INEX 2007 Retrieval Task and Result Submission Specification

Charles L. A. Clarke, Jaap Kamps, Mounia Lalmas

Sunday, June 3, 2007

What's New in 2007? The INEX 2007 Adhoc track sees a continuation of three retrieval tasks: the Focused Task, the Relevant in Context Task, and the Best in Context Task. The main change at INEX 2007 is the liberalization to arbitrary passages. That is, a retrieval result can be either an XML element, or an arbitrary passage (a sequence of textual content either from within an element, or spanning a range of elements).

1 Retrieval Task

The retrieval task to be performed by the participating groups of INEX 2007 is defined as the adhoc retrieval of XML elements or passages. In information retrieval (IR) literature, adhoc retrieval is described as a simulation of how a library might be used, and it involves the searching of a static set of documents using a new set of topics. While the principle is the same, the difference for INEX is that the library consists of XML documents, the queries may contain both content and structural conditions and, in response to a query, arbitrary XML elements may be retrieved from the library. Moreover, at INEX 2007, we also allow the submission of arbitrary passages.

The general aim of an IR system is to find *relevant information* for a given topic of request. In the case of XML retrieval there is, for each article containing relevant information, a choice from a whole hierarchy of different elements or passages to return. Hence, within XML retrieval, we regard as *relevant results* those results that both

- contain relevant information (the result exhaustively discusses the topic), but
- contain as little non-relevant information as possible (the result is specific for the topic).

For example, if an XML element contains another element but they have the same amount of relevant text, the shorter element is strictly more specific and a preferred result. The same holds for different passages covering the same amount of relevant text.

Within the adhoc XML retrieval task we define the following three sub-tasks:

1. Focused Task asks systems to return a ranked list of elements or passages to the user.
2. Relevant in Context Task asks systems to return relevant elements or passages clustered per article to the user.
3. Best in Context Task asks systems to return articles with one best entry point to the user.

1.1 Focused Task

1.1.1 Motivation for the Task

The scenario underlying the Focused Task is to return to the user a ranked-list of element or passages for her topic of request. The INEX 2007 Focused Task asks systems to find the most focused results that satisfy a information need, without returning "overlapping" elements. That is, for a given topic, no retrieval result in the result set may contain text already contained in another result. Or, in terms of the XML tree, no element in the result set should be a child or descendant of another element. The task makes a number of assumptions:

Display the results are presented as a ranked-list of results to the user.

Users view the result list top-down, one-by-one. Users do not want overlapping results in the result-list, and if equally relevant prefer shorter results over longer ones.

What we hope to learn from this task is: How important is the XML document-structure? How effective are XML element retrieval approaches relative to pure passage retrieval? How do structural constraints in the query help retrieval?

1.1.2 Results to Return

The aim of the FOCUSED TASK is to return a ranked-list of elements or passages, where no result may be overlapping with any other result. Since ancestors elements and longer passages may also be relevant (be it to a lesser or greater extent) it is a challenge to chose the results appropriately. *Please note that submitted runs containing overlapping results will be disqualified.*

Summarizing: FOCUSED TASK returns results ranked in relevance order (where specificity is rewarded). Overlap is **not** permitted in the submitted run.

1.2 RELEVANT IN CONTEXT TASK

1.2.1 Motivation for the Task

The scenario underlying the RELEVANT IN CONTEXT TASK is to return the relevant information (captured by a set of elements or passages) within the context of the full article. As a result, an article devoted to the topic of request, will contain a lot of relevant information across many elements. The INEX 2007 RELEVANT IN CONTEXT TASK asks systems to find a set of results that corresponds well to (all) relevant information in each article. The task makes a number of assumptions:

Display results will be grouped per article, in their original document order, providing access through further navigational means.

Users consider the article as the most natural unit, and prefer an overview of relevance in their context.

What we hope to learn from this task is: How does the user-oriented RELEVANT IN CONTEXT TASK differ from FOCUSED TASK? What techniques are effective at locating relevance within articles? How do structural constraints in the query help retrieval?

