Structural Relevance: A Common Basis for the Evaluation of Structured Document Retrieval

M. S. Ali
University of Toronto
sali@mie.utoronto.ca

Mariano P. Consens
University of Toronto
consens@mie.utoronto.ca

Gabriella Kazai
Microsoft Research
gabkaz@microsoft.com

Mounia Lalmas
Queen Mary, U. of London
mounia@dcs.qmul.ac.uk

ABSTRACT

This paper presents a unified framework for the evaluation of a range of structured document retrieval (SDR) approaches and tasks. The framework is based on a model of tree retrieval, evaluated using a novel extension of the Structural Relevance (SR) measure. The measure replaces the assumption of independence in traditional information retrieval (IR) with a notion of *redundancy* that takes into account the user navigation inside documents while seeking relevant information. Unlike existing metrics for SDR, our proposed framework does not require the computation of an ideal ranking which has, thus far, prevented the practical application of such measures. Instead, SR builds on a Markovian model of user navigation that can be estimated through the use of *structural summaries*. The results of this paper (supported by experimental validation using INEX data) show that SR defined over a tree retrieval model can provide a common basis for the evaluation of SDR approaches across various structured search tasks.

Categories and Subject Descriptors

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

General Terms

Measurement

Keywords

Strucutural Relevance, XML

1. INTRODUCTION

Structured document retrieval (SDR) approaches aim at returning parts of documents that are focused on the user's information need [3]. Such approaches have been investigated for the past six years at the INitiative for the Evalutaion of XML Retrieval (INEX)[1], which has also provided a major forum for the growing body of work in SDR. To date, most proposals in SDR have been concerned with the retrieval of flat text units such as passages [13] and elements [16]. An important omission has been the retrieval of trees from structured documents, an approach which allows the embodiment of the document's structure in the results, while also explicating the user's navigation of the document tree.

The need for retrieving trees has been stated in numerous proposals in the literature. For example, XML query languages such as XQueryFT [2] include support for queries with tree-based structural constraints and specifications for tree-based results. Searching via matching trees to document fragments has been explored in [4]. The combination of passages and elements to represent trees for returning multiple ranges of information from XML documents has been proposed in [6]. In addition, there exists a number of systems, including XRANK [11], XKeyword [12], XXL [24], and XIRQL [10], that provide keyword search on XML and output ranked lists of XML document subtrees.

```
1  <book>
2   <fm>
3     <description type="main">
4       <name type="title">Moby Dick</name>
5       <meta type="etymology">whale [..20 chars]</meta>
6       <meta type="introduction">biography [..15 chars]</meta>
      </description>
7     <description type="creator">
8       <name type="author">Herman Melville</name>
9       <name type="publisher">Plain Label Books</name>
      </description>
10    <description type="toc">
11      <name type="ch">Chapter 1: Looming</name>
12      <name type="ch">Chapter 2: The Carpet-bag</name>
13      <name type="ch">Chapter 3: The Spouter-Inn</name>
      </description>
    </fm>
14  <body>
15    <chapter>whaling ship [..50 chars]</chapter>
16    <chapter>captain [..100 chars]</chapter>
17    <chapter>[..50 chars]</chapter>
    </body>
  </book>
```

Figure 1: Extract from a book with XML markup

To illustrate tree retrieval, consider the problem of retrieving results for a query from a book marked up in XML; an extract of which is shown in Figure 1. The correspond-

[1]http://www.inex.otago.ac.nz/

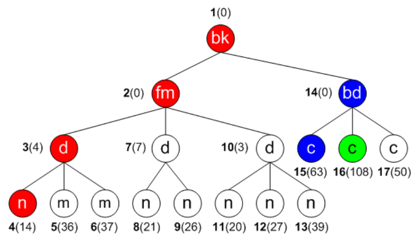

Figure 2: Tree structure of book in Figure 1

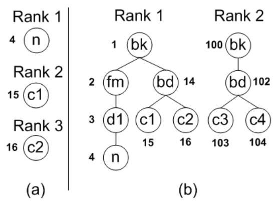

Figure 3: Trees that model different SDR approaches; (a) document/element/passage, (b) tree

ing tree structure is shown in Figure 2; the tags have been abbreviated as follows: `book` (bk), `body` (bd), `description` (d), `name` (n), `meta` (m), and `chapter` (c). The figure shows a unique identifier beside each element node, and the character length of the element content (in brackets). For instance, the element `/bk[1]/fm[1]/d[3]/n[2]` (in XPath notation) corresponds to the node 12 of length 27. The query "ship captain in Moby Dick" matches terms in different structural parts of the book extract, i.e., node 4 (match on "Moby Dick"), node 15 (on "ship") and node 16 (on "captain") - these nodes are shown as shaded circles in Figure 2. Given this, a question faced by retrieval systems is the question of which element(s)/passage(s) to return to the user? The answer depends on the definition of the retrieval task and the SDR approach employed. For example, a document retrieval system may return the root node, in which case users would need to browse the whole book. A focused retrieval system, e.g. [16], may return the individual leaf nodes at separate ranks. This provides the user with more specific information, but at the cost of having to examine three separate results at different rank positions. Alternatively, using a tree retrieval framework, a partial subtree of the document could be returned, which would provide the user both content and context in a single result. Figure 3(b) shows a subtree returned from our example book at the first rank, and a subtree from another document in the second rank. With tree retrieval, a single result can be returned affording direct and focused access to the desired content. In addition, both of the previous results can be modeled as subtrees, as shown in Figure 3 where single nodes (singletons) represent focused results.

While there is growing interest in tree retrieval, there is a notable gap when it comes to the evaluation of tree retrieval approaches. This has been voiced, for example, by the authors of [10], calling for more sophisticated approaches for the evaluation of complex queries, where query terms may appear in different contexts within a document. The need for a tree-based measure was also noted in [9] for comparing the results of Web page aggregation systems. Finally, the authors of EXTIRP [8], a document fragment based search system, were unable to evaluate their system due to the lack of an evaluation measure to estimate the relevance of the trees output by their system.

The most closely related evaluation frameworks have been developed in the context of INEX. However, it is widely known that the evaluation of the range of SDR approaches has challenged INEX since its beginnings. The challenges surrounding this topic has resulted in a great number of publications that examine the critical factors affecting SDR evaluation [25] and many more that propose solutions in the form of new evaluation metrics, e.g., [19, 15, 20, 17]. To date, however, there exists no single, unified framework that allows for the evaluation of all the different types of SDR tasks and approaches. A further key outstanding issue of existing proposals is that a good basis for measuring the effectiveness of systems that return trees has not yet been found.

