>
<!ELEMENT keywords (#PCDATA)>
<!ATTLIST inex_topic
  topic_id   CDATA  #REQUIRED
  query_type CDATA  #REQUIRED
>
```

5. Procedure for topic development

Each participating group will have to submit **3 CO and 3 CAS** queries by the **30 May 2003** by filling in the Candidate Topic Form (one per topic) at

http://inex.is.informatik.uni-duisburg.de:2003/internal/TopicSubmission.html

This section outlines the procedures involved in the development of candidate topics. There are four steps in creating topics for a test collection: 1) creating the initial topic statements, 2) exploring the collection, 3) selecting final set of topics, and 4) refining the topic statements.

5.1. Initial topic statements

In this step, you should create a one or two sentence description of the information you are seeking. This should be a simple description of the information need without regard to retrieval system capabilities or document collection peculiarities. This should be recorded in the topic description field.

Use either a printout or directly the on-line version of the Candidate Topic Form to record all information on a topic you are creating.

5.2. Collection exploration

In this step the initial topic statements are used to explore the document collection in order to obtain an estimate of the number of relevant documents/elements in the collection and to evaluate whether this topic can be judged consistently in the assessment phase. You may use any retrieval engine for this task, including your own or HyRex (HyRex can be accessed via http://inex.is.informatik.uni-duisburg.de:2003/internal/#topics).

Using the Candidate Topic Form record the set of keywords that you use for retrieval (make sure to record all the keywords from all iteration of your search or if you use query expansion strategies the query terms generated by the process). You should try and make your search queries (e.g. set of keywords) as expressive as possible for the kind of documents you wish to retrieve: think of the words that would make good scan words when assessing, and use those as your query keywords.

Next, judge the top 25 documents/components of your retrieval result. Using the Candidate Topic Form record the number of found relevant components and the XPath path representing each relevant element. If you have found less than 2 or more than 20 relevant components within the top 25 results, you should abandon the topic and start with a new one! If you have found at least 2 relevant components and no more than 20, perform a feedback search (don't forget to record the terms (if any) that you decide to add to your query keywords). Judge the top 100 (some of them you will have judged already), and record the number of relevant documents/components in Candidate Topic Form.

Finally write your detailed explanation on what makes a document/component relevant and record this in the narrative field of the topic. Make sure your description is as exhaustive as possible as there will be a couple of months gap before you will return to the topic for relevance assessments. The expectation is that by judging 100 documents/components you will have determined how you will judge the topic in the assessment phase. The narrative of the topic should reflect this.

To assess the relevance of a retrieved document/component use the following working definition: mark a document/component relevant if it would be useful if you were writing a report on the subject of the topic, or if it contributes towards satisfying your information need. Each document/component should be judged on it own merits. That is, a document/component is still relevant even if it is the thirtieth document/component you have seen with the same information. It is crucial to obtain exhaustive relevance judgements. It is also very important that your judgement of relevance is consistent throughout this task.

5.3. Refining topic statements

Refining the topic statement means finalising the topic title, description, keywords and narrative. Note that it should be possible to use each of the four parts of a topic in a stand-alone fashion (e.g. using only the title for retrieval, or only the description for filtering etc.).

Once you finished, submit the on-line Candidate Topic Form at

http://inex.is.informatik.uni-duisburg.de:2003/internal/TopicSubmission.html.

Make sure you submit all **6** candidate topics no later than the 30 May 2003.

5.4. Topic selection

From the received candidate topics, we (the clearinghouse) will then decide which topics to use such that a wide range of likely number of relevant documents is included. The data obtained from the collection exploration phase will be used as input to the topic selection process. We will then distribute final set of topics back to you to be used for the retrieval and evaluation.

We would like to thank you for your contribution.

Acknowledgements

The topic format proposed in this document is based on the outcome of a working group set up during the INEX 2002 workshop in Dagstuhl and the intense discussions on the INEX 2003 mailing list. We are very grateful for their contribution.

Authors:
Gabriella Kazai, Mounia Lalmas and Saadia Malik
06 May, 2003
Updated 12 May 2003