Controlling Overlap in Content-Oriented XML Retrieval

Charles L. A. Clarke
School of Computer Science, University of Waterloo, Canada
claclark@plg.uwaterloo.ca

ABSTRACT

The direct application of standard ranking techniques to retrieve individual elements from a collection of XML documents often produces a result set in which the top ranks are dominated by a large number of elements taken from a small number of highly relevant documents. This paper presents and evaluates an algorithm that re-ranks this result set, with the aim of minimizing redundant content while preserving the benefits of element retrieval, including the benefit of identifying topic-focused components contained within relevant documents. The test collection developed by the INitiative for the Evaluation of XML Retrieval (INEX) forms the basis for the evaluation.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Storage and Retrieval—*Information Search and Retrieval*

General Terms

Algorithms, Measurement, Performance, Experimentation

Keywords

XML, Ranking, Information Retrieval

1. INTRODUCTION

The representation of documents in XML provides an opportunity for information retrieval systems to take advantage of document structure, returning individual document components when appropriate, rather than complete documents in all circumstances. In response to a user query, an XML information retrieval system might return a mixture of paragraphs, sections, articles, bibliographic entries and other components. This facility is of particular benefit when a collection contains very long documents, such as product manuals or books, where the user should be directed to the most relevant portions of these documents.

```
<article>
   <fm>
      <atl>Text Compression for
           Dynamic Document Databases</atl>
      <au>Alistair Moffat</au>
      <au>Justin Zobel</au>
      <au>Neil Sharman</au>
      <abs><p><b>Abstract</b> For ...</p></abs>
   </fm>
   <bdy>
      <sec><st>INTRODUCTION</st>
         <ip1>Modern document databases...</ip1>
         <p>There are good reasons to compress...</p>
      </sec>
      <sec><st>REDUCING MEMORY REQUIREMENTS</st>...
         <ss1><st>2.1 Method A</st>...
      </sec>
   ...
   </bdy>
</article>
```

Figure 1: ***A journal article encoded in XML.***

Figure 1 provides an example of a journal article encoded in XML, illustrating many of the important characteristics of XML documents. Tags indicate the beginning and end of each element, with elements varying widely in size, from one word to thousands of words. Some elements, such as paragraphs and sections, may be reasonably presented to the user as retrieval results, but others are not appropriate. Elements *overlap* each other — articles contain sections, sections contain subsections, and subsections contain paragraphs. Each of these characteristics affects the design of an XML IR system, and each leads to fundamental problems that must be solved in an successful system. Most of these fundamental problems can be solved through the careful adaptation of standard IR techniques, but the problems caused by overlap are unique to this area [4, 11] and form the primary focus of this paper.

The article of figure 1 may be viewed as an XML tree, as illustrated in figure 2. Formally, a collection of XML documents may be represented as a forest of ordered, rooted trees, consisting of a set of nodes $\mathcal{N}$ and a set of directed edges $\mathcal{E}$ connecting these nodes. For each node $x \in \mathcal{N}$, the notation $x.parent$ refers to the parent node of x, if one exists, and the notation $x.children$ refers to the set of child nodes

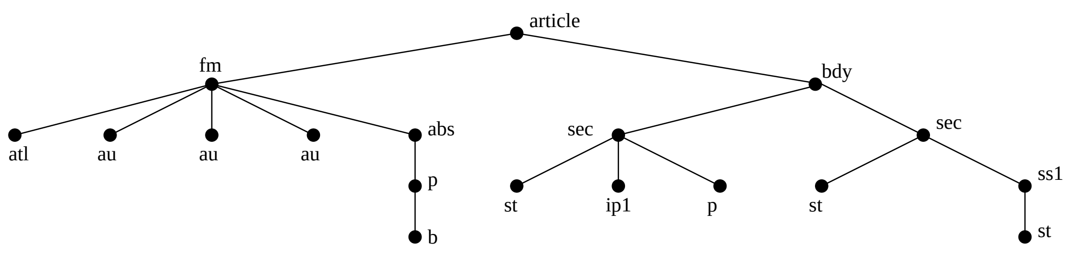

Figure 2: ***Example XML tree.***

of x. Since an element may be represented by the node at its root, the output of an XML IR system may be viewed as a ranked list of the top-m nodes.

The direct application of a standard relevance ranking technique to a set of XML elements can produce a result in which the top ranks are dominated by many structurally related elements. A high scoring section is likely to contain several high scoring paragraphs and to be contained in an high scoring article. For example, many of the elements in figure 2 would receive a high score on the keyword query "`text index compression algorithms`". If each of these elements are presented to a user as an individual and separate result, she may waste considerable time reviewing and rejecting redundant content.