1.2.2 Results to Return

The aim of the RELEVANT IN CONTEXT TASK is to first identify relevant articles (the fetching phase), and then to identify the relevant results within the fetched articles (the browsing phase). In the fetching phase, articles should be ranked according to their topical relevance. In the browsing phase, we have a set of results that cover the relevant information in the article. The `/article[1]` element itself need not be returned, but is implied by any result from a given article. Since the content of an element is fully contained in its parent element and ascendants, the set may **not** contain overlapping elements. Also passage results may **not** be overlapping. *Please note that submitted runs containing results from interleaved articles will be disqualified, as will submitted runs containing overlapping results.*

Summarizing: RELEVANT IN CONTEXT TASK returns a ranked list of articles. For each article, it returns an unranked **set** of results, covering the relevant material in the article. Overlap is not permitted.

1.3 BEST IN CONTEXT TASK

1.3.1 Motivation for the Task

The scenario underlying the BEST IN CONTEXT TASK is to find the best-entry-point for starting to read articles with relevance. As a result, even an article completely devoted to the topic of request, will only have one best starting point to read. The INEX 2007 BEST IN CONTEXT TASK asks systems to find the XML elements or passages that corresponds to these best-entry-points. The task makes a number of assumptions:

Display single result per article.

Users consider the article as the most natural unit, and prefer to be guided to the best point to start to read the most relevant content.

What we hope to learn from this task is: How does the BEST IN CONTEXT TASK differ from the RELEVANT IN CONTEXT TASK? How do best-entry points relate to the relevance of elements (FOCUSED TASK)? How do structural constraints in the query help retrieval?

1.3.2 Results to Return

The aim of the BEST IN CONTEXT TASK is to first identify relevant articles (the fetching phase), and then to identify the element corresponding to the best entry points for the fetched articles (the browsing phase). In the fetching phase, articles should be ranked according to their topical relevance. In the browsing phase, we have a single element or passage whose first content corresponds to the best entry point for starting to read the relevant information in the article. Note that there is no implied end-point: if (the start of) a paragraph is returned, it's not indicating that the reader should stop at the end of the paragraph. Similarly, although we request a complete passage with start and end-point out of practical convenience, the end-point of a passage result is ignored in the evaluation. The `/article[1]` element itself may be returned in case it is the best entry point, otherwise it will implied by any result from a given article. *Please note that submitted runs containing multiple results per article will be disqualified.*

Summarizing: BEST IN CONTEXT TASK returns a ranked list of articles. For each article, it returns a **single** result, representing the best entry point for the article with respect to the topic of request.

1.4 Passage Retrieval and Structured Queries

Within the INEX 2007 Adhoc retrieval tasks, we invite participants to experiment with two sets of different retrieval approaches:

- XML element retrieval versus passage retrieval.
- Standard keyword query (CO) retrieval versus structured query (CAS) retrieval.

1.4.1 Elements or arbitrary passages

XML element retrieval makes use of the document structure to determine the potential units of retrieval. The document structure is capturing both semantic labels (think of a link, a figure, or a figure caption) or ways of structuring the text (think of the sectioning structure). For example, in case of the XML Wikipedia collection, the sectioning structure of the document reflects the author's view on which parts of the text naturally group together and form a coherent subpart of the article. But how useful is the document structure to single out the relevant text for a particular search request? For example, a section element may be too short, and the relevant text should also contain the last paragraph of a preceding section. Or, alternatively, a paragraph element may be too long since all relevant information is contained in the first two sentences. To answer such questions, INEX 2007 also allows arbitrary passages—a result starting anywhere in the content of an element, and ending anywhere in the content of the same or another element later in document order.

At INEX 2007 there is no separate passage retrieval task, but for all three tasks arbitrary passages may be returned instead of elements. Although both types of retrieval may be used for each task, the best performing element and passage runs will also be reported. The type of results, XML elements or arbitrary passages, is recorded in submission format.