In this paper, we consider the evaluation of SDR systems in the wider context by proposing an evaluation framework based on tree retrieval, providing a single, unified solution that encompasses all SDR approaches and tasks. We view trees as an underlying model for SDR which allows the unification of the different SDR approaches, incorporating both document-centric and data-centric views on XML, as well as traditional information retrieval (IR). Another key advantage of our framework is that it alleviates the need for an ideal ranking, which is a major limitation of existing SDR metrics. Based on the model of tree retrieval illustrated in the above example, we develop a novel tree retrieval evaluation framework based on an extension of the Structural Relevance (SR) measure [1]. The initial version of SR, developed in [1], is limited to the evaluation of element retrieval tasks. In this paper we extend SR for the evaluation of tree retrieval. An important aspect of SR is that it uses a Markovian model of user navigation which can be estimated through the use of *structural summaries* [7]. It is this property of SR that alleviates the need for an ideal ranking in our framework.

The paper is structured as follows. In Section 2, we describe related work for the evaluation of SDR and show their limitations. Section 3 discusses desiderata for tree retrieval. We present Structural Relevance in Section 4, and apply it to the evaluation of tree retrieval in Sections 5 and 6. Section 7 contains experimental results that show the accuracy and fidelity of SR for two element retrieval tasks, as well as the fidelity of SR for a tree retrieval task. We conclude in Section 8.

2. EXISTING SDR MEASURES

The first metric to take into account the user's navigation along an XML document's structure from a returned element was proposed in [19]. The measure of Expected Ratio of Relevant Documents (*ERR*) is based on an hypothetical user behavior to infer the relevance of elements modeled as entry points into the document. *ERR* is defined as the expected number of relevant XML elements a user sees when consulting the list of the first k returned results divided by the expected number of relevant XML elements a user sees whilst exploring the whole collection.

An extension of the probabilistic PRecall measure [22], where users' navigation is defined stochastically, is the measure of Precision-Recall with User Modeling (PRUM) [20].

The measure is defined as the probability that the user sees a previously unseen ideal element when consulting the context of a retrieved element, assuming that the user wants to see a given number of ideal elements. An ideal element is the most relevant node on a branch in a document and is considered to be the preferred result by users.PRUM estimates the probability that the user visits a given document context to see an ideal element. It employs probability estimations of a user's navigation to ultimately determine the probability of a node being seen by the user. The navigation probability depends on the document's structure and assumptions about the user's interaction. The more structurally related elements that have been returned at earlier ranks, the more likely that users would have seen an element from the same document at further down the ranking, thus the more the element's score is reduced.

A related measure is Expected Precision-Recall with User Modelling (or EPRUM) [18], which substantially reduces the complexity of PRUM and also allows for graded relevance. It defines precision as a ratio of two minimum values: the minimum rank where a given recall is reached over all the possible rankings and the minimum rank where the same recall is reached by the evaluated ranking. For a given recall level, precision in EPRUM is thus defined as the percentage of effort (in minimum number of consulted ranks) a user would have to expend when consulting an ideal ranking with respect to the effort when consulting the evaluated ranking.

The family of the eXtended Cumulated Gain (XCG) measures [15] are based on the concepts of gain and effort that a user obtains and expend while traversing a ranked list. One of the official metrics for INEX 2006 was normalized XCG (nXCG), which is a ratio of the gain the user accumulates examining the system ranking and the gain the user would accumulate when traversing the ideal ranking. In this study, we consider nXCG as the most appropriate measure with which to test our proposal because it explicitly considers both user navigation and redundancy in the evaluation of SDR systems. In addition, nXCG has been validated with extensive multi-year experimental results, and is considered an established benchmark metric for SDR tasks.

All of the metrics described above are based on some model of user browsing behavior and take into account the user's browsing history when estimating performance. While they are specific to element retrieval, they provide insight into how users prefer to seek relevant information by navigating within a retrieved documemt.Of particular importance is the user browsing model defined in PRUM, where users follow structural paths between nodes e and f in a document with a given probability $P(e \rightsquigarrow f)$. In this paper we build on this model - albeit in a different way (see Section 4.1).

A major limitation of the above measures is that they rely on an ideal ranking. The derivation of such a ranking is, however, a non-trivial problem that is still open to research [20, 18, 17]. In addition, studies have shown that performance measured by XCG can be sensitive to how the ideal ranking is built [14].

In summary, existing measures, and their possible extensions for tree retrieval, cannot provide a basis for evaluating tree retrieval because of their reliance on an ideal ranking. The issue of ideality becomes even more complex in the context of trees, necessitating the development and validation of how to assess ideal trees. It is not actually clear whether there exists an ideal tree, in the same way that, e.g., ideal elements exist. In addition, trees are not flat text elements, so we cannot infer their ideality from existing assessments from either element or passage retrieval. In the next section, we investigate the tree retrieval task to better understand the requirements for a tree retrieval metric.

3. DESIDERATA FOR THE EVALUATION OF TREE RETRIEVAL

In order to establish our desiderata (Section 3.3), we first need to take a closer look at the assumptions behind the tree retrieval task (Section 3.1) and the user's browsing behaviour (Section 3.2).

3.1 Tree Retrieval Task

Tree retrieval is the task of returning trees that provide the user with access to nodes of an XML document that are relevant to their information need. It is an approach to SDR; as are element and passage retrieval. Tree retrieval differs from other SDR approaches because trees can represent information items that go beyond flat text units, such as documents, elements or passages. As with other SDR tasks, the task of returning trees to satisfy an information need builds on a more complex notion of relevance that extends beyond the classical content-based criterion. The relevance of a tree depends on both its content and its context. Tree retrieval involves not only finding relevant information, but also finding trees that afford users access to this information. Finally, tree retrieval considers a user's browsing in the document. Users find relevant information by either visiting the nodes in a returned tree, or by browsing the document from the nodes of the tree. The relevance of a tree depends not only on its content and structure, but also on the content and structure of other trees in the same document. Different trees from the same document may afford users access to the same content. This *redundancy* between trees (and, similarly for elements and passages) is a pivotal difference between evaluation in classical IR and tree retrieval (and SDR, in general).