One possible solution is to report only the highest scoring element along a given path in the tree, and to remove from the lower ranks any element containing it, or contained within it. Unfortunately, this approach destroys some of the possible benefits of XML IR. For example, an outer element may contain a substantial amount of information that does not appear in an inner element, but the inner element may be heavily focused on the query topic and provide a short overview of the key concepts. In such cases, it is reasonable to report elements which contain, or are contained in, higher ranking elements. Even when an entire book is relevant, a user may still wish to have the most important paragraphs highlighted, to guide her reading and to save time [6].

This paper presents a method for *controlling overlap*. Starting with an initial element ranking, a re-ranking algorithm adjusts the scores of lower ranking elements that contain, or are contained within, higher ranking elements, reflecting the fact that this information may now be redundant. For example, once an element representing a section appears in the ranking, the scores for the paragraphs it contains and the article that contains it are reduced. The inspiration for this strategy comes partially from recent work on structured documents retrieval, where terms appearing in different fields, such as the title and body, are given different weights [20]. Extending that approach, the re-ranking algorithm varies weights dynamically as elements are processed.

The remainder of the paper is organized as follows: After a discussion of background work and evaluation methodology, a baseline retrieval method is presented in section 4. This baseline method represents a reasonable adaptation of standard IR technology to XML. Section 5 then outlines a strategy for controlling overlap, using the baseline method as a starting point. A re-ranking algorithm implementing this strategy is presented in section 6 and evaluated in section 7. Section 8 discusses an extended version of the algorithm.

2. BACKGROUND

This section provides a general overview of XML information retrieval and discusses related work, with an emphasis on the fundamental problems mentioned in the introduction. Much research in the area of XML retrieval views it from a traditional database perspective, being concerned with such problems as the implementation of structured query languages [5] and the processing of joins [1]. Here, we take a "content oriented" IR perceptive, focusing on XML documents that primarily contain natural language data and queries that are primarily expressed in natural language. We assume that these queries indicate only the nature of desired content, not its structure, and that the role of the IR system is to determine which elements best satisfy the underlying information need. Other IR research has considered mixed queries, in which both content and structural requirements are specified [2, 6, 14, 17, 23].

2.1 Term and Document Statistics

In traditional information retrieval applications the standard unit of retrieval is taken to be the "document". Depending on the application, this term might be interpreted to encompass many different objects, including web pages, newspaper articles and email messages.

When applying standard relevance ranking techniques in the context of XML IR, a natural approach is to treat each element as a separate "document", with term statistics available for each [16]. In addition, most ranking techniques require global statistics (e.g. inverse document frequency) computed over the collection as a whole. If we consider this collection to include all elements that might be returned by the system, a specific occurrence of a term may appear in several different "documents", perhaps in elements representing a paragraph, a subsection, a section and an article. It is not appropriate to compute inverse document frequency under the assumption that the term is contained in all of these elements, since the number of elements that contain a term depends entirely on the structural arrangement of the documents [13, 23].

2.2 Retrievable Elements

While an XML IR system might potentially retrieve any element, many elements may not be appropriate as retrieval results. This is usually the case when elements contain very little text [10]. For example, a section title containing only the query terms may receive a high score from a ranking algorithm, but alone it would be of limited value to a user, who might prefer the actual section itself. Other elements may reflect the document's physical, rather than logical, structure, which may have little or no meaning to a user. An effective XML IR system must return only those elements that have sufficient content to be usable and are able to stand alone as independent objects [15, 18]. Standard document components such as paragraphs, sections, subsections, and abstracts usually meet these requirements; titles, italicized phrases, and individual metadata fields often do not.

2.3 Evaluation Methodology

Over the past three years, the INitiative for the Evaluation of XML Retrieval (INEX) has encouraged research into XML information retrieval technology [7, 8]. INEX is an experimental conference series, similar to TREC, with groups from different institutions completing one or more experimental tasks using their own tools and systems, and comparing their results at the conference itself. Over 50 groups participated in INEX 2004, and the conference has become as influential in the area of XML IR as TREC is in other IR areas. The research described in this paper, as well as much of the related work it cites, depends on the test collections developed by INEX.

Overlap causes considerable problems with retrieval evaluation, and the INEX organizers and participants have wrestled with these problems since the beginning. While substantial progress has been made, these problem are still not completely solved. Kazai et al. [11] provide a detailed exposition of the overlap problem in the context of INEX retrieval evaluation and discuss both current and proposed evaluation metrics. Many of these metrics are applied to evaluate the experiments reported in this paper, and they are briefly outlined in the next section.

3. INEX 2004

Space limitations prevent the inclusion of more than a brief summary of INEX 2004 tasks and evaluation methodology. For detailed information, the proceedings of the conference itself should be consulted [8].

3.1 Tasks

For the main experimental tasks, INEX 2004 participants were provided with a collection of 12,107 articles taken from the IEEE Computer Societies magazines and journals between 1995 and 2002. Each document is encoded in XML using a common DTD, with the document of figures 1 and 2 providing one example.