1.4.2 Structured Queries

Queries with content-only conditions (CO queries) are requests that ignore the document structure and contain only content related conditions, e.g. only specify what an element 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 may not care about the structure of the result components or may not be familiar with the exact structure of the XML documents. CAS queries are more expressive topic statements that contain explicit references to the XML structure, and explicitly specify the contexts of the user's interest (e.g. target elements) and/or the context of certain search concepts (e.g. containment conditions). More precisely, a CAS query contains two kinds of structural constraints: where to look (i.e. the support elements), and what to return (i.e. the target elements). The structural constraints are considered as structural hints, and similar to CO queries the elements will be assessed using the ⟨`narrative`⟩ part of the topics. Runs using CO queries and runs using CAS queries will be merged to create the assessment pool (this will in fact improve the pool quality).

At INEX 2007 there is no separate CAS task, but the vast majority of topics have both a keyword CO query and a structured CAS query.[1] As noted above, for all the tasks, we want to find out if, when and how the structural constraints in the query have an impact on retrieval effectiveness. Although both types of queries may be used for each task, mixing runs with both query types, also the best performing CAS query runs (restricted to topics containing a CAS query), and the best CO query runs will be reported. The use of CO/CAS query fields is recorded in submission format.

2 Result Submission

Fact sheet:

- For all three tasks, we allow up to 3 CO submissions, and up to 3 CAS submissions. That is, a participant can never submit more than 18 runs in total. For all runs, we allow element or passage results.
- All participants are invited to submit title-only runs (free choice between the CO title and the CAS title, and element or passage results) in a common submission format (details below), which allows up to 1,500 results per topic.
- There are additional requirements on the submissions the tasks:
  - FOCUSED TASK: for the same topic, results may not be overlapping.
  - RELEVANT IN CONTEXT TASK: articles may not be interleaved, and results may not be overlapping.
  - BEST IN CONTEXT TASK: only **one single** result per article is allowed.

  Runs that violate these requirements in any way, will be disqualified.

2.1 INEX 2007 Topics

There is only one set of topics to be used for all adhoc retrieval tasks at INEX 2007. The format of the topics is defined in the following DTD:

```
<!ELEMENT inex_topic
  (title,castitle?,mmtitle?,description,narrative)>
<!ATTLIST inex_topic
  id     CDATA #REQUIRED
  ct_no  CDATA #REQUIRED
>
<!ELEMENT title            (#PCDATA)>
<!ELEMENT castitle         (#PCDATA)>
<!ELEMENT mmtitle          (#PCDATA)>
<!ELEMENT description      (#PCDATA)>
<!ELEMENT narrative        (#PCDATA)>
```

[1]Of course, any CO query can be directly rephrased as a CAS query `//*[about(.,"`*CO query*`")]` using the tag wildcard `*` that matches any element.

The submission format will record the precise topic fields that are used in a run. Participants are allowed to use all fields, but only runs using either the ⟨title⟩, ⟨castitle⟩, or ⟨description⟩ fields, or a combination of these, will be regarded as truly *automatic*, since the additional fields will not be available in operational settings.

The ⟨title⟩ part of the INEX 2007 topics should be used as queries for the CO submissions. The ⟨mmtitle⟩ part of the INEX 2007 topics is dedicated for use in the INEX 2007 Multimedia Track. The ⟨castitle⟩ part of the INEX 2007 topics should be used as queries for the CAS submissions. In the number of runs allowed to be submitted, runs using more fields than the ⟨title⟩ (or ⟨castitle⟩) will still be regarded as an CO (or CAS) submission.

Since the comparative analysis of CO and CAS queries is a main research question at INEX 2007, we encourage participants to submit runs using only the ⟨title⟩ field (CO query) or only the ⟨castitle⟩ field (CAS query). We do not outlaw the use of the other topic fields, to allow participants to conduct their own experiments involving them, and since such deviating runs may in fact improve the quality of the assessment pool.

2.2 Runs

For each of the three tasks, we allow up to 3 CO submissions, and up to 3 CAS submissions. The results of one run must be contained in one submission file (i.e. up to 18 files can be submitted in total). A submission may contain up to 1,500 retrieval results for each of the INEX topics included within that task.

There are however a number of additional task-specific requirements.