More formally, we model the task of a tree retrieval system as follows:

DEFINITION 3.1 (TREE RETRIEVAL). *The task of a tree retrieval system is to output a ranked list of document subtrees $R = \{t_1, t_2, \ldots, t_k\}$, where R is the ranked list, k is the length of the list, and t_i is a subtree retrieved from a collection C of trees.*

Each document in the collection is represented as a tree:

DEFINITION 3.2 (TREE). *A tree is a connected graph $T = \{T_V, T_E\}$ where T_V is a set of nodes, T_E is a set of edges between pairs of nodes from T_V, $T_E \equiv \{\{x, y\}, x \in T_V, y \in T_V, x \neq y\}$, where there are no cycles in the graph.*

Each node in a document's tree represents a document component. The set of nodes connected by edges to a given node in the tree represent the context of that node. A subset of a tree's nodes that are connected by edges is a subtree. Formally, this is referred to as an induced subtree:

DEFINITION 3.3 (INDUCED SUBTREE). *An induced subtree t of tree T is given as (t_v, t_e) such that the set of nodes*

$t_v \subset T_V$, with a set of edges t_e, must form a tree according to paths defined in the set of edges T_E of the tree T. We refer to induced subtrees simply as subtrees. A subtree is also a tree. Moreover, when we refer to the subtree t as a set, it refers to the set of nodes t_v in the subtree.

3.2 Information Seeking Behavior

As described above, tree retrieval systems return ranked lists of subtrees to the user. Users *consult* the system output going from one rank to the next, *visiting* the nodes of a retrieved subtree. The user stops consulting the output when their information need has been satisfied. A visited node is considered to have been *seen* by the user. After seeing a node, the user may visit additional nodes by *browsing* out of the node into the rest of the document's tree. If all nodes in a tree have been visited by the user then we say that the tree has been *viewed.* This process of visiting, browsing and seeing content (in nodes) within documents to seek relevant information is called the *user navigation.* During this process the user may encounter already seen nodes, thus redundant content. If the user tolerates redundant content, then the relevant content that is seen a number of times remains relevant to the user. If the user does not tolerate redundant content, then relevant content is considered non-relevant if already seen.

For example, consider the ranking $R = \{t_r, t_b\}$, where t_r is the subtree formed by the nodes 1, 2, 3, and 4 in Figure 2 and t_b is the subtree containing the nodes 14 and 15. In consulting R for relevant information, the user first views t_r, visiting each node in t_r. On each visit, the user might seek additional relevant information by browsing out of a node in t_r into the rest of the document. During browsing, the user may visit (and thereby see) one of the nodes in t_b. After the user has viewed t_r (seen all of the nodes in the subtree), he returns to the ranking and goes to view the next result: t_b, visiting each node in t_b. If the user has already visited some of these nodes then parts of t_b could be *redundant* to the user. If the user does not tolerate redundant information, then any relevant information in an already seen node in t_b would become non-relevant to the user when visiting the nodes t_b. This way, the user's *browsing history* affects the user's perceived relevance of a result subtree.

3.3 Requirements for the Evaluation of Tree Retrieval

The evaluation of tree retrieval systems rests on three fundamental requirements:

(i) the relevance of retrieved trees in the output are not independent and depend on whether users tolerate redundancy,

(ii) the purpose of the system is to retrieve trees that afford a user access to relevant information by directly visiting a node in the tree or through navigating from a visited node into the rest of the document, and

(iii) the same relevant information may be expressed in trees of varying structure.

Traditional IR measures cannot be used for tree retrieval due to the dependencies in the output (first requirement). Existing SDR metrics such as XCG and PRUM, if they were extended to trees, would meet the first (i.e., dependencies) and second requirements (i.e., access to relevant information). However, the third requirement of variable-structured trees is not possible to meet when a metric requires an ideal ranking of trees (see Section 2).

We propose structural relevance (SR) as an evaluation measure for tree retrieval that meets the above requirements. In the next section, we show how SR incorporates redundancy and user navigation into an evaluation framework while avoiding the issue of ideality for trees by inferring their relevance value from the assessments of their nodes. SR uses a graph-based model of user navigation that is flexible and can accommodate both exact and approximated models of how users navigate while browsing structured documents in search of relevant information.

4. STRUCTURAL RELEVANCE

Structural relevance (SR) is a measure of the relevance of the results given that the user may find some of the results in the output redundant. SR depends on the user's browsing history, which is the set of trees that a user has viewed. We calculate SR as the expected relevance value of a ranked list given that the user does *not* find the results redundant:

$$SR(R) = \sum_{i=1}^{k} rel_{tree}(t_i) \cdot (1 - p(t_i; R[t_{i-1}])) \quad (1)$$

where the system output $R = \{t_1, t_2, \ldots, t_k\}$ is a ranked list of k subtrees, the browsing history $R[t_{i-1}] = \bigcup_{j=1}^{i-1} t_j$ is the set of subtrees that are ranked higher than the i-th subtree $t_i \in R$, $rel_{tree}(t_i) \in [0, 1]$ is the relevance value of subtree t_i, and the redundancy $p(t_i; R[t_{i-1}])$ is the probability that the nodes in subtree t_i are seen more than once by the user.

A node is the same as a singleton tree. To determine the redundancy between trees, we first define it for singletons and then extend this basic case to the general case of trees. Section 4.1 defines redundancy between nodes as a probability based on how a user fulfills their information need from search results and navigates the collection to seek relevant information. Section 4.2 extends redundancy to trees by extending the user navigation probability between two nodes to two sets of nodes.

4.1 Redundancy of Nodes

We define the redundancy of a node to a user as whether the content of the node has been previously seen by the user in their browsing history of search results. We measure it by finding the probability of whether the content has been seen more than once while the user was seeking to fulfill their information need, given the search results and how the user navigates within the collection to find relevant information.

DEFINITION 4.1 (USER NAVIGATION BETWEEN NODES). *The probability that the content in a node is seen by a user while visiting a different node is*

$$p(e; f) = P(e\ seen | \ Visit\ to\ f) \quad (2)$$

If e and f are from two different documents then e is not seen and $p(e; f) = 0$; if $e = f$ then e is seen with certainty and $p(e; f) = 1$; otherwise, e and f are from the same document, so e is seen with some probability $0 \leq p(e; f) \leq 1$.

SR uses the same probability for the user's navigation between nodes $p(e; f) = P(f \rightsquigarrow e)$ as PRUM [20]. SR differs from PRUM in that SR measures the effect of redundancy on the relevance value of hits in the output, whereas, PRUM measures the near-misses vis à vis the number of ideal elements that a user sees from the output.

Next, we show how $p(e; f)$ is used to estimate $p(e_i; R[e_{i-1}])$, which is the probability that node e_i is seen more than once by the user in the ranking R. We first give an example to illustrate this estimate. $p(e_i; R[e_{i-1}])$ is the same component as found in Equation 1, but applied to nodes (singleton trees).

