Appears in "Workshops of the 24th British National Conference on Databases" (BNCODwebim).

Improving Content-Oriented XML Retrieval by Exploiting Small Elements

Philipp Dopichaj
University of Kaiserslautern
Gottlieb-Daimler-Str., 67663 Kaiserslautern, Germany
dopichaj@informatik.uni-kl.de

Abstract

XML element retrieval aims at finding the best elements satisfying a user's information need. Elements spanning only a few words, like titles or italicized phrases, are not in themselves useful results, but they can support the relevance of their enclosing elements. For example, if a section's title contains the key words from the user's query, the title itself is unlikely to be a useful result, but the section is very likely to be useful. This paper provides an overview of methods for exploiting small elements for better retrieval results, highlighting their respective advantages and disadvantages. Using the INEX testbed, we show that small elements can indeed provide useful retrieval hints, and we evaluate the trade-offs.

1. Introduction

In content-oriented XML retrieval, documents are not considered atomic entities as they are in traditional text-based information retrieval: A retrieval result can not only contain complete documents, but also elements such as chapters or paragraphs. This is convenient for the user, who does not have to examine large chunks of irrelevant context information to find the possibly small units of information he seeks, but it gives rise to the new issue of handling overlap [9]. Due to the hierarchical structure of XML documents (like document–chapters–sections–paragraphs), one matching result may be embedded in another one. For example, if a section contains the keywords from the user's query, the enclosing chapter and document also contain these keywords, so they also match the query (possibly to a lower degree).

Among these overlapping elements are elements that are only a few words long, for example, running titles or italicized phrases. These elements contain too little text to be useful retrieval results on their own, but they can provide valuable hints about their enclosing elements.

1.1. Contribution

In this paper, we discuss two approaches to improving retrieval quality by using these small elements: One is based on using the element names as a hint; the names in the documents are chosen by their authors, so this should be the most accurate method. On the other hand, this approach requires preparative work of the search engine's administrator as well as good knowledge of the documents' schema, so it cannot be used in all circumstances. Because of this, we explore a second approach that only looks at general meta data like length and position of the elements. It works by examining small sub-trees from the retrieval result for adjusting the scores to better match the searcher's intentions. The method makes no assumptions about the schemas of the documents, as long as the logical structure of the text corresponds to the tagged structure of the XML files. Because of this, it should be possible to use it on heterogeneous collections of textual documents that fulfill this basic requirement; we do not expect the method to be useful for data-centric XML. In order to show that these methods can indeed improve retrieval quality, we evaluate their performance on the INEX 2005 data set, showing significant improvements.

Naïve implementations of these methods can exhibit bad performance, making their use infeasible for larger document collections. We address this problem by presenting adaptations that retain the good characteristics of the original methods and at the same time reduce the overall run time and index size.

1.2. Related Work

Overlapping results are natural in the context of semi-structured retrieval, so XML retrieval engines have to deal with this. On the one hand, overlapping results can be a nuisance to the user, so the retrieval engine should strive to deliver as little redundant content as possible. Kazai et al. [9] discuss the overlap problem in detail. Kekäläinen et al. [10] submitted retrieval runs with varying degrees of overlap to the workshop of the Initiative for the Evaluation of

XML Retrieval, INEX 2004, and found that returning overlapping elements appears to improve the score. INEX 2005 features a new task that explicitly disallows the submission of overlapping documents, CO.Focused [12].

On the other hand, overlapping results can be used to improve the quality of the retrieval results. Several researchers make use of the context of an element in order to improve retrieval quality. Ogilvie and Callan [14] use the context of an element by incorporating the parent's language model. Arvola et al. [1] use the context of retrieval nodes by incorporating evidence from other nodes in the same document into the retrieval status value (RSV) of a node. Their approach differs from ours in that they use the scores of the enclosing elements to alter the score of a node; only the score of these nodes is taken into account, and only the ancestors are considered.

