EXTIRP 2004: Towards Heterogeneity

Miro Lehtonen

Department of Computer Science,
P. O. Box 68 (Gustaf Hällströmin katu 2b),
FI–00014 University of Helsinki, Finland
Miro.Lehtonen@cs.Helsinki.FI

Abstract. The effort around EXTIRP 2004 focused on the heterogeneity of XML document collections. The subcollections of the heterogeneous track (het-track) did not offer us a suitable testbed, but we successfully applied methods independent of any document type to the original INEX test collection. By closing our eyes to the element names defined in the DTD, we created comparable runs and discovered improvement in the results. This was anticipated evidence for our hypothesis that we do not need to know the element names when indexing the collection or when returning full-text answers to the Content-Only type queries. Some problematic areas were also identified. One of them is score combination which enables us to combine elements of any size into one ranked list of results given that we have the relevance scores of the leaf-level elements. However, finding a suitable score combination method remains part of our future work.

1 Introduction

One of our goals for the INEX 2004 project was to adapt our system to support heterogeneous XML collections without losing the retrieval accuracy achieved in 2003. Our system for XML retrieval — EXTIRP — has now been successfully modified: it is independent of any document type and, based on tests with the topics of 2003, the accuracy has even improved. However, not all components of EXTIRP adjusted to the changes equally well. For example, the score combination algorithm of EXTIRP showed its weakness in a significant decline in average precision. Consequently, the best result sets were those that consisted of disjoint answers at the finest level of granularity. Another factor having a negative impact on EXTIRP 2004 was the lack of manpower, due to which we gave up a previous success story: query expansion.

During the course of the year, the research on heterogeneous XML collections was added to the number of fields where documents as a whole are too big for their users and use cases, e.g. to be indexed for information retrieval or displayed in a browser window, and where algorithms are needed for dividing the documents into fragments. We analysed a selection of existing algorithms that detect fragments inside documents and observed which properties could be applied to heterogeneous XML documents. Based on the analysis, we developed a novel method for dividing the collection into equi-sized fragments.

N. Fuhr et al. (Eds.): INEX 2004, LNCS 3493, pp. 372–381, 2005.

This paper is organised as follows. XML terminology and related vocabulary have had various interpretations in the history of INEX. Section 2 clarifies the terminology that is necessary in order to fully understand the rest of the paper. In Section 3, EXTIRP is briefly described. The problem of documents that are too big to be easily handled is addressed in Section 4, and the challenge of score combination in Section 5. Our runs for 2004 topics are described in Section 6 after which we draw conclusions in Section 7.

2 Common Misconceptions

The purpose of this section is to make our paper accessible to readers who are not familiar with XML terminology. Because of the confusion with vocabulary, the INEX document collection is often misrepresented; see [1, 2]. We will now explain what the collection looks like to XML-oriented people in order to have a common terminological basis with the reader.

XML documents in INEX. An *XML Document* is the biggest logical unit of XML. It can be stored in several XML files which are part of the physical structure of the document. When parsed into DOM[1] trees, each XML document only has one Document Node in the tree. The DOM trees representing the whole INEX collection have 125 Document Nodes because the collection consists of 125 XML documents. Each XML document contains one volume of an IEEE journal. It is misleading to look at the physical structure of the document collection which includes over 12,000 files, as most of the files contain an external entity instead of a whole XML document.

INEX articles. The concept of a *document* has changed because of XML. A document is no longer considered the atomic unit of retrieval. However, XML should have no effect on the concept of an *article*. It is true that there are 12,107 `article` elements in the document collection, but the number of actual articles is smaller. According to the common perception, many article elements do not have article content. Instead, they contain a number of other papers such as errata, lists of reviewers, term indices, or even images without any text paragraphs.

INEX tags. The specification for XML[2] defines three different kind of tags: start tags, end tags, and empty element tags. A DTD does not define any tags, but it does define *element types*. In the XML documents, though, each non-empty element contains two different tags. Counting the different tags in the collection (361) is very different from counting the different element type definitions in the DTD (192), different element types inside the article elements (178), or different element types in the whole collection (183).

Number of content models. The content models of the collection are defined in the DTD. Each element type definition contains the name of the element

[1] http://www.w3.org/DOM/
[2] http://www.w3.org/TR/REC-xml/

followed by its content model. Altogether 192 content models are defined in the DTD, but only 65 of those are different. Of the 65 different content models, only 59 appear in the articles of the collection. For example, the content models of element types `journal` and `books` are not allowed in article content, and elements such as `couple`, `line`, and `stanza` are not included in the collection at all.