Example 4.2. Consider $R = \{e_4, e_{15}, e_{16}\}$, a ranked list of nodes, which correspond to nodes 4, 15 and 16 in Figure 2, respectively. We assume that the user will first visit e_4, then e_{15}, and finally e_{16}. Upon the user's visit to node e_{16}, the user's browsing history will contain the higher-ranked nodes $\{e_4, e_{15}\}$. The probability that the content in e_{16} *has been seen* while visiting e_4 is $p(e_{16}; e_4)$ (and similarly for a visit to node e_{15}). So, for node e_{16} to *not be seen*, prior to visiting it, is the probability $(1-p(e_{16}; e_4)) \cdot (1-p(e_{16}; e_{15}))$. The probability that it has been seen is $p(e_{16}; \{e_4, e_{15}\}) = 1 - (1 - p(e_{16}; e_4)) \cdot (1 - p(e_{16}; e_{15}))$.

We can easily generalize with respect to a ranked list of nodes, thus defining $p(e_i; R[e_{i-1}])$ as follows:

$$p(e_i; R[e_{i-1}]) \quad = \quad 1 - \left(\prod_{e_j \in R[e_{i-1}]} 1 - p(e_i; e_j) \right) \qquad (3)$$

where R is a ranked list of nodes, $p(e_i; R[e_{i-1}])$ is the probability that the user will see the content in the i-th node e_i in R more than once, i.e. the redundancy of node e_i in the ranked list R, the browsing history $R[e_{i-1}]$ is the set of nodes in R ranked higher than e_i, and $p(e_i; e_j)$ is the user navigation probability between nodes e_i and e_j as defined in Equation 2. This completes our definition of the redundancy of nodes.

4.2 Redundancy of Trees

In this section, we extend the notions presented in the previous section for the redundancy of nodes to determine the redundancy of trees in a ranked output of subtrees. We base the redundancy between trees on the redundancy between *sets* of nodes. We thus define the redundancy of trees as the probability that the user will see all of the content in all of the nodes of a subtree given a browsing history of subtrees from a ranked list of subtrees.

We rewrite Equation 3, which is for the redundancy of a node, for the redundancy of a subtree in a ranked list of subtrees as:

$$p(t_i; R[t_{i-1}]) \quad = \quad 1 - \prod_{t_j \in R[t_{i-1}]} 1 - p(t_i; t_j) \qquad (4)$$

where R is a ranked list of subtrees, t_i is the i-th subtree in R, the browsing history $R[t_{i-1}]$ is the set of subtrees ranked higher than t_i, and $p(t_i; t_j)$ is the user navigation between trees, which we determine with the next theorem.

The user navigation between trees is measured using the probability that all nodes in a subtree are seen given that the user visited the nodes in a previously viewed subtree.

Theorem 4.3 (User Navigation Between Trees). *The probability that a user, who seeks relevant information using independent visits to nodes to review a tree, has seen all of the content in the nodes in tree t_i after having visited the nodes in tree t_j is*

$$p(t_i; t_j) = \frac{\sum_{f \in t_j} \sum_{e \in t_i} p(e; f)}{|t_i| \cdot |t_j|} \qquad (5)$$

where t_i and t_j are two induced subtrees from the collection, $e \in t_i$ and $f \in t_j$ are nodes, $|t|$ is the number of nodes in tree t and the probability $p(e; f)$ is the user navigation probability that the content in node e will be seen by visiting node f as defined in Equation 2.

Proof. Let the two induced subtrees t_i and t_j from tree T be defined as per Definition 3.3. Assume that each visit to a node by the user is independent. From a visit to node f in subtree t_j, the expected number of distinct nodes from subtree t_i that the user would see is:

$$E[t_i; f] \quad = \quad \sum_{e \in t_i} p(e; f) \qquad (6)$$

where $p(e; f)$ is the user navigation probability that the user will see node e if they visit node f as defined in Equation 2. For each node in t_j, Equation 6 has a maximum value of $|t_i|$. We will refer to t_j as the previous subtree, and t_i as the current subtree. The number of nodes seen in the current subtree from the previous subtree is $\sum_{f \in t_j} E[t_i; f]$. The maximum number of nodes seen is $|t_i| \cdot |t_j|$. The proportion of the nodes in the current subtree that were seen from the previous subtree is $p(t_i; t_j) = \sum_{f \in t_j} E[t_i; f]/(|t_j| \cdot |t_i|)$. This is the probability that the nodes in the current subtree have been seen from the previous subtree. Substituting Equation 6 for $E[t_i; f]$, the probability becomes $p(t_i; t_j) = (\sum_{f \in t_j} \sum_{e \in t_i} p(e; f))/(|t_i| \cdot |t_j|)$. This completes the proof. $\square$

It should be noted that Theorem 4.3 for trees reduces to redundancy of nodes (Equation 2) for singleton trees.

Example 4.4. Consider a system that returns two book XML subtrees $R = \{t_l, t_r\}$ from the same document, see Figure 4. Assume that the user navigation probability is $p(e; f) = 0.5$ for all nodes when $e \neq f$. There are 30 pairwise possible ways to see nodes in t_r from t_l because they contain 6 and 5 nodes each, respectively. The nodes 1 (`bk`), 2 (`fm`), and 14 (`bd`) are common to both subtrees. Consider the terms in the numerator in Equation 5 for node 1 in t_r. We get $p(e_1; e_1) + p(e_1; e_2) + p(e_1; e_{14}) + p(e_1; e_3) + p(e_1; e_{15}) + p(e_1; e_4) = 1 + 5 \cdot 0.5$. This will be the same for nodes 2 and 14 in t_r. For node 7 (`d2`) in t_r, we get $p(e_7; e_1) + p(e_7; e_2) + p(e_7; e_{14}) + p(e_7; e_3) + p(e_7; e_{15}) + p(e_7; e_4) = 6 \cdot 0.5$. This will be the same for node 17 in t_r. With Theorem 4.3, we get $p(t_r; t_l) = (3 \cdot (1 + 5 \cdot 0.5) + 2 \cdot (6 \cdot 0.5))/30 = 0.45$, which is probability that all of the content in t_r will be seen given that the user visited t_l.

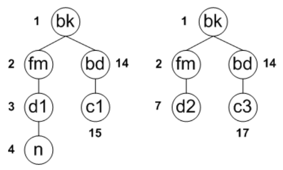

Figure 4: Retrieved Book XML subtrees

In this section, we have defined how the redundancy between trees, i.e. $p(t_i; t_j)$ is calculated (Theorem 4.3), and this can be used to calculate the redundancy of a tree with respect to a ranked list of trees, i.e. $p(t_i; R[t_{i-1}])$. The next step is to calculate $p(e; f)$, which was introduced in Equation 2 and used to calculate $p(t_i; t_j)$.