Using small elements to affect a document's RSV is not new; in 1993, Salton et al. [17] use textual context for improving results for passage retrieval; this includes considering titles and section headings, and in 1997, Cutler et al. [2] investigate the use of heading tags in HTML document retrieval; the physical structure of HTML documents does not match the logical structure, so element retrieval is not relevant here. In the context of XML element retrieval, Robertson et al. [16] propose to inherit the RSVs of fields, for example, a complete article's RSV is influenced by the RSV of its title. Ramírez et al. [15] use small elements with high RSVs to adapt the RSV of the enclosing element, for example, a match in the section title will increase the corresponding section's RSV. Both approaches require preparation and knowledge of the DTD, because all these relationships must be modeled explicitly.

2. Using Small Elements

As we have seen in the previous section, there are several approaches to using the information contained in small elements. We use the following types of small elements for adjusting the RSVs:

Inline Elements: We assume that if an author encloses single words or short phrases in markup, this markup should be interpreted as some form of emphasis. For example, single words are sometimes italicized when they are defined in the text; the surrounding element should be rewarded if the user searches for this term.

Titles: Section or chapter titles can be seen as very concise summaries of the document part they belong to; thus, hits in titles should be rewarded. In many XML document formats (for example, DocBook or the format of the IEEE collection), titles are the very first child element.

The most straightforward way of using the information about the small elements is to index *all* elements in the collection – even small elements that are not good retrieval results on their own. Retrieval is then performed by using a traditional IR engine using the vector space model and adjusting the RSVs in a post-processing step, based on the small elements. Fig. 1 describes the retrieval process in more detail, along with examples of intermediate results.

There are several options for determining which elements are inline or title elements; we will focus on two basic approaches, based on the names of the elements respectively their lengths. Naïve implementations of these methods are slow and resource-hungry, so we describe an optimization that reduces both the space usage of the index and the retrieval time without negatively affecting retrieval quality; we will address this in Section 3.

2.1. Name-based

A collection-specific approach is to use the element names for adjusting the enclosing elements' RSVs. This approach has the advantage that it uses the semantic information provided by the document author, but it requires tailoring to the schema that is used in the document collection.

Deciding which element types to use and how to use them (how many levels should we go up if we have a hit in an `st` element?) is not trivial. Ramírez et al. [15] obtained a set of rules for the INEX 2005 collection by analyzing the assessments of previous retrieval experiments on the same test collection; in practice, this kind of relevance information will not be available.

We use the following list of elements (based on the list in [15]): `st` (section title), `it` (italics), `fig` (figure), and `fgc` (figure caption). We do not use propagation links because the experiments indicate that they do not help improve retrieval quality. We double an element's score if at least one of its children has a non-zero RSV and is of one of the given types; the support elements themselves are removed from the final result.

2.2. Length-based

For our submissions to INEX 2005, we used a different approach [3]: We assume that *all* short elements embedded in a longer text contain words that are relevant for the enclosing element. Obviously, this means that we completely discard the semantic information contained in the element names and run the risk of using short elements that are not used for emphasis. On the other hand, we do not need to manually determine the set of suitable element names, which makes us somewhat independent of the document schemas; as our experiments show, this heuristic approach works as well as the name-based approach for the