At INEX 2004, the two main experimental tasks were both adhoc retrieval tasks, investigating the performance of systems searching a static collection using previously unseen topics. The two tasks differed in the types of topics they used. For one task, the "content-only" or CO task, the topics consist of short natural language statements with no direct reference to the structure of the documents in the collection. For this task, the IR system is required to select the elements to be returned. For the other task, the "content-and-structure" or CAS task, the topics are written in an XML query language [22] and contain explicit references to document structure, which the IR system must attempt to satisfy. Since the work described in this paper is directed at the content-only task, where the IR system receives no guidance regarding the elements to return, the CAS task is ignored in the remainder of our description.

In 2004, 40 new CO topics were selected by the conference organizers from contributions provided by the conference participants. Each topic includes a short keyword query, which is executed over the collection by each participating group on their own XML IR system. Each group could submit up to three experimental runs consisting of the top $m = 1500$ elements for each topic.

3.2 Relevance Assessment

Since XML IR is concerned with locating those elements that provide complete coverage of a topic while containing as little extraneous information as possible, simple "relevant" vs. "not relevant" judgments are not sufficient. Instead, the INEX organizers adopted two dimensions for relevance assessment: The *exhaustivity* dimension reflects the degree to which an element covers the topic, and the *specificity* dimension reflects the degree to which an element is focused on the topic. A four-point scale is used in both dimensions. Thus, a (3,3) element is highly exhaustive and highly specific, a (1,3) element is marginally exhaustive and highly specific, and a (0,0) element is not relevant. Additional information on the assessment methodology may be found in Piwowarski and Lalmas [19], who provide a detailed rationale.

3.3 Evaluation Metrics

The principle evaluation metric used at INEX 2004 is a version of mean average precision (MAP), adjusted by various *quantization functions* to give different weights to different elements, depending on their exhaustivity and specificity values. One variant, the *strict quantization* function gives a weight of 1 to (3,3) elements and a weight of 0 to all others. This variant is essentially the familiar MAP value, with (3,3) elements treated as "relevant" and all other elements treated as "not relevant". Other quantization functions are designed to give partial credit to elements which are "near misses", due to a lack or exhaustivity and/or specificity. Both the *generalized quantization* function and the *specificity-oriented generalization* (sog) function credit elements "according to their degree of relevance" [11], with the second function placing greater emphasis on specificity. This paper reports results of this metric using all three of these quantization functions. Since this metric was first introduced at INEX 2002, it is generally referred as the "inex-2002" metric.

The inex-2002 metric does not penalize overlap. In particular, both the generalized and sog quantization functions give partial credit to a "near miss" even when a (3,3) element overlapping it is reported at a higher rank. To address this problem, Kazai et al. [11] propose an *XML cumulated gain* metric, which compares the cumulated gain [9] of a ranked list to an ideal gain vector. This ideal gain vector is constructed from the relevance judgments by eliminating overlap and retaining only best element along a given path. Thus, the XCG metric rewards retrieval runs that avoid overlap. While XCG was not used officially at INEX 2004, a version of it is likely to be used in the future.

At INEX 2003, yet another metric was introduced to ameliorate the perceived limitations of the inex-2002 metric. This "inex-2003" metric extends the definitions of precision and recall to consider both the size of reported components and the overlap between them. Two versions were created, one that considered only component size and another that considered both size and overlap. While the inex-2003 metric exhibits undesirable anomalies [11], and was not used in 2004, values are reported in the evaluation section to provide an additional instrument for investigating overlap.

4. BASELINE RETRIEVAL METHOD

This section provides an overview of baseline XML information retrieval method currently used in the MultiText IR system, developed by the Information Retrieval Group at the University of Waterloo [3]. This retrieval method results from the adaptation and tuning of the Okapi BM25 measure [21] to the XML information retrieval task. The MultiText system performed respectably at INEX 2004, placing in the top ten under all of the quantization functions, and placing first when the quantization function emphasized exhaustivity.

To support retrieval from XML and other structured document types, the system provides generalized queries of the form:

`rank` X `by` Y

where X is a sub-query specifying a set of document elements to be ranked and Y is a vector of sub-queries specifying individual retrieval terms.

For our INEX 2004 runs, the sub-query X specified a list of retrievable elements as those with tag names as follows:

```
abs app article bb bdy bm fig fm ip1
li p sec ss1 ss2 vt
```

This list includes bibliographic entries (`bb`) and figure captions (`fig`) as well as paragraphs, sections and subsections. Prior to INEX 2004, the INEX collection and the INEX 2003 relevance judgments were manually analyzed to select these tag names. Tag names were selected on the basis of their frequency in the collection, the average size of their associated elements, and the relative number of positive relevance judgments they received. Automating this selection process is planned as future work.

For INEX 2004, the term vector Y was derived from the topic by splitting phrases into individual words, eliminating stopwords and negative terms (those starting with "-"), and applying a stemmer. For example, keyword field of topic 166

```
+"tree edit distance" + XML -image
```

became the four-term query

```
"$tree" "$edit" "$distance" "$xml"
```

where the "`$`" operator within a quoted string stems the term that follows it.