For the FOCUSED TASK, it is not allowed to retrieve elements or passages that contain text already retrieved by another result. For example, within the same article, the element `/article[1]/section[1]` is disjoint from `/article[1]/section[2]`, but overlapping with all ancestors (e.g., `/article[1]`) and all descendants (e.g., `/article[1]/section[1]/p[1]`).

For the RELEVANT IN CONTEXT TASK, articles may not be interleaved. That is, if a result from article `a` is retrieved, and then a result from a different article `b`, then it is not allowed to retrieve further results from article `a`. Additionally, it is not allowed to retrieve results than contain text already retrieved by another result (similar to the FOCUSED TASK). Note also that for this task the `/article[1]` result is implied by any result from the article, and need not be returned.

For the BEST IN CONTEXT TASK, only a single element or passage per article is allowed. To allow for a single submission format, we request a complete passage result, but the end-point will be ignored in the evaluation. The `/article[1]` element may be returned in case it is regarded as the best place to start reading, otherwise it is implied by any other result from this article.

2.3 Submission format

For relevance assessments and the evaluation of the results we require submission files to be in the format described in this section. The submission format for all tasks is defined in the following DTD:

```
<!ELEMENT inex-submission (topic-fields, description, collections, topic+)>
<!ATTLIST inex-submission
  participant-id CDATA #REQUIRED
  run-id         CDATA #REQUIRED
  task           (Focused | RelevantInContext | BestInContext ) #REQUIRED
  query          (automatic | manual) #REQUIRED
  result-type    (element | passage) #REQUIRED
>
<!ELEMENT topic-fields EMPTY>
<!ATTLIST topic-fields
  title             (yes|no) #REQUIRED
  mmtitle           (yes|no) #REQUIRED
  castitle          (yes|no) #REQUIRED
  description       (yes|no) #REQUIRED
  narrative         (yes|no) #REQUIRED
>
<!ELEMENT description (#PCDATA)>
<!ELEMENT topic (result*)>
```

```
<!ATTLIST topic topic-id CDATA #REQUIRED >
<!ELEMENT collections (collection+)>
<!ELEMENT collection (#PCDATA)>
<!ELEMENT result (in?, file, (path|passage), rank?, rsv?)>
<!ELEMENT in (#PCDATA)>
<!ELEMENT file(#PCDATA)>
<!ELEMENT path (#PCDATA)>
<!ELEMENT passage EMPTY>
<!ATTLIST passage
  start            (#PCDATA) #REQUIRED
  end              (#PCDATA) #REQUIRED
>
<!ELEMENT rank (#PCDATA)>
<!ELEMENT rsv (#PCDATA)>
```

Each submission must contain the participant ID of the submitting institute (available at the INEX website `http://inex.is.informatik.uni-duisburg.de/2007/ShowParticipants.html`), a run ID (which must be unique for the submissions sent from one organization – also please use meaningful names as much as possible), the identification of the task (e.g. Focused, etc), the identification of whether the query was constructed automatically or manually from the topic, and whether XML elements or passages have been retrieved. Furthermore, the used topic fields must be indicated in the ⟨`topic-fields`⟩ tag. Moreover, each submitted run must contain a description of the retrieval approach applied to generate the search results. A submission contains a number of topics, each identified by its topic ID (as provided with the topics).

For compatibility with the heterogeneous collection track, the ⟨`collections`⟩ tag is mandatory. There should be with ⟨`collections`⟩ at least one ⟨`collection`⟩ tag, which is by default set to ”wikipedia” for the adhoc track. The ⟨`in`⟩ tag is optional for the adhoc track (⟨`in`⟩ states from which collection each result comes from).