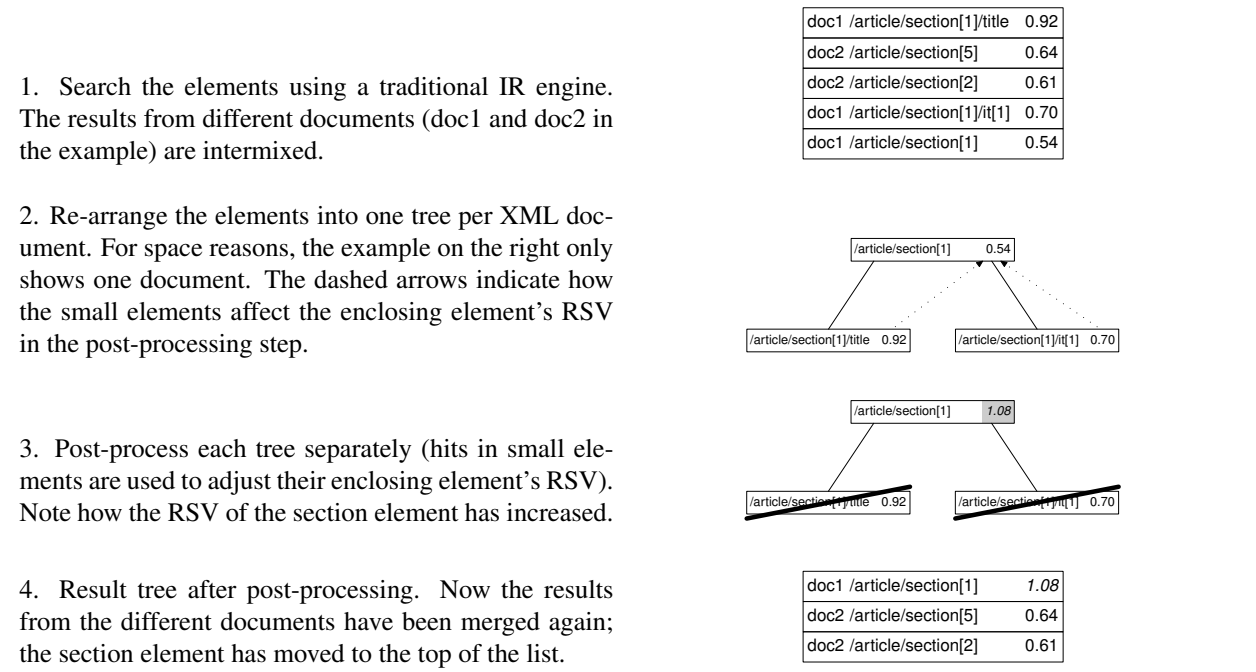

Figure 1: Intermediate results while searching

INEX 2005 test collection.

We use the following heuristics:

- All elements containing at most 40 words long are considered inline elements. The RSVs of parent elements containing inline hits is multiplied by 1.5.
- All elements that occur at the very start of their enclosing element, are at most 40 words long, and whose parent is at least 80 words long are considered title elements. This is a special case of the inline heuristic, but titles are a better indication of important words than inline elements, so we double the parent element's score if there is a hit in a title.

Although 40 words may seem like a high threshold, it has proved to lead to good results; the method is insensitive to small changes to this value, but how to determine a good value without experiments is still an open question.

3. Performance Improvements

Both of the methods from the preceding section can be seen as post-processing steps on the results of a traditional IR search engine. In order to exploit the small elements, they need to be stored in the index as separate "documents", leading to both a higher demand on hard-disk space and more "documents" that need to be searched (in both the collection of IEEE articles used for INEX 2005 and the Wikipedia collection used for INEX 2006, more than 90 % of all elements are at most 40 tokens, see Table 1).

As a remedy for this problem, we can omit the short elements from the index and emphasize their contents in the enclosing elements' text. This approach has the advantage of dramatically reducing the number of elements in the index, but the information about the element boundaries is lost. One important consequence is that the parameter (how long can an inline element be?) can be changed at run time for dynamic, but requires re-indexing for static.

For static indexing, the text of the small sub-elements is in effect appended to the enclosing element's text, resulting in an increased weight of the contained terms (see Figure 2).

As we can see in Table 1, although the number of fragments is reduced by more than 90 %, the total size of the index is only reduced by 40–50 %. Most important, however, is that the retrieval time goes down by more than 60 % without adversely affecting retrieval quality (as we shall see in Section 4).