Our implementation of Okapi BM25 is derived from the formula of Robertson et al. [21] by setting parameters $k_2 = 0$ and $k_3 = \infty$. Given a term set Q, an element x is assigned the score

$$\sum_{t \in Q} w^{(1)} q_t \frac{(k_1 + 1)x_t}{K + x_t} \tag{1}$$

where

$$\begin{aligned}
w^{(1)} &= \log\left(\frac{D - D_t + 0.5}{D_t + 0.5}\right) \\
D &= \text{number of documents in the corpus} \\
D_t &= \text{number of documents containing } t \\
q_t &= \text{frequency that } t \text{ occurs in the topic} \\
x_t &= \text{frequency that } t \text{ occurs in } x \\
K &= k_1((1 - b) + b \cdot l_x / l_{avg}) \\
l_x &= \text{length of } x \\
l_{avg} &= \text{average document length}
\end{aligned}$$

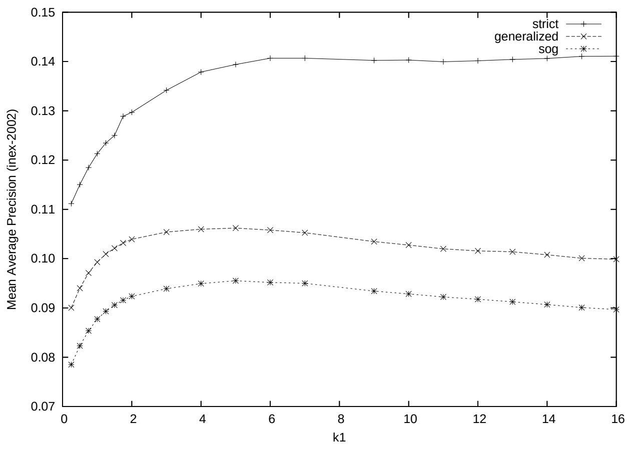

Figure 3: *Impact of k_1 on inex-2002 mean average precision with $b = 0.75$ (INEX 2003 CO topics).*

Prior to INEX 2004, the INEX 2003 topics and judgments were used to tune the b and k_1 parameters, and the impact of this tuning is discussed later in this section.

For the purposes of computing document-level statistics (D, D_t and l_{avg}) a document is defined to be an `article`. These statistics are used for ranking all element types. Following the suggestion of Kamps et al. [10], the retrieval results are filtered to eliminate very short elements, those less than 25 words in length.

The use of `article` statistics for all element types might be questioned. This approach may be justified by viewing the collection as a set of articles to be searched using standard document-oriented techniques, where only articles may be returned. The score computed for an element is essentially the score it would receive if it were added to the collection as a new document, ignoring the minor adjustments needed to the document-level statistics. Nonetheless, we plan to examine this issue again in the future.

In our experience, the performance of BM25 typically benefits from tuning the b and k_1 parameters to the collection, whenever training queries are available for this purpose. Prior to INEX 2004, we trained the MultiText system using the INEX 2003 queries. As a starting point we used the values $b = 0.75$ and $k_1 = 1.2$, which perform well on TREC adhoc collections and are used as default values in our system. The results were surprising. Figure 3 shows the result of varying k_1 with $b = 0.75$ on the MAP values under three quantization functions. In our experience, optimal values for k_1 are typically in the range 0.0 to 2.0. In this case, large values are required for good performance. Between $k_1 = 1.0$ and $k_1 = 6.0$ MAP increases by over 15% under the strict quantization. Similar improvements are seen under the generalized and sog quantizations. In contrast, our default value of $b = 0.75$ works well under all quantization functions (figure 4). After tuning over a wide range of values under several quantization functions, we selected values of $k = 10.0$ and $b = 0.80$ for our INEX 2004 experiments, and these values are used for the experiments reported in section 7.

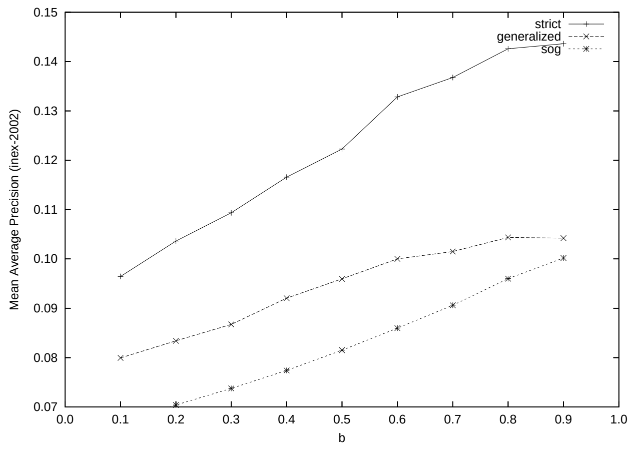

Figure 4: *Impact of b on inex-2002 mean average precision with $k_1 = 10$ (INEX 2003 CO topics).*

5. CONTROLLING OVERLAP

Starting with an element ranking generated by the baseline method described in the previous section, elements are re-ranked to control overlap by iteratively adjusting the scores of those elements containing or contained in higher ranking elements. At a conceptual level, re-ranking proceeds as follows:

1. Report the highest ranking element.
2. Adjust the scores of the unreported elements.
3. Repeat steps 1 and 2 until m elements are reported.

