The Utrecht Blend: Basic Ingredients for an XML Retrieval System

Roelof van Zwol, Frans Wiering, and Virginia Dignum

Centre for Content and Knowledge Engineering,
Utrecht University,
Utrecht, The Netherlands
{roelof, frans.wiering, virginia}@cs.uu.nl

Abstract. Exploiting the structure of a document allows for more powerful information retrieval techniques. In this article a basic approach is discussed for the retrieval of XML document fragments. Based on a vector-space model for text retrieval we aim at investigating various strategies that influence the retrieval performance of an XML-based IR system.

The first extension of the system uses a schema-based approach that assumes that authors tag their text to emphasise on particular pieces of content that are of importance. Based on the schema used by the document collection, the system can easily derive the children of mixed content nodes. Our hypothesis is that those child nodes are more important than other nodes.

The second approach discussed here is based on a horizontal fragmentation of the inverse document frequencies, used by the vector space model. The underlying assumption states that the distribution of terms is related to the semantical structure of the document. However, we observed that the IEEE collection is not a good example of semantic tagging.

The third approach investigates how the performance of the retrieval system can improve for the 'Content Only' task by using a set of a-priori defined cut-off nodes that define 'logical' document fragments that are of interest to a user.

1 Introduction

The upcoming XML standard as a publishing format provides many new challenges. One of these challenges, the scope of INEX [2], is the retrieval of structured documents. This requires new techniques that extend current developments in text retrieval. Not only should an XML retrieval system be equipped with an adequate text retrieval strategy, it is also required that the system is capable to include the document structure into the retrieval process.

The structure of the XML document is not only used to refine the query formulation process, it also allows to retrieve more accurate the relevant pieces of information that a user is interested in. For the ad hoc track of INEX, two tasks are defined that examine these aspects: the *Content Only* (CO) task and the '*Vague Content and Structure* (VCAS) task [5]. The aim of both tasks is to retrieve relevant document fragments. The difference lays in the query formulation. The CO task uses only a keyword specification, as commonly used for text retrieval and the well-known Internet search engines. The VCAS task, however, alsouses the document structure for the query formulation, using the NEXI specification [9]. NEXI is an Xpath-like query language, that allows both structural and content-based constraints to be specified for a typical user information request on a structured document collection.

The challenge is thus to build the best content-based XML retrieval system that allows for the retrieval of relevant text fragments, while taking the structure of the XML documents into account. Our personal aim is more modest, since we are primarily interested in the effect of our hypothesis on the retrieval performance of an XML retrieval system. Therefore we have built a retrieval system, that is based on the vector space model for text retrieval and use a strict interpretation of the structural constraints, formerly referred to as the *strict content and structure* (SCAS) task [3].

We have three hypothesis that we want to put to the test. First of all our aim is to investigate whether the retrieval performance of our default XML retrieval system can be improved by taking into account that the author uses markup (structure) to emphasise on particular pieces of text that are of extra importance, i.e. bold/italic text, itemised lists, or enumerations. Focusing on the XML structure, examples of these text fragments are typically found within *mixed-content* nodes. The content model of a mixed-content node contains a mixture of text and child-elements. Using the DTD or XML-schema definition the content type of nodes can easily be determined. In this article we refer to this as the *schema-based* run.

Another hypothesis that we want to investigate here, takes into account that some terms will occur more often within certain XML document fragments, than in other document fragments. We expect that adjusting the term weights by taking this distribution into account will increase the performance of the ranking of the retrieval strategy. This hypothesis has already been tested successfully in the context of XML and semantical schemas [10]. The vector space model consists of two components: a document statistic, i.e. the term frequency (tf), and a collection statistic, i.e. the inverse document frequency (idf). These two statistics are calculated for each term in the document collection. However, the inverse document frequencies are no longer calculated over the entire document, but for small text fragments. Assume now that some terms occur less frequently in abstract, than in other parts of the document. As a result the idf, and thus the term weight, of those terms is valued relatively low compared to other terms in the abstract. Using a fragmented document frequency, where the idf is calculated per XML element name corrects this problem. Our experience is that for semantically tagged XML documents an increase in retrieval performance can be achieved, when the query consists of two or more query terms [10]. We refer to this approach as the *fdf* run.