For large collections, determining $p(e; f)$ for all pairs of nodes is not feasible. For our simple example above, in general to apply SR, we would require 30 pair-wise user navigation probabilities. In the next section, we simplify the user navigation probabilities $p(e; f)$ by showing that they can be approximated using steady-state probabilities if we view user navigation as a Markovian process.

5. USER NAVIGATION

User navigation probabilities must be defined between all pairs of nodes in the collection, which can be expensive to determine. In Section 5.1, we show that the user navigation probability can be approximated as a memoryless process of independent steps along a set of edges in a graph using a Markov model. Then, in Section 5.2, we reduce the number of pair-wise probabilities needed to find SR by partitioning the nodes in the collection of structured documents.

5.1 Approximating User Navigation

In this section, we approximate the user navigation probability, $p(e; f)$, which is the probability that node e is seen given that the user visited node f, by assuming that it is a discrete Markov process. A discrete Markov process is defined as the probability of a process entering a state, given that the system was previously in a known state, where each transition of state in the system is assumed to be independent of the past history of the system. User navigation, as defined in SR, can thus be viewed as a discrete Markov process.

A user navigates between nodes by following paths in a collection C of trees. This probability $p(e; f)$ is defined for all pairs of nodes in the collection. This can be represented as a transition matrix with the nodes in the collection as the axes of the matrix. From any given node f, the total probability to visit all other nodes in the collection of trees C in a single transition is $\sum_{T \in C} \sum_{e \in T} p(e; f) = 1$. To navigate from f to e, a user may require more than one transition to reach node e.

As a Markov process, the probability that, at any given time, a user is in node e, while navigating nodes in the collection to find relevant information, is called the steady-state probability $\pi_{(e)}$ of node e. The steady-state probability is the proportion of the time that a user *spends in* node e while navigating, not the probability that they *navigated to* node e.

Let us assume that to *navigate to* node e a user must *navigate from* another node that is not node e. This means that if a user visits the document that contains e from a node that is not e, the probability that node e will be seen corresponds to the probability that the user is *not* already in e. Thus, we approximate the probability $p(e; f)$ as:

$$p(e; f) \quad \approx \quad 1 - \pi_{(e)} \tag{7}$$

where e and f are nodes in the same document, $1 - \pi_{(e)}$ is the steady-state probability that the user is not browsing in node e, and $e \neq f$. If e and f are from different documents, then $p(e; f) = 0$. If $e = f$, then $p(e; f) = 1$. Thus, we satisfy Definition 4.1.

This avoids the need for pair-wise user navigation probabilities, because the approximation (Equation 7) is constant for a node, given all other nodes in the same document. Another benefit is that, with Markov models, the user navigation probabilities can then be determined using weighting schemes on the edges between nodes.

There are many ways to calculate steady-state probabilities in Markov processes. For SR, a commonly occurring weighting scheme is called a time-reversible Markov process. This is the case where the weight of the edges between nodes are equal and bi-directional such that $w_{ef} = w_{fe}$, where w_{ef} is the weight of the edge between the nodes e and f. The steady-state probabilities for a time-reversible Markov process is computed as:

$$\pi_{(e)} = \frac{\sum_f w_{ef}}{\sum_f \sum_g w_{fg}} \tag{8}$$

where $e, f, g \in T_V$ for tree T.

Some examples of statistics that could be used as weights are the character length of the content in the node, the number of times that participants in a user study were observed to jump between nodes, the time spent browsing in a node, or structural summary statistics such as the number of incoming paths to or outgoing from a node. In the next section, we show how summary statistics are used to calculate the steady-state probabilities. In Section 7, we experimentally validate this approach of using summary statistics for a range of SDR tasks.

In this section, we have shown how to simplify the determination of the user navigation probabilities using steady-state probabilities. But, the computation of the probabilities remains defined over the number of nodes in the collection, which is usually extremely high. In the next section, the computation of SR is further simplified by partitioning the nodes in the collection and showing that the number of user navigation probabilities can be reduced to the number of partitions.

5.2 Summary Models of User Navigation

In this section, we propose structural summaries (see [7]) as a way to further simplifying the calculation of SR.

User navigation between nodes $p(e; f)$ is defined for all possible pairs of nodes in the collection. Through user studies, navigation can be determined, but the studies are costly and it is typically not feasible to test user navigation beyond more than a subset of all of the possible pairs of nodes in the collection. In practice, the user navigation must be inferred for the entire collection based on a limited sample of measurements observed in a user study. This is analogous to considering user navigation as transitions over *partitions* of the nodes in the collection.

Much of the existing work in SDR has been applied to XML retrieval. In the following, we show how to partition a collection using XML as our basis of discussion. XML documents are often represented as trees, with the nodes being XML elements and the edges being the XML label paths. We note that these results, using XML summaries, are easily applied to other structured document IR domains.

XML structural summaries are graphs representing relationships between sets of XML elements with a common

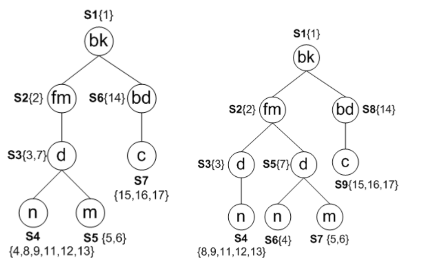

Figure 5: p^* (left) and $p^*|c$ (right) summaries

structure (paths, subtrees, etc.). AxPRE summaries [7] define a broad range of the different summaries available in the literature. They are created using an axis path regular expression language that is capable of describing a plethora of partitioning schemes. For example, a p^* summary partitions the XML elements based on their incoming paths, since p^* is the axis path regular expression describing paths of parent (p) axis traversals. A refinement to this is a $p^*|c$ summary, which partitions elements based on their incoming paths and the labels of their children, since $p^*|c$ is the axis path regular expression describing paths of parent axis traversals unioned with a single child (c) axis step. Conversely, a less refined c summary, partitioning elements only by the labels of their child elements, has the same level of detail as the information given in the content model of the document's DTD or XML Schema.

Figure 5 shows two different summary graphs (p^* and $p^*|c$ summaries, respectively) of the XML document represented in Figure 2. Each partition in the summary is associated with a node in the summary graph, and is referred to using a summary node identifier (SID). Figure 5 is labeled with the SID in bold and the extent of the partition in curly braces. For instance, `/bk/fm/d/m` in the p^* summary in Figure 5 (left) is SID $S5$ whose extent is nodes 5 and 6. The number of elements in each partition is the *extent size*.