One approach to adjusting the scores of unreported elements in step 2 might be based on the Okapi BM25 scores of the involved elements. For example, assume a paragraph with score p is reported in step 1. In step 2, the section containing the paragraph might then have its score s lowered by an amount $\alpha \cdot p$ to reflect the reduced contribution the paragraph should make to the section's score.

In a related context, Robertson et al. [20] argue strongly against the linear combination of Okapi scores in this fashion. That work considers the problem of assigning different weights to different document fields, such as the title and body associated with Web pages. A common approach to this problem scores the title and body separately and generates a final score as a linear combination of the two. Robertson et al. discuss the theoretical flaws in this approach and demonstrate experimentally that it can actually harm retrieval effectiveness. Instead, they apply the weights at the term frequency level, with an occurrence of a query term t in the title making a greater contribution to the score than an occurrence in the body. In equation 1, x_t becomes $\alpha_0 \cdot y_t + \alpha_1 \cdot z_t$, where y_t is the number of times t occurs in the title and z_t is the number of times t occurs in the body.

Translating this approach to our context, the contribution of terms appearing in elements is dynamically reduced as they are reported. The next section presents and analysis a simple re-ranking algorithm that follows this strategy. The algorithm is evaluated experimentally in section 7. One limitation of the algorithm is that the contribution of terms appearing in reported elements is reduced by the same factor regardless of the number of reported elements in which it appears. In section 8 the algorithm is extended to apply increasing weights, lowering the score, when a term appears in more than one reported element.

6. RE-RANKING ALGORITHM

The re-ranking algorithm operates over XML trees, such as the one appearing in figure 2. Input to the algorithm is a list of n elements ranked according to their initial BM25 scores. During the initial ranking the XML tree is dynamically re-constructed to include only those nodes with non-zero BM25 scores, so n may be considerably less than $|\mathcal{N}|$. Output from the algorithm is a list of the top m elements, ranked according to their adjusted scores.

An element is represented by the node $x \in \mathcal{N}$ at its root. Associated with this node are fields storing the length of element, term frequencies, and other information required by the re-ranking algorithm, as follows:

$x.\vec{f}$	—	term frequency vector
$x.\vec{g}$	—	term frequency adjustments
$x.l$	—	element length
$x.score$	—	current Okapi BM25 score
$x.reported$	—	boolean flag, initially false
$x.children$	—	set of child nodes
$x.parent$	—	parent node, if one exists

These fields are populated during the initial ranking process, and updated as the algorithm progresses. The vector $x.\vec{f}$ contains term frequency information corresponding to each term in the query. The vector $x.\vec{g}$ is initially zero and is updated by the algorithm as elements are reported.

The *score* field contains the current BM25 score for the element, which will change as the values in $x.\vec{g}$ change. The score is computed using equation 1, with the x_t value for each term determined by a combination of the values in $x.\vec{f}$ and $x.\vec{g}$. Given a term $t \in Q$, let f_t be the component of $x.\vec{f}$ corresponding to t, and let g_t be the component of $x.\vec{g}$ corresponding to t, then:

$$x_t = f_t - \alpha \cdot g_t \qquad (2)$$

For processing by the re-ranking algorithm, nodes are stored in priority queues, ordered by decreasing score. Each priority queue PQ supports three operations:

PQ.front()	—	returns the node with greatest score
PQ.add (x)	—	adds node x to the queue
PQ.remove(x)	—	removes node x from the queue

When implemented using standard data structures, the *front* operation requires $O(1)$ time, and the other operations require $O(\log n)$ time, where n is the size of the queue.

The core of the re-ranking algorithm is presented in figure 5. The algorithm takes as input the priority queue S containing the initial ranking, and produces the top-m re-ranked nodes in the priority queue F. After initializing F to be empty on line 1, the algorithm loops m times over lines 2-15, transferring at least one node from S to F during each iteration. At the start of each iteration, the unreported node at the front of S has the greatest adjusted score, and it is removed and added to F. The algorithm then traverses the

```
1   $F \leftarrow \emptyset$
2   **for** $i \leftarrow 1$ **to** $m$ **do**
3       $x \leftarrow S.front()$
4       $S.remove(x)$
5       $x.reported \leftarrow true$
6       $F.add(x)$
7
8       **foreach** $y \in x.children$ **do**
9           $Down\ (y)$
10      **end do**
11
12      **if** $x$ is not a root node **then**
13          $Up\ (x, x.parent)$
14      **end if**
15  **end do**
```