One side-effect of the omission of small elements from the index is that the component frequency (the number of elements a given term occurs in, analogous to document frequency in traditional IR) becomes more meaningful: If all elements are indexed, the component frequency is overemphasized; now that only larger elements are indexed, the elements in the index are more likely to be sensible retrieval units. (This does not appear to have a noticeable effect on retrieval quality, however.)

```
<section>
<title>Section title</title>
<p>Longer paragraph with <emph>emphasized</emph> sub-elements. All elements shorter than three words will be appended to the parent.</p>
<p>Another paragraph, slightly shorter.</p>
</section>
```

(a) Example document

- **/section:**
  Section title
  Longer paragraph with emphasized sub-elements. All elements shorter than three words will be appended to the parent.
  Another paragraph, slightly shorter.
- **/section/title:**
  Section title
- **/section/p[1]:**
  Longer paragraph with emphasized sub-elements. All elements shorter than three words will be appended to the parent.
- **/section/p[1]/emph:**
  emphasized
- **/section/p[2]:**
  Another paragraph, slightly shorter.

(b) Indexed fragments for the example document

- **/section:**
  Section title
  Longer paragraph with emphasized sub-elements. All elements shorter than three words will be appended to the parent.
  Another paragraph, slightly shorter.
  Section title
- **/section/p[1]:**
  Longer paragraph with emphasized sub-elements. All elements shorter than three words will be appended to the parent.
  emphasized
- **/section/p[2]:**
  Another paragraph, slightly shorter.

(c) Indexed fragments for the example document; appended text is in *italics*. Note that the text of small elements is only appended to the parent element, not all ancestors.

Figure 2: Example of static indexing. All elements containing two words or less are regarded as inline elements.

Table 1: Static versus dynamic: Effects on index size and search time, INEX 2005 collection (accumulated disk usage of XML documents: 742 MB). A list of stop words was used in both cases. The average search time is for the official set of topics in that year.

	Number of indexed fragments	Index size (MB)	Avg. search time
IEEE collection (INEX 2005)			
Document index	16,801	95	0.8 seconds
Dynamic	7,833,451	750	16 seconds
Static/length-based	738,649	450	6 seconds
	(9 %)	(60 %)	(38 %)
Static/name-based	6,915,154	711	8 seconds
Wikipedia collection (INEX 2006)			
Document index	659,388	444	4 seconds
Dynamic	47,753,280	3,329	28 seconds
Static/length-based	3,602,117	1,553	10 seconds
	(8 %)	(47 %)	(36 %)

4. Evaluation of Retrieval Quality

We participated in the INEX 2005 workshop with a more general version of the dynamic length-based result enhancements described in this paper [3]. The INEX (Initiative for the Evaluation of XML Retrieval) workshop[1] [5] is a yearly event for evaluating the effectiveness of XML retrieval systems. It is comparable to TREC in traditional information retrieval. The test collection consists of more than 15 000 articles from the IEEE Computer Society's journals and transactions.

4.1. Setup

Every year, the INEX participants submit topics and assess the pooled retrieval results submitted by all participants. In INEX, there are two dimensions to relevance, *specificity* and *exhaustivity*. Specificity denotes what fraction of a retrieval result is relevant to the topic at hand (this is done by highlighting the relevant parts), and exhaustivity measures to what degree the information need is fulfilled (on a scale from 0 meaning "not exhaustive" to 2 for "highly exhaustive").

The evaluation of the retrieval quality is performed using the extended cumulative gain (xCG) metric introduced in INEX 2005 [9, 8], an extension of the cumulative gain (CG) metric by Järvelin and Kekäläinen [7]. The basic idea of CG is to obtain a graded relevance assessment for each retrieval result, and then to determine the quality of a given ranked list of retrieval results up to a specific rank by summation of the relevance assessment. For example, if the first three results had the relevance scores 3, 0, and 1, the CG values would be 3, $3+0=3$, and $3+0+1=4$ for the first three cut-off points. The normalized version nCG simply divides the CG values for a given system by the values of an ideal run constructed from the relevance assessments.