The summary provides a convenient way to represent different graphs that characterize user navigation in the collection. The simplifying assumption is that nodes in the same partition share the same user navigation probability with any node in a different partition, such that $p(e_1; f) = p(e_2; f)$ where $e_1, e_2 \in S_e$ are nodes in partition S_e and $f \in S_f$ is a node in partition $S_f \neq S_e$. So, using the approximation in Equation 7, $p(e_1; f) = p(e_2; f) \approx 1 - \pi_{(e_2)}$. To find $\pi_{(e_2)}$, we consider the nodes in partition S_e as a single node in the Markov model, such that $\pi_{S_e} = \pi_{(e_1)} = \pi_{(e_2)}$ and therefore $p(e; f) \approx 1 - \pi_{(e)} = 1 - \pi_{S_e}$.

We can select summary node statistics, such as the extent size or content length of nodes, to weight the edges of the summary graph to estimate the user navigation probabilities.

Example 5.1. What is the probability that a visit to a node in the document shown in Figure 2 would result in the user seeing the content in `/bk/fm/d[2]` (i.e., node 7 or e_7)? Let us assume that user navigation between nodes can depend either on the extent size or the character length of content in the nodes in the collection. A transition matrix weighted by the extent size (number of elements in the summary node partitions) for the incoming p^* summary, and by the character length of nodes for the incoming-outgoing $p^*|c$ summary in Figure 5 is shown in Tables 1 and 2, respectively, where π^{in} and π^{io} represent steady-state probabilities as shown in Equation 7. The steady-state probabilities are calculated using Equation 8. If the user visits e_7 then, from Equation 2, the probability is $p(e_7; e_7) = 1$. Using summary model p^*, for node $f \neq e_7$, the user will see e_7 with a probability of $p(e_7; f) \approx 1 - \pi^{in}_{(e_7)} = 1 - \pi^{in}_{S3} = 0.94$. Similarly, using the $p^*|c$ summary, we get $p(e_7; f) \approx 1 - \pi^{io}_{(e_7)} = 1 - \pi^{io}_{S5} = 1 - 0.1 = 0.9$. This example demonstrates how the calculation of SR can be simplified by selecting different user models vis à vis a partitioning/weighting scheme of the nodes in the collection (e.g., a summary model).

To conclude, in SR, user navigation between nodes can be pair-wise probabilities like those used in PRUM, or it can be approximated based on steady-state probabilities. In SR, the number of probabilities required for characterizing user navigation between nodes can be greatly reduced by partitioning the collection. The user navigation between nodes can be determined by introducing edges between partitions and weighting the edges with factors such as the content length of nodes, the extent size, or the number of times that a partition was visited by participants in a user study. This way, both transition probabilities $p(e; f)$ and steady-state probabilities $\pi_{(e)}$ can be found using well-known solutions for discrete Markov processes [23].

We have shown how to approximate user navigation probabilities and to reduce the number of probabilities required to calculate SR. In the next section, we apply SR to evaluation in tree retrieval.

6. APPLYING SR

SR can be used in precision-based evaluation by replacing the number of relevant hits with SR. So, for instance, structural relevance in precision would be $SRP = SR/k$ where SR is given by Equation 1 and k is the size of the ranked list output. This way, SRP can be used to evaluate the effectiveness of systems at rank cut-offs, which is what we follow in this paper for our experiments reported in Section 7. It can be easily proven (not shown here) that SRP reduces to classical precision when $p(t_i; R[t_{i-1}]) = 0$ for any tree t_i in the ranked list, which captures the case where the user tolerates redundancy in the output.

The remaining task is the determination of $rel_{tree}(t)$ for a given tree t in Equation 1. This is the relevance value of a tree t, independent of whether it has redundant information compared to trees retrieved at earlier ranks.

The relevance of a tree is difficult to assess. Not only is it a complex task (in the context of INEX, it has been shown that the complexity of the assessment process affects the reliability of the assessment data [21]), but there are typically many different trees that can equivalently represent the same instance of relevant information. It is therefore more practical and in fact recommended to determine the relevance of a tree based on the relevance of its nodes. How to assess the relevance of nodes in SDR has been studied within INEX, where now a stable and reliable process has been established[2]. Let us assume that for a given query, each node e has a relevance value $rel(e)$, where for a fully relevant node, $rel(e) = 1$; for a fully irrelevant node, $rel(e) = 0$; and for all other cases, $0 < rel(e) < 1$.

[2]How to obtain assessments at node level is not discussed in this paper. Possible approaches are based on the methodol-

	S1	S2	S3	S4	S5	S6	S7
S1	0	1	1	3	3	6	2
S2	1	0	0	3	0	6	2
S3	1	0	0	0	3	0	0
S4	3	3	0	0	0	6	2
S5	3	0	3	0	0	0	0
S6	6	6	0	6	0	0	0
S7	2	2	0	2	0	0	0
π^{in}	0.23	0.17	0.06	0.20	0.09	0.26	0.09

Table 1: Steady-state probabilities in p^* summary

Thus for a tree t, we define $rel_{tree}(t)$ in terms of the relevance $rel(e)$ of its nodes $e \in t$. Several formulations are possible, for example including the size of the node, its depth, etc. In the experiments reported in this paper, we adopt a simple formulation, which is the average relevance of the nodes:

$$rel_{tree}(t) = \frac{\sum_{e \in t} rel(e)}{|t|} \tag{9}$$

where t is a subtree, e is a node in the subtree, $|t|$ is the number of nodes in the subtree, and $rel(e)$ is the assessed relevance value of the node e.