DTD-independent methods. One of the goals of the het-track has been the development of DTD-independent methods. However, methods that are independent of the DTD are necessary only when no DTD is available. The problem of documents with several different DTDs or schema definitions is far more common. Methods that are independent of the document type may require the presence of a DTD or a schema definition in order to make the most of it. For example, link relations inside one document are easily resolved with a DTD by reading which attributes are of the ID type and which are of the type IDREF or IDREFS. Moreover, the independence of the DTD does not exclude the dependence on a schema definition. Because the lack of a DTD does not increase the amount of heterogeneity in the collection but different document types do, methods independent of document types suffice for heterogeneous XML collections.

3 System Overview

EXTIRP specialises in full-text search of XML documents and does not support any structural conditions in the queries. Only full-text is indexed, and only full-text is queried. It is thus natural to focus on CO-type topics when evaluating our system.

EXTIRP uses two static indices: an inverted word index and an inverted phrase index. The *key* in the word index is the identifier of a stemmed word and the corresponding *value* contains a list of XML fragment identifiers indicating where in the collection the word occurs. The phrase index contains similar information about those phrases that are considered *Maximal Frequent Sequences* [3].

Before the static indices are built, we divide the document collection into document fragments which results in a collection of disjoint fragments. Each fragment represents the smallest atomic unit of content that can be retrieved. In other words, only the finest level of granularity is indexed. The fragments are not leaf nodes in the document tree but whole sub-trees that contain element and text nodes. How the indexed fragments are selected is described in Section 4 in more detail.

Upon query processing, two normalised similarity scores are computed for each fragment: word similarity and phrase similarity. These two scores are aggregated into a Retrieval Status Value (RSV), according to which the fragments are ranked. At this point, the result list contains a ranked list of relatively small answers for each query. By combining the scores of these fragments, we can replace them with bigger fragments in the list. For example, the score of a section is computed using the scores of each child element, e.g. a paragraph, inside the

section. If the section seems more relevant than the paragraphs in the light of RSVs, the paragraph-size fragments are replaced with the particular section-size fragment and ranked accordingly. By combining the scores of adjacent fragments, all the scores are propagated upward in the document hierarchy all the way to the article level. This process has turned out to be remarkably challenging with the fragment collections of 2004.

A more detailed description of the 2003 version of EXTIRP was presented in the INEX Workshop in 2003 [4]. A novelty in EXTIRP 2004 is its independence of any document type. The major changes from 2003 are described in Section 4.2.

4 Division into Fragments

The INEX test collection of IEEE journals is organised into 125 volumes, each of which is represented by the logical unit of storage called an XML document. The character sizes of the volumes vary between 405,832 and 7,385,546 characters, which in most cases is considered too coarse of a granularity in XML retrieval. A finer granularity level has to be reached if smaller units for indexing and retrieval are required.

In recent research, several different purposes have been presented for dividing structured documents into fragments. We will first look into the state of the art and see whether these algorithms could be applied to XML retrieval. Then we will describe how EXTIRP selects the fragments to be indexed.

4.1 Related Work

Ramaswamy et al. presented a fragment detection algorithm motivated by performance issues [5]. Web pages were divided into fragments according to certain criteria, e.g. how many other fragments share its content, whether it is maximal or content of another fragment, and how frequently the content is updated in comparison with other fragments on the web page. A minimum size was also set on the qualifying fragments which were considered cost-effective cache units. This algorithm is not directly applicable to a static collection of XML documents because we cannot measure any lifetime characteristics in an unchanging set of documents. Moreover, the potential fragments in the INEX test collection are unique and not shared among other fragments. Some noteworthy details in their studies include the *minimum size of a detected fragment* which is used to exclude the smallest segments of web pages from being detected as candidate fragments. The values of 30 bytes and 50 bytes seemed reasonable in their experiments. Note also the unit of the minimum size: it is not the number of words, terms, or tokens but simply the bytesize which roughly corresponds to the number of characters in the element content.