For each topic a maximum of 1,500 results may be included per task. A result element is described by a file name and an element path, and it may include rank and/or retrieval status value (rsv) information. For the adhoc retrieval task, ⟨`collection`⟩ is set to ”wikipedia”. Here is a sample submission file for the FOCUSED TASK:

```
<inex-submission participant-id="12" run-id="VSM_Aggr_06"
  task="Focused" query="automatic" result-type="element">
  <topic-fields title="no" castitle="yes" description="no"
    narrative="no"/>
  <description>Using VSM to compute RSV at leaf level combined with
   aggregation at retrieval time, assuming independence and using
   augmentation weight=0.6. Top-down removal of overlapping
   elements</description>
  <collections>
    <collection>wikipedia</collection>
  </collections>
  <topic topic-id="01">
    <result>
      <file>9996</file>
      <path>/article[1]/name[1]</path>
      <rsv>0.67</rsv>
    </result>
    <result>
      <file>9996</file>
      <path>/article[1]/body[1]/p[1]</path>
      <rsv>0.1</rsv>
    </result>
    [ ... ]
  </topic>
  <topic topic-id="02">
    [ ... ]
  </topic>
  [ ... ]
</inex-submission>
```

Rank and RSV The rank and rsv elements are provided for submissions based on a retrieval approach producing ranked output. The ranking of the result elements can be described in terms of:

- Rank values, which are consecutive natural numbers, starting with 1. Note that there can be more than one element per rank.
- Retrieval status values (RSVs), which are positive real numbers. Note that there may be several elements having the same RSV value.

Either of these methods may be used to describe the ranking within a submission. If both rank and rsv are given, the rank value is used for evaluation. These elements may be omitted from a submission if a retrieval approach does not produce ranked output. In case there is no complete ranking specified by the submission, the results are evaluated in arbitrary order.

File and path Since XML retrieval approaches may return arbitrary results from the documents of the INEX collection, we need a way to identify these nodes without ambiguity.

We will first explain XML element results, which are identified by means of a file name and an element (node) path specification, which must be given in XPath syntax. The file names in the Wikipedia collection uniquely define an article, so there is no need for including the directory in which the file resides (in contrast with the earlier IEEE collection). The extension .xml must be left out. Example:

```
<file>9996</file>
```

Element paths are given in XPath syntax. To be more precise, only fully specified paths are allowed, as described by the following grammar:

Path	::= ’`/`’ *ElementNode Path* | ’`/`’ *ElementNode* | ’`/`’ *AttributeNode*
ElementNode	::= ElementName *Index*
AttributeNode	::= ’`@`’ AttributeName
Index	::= ’`[`’ integer ’`]`’

Example:

```
<path>/article[1]/body[1]/section[1]/p[1]</path>
```

This path identifies the element which can be found if we start at the document root, select the first ”article” element, then within that, select the first ”body” element, within which we select the first ”section” element, and finally within that element we select the first ”p” element. Important: XPath counts elements starting with 1 and takes into account the element type, e.g. if a section had a title and two paragraphs then their paths would be given as: `title[1]`, `p[1]` and `p[2]`.

A result element may then be identified unambiguously using the combination of its file name and element path. Example:

```
<result>
  <file>9996</file>
  <path>/article[1]/body[1]/section[1]/p[1]</path>
  <rsv>0.9999</rsv>
</result>
```

Passage paths are given in the same XPath syntax, but allow for an optional character-offset.

PassagePath	::= *Path* | *Path* ’/text()’ *Index* `.`’ *Offset*
Offset	::= integer

The following example is effectively equivalent to the example element result above. Since we want to start and end at an element boundary, we can use the same path expressions as above.

```
<result>
  <file>9996</file>
  <passage start="/article[1]/body[1]/section[1]/p[1]"
     end="/article[1]/body[1]/section[1]/p[1]"/>
  <rsv>0.9999</rsv>
</result>
```

Note the the start attribute will refer to the beginning of an element (or its first content), and the end attribute will refer to the ending of an element (or its last content).

In the next example, however, the result starts at the second sentence of the paragraph and continues until a list item in list below the paragraph.

```
<result>
  <file>9996</file>
  <passage start="/article[1]/body[1]/section[1]/p[1]/text()[1].85"
     end="/article[1]/body[1]/section[1]/normallist[1]/item[2]/text()[2].106"/>
</result>
```

The offset can be placed anywhere in the text node starting from 0 (first character) to the *node-length* (last character).

2.4 Example: Identifying results

This section uses an example document to explain how results are to be identified.