The CG metric requires a single relevance value, so xCG introduces quantization functions for calculating a combined value from the exhaustivity and specificity values: A strict version that only accepts the best results ($e=2$ and $s=1$), and a generalized version that gives partial credit to less than perfect results by multiplying e and s.

As a baseline, we used the Lucene retrieval engine[2] in version 1.9 with standard (Porter) stemming and indexed each element as a separate document [4]. Lucene performs ranking like method 2 in Lee et al. [11], using *tf.idf* and normalization based on document length alone.

We executed the evaluation using the official INEX 2005 CO topics with the corresponding relevance assessments and the EvalJ evaluation package[3].

4.2. Evaluation

Apart from retrieval runs based on the methods described in this paper, we also included the following retrieval runs in our evaluation:

University of Kaiserslautern: (INEX submission CO_Pattern_Thorough_NoERG) This run (our submission) was the INEX 2005 submission with the best early precision for CO.Thorough with generalized evaluation. It is based on a preliminary version of the research presented in this paper [3].

[1] see `http://inex.is.informatik.uni-duisburg.de/`

[2] see `http://lucene.apache.org`

[3] see `http://evalj.sourceforge.net/`

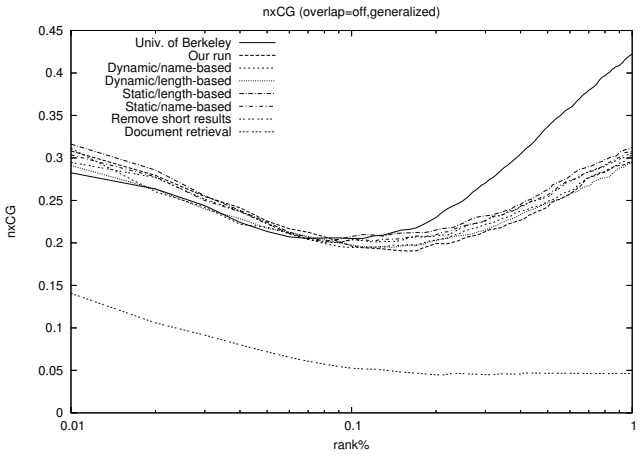

Figure 3: Evaluation results for CO.Thorough, nxCG with generalized quantization. The *x* axis (cut-off point as a fraction of the full recall base of 1,500 results) uses a logarithmic scale to better show the retrieval quality at low cut-off points.

University of Berkeley: (INEX submission CO_T2FB_PIV50_THR) This run was the INEX 2005 submission with the second best early precision for CO.Thorough with generalized evaluation.

Document retrieval: For reference, we included this run based on traditional IR (the granularity is restricted to documents); for obvious reasons, it achieved bad retrieval quality as an element retrieval system, but it is useful for showing that the increase in retrieval time is offset by an increase in retrieval quality.

Remove short results: In order to show that the increased retrieval quality of the static method is not only caused by the omission of the short elements from the index, we also include a run that excludes the same results, but without re-weighting (this is basically traditional IR on all elements that are at least 40 tokens long).

As we can see in Table 2 and Fig. 3, the methods using small elements to enhance retrieval results all fare well compared to the official INEX 2005 submissions. As expected, pure document retrieval stays far behind all the element retrieval methods; the "remove short results" run has surprisingly good results compared to the runs that actually exploit the structure, but it is still improved on by the runs exploiting small elements.

Recall appears to be a problem for all our runs: At the full 1,500 results cut-off point, our runs have an nxCG of about 0.30, whereas the run of the University of Berkeley has 0.42. Considering that this applies to all our runs (not only the runs exploiting small elements), we assume that this is not caused by using small elements, but a shortcoming of our basic retrieval engine.

5. Conclusions and Future Work

We have shown that small elements can improve retrieval quality for XML element retrieval. Although the naïve way of implementing them is very resource-hungry (one "document" per element), we have shown that it is possible to improve performance significantly by modifying the index entries of the parents of the small elements, while still retaining the gain in retrieval quality. Even with these improvements, retrieval time is still an order of magnitude higher than that of document retrieval, so there is still room for improvement.