In 1999, Jon Kleinberg introduced the HITS algorithm [6] that categorises web pages into *hubs* and *authorities*. A page with a good collection of links has a high hub score whereas a popular page or an authoritative source of information has a high authority score. However, hubs rarely contain links related to a single topic. On the contrary, hubs can cover wide ranges of topics which makes them

mixed hubs. Chakrabarti developed an algorithm that disaggregates web pages considered mixed hubs into coherent regions by segmenting their DOM trees [7]. He uses the HITS algorithm for topic distillation by computing the hub score for each subtree in the DOM tree instead of computing the hub score for the whole document. The resulting fragments are pure hubs that are highly specific answers to appropriate queries. Topic distillation is a proper but not sufficient criterion for dividing XML documents into fragments. Applying Chakrabarti's hyperlink-based algorithm directly to any of the INEX test collections is inappropriate in the absence of hyperlinks. Nevertheless, the potential for the application of the HITS algorithm lies in the citations and cross-references which can be seen as links between different subtrees representing articles, sections, as well as other document fragments in the collection.

Another need for document segmentation comes from devices that can only display a small amount of information at a time, either because of a small display size or a low resolution. Hoi et al. developed a document segmentation and presentation system (DSPS) that automatically divides a web page into logical segments based on the display size, and document structure and content [8]. The segments have their own branches in a *content tree* into which HTML documents are first converted. The HTML tags are classified into different categories and interpreted accordingly. Applying this algorithm to arbitrary XML documents requires a thorough analysis of the document type. The maximum size of an HTML segment is also rather small. A major difference from the algorithms of Ramaswamy and Chakrabarti is that the resulting HTML segments do not cover all of the original documents: Segments that are very small or very different from the adjacent segments are removed from the content tree. The idea of discarding irrelevant content can be successfully applied to XML retrieval and indexing XML documents, as well.

4.2 Size-Based Division

In 2003, the indexed fragments were selected by the name of the corresponding XML element. After carefully studying the DTD of the collection, we could see which element types represented section-level fragments (sec, ss1, ss2, etc.) and which element types were common at the paragraph-level (p, ip1, etc.). Similar approaches have been common among other participants. For example, the selection of index nodes in the HyREX system is strictly based on element names [9]. Approaches relying on element names do not scale well to fit the needs of heterogeneous document collections. Analysing each DTD is hardly an option as the number of document types increases. Furthermore, it will shortly be shown that better results can be achieved with methods that are independent of the document type.

The EXTIRP algorithm for dividing XML documents into fragments has two parameters: the maximum and minimum size of an indexed fragment. The fragments are selected by traversing the document tree in preorder. If the current node is small enough and qualifies as a full-text fragment, it will added to the fragment collection, after which the following node is tested. The fragments in the fragment collection are disjoint because all the subtrees of qualifying fragments