XML We use a small excerpt from `12.xml`, a page on "Anarchism":

```
<item>
<collectionlink xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple"
  xlink:href="58198.xml">
Mikhail Bakunin
</collectionlink>,
<emph2>
<outsidelink xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple"
  xlink:href="http://dwardmac.pitzer.edu/Anarchist_Archives/bakunin/">
God and the State
</outsidelink></emph2>,
<emph2>
The Paris Commune and the Idea of the State
</emph2>, others
</item>
```

This is in fact element `/article[1]/body[1]/section[4]/section[2]/normallist[1]/item[3]` which was relevant for INEX 2006 topic 306.

XML and whitespace XML is very flexible in its handling of whitespace, i.e., the following two documents are usually regarded as identical.

```
<a>
   <b/>                          <a><b/></a>
</a>
```

However, strictly speaking the document on the left contains whitespace content (newlines, tabs, spaces) which is not present in the document on the right. That is, the element `<a>` in the document on the left contains first a newline and some spaces, then an empty `<b>` element, and then again a newline.

The whitespace which is only inserted to make the XML file human readable has to be ignored, but white-space used to format the content of an element should be retained.

DOM tree We view the XML document using the Document Object Model (DOM), which results in a tree of root, element, attribute, text, and entity nodes like:

```
root
  |___element 'item'
        |___element 'collectionlink'
        |     |  \___attribute 'type' [...]
        |     |  \___attribute 'href' [...]
        |     |___text '\nMikhail Bakunin\n'
        |___text ', \n'
        |___element 'emph2'
```

```
|     |___element 'outsidelink'
|          |  \___attribute 'type' [...]
|          |  \___attribute 'href' [...]
|          |___text '\nGod and the State\n'
|___text ', \n'
|___element 'emph2'
|     |___text '\nThe Paris Commune and the Idea of the State\n'
|___text ', others \n'
```

In the DOM tree model, all content or text is in special text nodes.

Again, strictly speaking there would be an additional text node between the start tag of `<item>` and the start tag of `<collectionlink>` containing only newline whitespace. We completely ignore text nodes containing *only* whitespace (spaces, newlines, tabs).[2] As a result the `<item>` element has three text nodes as direct children. Note that all whitespace in other text nodes is preserved, including the newlines (indicated with `\n` in the tree).

Locating elements and text nodes We can use the XPath style location paths for locating elements. Since all content is in text nodes, passages start either on an element or inside a text node. For example, if we want our passage to start inside the first `<collectionlink>` we can locate the text node by `/item[1]/collectionlink[1]/text()[1]` and use any offset between 0 and 17 (the total character length including the newlines). For example, the last name `Bakunin` starts at offset `9`, and the passage

```
<passage start="/item[1]/collectionlink[1]/text()[1].9"
   end="/item[1]/collectionlink[1]/text()[1].16"/>
```

would precisely select the whole last name.

Mapping local offsets to global offsets The offsets used in the passages are all local to the text nodes. But since all textual content is in text nodes, we can use these to generate global offsets on the concatenation of all text nodes. Below we list all elements and text nodes with global start and end offsets for the example document:

```
/item[1] 0 97
/item[1]/collectionlink[1] 0 17
/item[1]/collectionlink[1]/text()[1] 0 17
/item[1]/text()[1] 17 20
/item[1]/emph2[1] 20 39
/item[1]/emph2[1]/outsidelink[1] 20 39
/item[1]/emph2[1]/outsidelink[1]/text()[1] 20 39
/item[1]/text()[2] 39 42
/item[1]/emph2[2] 42 87
/item[1]/emph2[2]/text()[1] 42 87
/item[1]/text()[3] 87 97
```

The length of (the content of) an element, in characters, is simply the end-offset minus the start-offset. For example, the `<item>` element contains 97 characters.

2.5 Result Submission Procedure

To submit a run, please use the following link: `http://inex.is.informatik.uni-duisburg.de/2007/` Then go to Tasks/Tracks, Adhoc, Submissions. The online submission tool will be available soon.

[2] Or in alternative terminology, these text nodes contain *element content whitespace*.