The third hypothesis focuses on the CO task. For the CO task it is not specified in the query, which document fragments should be returned by the system. Returning entire documents as the result of a query will result in a low performance according to the specificity quantisation [4], since it is likely that only small portions of the XML document will contain relevant information. To deal with this we have defined a cutoff node set, that consists of XML elements that provide a partial logical view on the XML document. When retrieving XML document fragments this node set is used to return smaller fragments, that have a higher specificity of the content in relation to the query terms. We refer to this strategy as the *cutoff* run.

1.1 Organisation

In the remainder of this article we first discuss the approach used to index the XML collection in Section 2. In Section 3 the different retrieval strategies for querying XML documents is discussed for the different runs that we have submitted for INEX 2004. The results of our system are presented in Section 4, together with the unofficial runs that we computed with improved performance of the vector space model. Finally we come to our conclusions in Section 5.

2 Indexing the XML Collection

To index the IEEE XML document collection the XML structure of each document is analysed and a text retrieval strategy is implemented. In Section 2.1 the indexing of the index structure is discussed, while in Section 2.2 the text retrieval component is described.

2.1 Processing XML Structures

To index the XML collection the structure of each document is analysed as follows. The nodes are numbered using the method described in Table 1. This resembles an approach adopted by others [6], however we have chosen not to number the individual terms within a text fragment, but to refer to a text fragment as a whole.

Furthermore we keep track of parent-child relations for each node. All node information is stored in the `Element` table, as shown in Table 2. This table

Table 1. XML example illustrating the numbering of nodes

<ElementA>^{1}
    TextFragmentA^{2}
    <ElementB>^{3}TextFragmentB^{4}</ElementB>^{5}
    <ElementC>^{6}TextFragmentC^{7}</ElementC>^{8}
    <ElementB/>^{9}
    TextFragmentD^{10}
</ElementA>^{11}

Table 2. Internal data structure

`Document`

id	uri

`Element`

id	name	parent	document	path	start	end

`Textfragment`

id	parent	position	length	document

`Term`

content	fragment	tf	tfidf

contains the following information about element nodes: A unique id, the element name, a reference to its parent, a pointer to the document containing the element, and the unique path leading to the element node. Finally, for each element node the start and end positions are stored, as explained above.

Whenever the indexer encounters a text fragment, a new id is generated and stored in the table `TextFragment`. A reference to the parent node, its position in the document, the number of terms, i.e. the length, and a pointer to the document URI is stored. The text fragment is then handed to the text indexer.

2.2 Processing Text Fragments

The text retrieval component of our indexing system is based on vector space model [1]. This component analyses the rather small text fragments according to the following steps:

- **pre-processing.** A number of basic text operations are called during the pre-processing step. Among these are lexical cleaning, stop word removal and stemming [1].
- **indexing.** Using a bag of terms approach the frequencies of the terms occurring in the text fragment are calculated. After processing a text fragment, all the terms are stored in the `Term` table. For each term, its content, a reference to the corresponding text fragment and the term frequency is stored in the database.
- **post-processing.** Once all documents have been indexed the collection statistics are calculated. For each unique term in the collection the inverse document frequency is calculated as:

$$idf(t) = log(\frac{N}{n(t)}), \tag{1}$$

with N being the total number of unique terms, and $n(t)$ the number of text fragments in which term t occurs.