Example 6.1. Consider again Example 4.4. We want to find the precision (SRP) for the ranked list of subtrees $R = \{t_l, t_r\}$ shown in Figure 4 using the transition matrices of the p^* and $p^*|c$ summaries shown in Tables 1 and 2, respectively. Assume that the relevance of each subtree is $rel_{tree}(t) = 1$ and that the user does not tolerates seeing redundant content. From Equation 1, $SR = \sum_{t_i \in R} rel_{tree}(t_i) \cdot (1 - p(t_i; R[t_{i-1}])$. The redundancy $p(t_r; \{t_l\})$ is derived from the answer in Example 4.4 using the steady-state probabilities in Table 1 where $p(t_r; t_l) \approx 1-(1-(6\cdot\pi_{(e_3)}+6\cdot\pi_{(e_{17})}+5\cdot\pi_{(e_2)}+5\cdot\pi_{(e_{14})}+5\cdot\pi_{(e_1)})/(5\cdot 6))$. Using Table 1, we get $\pi_{(e_3)} = \pi^{in}_{S3} = 0.06, \pi_{(e_{17})} = \pi^{in}_{S7} = 0.09, \pi_{(e_2)} = \pi^{in}_{S2} = 0.17, \pi_{(e_{14})} = \pi^{in}_{S6} = 0.26, \pi_{(e_1)} = \pi^{in}_{S1} = 0.23$. We get that the expected relevance value of the output will be $SR^{in} = 1.138$ with an overall precision (based on rank positions) of $SRP^{in} = SR^{in}/2 = 0.569$, where we model the user navigation as single-step node transitions based on incoming paths, and weight the model with the number of occurrences of the node's label path in the collection. Similarly, using Table 2, we get $\pi_{(e_3)} = \pi^{io}_{S3} = 0.15, \pi_{(e_{17})} = \pi^{io}_{S9} = 0.24, \pi_{(e_2)} = \pi^{io}_{S2} = 0.01, \pi_{(e_{14})} = \pi^{io}_{S8} = 0.24, \pi_{(e_1)} = \pi^{io}_{S1} = 0$. So, $SR^{io} = 1.878$ with an overall precision of $SRP^{io} = SR^{io}/2 = 0.939$. For simplicity here, we ignore the manner in which the results will be presented to the user. Ideally, these weights would be derived directly from browsing behaviour observed in a controlled user study. Here, we have adopted a model of a user who randomly navigates structural paths that are likely to lead to content-laden nodes. The user navigates based on incoming paths and children of nodes, with the propensity to navigate into nodes derived by weighting edges with the content length of child nodes.

This completes our description of our proposed extension of SR to tree retrieval. In the next section, we show experimental results for the measure of SRP.

ogy followed by INEX, see [21].

	S1	S2	S3	S4	S5	S6	S7	S8	S9
S1	0	0	0	0	0	0	0	0	0
S2	0	0	4	0	7	0	0	0	0
S3	0	4	0	133		0	0	0	0
S4	0	0	133	0	0	0	0	0	0
S5	0	7	0	0	0	14	73	0	0
S6	0	0	0	0	14	0	0	0	0
S7	0	0	0	0	73	0	0	0	0
S8	0	0	0	0	0	0	0	0	221
S9	0	0	0	0	0	0	0	221	0
π^{io}	0.00	0.01	0.15	0.15	0.10	0.02	0.08	0.24	0.24

Table 2: Steady-state probabilities in $p^*|c$ summary

7. EXPERIMENTS AND RESULTS

Our goal here is to evaluate our proposed measure of structural relevance in precision (SRP). We are interested in two aspects: the accuracy and fidelity of SRP. Accuracy refers to whether a metric evaluates accurately what it is designed to measure in the system (i.e., evaluation of systems that return redundant subtrees). The fidelity of a measure relates to how consistently it evaluates specific qualities of the system. We evaluate these properties based on the retrieval scenarios studied at INEX and using the INEX test set.

Although this paper argues for the need to go beyond passages or elements in SDR evaluation, the majority of the work on the experimental application and validation of measures for SDR falls into these simpler scenarios. With this in mind, both element and passage retrieval systems have been researched extensively at INEX which provides us with years' worth of experimental data where several measures have been applied to evaluate the retrieval effectiveness of SDR systems.

We evaluate three tasks in ad-hoc element retrieval: Thorough, Focused and Relevant in Context. In Thorough retrieval, the user prefers to see all relevant results (that may overlap). Overlap refers to nodes from the same document that are on the same branch in the document [5]. In Focused retrieval, the user prefers a single result from a set of related elements (i.e., from a branch of the document tree), where results must not overlap. The Relevant in Context task is similar to the Focused task, but results from the same document are clustered into single rank positions in the output.

7.1 Experimental Setup

In this study, system performance was measured using SR (Section 4) and XCG (Section 2). The resultant system rankings from SR (using SRP) and XCG (using nXCG) were compared to determine whether SR was accurate. System ranking in SR was based on mean-averaged SRP across rank cut-offs with a p^* summary (Section 5.2) of the INEX 2006 Wikipedia collection weighted by the extent size. To calculate nXCG, the official metric of INEX 2006, the generalised quantization method was used and the overlap parameter of α was set to 1 for all runs (i.e., overlap was punished and near-misses rewarded). The accuracy of SR was tested in the INEX 2006 Ad-Hoc Thorough and Focused Tasks for 26 systems across 114 topics. Our results are reported in Section 7.2

To determine whether the fidelity of SR was consistent, we tested SR by systematically perturbing and then comparing the evaluations of runs from both the INEX 2006 Thorough Task and the INEX 2007 Relevant in Context Task. For the Relevant in Context Task, we considered the clustering of results from the same document as induced subtrees. We checked the consistency by seeing whether SR would properly evaluate the predicted improvement in the

A. Accuracy - Thorough Task				
	k=10	**k=100**	**k=500**	**k=1000**
	0.069	0.070	0.079	0.129
B. Accuracy - Focused Task				
	k=5	**k=10**	**k=25**	**k=50**
Overall	0.405	0.207	0.545	0.953
Relative	0	0.005	0.028	0.0002
C. Fidelity - Relevant-in-Context (RiC) Task				
	k=10	**k=20**	**k=50**	**k=100**
	8×10^{-33}	8×10^{-33}	9×10^{-33}	4.35×10^{-30}

Table 3: Evaluation (p-values) of SRP across tasks

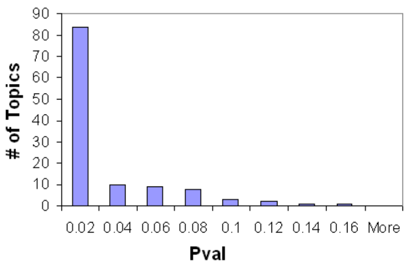

Figure 6: Fidelity of SRP in Thorough Task

output due to different systematic perturbations. The perturbations were as follows: (i) ranked higher the relevant results, (ii) ranked lower the relevant results, (iii) removed overlap, (iv) ranked higher the leaf nodes, and (v) ranked higher the mid-branch nodes. The INEX 2007 Relevant in Context Task was evaluated for 10 systems, across 104 topics. The expected ranking of perturbed runs, for both tasks, was $S_{(i)} \geq S_{(iii)} \geq S_0 \geq S_{(iv)} \geq S_{(v)} \geq S_{(ii)}$, where S_0 is the SRP score of the original system output, and $S_{(\cdot)}$ corresponds to the SRP score of a perturbed run. Our results are reported in Section 7.3.