Figure 5: *Re-Ranking Algorithm* — As input, the algorithm takes a priority queue S, containing XML nodes ranked by their initial scores, and returns its results in priority queue F, ranked by adjusted scores.

```
1   $Up(x, y) \equiv$
2       $S.remove(y)$
3       $y.\vec{g} \leftarrow y.\vec{g} + x.\vec{f} - x.\vec{g}$
4       $recompute\ y.score$
5       $S.add(y)$
6       **if** $y$ is not a root node **then**
7           $Up\ (x, y.parent)$
8       **end if**
9
10  $Down(x) \equiv$
11      **if not** $x.reported$ **then**
12          $S.remove(x)$
14          $x.\vec{g} \leftarrow x.\vec{f}$
15          $recompute\ x.score$
16          **if** $x.score > 0$ **then**
17              $F.add(x)$
18          **end if**
19          $x.reported \leftarrow true$
20          **foreach** $y \in x.children$ **do**
21              $Down\ (y)$
22          **end do**
23      **end if**
```

Figure 6: *Tree traversal routines called by the re-ranking algorithm.*

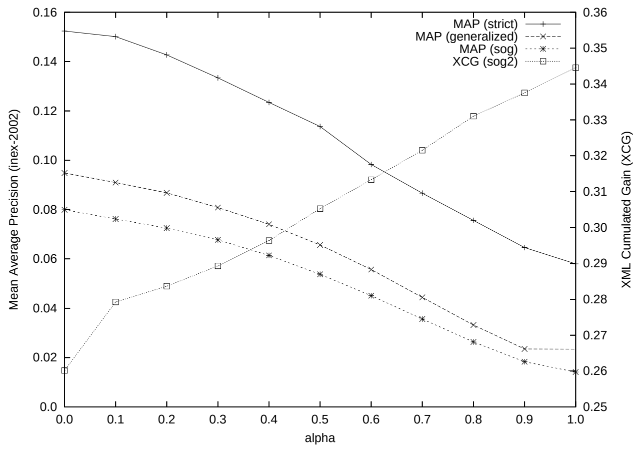

Figure 7: *Impact of α on XCG and inex-2002 MAP (INEX 2004 CO topics; assessment set I).*

node's ancestors (lines 8-10) and descendants (lines 12-14) adjusting the scores of these nodes.

The tree traversal routines, *Up* and *Down* are given in figure 6. The *Up* routine removes each ancestor node from S, adjusts its term frequency values, recomputes its score, and adds it back into S. The adjustment of the term frequency values (line 3) adds to $y.\vec{g}$ only the previously unreported term occurrences in x. Re-computation of the score on line 4 uses equations 1 and 2. The *Down* routine performs a similar operation on each descendant. However, since the contents of each descendant are entirely contained in a reported element its final score may be computed, and it is removed from S and added to F.

In order to determine the time complexity of the algorithm, first note that a node may be an argument to *Down* at most once. Thereafter, the *reported* flag of its parent is *true*. During each call to *Down* a node may be moved from S to F, requiring $O(\log n)$ time. Thus, the total time for all calls to *Down* is $O(n \log n)$, and we may temporarily ignore lines 8-10 of figure 5 when considering the time complexity of the loop over lines 2-15. During each iteration of this loop, a node and each of its ancestors are removed from a priority queue and then added back into a priority queue. Since a node may have at most h ancestors, where h is the maximum height of any tree in the collection, each of the m iterations requires $O(h \log n)$ time. Combining these observations produces an overall time complexity of $O((n + mh) \log n)$.

In practice, re-ranking an INEX result set requires less than 200ms on a three-year-old desktop PC.

7. EVALUATION

None of the metrics described in section 3.3 is a close fit with the view of overlap advocated by this paper. Nonetheless, when taken together they provide insight into the behaviour of the re-ranking algorithm. The INEX evaluation packages (`inex_eval` and `inex_eval_ng`) were used to compute values for the inex-2002 and inex-2003 metrics. Values for the XCG metrics were computed using software supplied by its inventors [11].

Figure 7 plots the three variants of inex-2002 MAP metric together with the XCG metric. Values for these metrics