Later on, we also used a normalised tf factor [8]. The ntf factor reduces the range of the contributions from the term frequency of a term. This is done by compressing the range of the possible tf factor values. The ntf factor is used with the belief that mere presence of a term in a text should have a default weight. Additional occurrences of a term could increase the weight of the term to some maximum value. To compute this factor we used:

$$ntf(t) = 0.5 \; + \; 0.5 \; * \frac{tf(t)}{max \; tf(t)} \tag{2}$$

$tf(t)$ contains the raw term frequency for the term, while $max \; tf(t)$ provides the maximum term frequency found in that text fragment.

The tfidf for each term in `Term` is then calculated as:

$$tfidf(t) = \frac{tf(t) * idf(t)}{l} \tag{3}$$

Where l is the length of the text fragment. Please note that this is not a standard way to normalise the term weights for the length of the text fragments.

3 Querying the XML Collection

For INEX we submitted six runs, as discussed below. They all use the same vector space model, with the exception of the fdf runs. Furthermore, we believe that this implementation of the vector-space model leaves plenty of room for improvement. When discussing the results, we will show some simple modifications that improve the retrieval performance of our system. Our interest in this experiment focuses mainly on the effect of using different XML-based mechanisms for calculating the relevances of the document fragments retrieved by our system. The official runs computed for the INEX 2004 topic set are described below.

3.1 Content and Structured XML Retrieval

The so called vague content and structure (VCAS) topics are defined using the NEXI specification [9]. Our system implements the NEXI grammar for these types of topics and evaluates the NEXI queries by following the path expressions and narrowing down the possible set of results. In fact our system enforces that the path constraints defined by the topic are computed in a strict fashion, according to the SCAS specification. We computed the following three runs for the VCAS ad hoc task:

– **33-VCAS-default.** Our default approach to compute a ranking of the retrieved documents simply determines a set of possible document fragments for the first structural constraint, and assigns a textual relevance of '0' to them. If a filter clause is available, this set is narrowed down, according to the conditions defined in the filter. If an *about*-clause is defined within that filter, a relevance ranking of the document fragments is obtained by the system.

Table 3. NEXI example: INEX 2004, topic 132

$$//article[about(.//abs, classification)]//sec[about(., experiment\ compare)]$$

This basic approach is followed for all VCAS runs submitted. The variance between the runs is determined by the implementation of the *about*-clause. Consider for example the following NEXI-query, presented in Table 3.

During the first step a set of `article`-fragments is retrieved, having a relevance score of '0'. The next step is to evaluate the *about*-filter, narrowing down the set of articles to those containing an `abstract`, which contains the word '*classification*'. The relevances computed by the about function are then summed and associated with the corresponding `article`-fragments. For this set, the second path-constraint is computed, which in this case results in a set of `sec`-nodes, which inherit the relevances computed for the parent `article` nodes. Again the about-filter is evaluated and the relevances are added to the existing relevance scores of the retrieved `sec` nodes.

For the default run the relevances for the document fragment are simply calculated by filtering all the relevant terms from the `TERM` table, using only the *positive* query terms. The relevance for each document fragment, defined in the offset of the about clause, is then calculated by summing over the terms of the text fragments that are contained within the start- and end position of the document fragment.

– **33-VCAS-schema.** The structural constraints for this run are computed similar to the default run. However the about function uses a weighing function, that increases the weight of those nodes which are considered of more importance.

  The underlying hypothesis is that authors writing text use markup to emphasise on particular pieces of content that they find of more importance. Simple examples are those text fragments containing bold and italic text. A reader's attention is automatically drawn whenever a bold or italic text fragment is seen. In XML, this markup is typically found within *mixed-content* nodes. Mixed content nodes are nodes that allow both text fragments and additional markup to be used in a mixed context. In our case, we are interested in the set of child nodes found within such *mixed-content* nodes. Using the DTD, or XML-schema definition this node set can be easily computed. To compute the relevances of the XML document fragments the system first has to derive the set with text fragments containing relevant terms. If one or more ancestor nodes are contained in the set with mixed-content nodes a multiplication factor, i.e. 2, 4, 8, or . . ., is added to the weight of that text fragment, depending on the number of mixed-content nodes that are found. Next, the relevance for each document fragment is calculated by summing over the terms of the text fragments that are contained within the start- and end position of the document fragment.