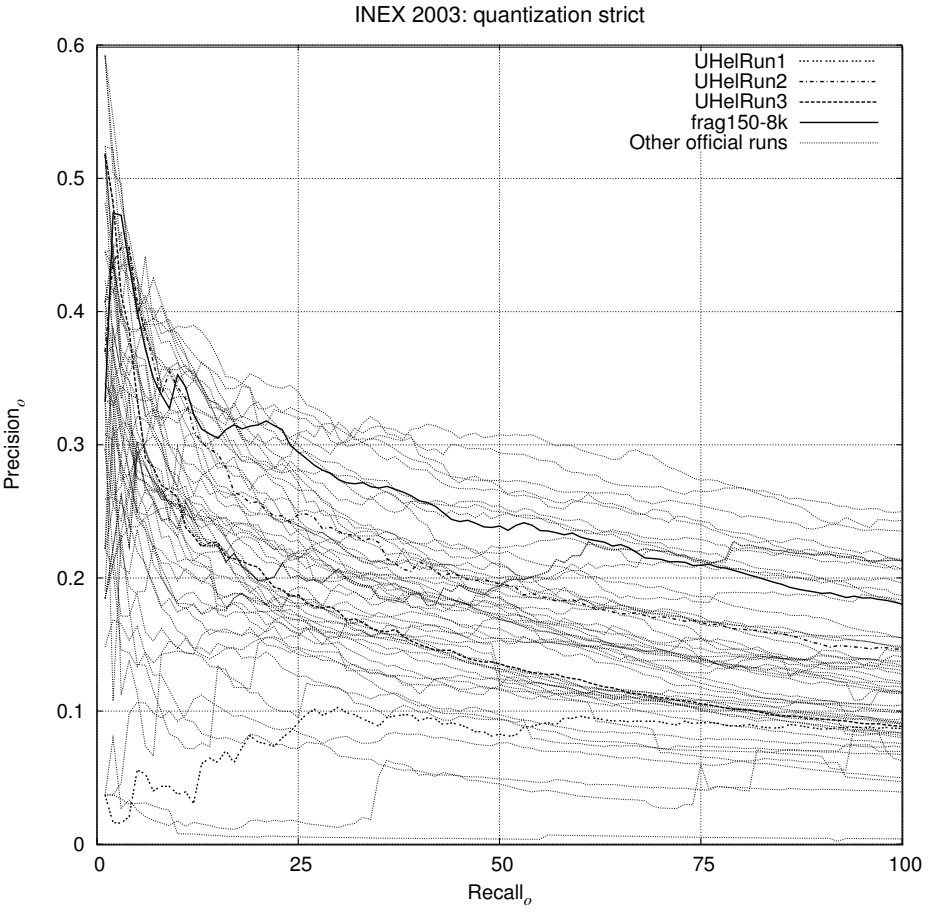

Fig. 1. Precision$_o$ of all the official runs of 2003 at recall levels 1–100. The runs UHelRun1–UHelRun3 are the official submissions of University of Helsinki in 2003, and the thick continuous line represents the current performance level of EXTIRP without score combination or query expansion

are skipped. If the current node does not qualify, its children will be tested until they are either small enough or too small. Consequently, irrelevant fragments, e.g. those that are unlikely answers to any full-text query, are discarded in a similar fashion to that of the DSPS by Hoi et al. removes irrelevant segments [8]. The algorithm on the whole is independent of any document type and also applicable to the INEX test collection.

We compare the precision of four different runs drawn with a solid line in Figure 1. Three of the runs were our official submission in 2003, and the fourth one is based on a fragment collection with the minimum size of a fragment set to 150 characters and maximum to 8,000 characters in text nodes. The other curves represent the official submissions of other participants for the CO topics. Only the first 100 recall answers of each run are considered here. The run that was

based on EXTIRP 2004 methods shows the best performance (see the thickest solid line). Our official runs of 2003 have a significantly lower precision at most recall levels. The curves for the generalised quantisation show similar results.

More evidence for the results can be found in Table 1 where the Generalised Recall measure is shown for each run. The online evaluation tool[3] was used for computing the GR score. No score combination method is applied to the disjoint fragments in the run 'Fragments150-8k', which shows in the 0.0 List-Based Overlap (LBO).

Table 1. A run with size-based division and no score combination or query expansion compared with the official runs of University of Helsinki

Run	LBO	strict -o	strict -s	generalised -o	generalised -s	GR
UHel-Run1	39.6	0.0484	0.0358	0.0340	0.0270	9.70
UHel-Run2	28.0	0.1135	0.0866	0.0716	0.0586	11.63
UHel-Run3	10.5	0.1058	0.0787	0.0537	0.0418	8.60
Fragments150-8k	0.0	0.1170	0.0924	0.0831	0.0658	27.68

5 Score Combination

In the retrieval model of EXTIRP, similarity-based scores are computed for XML elements of the finest level of granularity. If bigger answers are desired, the comparable scores need to be computed for all the ancestor elements of the minimal fragments in the document tree. We have utilised the equation (1) for score combination and propagated the scores upward all the way to the article element.

$$score(p) = \frac{\sum score(c)}{size(p)^{UPF}} \tag{1}$$

The adjustable parameter in the equation is the Upward Propagation Factor (UPF), with which we can prioritise answers or a particular size. The score of the parent element p is sum of the scores of its child elements c when the UPF has a value of 0. The obvious shortfall of the value of 0 is that elements with big quantities of content are unjustly favoured. With bigger UPF values, we can even out this effect and focus on the quality of the content instead. The score of the child element is the RSV which was computed earlier. The size of an element can be measured with a variety of metrics. Our implementation defines the size as the bytesize of the fragment. Other possible size units include the number of tokens, number of words, number of characters, etc.

In order to test the score combination method, we created a fragment collection of disjoint fragments with the maximum size set to 20,000 and minimum size to 200 characters. After the RSV was computed for each fragment, the score

[3] http://inex.lip6.fr/2004/metrics/

combination method was applied. The results with different values for the UPF are shown in Table 2. There is no room for questioning the negative effect of our score combination method, because the best average precision was achieved without score combination and it is clearly better than the other scores. Tests with other fragment collections gave similar results.