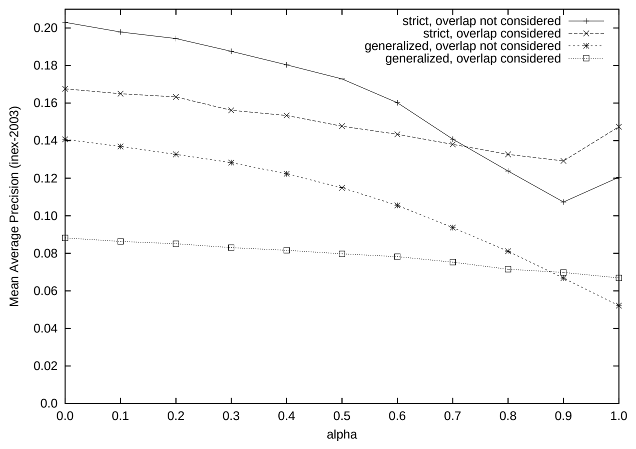

Figure 8: *Impact of α on inex-2003 MAP (INEX 2004 CO topics; assessment set I).*

are plotted for values of α between 0.0 and 1.0. Recalling that the XCG metric is designed to penalize overlap, while the inex-2002 metric ignores overlap, the conflict between the metrics is obvious. The MAP values at one extreme ($\alpha = 0.0$) and the XCG value at the other extreme ($\alpha = 1.0$) represent retrieval performance comparable to the best systems at INEX 2004 [8, 12].

Figure 8 plots values of the inex-2003 MAP metric for two quantizations, with and without consideration of overlap. Once again, conflict is apparent, with the influence of α substantially lessened when overlap is considered.

8. EXTENDED ALGORITHM

One limitation of the re-ranking algorithm is that a single weight α is used to adjust the scores of both the ancestors and descendants of reported elements. An obvious extension is to use different weights in these two cases. Furthermore, the same weight is used regardless of the number of times an element is contained in a reported element. For example, a paragraph may form part of a reported section and then form part of a reported article. Since the user may now have seen this paragraph twice, its score should be further lowered by increasing the value of the weight.

Motivated by these observations, the re-ranking algorithm may be extended with a series of weights

$$1 = \beta_0 \geq \beta_1 \geq \beta_2 \geq ... \geq \beta_M \geq 0.$$

where β_j is the weight applied to a node that has been a descendant of a reported node j times. Note that an upper bound on M is h, the maximum height of any XML tree in the collection. However, in practice M is likely to be relatively small (perhaps 3 or 4).

Figure 9 presents replacements for the *Up* and *Down* routines of figure 6, incorporating this series of weights. One extra field is required in each node, as follows:

$$x.j \quad \text{—} \quad \text{down count}$$

The value of $x.j$ is initially set to zero in all nodes and is incremented each time *Down* is called with x as its argument. When computing the score of node, the value of $x.j$ selects

```
1   $Up(x, y) \equiv$
2       if not $y.reported$ then
3           $S.remove(y)$
4           $y.\vec{g} \leftarrow y.\vec{g} + x.\vec{f} - x.\vec{g}$
5           $recompute\ y.score$
6           $S.add(y)$
8           if $y$ is not a root node then
9               $Up\ (x, y.parent)$
10          end if
11      end if
12
13  $Down(x) \equiv$
14      if $x.j < M$ then
15          $x.j \leftarrow x.j + 1$
16          if not $x.reported$ then
17              $S.remove(x)$
18              $recompute\ x.score$
19              $S.add(x)$
20          end if
21          foreach $y \in x.children$ do
22              $Down\ (y)$
23          end do
24      end if
```

Figure 9: *Extended tree traversal routines.*

the weight to be applied to the node by adjusting the value of x_t in equation 1, as follows:

$$x_t = \beta_{x.j} \cdot (f_t - \alpha \cdot g_t) \qquad (3)$$

where f_t and g_t are the components of $x.\vec{f}$ and $x.\vec{g}$ corresponding to term t.

A few additional changes are required to extend *Up* and *Down*. The *Up* routine returns immediately (line 2) if its argument has already been reported, since term frequencies have already been adjusted in its ancestors. The *Down* routine does not report its argument, but instead recomputes its score and adds it back into S.

A node cannot be an argument to *Down* more than $M + 1$ times, which in turn implies an overall time complexity of $O((nM + mh) \log n)$. Since $M \leq h$ and $m \leq n$, the time complexity is also $O(nh \log n)$.

9. CONCLUDING DISCUSSION

When generating retrieval results over an XML collection, some overlap in the results should be tolerated, and may be beneficial. For example, when a highly exhaustive and fairly specific (3,2) element contains a much smaller (2,3) element, both should be reported to the user, and retrieval algorithms and evaluation metrics should respect this relationship. The algorithm presented in this paper controls overlap by weighting the terms occurring in reported elements to reflect their reduced importance.

Other approaches may also help to control overlap. For example, when XML retrieval results are presented to users it may be desirable to cluster structurally related elements together, visually illustrating the relationships between them. While this style of user interface may help a user cope with overlap, the strategy presented in this paper continues to be applicable, by determining the best elements to include in each cluster.

At Waterloo, we continue to develop and test our ideas for INEX 2005. In particular, we are investigating methods for learning the α and β_j weights. We are also re-evaluating our approach to document statistics and examining appropriate adjustments to the k_1 parameter as term weights change [20].