Various parameters, such as the maximum length of an inline element, have to be tuned to the problem; although values of 30–40 appear to work well for the IEEE collection, it is doubtful whether this holds true for other collections. For optimum retrieval quality, it will be necessary to fine-tune these values based on training data or in a feedback process (preferably automatically using machine learning techniques).

References

[1] P. Arvola, M. Junkkari, and J. Kekäläinen. Generalized contextualization method for XML information retrieval. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, 2005.

[2] M. Cutler, Y. Shih, and W. Meng. Using the structure of HTML documents to improve retrieval. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, 1997.

[3] P. Dopichaj. The University of Kaiserslautern at INEX 2005. In Fuhr et al. [5].

[4] B. Eger. Entwurf und Implementierung einer XML-Volltext-Suchmaschine. Master's thesis, University of Kaiserslautern, 2005.

[5] N. Fuhr, M. Lalmas, S. Malik, and G. Kazai, editors. *Advances in XML Information Retrieval and Evaluation: Fourth Workshop of the INitiative for the Evaluation of XML Retrieval (INEX 2005)*. Springer, 2006.

[6] N. Fuhr, M. Lalmas, S. Malik, and Z. Szlávik, editors. *Advances in XML Information Retrieval: Third International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX 2004)*. Springer, 2005.

[7] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4):422–446, 2002.

[8] G. Kazai and M. Lalmas. INEX 2005 evaluation metrics. In Fuhr et al. [5], pages 16–29.

[9] G. Kazai, M. Lalmas, and A. P. de Vries. The overlap problem in content-oriented XML retrieval evaluation. In M. Sanderson, K. Järvelin, J. Allan, and P. Bruza, editors,

Table 2: Retrieval results, based on the INEX 2005 collection, CO.Thorough, quantization generalized. The first two runs are official submissions, the other runs are specific to this paper.

Retrieval run	nxCG@10	nxCG@25	nxCG@1500
University of Berkeley	0.2820	0.2654	0.4223
University of Kaiserslautern	0.3037	0.2771	0.2958
Remove short results	0.2854	0.2714	0.3056
Document retrieval	0.1336	0.1027	0.0464
Dynamic/length-based	0.3023	0.2582	0.2944
Dynamic/name-based	0.2947	0.2552	0.3030
Static/length-based	0.3078	0.2858	0.3122
Static/name-based	0.3019	0.2704	0.3095

SIGIR 2004: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 72–79. ACM, 2004.

[10] J. Kekäläinen, M. Junkkari, and P. Arvola. TRIX 2004 – struggling with the overlap. In Fuhr et al. [6], pages 127–139.

[11] D. L. Lee, H. Chuang, and K. Seamons. Document ranking and the vector-space model. *IEEE Software*, 14(2):67–75, March 1997.

[12] S. Malik, G. Kazai, M. Lalmas, and N. Fuhr. Overview of INEX 2005. In Fuhr et al. [5], pages 1–15.

[13] Y. Mass and M. Mandelbrod. Retrieving the most relevant XML components. In *INEX 2003 Workshop Proceedings*, 2003.

[14] P. Ogilvie and J. Callan. Hierarchical language models for XML component retrieval. In Fuhr et al. [6], pages 224–237.

[15] G. Ramírez, T. Westerveld, and A. P. de Vries. Using small XML elements to support relevance. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2006)*, 2006.

[16] S. Robertson, W. Lu, and A. MacFarlane. XML-structured documents: Retrievable units and inheritance. In *Flexible Query Answering Systems, 7th International Conference, FQAS 2006, Proceedings*, pages 121–132. Springer, 2006.

[17] G. Salton, J. Allan, and C. Buckley. Approaches to passage retrieval in full text information systems. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR 1993)*, pages 49–58, 1993.