- **33-VCAS-fdf.** This run uses an alternative way of calculating the term weights. The vector space model uses a combination of two statistics to calculate the term weights, i.e. the term frequencies and the inverse document frequencies. The inverse document frequency is a collection measure, that determines how frequently a term occurs in different documents of the collection. For the 'fragmented document fragments'-run (fdf) we have used a fragmented version of the inverse document frequencies (ifdf).

  The underlying assumption for this fragmentation is that if the XML structure of the document is not merely based on presentation, but defines a semantic structure for the content contained in the document, it is likely that some terms, associated with the semantic structure will appear more often in certain document fragments than other terms.

  For example, in text fragments discussing cultural information about a destination, the term '*church*' is more likely to appear, than in text fragments that discuss sports activities[1]. Consider now the following information request: 'Find information about basketball clinics in former churches', the term church is an important query term in this search, however the *idf* for the query term '*church*' will be relatively low if the document collection contains both cultural- and sports descriptions of destinations. We have found that the retrieval performance improves significantly [10], when using the fdf approach. The retrieval strategy, based on the ifdf, is capable of ranking the relevant documents higher in the ranking, if the query consists of two or more query terms. In fact, increasing the amount of query terms will result in a higher retrieval performance.

3.2 Content Only XML Retrieval

For the CO task we have defined four runs.

- **33-CO-default.** The content only runs are mainly driven by the text retrieval component. The positive query terms defined for each content only topic are used to find relevant text fragments. The term weights found in each text fragment are summed over the corresponding parent node of each text fragment.

  In the next step the result set is grouped and summed per document. As a result the smallest common document fragment that can be retrieved for each document is returned as the result of a query. This approach ensures that no redundancy is possible between the document fragments retrieved by the system.

  This approach has two advantages: no redundancy in the retrieved document fragments, and the retrieved fragments should score high on the ex-

[1] This example is based on the Lonely Planet collection, where the tagging of content is semantically organised[10].

haustiveness measure. This also introduces the drawback of this approach: together with the relevant information a lot of 'garbage' is retrieved, resulting in poor performance from a specificity point of view.

– **33-CO-schema.** This run is a combination of runs 33-CO-default and 33-VCAS-schema. It uses the multiplication scheme for the children of the mixed-content nodes, and the combinational logic as defined for the default approach described above. In this way, for each document the smallest document fragment is returned that contains all relevant text fragments.
– **33-CO-cutoff.** From a user point of view not all document fragments that can be retrieved are logical units. To facilitate this, we have defined a set of nodes that provide the users logical document fragments. The aim here is to find a balance between the exhaustiveness and specificity measures. For the IEEE collection we have defined a cutoff-node set containing five nodes: $fm, abs, sec, bib, article$. The article element forms the root node of many documents and should always be there, to prevent losing documents from the result set.

  After retrieving the relevant text fragments, the parent nodes are retrieved and (child) results merged into larger document fragments, until a node is found that is contained in the set with cutoff-nodes.
– **33-CO-fdf.**[2] This run is also a combination of two other runs: 33-VCAS-fdf, and 33-CO-default. Instead of the default tfidf weights this run uses the tfifdf index, as explained in Section 3.1

4 Results

In this section we will first present the results CO task and then the results for the VCAS task. Before diving into the evaluation of the results, we have two issues to address that highly affected the retrieval performance. First of all, we did not allow overlapping elements in the result set, and secondly we used a SCAS interpretation, which also reduces the possible candidate elements for the resultset. All plots and measures were calculated using the on-line evaluation tool [7].

4.1 CO Task

We first discuss the results of the official run for the CO task in Section 4.1. To improve on the performance for the CO task we need a better retrieval strategy for the text retrieval component.