10. ACKNOWLEDGMENTS

Thanks to Gabriella Kazai and Arjen de Vries for providing an early version of their software for computing the XCG metric, and thanks to Phil Tilker and Stefan Büttcher for their help with the experimental evaluation. In part, funding for this project was provided by IBM Canada through the National Institute for Software Research.

11. REFERENCES

[1] N. Bruno, N. Koudas, and D. Srivastava. Holistic twig joins: Optimal XML pattern matching. In *Proceedings of the 2002 ACM SIGMOD International Conference on the Management of Data*, pages 310–321, Madison, Wisconsin, June 2002.

[2] D. Carmel, Y. S. Maarek, M. Mandelbrod, Y. Mass, and A. Soffer. Searching XML documents via XML fragments. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 151–158, Toronto, Canada, 2003.

[3] C. L. A. Clarke and P. L. Tilker. MultiText experiments for INEX 2004. In *INEX 2004 Workshop Proceedings*, 2004. Published in LNCS 3493 [8].

[4] A. P. de Vries, G. Kazai, and M. Lalmas. Tolerance to irrelevance: A user-effort oriented evaluation of retrieval systems without predefined retrieval unit. In *RIAO 2004 Conference Proceedings*, pages 463–473, Avignon, France, April 2004.

[5] D. DeHaan, D. Toman, M. P. Consens, and M. T. Özsu. A comprehensive XQuery to SQL translation using dynamic interval encoding. In *Proceedings of the 2003 ACM SIGMOD International Conference on the Management of Data*, San Diego, June 2003.

[6] N. Fuhr and K. Großjohann. XIRQL: A query language for information retrieval in XML documents. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 172–180, New Orleans, September 2001.

[7] N. Fuhr, M. Lalmas, and S. Malik, editors. *Initiative for the Evaluation of XML Retrieval. Proceedings of the Second Workshop (INEX 2003)*, Dagstuhl, Germany, December 2003.

[8] N. Fuhr, M. Lalmas, S. Malik, and Zoltán Szlávik, editors. *Initiative for the Evaluation of XML Retrieval. Proceedings of the Third Workshop (INEX 2004)*, Dagstuhl, Germany, December 2004. Published as *Advances in XML Information Retrieval*, Lecture Notes in Computer Science, volume 3493, Springer, 2005.

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

[10] J. Kamps, M. de Rijke, and B. Sigurbjörnsson. Length normalization in XML retrieval. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 80–87, Sheffield, UK, July 2004.

[11] G. Kazai, M. Lalmas, and A. P. de Vries. The overlap problem in content-oriented XML retrieval evaluation. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 72–79, Sheffield, UK, July 2004.

[12] G. Kazai, M. Lalmas, and A. P. de Vries. Reliability tests for the XCG and inex-2002 metrics. In *INEX 2004 Workshop Proceedings*, 2004. Published in LNCS 3493 [8].

[13] J. Kekäläinen, M. Junkkari, P. Arvola, and T. Aalto. TRIX 2004 — Struggling with the overlap. In *INEX 2004 Workshop Proceedings*, 2004. Published in LNCS 3493 [8].

[14] S. Liu, Q. Zou, and W. W. Chu. Configurable indexing and ranking for XML information retrieval. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 88–95, Sheffield, UK, July 2004.

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

[16] Y. Mass and M. Mandelbrod. Component ranking and automatic query refinement for XML retrieval. In *INEX 2004 Workshop Proceedings*, 2004. Published in LNCS 3493 [8].

[17] P. Ogilvie and J. Callan. Hierarchical language models for XML component retrieval. In *INEX 2004 Workshop Proceedings*, 2004. Published in LNCS 3493 [8].

[18] J. Pehcevski, J. A. Thom, and A. Vercoustre. Hybrid XML retrieval re-visited. In *INEX 2004 Workshop Proceedings*, 2004. Published in LNCS 3493 [8].

[19] B. Piwowarski and M. Lalmas. Providing consistent and exhaustive relevance assessments for XML retrieval evaluation. In *Proceedings of the 13th ACM Conference on Information and Knowledge Management*, pages 361–370, Washington, DC, November 2004.

[20] S. Robertson, H. Zaragoza, and M. Taylor. Simple BM25 extension to multiple weighted fields. In *Proceedings of the 13th ACM Conference on Information and Knowledge Management*, pages 42–50, Washington, DC, November 2004.

[21] S. E. Robertson, S. Walker, and M. Beaulieu. Okapi at TREC-7: Automatic ad hoc, filtering, VLC and interactive track. In *Proceedings of the Seventh Text REtrieval Conference*, Gaithersburg, MD, November 1998.

[22] A. Trotman and B. Sigurbjörnsson. NEXI, now and next. In *INEX 2004 Workshop Proceedings*, 2004. Published in LNCS 3493 [8].

[23] J. Vittaut, B. Piwowarski, and P. Gallinari. An algebra for structured queries in bayesian networks. In *INEX 2004 Workshop Proceedings*, 2004. Published in LNCS 3493 [8].