Table 2. Upward propagation applied to a fragment collection "Fragments200-20k"

UPF	strict -o	strict -s	generalised -o	generalised -s
2.0	0.0058	0.0057	0.0081	0.0079
1.0	0.0270	0.0288	0.0277	0.0305
0.7	0.0536	0.0437	0.0500	0.0446
0.6	0.0584	0.0443	0.0451	0.0379
0.5	0.0565	0.0408	0.0395	0.0318
0.4	0.0546	0.0379	0.0351	0.0275
0.2	0.0509	0.0351	0.0294	0.0219
—	0.0954	0.0728	0.0705	0.0562

The reasons for the deteriorating effect of the score combination possibly include our definition for the size of a fragment. Comparing the bytesizes is simple, but at the same time, we completely ignore the qualities of the element content. Fragments with mathematical equations or charts and tables contain significant amounts of XML markup — element tags and attributes — which do increase the bytesize of a fragment but do not improve the value of the content.

6 Our Runs

The results presented in earlier sections of this paper were not available at the time of run submission. We have, however, learned since then and found reasons for the decline in the precision of the three runs for the CO topics that we submitted. Three different factors have been identified:

Query expansion. Although query expansion turned out to improve the results in 2003, we did not have enough resources to repeat the success.

Score combination. As seen in Section 5, the results deteriorated after the score combination process. Adjusting the UPF did not have a great impact on the results.

Query processing. Only the title of the topic was used for the runs of 2004. In 2003, also the description of the topic was used for similarity computation. This change should not show in the overall results because it concerned all the participants of 2003, but it does show in the absolute precision curves.

We did not submit any runs for the het-track topics. As EXTIRP 2004 specialises in full-text search, it also does not index any data-oriented content. The

XML documents consisting of bibliographic data have no such full-text content that qualifies for the fragment index. It was not found meaningful to submit runs for a CO topic that would be identical with the corresponding runs for the ad-hoc track.

7 Conclusions

The research on EXTIRP 2004 was useful in that we developed methods for heterogeneous collections of XML documents. EXTIRP 2004 does not require any information about the document type, e.g. element names have no importance. A common approach to indexing the INEX test collection, e.g. that of EXTIRP 2003, has involved the definition of a set of elements that are considered *index nodes*. The novelty of EXTIRP 2004 is that the indexed fragments are selected by their size, and the tag name between brackets is disregarded. Size-based division is an appropriate technique in the case of the IEEE article collection, at least, but it is also capable of handling arbitrary XML collections. Whether a decent performance level can be achieved by running EXTIRP on heterogeneous XML will have to be tested in the future as suitable testbeds become available.

Acknowledgements

The author would like to thank Oskarie Heinonen and Roman Yangarber for technical help with Gnuplot and LaTeX, Diana Cousminer for proofreading, and the anonymous reviewers for their valuable comments.

References

1. Fuhr, N., Goevert, N., Kazai, G., Lalmas, M., eds.: INEX: Evaluation Initiative for XML retrieval - INEX 2002 Workshop Proceedings. DELOS Workshop, Schloss Dagstuhl (2003)
2. Fuhr, N., Lalmas, M.: Report on the INEX 2003 Workshop, Schloss Dagstuhl, 15-17 December 2003. SIGIR FORUM **38** (2004) 42–47
3. Ahonen-Myka, H.: Finding All Frequent Maximal Sequences in Text. In: Proceedings of the 16th International Conference on Machine Learning ICML-99 Workshop on Machine Learning in Text Data Analysis, Ljubljana, Slovenia, J. Stefan Institute, eds. D. Mladenic and M. Grobelnik (1999) 11–17
4. Doucet, A., Aunimo, L., Lehtonen, M., Petit, R.: Accurate Retrieval of XML Document Fragments using EXTIRP. In: INEX 2003 Workshop Proceedings, Schloss Dagstuhl, Germany (2003) 73–80
5. Ramaswamy, L., Iyengar, A., Liu, L., Douglis, F.: Automatic detection of fragments in dynamically generated web pages. In: 13th World Wide Web Conference (WWW - 2004). (2004) 443–454
6. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. Journal of the ACM **46** (1999) 604–632

7. Chakrabarti, S.: Integrating the document object model with hyperlinks for enhanced topic distillation and information extraction. In: Proceedings of the tenth international conference on World Wide Web, ACM Press (2001) 211–220
8. Hoi, K.K., Lee, D.L., Xu, J.: Document visualization on small displays. In: Proceedings of the 4th International Conference on Mobile Data Management (MDM 2003), Berlin, Germany, Springer-Verlag (2003) 262–278
9. Abolhassani, M., Fuhr, N., Malik, S.: HyREX at INEX 2003. In: INEX 2003 Workshop Proceedings. (2003) 49–56