Official Runs. Figure 1 gives an overview of the performance of our CO runs. The *CO-default*-run performed best when evaluated using the strict quantisation

[2] For the official INEX runs, this approach was left out, since only six runs per participant were permitted.

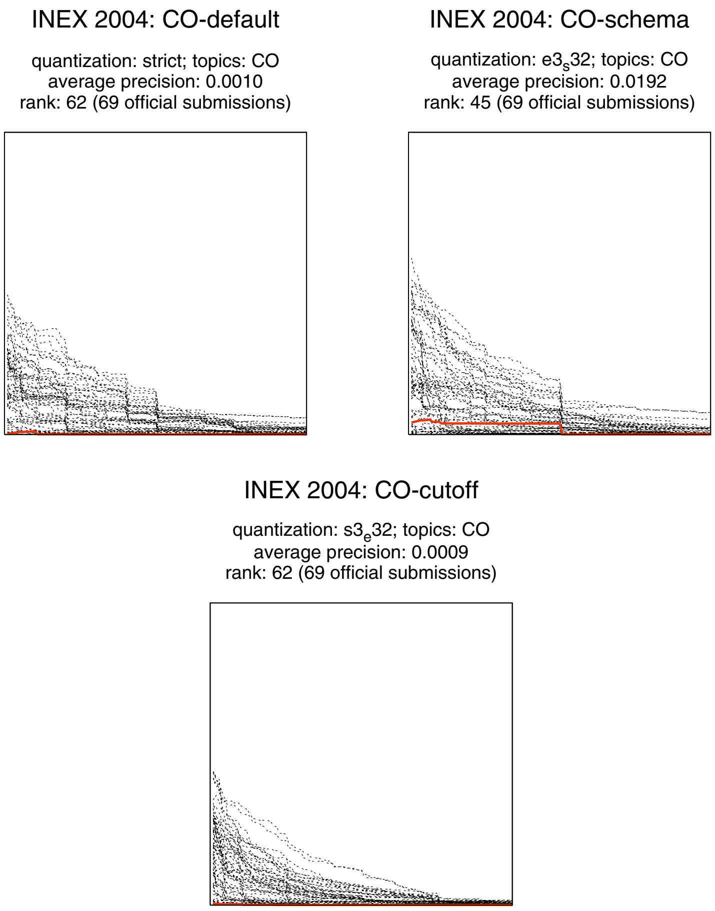


Fig. 1. Official runs for the CO task - best performances

measure. Slightly better performed the run *CO-schema*, while using the e3_s32 quantisation, which illustrates that this approach is best used, when searching for exhaustive document fragments. On the other hand, the *CO-cutoff*-run performed best for the s3_e32 quantisation measure. This was expected, since the aim of this approach was to return smaller logical document fragments, that would score better on the specificity scale.

These aspects are better illustrated in Figure 2, 3, 4, and 5. The average over all RP measures is showed in the top-left corner. On average, the best performance with the official runs was obtained with *CO-schema*, while the *CO-cutoff*-run performed worst. Surprisingly however, the run *CO-cutoff* performed best when looking at the expected ratio of relevance (bottom-right) for the generalised recall, and slightly better when evaluation is based on the specificity quantisation. The top-right graph shows that for the CO task, it makes sense to

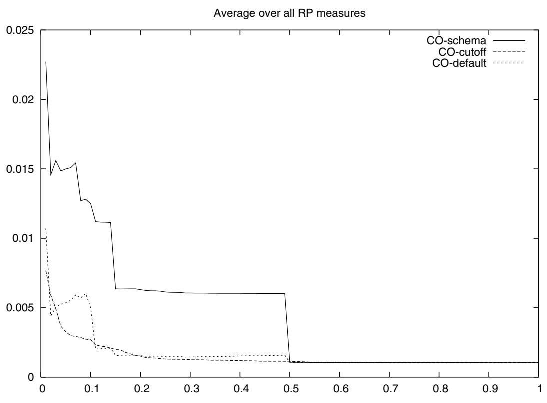


Fig. 2. Official runs for the CO task (a)

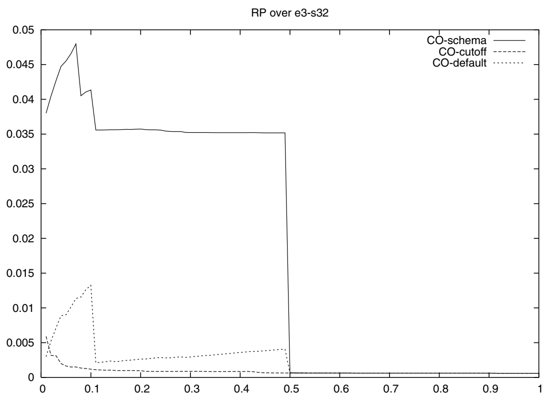


Fig. 3. Official runs for the CO task (b)

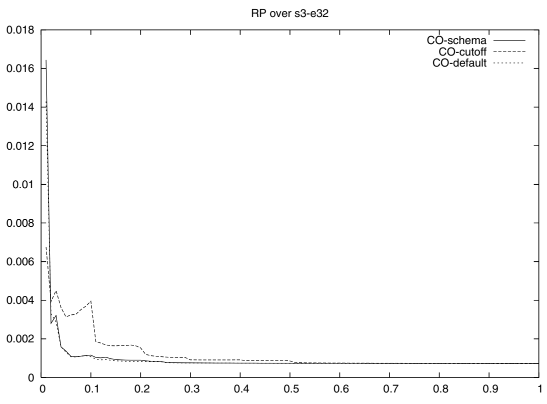


Fig. 4. Official runs for the CO task (c)

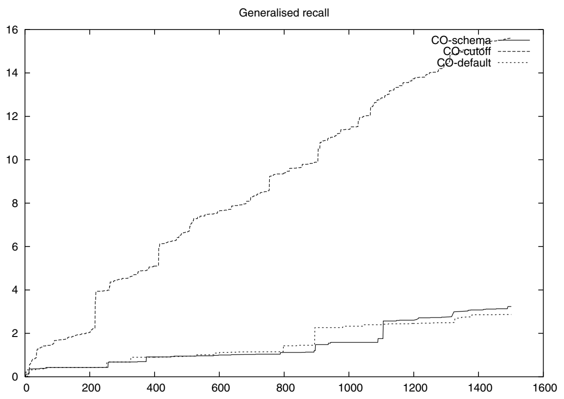
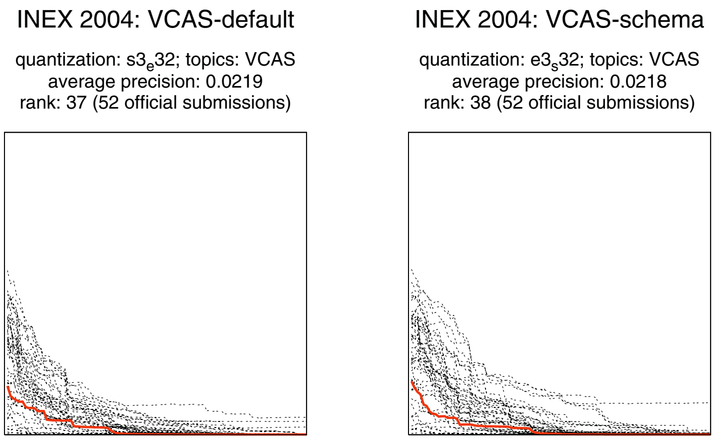


Fig. 5. Official runs for the CO task (d)



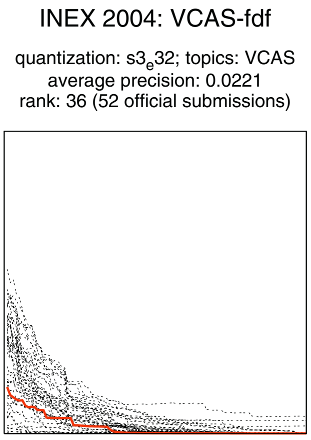


Fig. 6. Official runs for the VCAS task - Best performances

include the markup added by the author to emphasise certain terms in the text into the ranking process.

4.2 VCAS Task

Figure 6 gives an overview of the performance of our VCAS runs. The *VCAS-default*-run performed best when evaluated using the s3e32 quantisation measure. Not surprising, since the implementation of our system uses the strict content and structure approach. The same is true for the *VCAS-fdf*-run. For the *VCAS-schema*-run the best performance is gained using the exhaustiveness quantisation measure. The differences between the runs however are marginal.

5 Conclusions

Our goal within INEX was to investigate the influence of the three hypothesis on the retrieval performance. Obviously our system does not belong to the top performing systems. This is mainly caused by two important factors: we did not allow overlapping elements in the result set, and we used a SCAS interpretation, which also reduces the possible candidate elements for the result set.

The comparison the schema-based run with thedefault run, clearly showed that if authors use markup to emphasise on particular pieces of content that they find of more importance, it makes sense to increase the weights of those document fragments to improve the retrieval performance. The results show that more relevant document fragments are ranked higher in the result list.

On the other hand we can increase the specificity of the retrieved document fragments, by using a so called cutoff node set. The system then returns smaller document fragments that are more relevant for the given topic. Whereas our default run returns rather large document fragments, containing the all the relevant document fragments of one physical document.

Finally, the runs that were using the fragmented document frequencies (fdf) did not increase the retrieval performance of our system. We feel that this is mainly caused by the absence of a semantical markup of the content of the IEEE document collection. However, many of the XML document collections that are currently available are based on logical and presentation tagging, rather then semantical tagging. Extending these documents with a semantical markup, will allow for more meaningful structured document retrieval. Besides that we expect that the retrieval performance for the FDF run will also improve, as shown in the past for different document collections.

References

1. R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval.* ACM Press, 1999.
2. N. Fuhr, N. Kazai, and M. Lalmas. INEX: Initiative for the evaluation of XML retrieval. In *In Proceedings of the ACM SIGIR 2000 Workshop on XML and Information Retrieval*, 2000.

3. N. Fuhr, S. Malik, and M. Lalmas. Overview of the initiative for the evaluation of xml. In *In Proceedings of the Second INitiative for the Evaluation of XML Retrieval (INEX) Workshop*, pages 1–11, Decmber 2003.
4. G. Kazai. Report of the inex'03 metrics working group. In *In Proceedings of the Second INitiative for the Evaluation of XML Retrieval (INEX) Workshop*, pages 184–190, Dagstuhl, Germany, 2003.
5. M. Lalmas and S. Malik. Inex 2004 retrieval task and result submission specification, June 2004. http://inex.is.informatik.uni-duisburg.de:2004/internal/pdf/INEX04_Retrieval_Task.pdf.
6. J. A. List and A. P. de Vries. CWI at inex 2002. In *Proceedings of the First Workshop of the INitiative for the Evaluation of XML Retrieval (INEX)*, 2002.
7. S. Malik and M. Lalmas. http://inex.lip6.fr/2004/metrics/official.php, 2004.
8. G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.
9. A. Trotman and R. A. O'Keefe. The simplest query language that could possibly work. In *Proceedings of the Second Workshop of the INitiative for the Evaluation of XML retrieval (INEX)*, 2004.
10. R. van Zwol. *Modelling and searching web-based document collections.* Ctit ph.d. thesis series, Centre for Telematics and Information Technology (CTIT), Enschede, the Netherlands, 26 April 2002.