For the Thorough and Focused Tasks, we used single node assessments from the INEX 2006 element assessments. For the Relevant in Context Task, where we had the opportunity to consider more complex induced subtrees (as opposed to singletons only), we used the INEX 2007 element assessments with the relevance value function proposed in Equation 9.

To compare system rankings, we used Spearman's Rho[3]. We report the p-value which is the probability that two rankings are not correlated. A low p-value ($p_{val} < 0.1$) suggests correlation. In the Focused Task, to test for the accuracy of SRP, the Pearson correlation coefficient was also used (which is the general case of Spearman's Rho) to measure the relative change in the rank of systems across rank cut-offs.

7.2 Accuracy

Table 3(A) shows the results for the Thorough Task. The Thorough Task was evaluated at rank cut-offs of $k = 10, 100, 500, 1000$, and the Focused Task at rank cut-offs of $k = 5, 10, 25, 50$. These cut-offs corresponded to those used for the official results for the respective tasks at INEX 2006. In the Thorough Task, there was strong correlation ($p_{val} < 0.1$) between system rankings using nXCG and SRP for low cut-offs ($k = 10, 100, 500$).

Table 3(B) shows the results for the Focused Task (overall) and the relative change of systems across rank-cutoffs in the Focused Task (relative) . In the Focused Task, SRP was not correlated ($p_{val} >> 0.1$) to nXCG.

We see that nXCG and SRP are different. nXCG measures the effect of overlap, whereas SRP measures the effect of redundancy. Overlap and redundancy are related in that overlap is a specific case of redundancy. In the Thorough Task, overlapping results were allowed in the runs, whereas in the Focused Task, they were not. This resulted in a discrepancy between the two measures in the Focused Task. Therefore, in the Focused Task, we also looked at the relative change in rank for each system across rank cut-offs to see whether SRP changed its system rankings in the same way (although not necessarily to the same degree) as nXCG. We observed that the two measures had strong correlation ($p_{val} < 0.03$) between the relative change of the rank of each system across the rank cut-offs, see Table 3(B). This showed that SRP and nXCG were in agreement in how they measure element retrieval systems. As noted above, the two measures did not completely agree because SRP measures redundancy, whereas nXCG measures only overlap. In the Focused Task, nodes from the same document could be returned, as long as they were not on the same branch in the document. Hence, overlap was not present, but the user could still navigate (and access) the rest of the document. This case of redundancy (accounted for in SRP and thus SR) was not considered in nXCG (and thus XCG).

7.3 Fidelity

For the Thorough Task, Figure 6 shows that there was a strong correlation ($p_{val} < 0.05$) in the expected ranking of perturbed runs across topics. This means that given one of our perturbations of a run, we were able to predict the change in SRP relative to the SRP score of the original run. We conclude that, in this case, SRP appropriately evaluated the performance of the perturbed system outputs.

We evaluated clusters as subtrees for the Relevant in Context Task for outputs with up to $k = 100$ clusters. Table 3(C) shows that the resultant p-values were very low ($p_{val} < 1 \times 10^{-29}$) overall across topics and runs. Thus, SRP displayed excellent fidelity for measuring the Relevant in Context Task as subtrees.

These results are preliminary and will need to be validated in the future with submissions produced and assessments gathered explicitly for a tree retrieval task. The results presented here showed that the fidelity of SRP was consistent for both element retrieval (results for systems that output nodes in the Thorough Task) and tree retrieval (results for systems that output subtrees in the Relevant in Context Task). By modeling SDR tasks as tree retrieval, we believe that our proposed measure will result in a consistent evaluation across SDR retrieval tasks, where a tree-based model of retrieval is applicable.

8. CONCLUSIONS

SDR encompasses many tasks, many of which (if not all) can be considered as special cases of tree retrieval. There is a significant and growing interest in tree retrieval in many areas (e.g., web IR, XML databases, book search). Yet many of the proposals in the literature cannot be evaluated due to the lack of an adequate measure. The existing evalua-

[3]Kendall's Tau would also be an appropriate measure.

tion measures have been developed for specific SDR tasks, mainly in the context of the INEX initiative. In this paper we argued for the need of a general measure for tree retrieval, and we enumerated its requirements.

Our key contribution is the development and the experimental validation of Structural Relevance (SR) as an evaluation measure for the general task of tree retrieval. SR uses a flexible graph-based approach for user navigation in SDR to determine the effect of redundancy on the relevance of results. By assuming a Markovian model of navigation between nodes and applying structural summaries, the computation of SR can be made efficient and flexible across different user navigation assumptions. Furthermore, unlike existing measures in SDR that take into account users' navigation, SR does not require the computation of an ideal ranking.

We applied SR to the evaluation of specific tasks in element retrieval (Thorough, Focused and Relevant in Context tasks). We showed that SR was effective in evaluating these tasks. Compared to the official measures used in the INEX initiative (nXCG), we demonstrated the accuracy and fidelity of SR (implemented to measure precision and using one particular structural summary) across these tasks. We also identified a fundamental difference; while nXCG accounts for just overlap in the results, SR accounts for redundancy (of which overlap is one special case). Finally, we also presented initial experimental validation of the fidelity of SR applied to the general tree retrieval task. Further study is needed which is part of our future work. A comparison to the measures of PRUM [20] and the expected effort-precision/gain-recall measure of XCG [17] are also desired as these are based on similar models of user browsing as SR.

Other future work includes investigating assessments of SDR systems (required if we are to practically model system outputs using trees), the application of tree retrieval in new areas of SDR (such as book search, heterogeneous search, microformats, and semantic web retrieval), and developing similar measures for other IR domains such as email and multimedia.

9. ACKNOWLEDGMENTS

The completion of the research was made possible thanks to Bell Canada's support through its Bell University Laboratories R&D program.

10. REFERENCES

[1] M. S. Ali, M. P. Consens, and M. Lalmas. Structural Relevance in XML Retrieval Evaluation. In *SIGIR 2007 Workshop on Focused Retrieval*, pages 1–8, 2007.

[2] S. Amer-Yahia et al. XQuery 1.0 and XPath 2.0 Full-Text,W3C Working Draft 18 May 2007, 2007.

[3] R. Baeza-Yates and G. Navarro. Integrating contents and structure in text retrieval. *SIGMOD Rec.*, 25(1):67–79, 1996.

[4] D. Carmel, Y. Maarek, M. Mandelbrod, Y. Mass, and A. Soffer. Searching XML Documents via XML fragments. In *SIGIR 2003*, pages 151–158, 2003.

[5] C. Clarke. Controlling Overlap in Content-oriented XML Retrieval. In *SIGIR 2005*, pages 314–321, 2005.

[6] C. Clarke. Range results in XML retrieval. In *INEX 2005*, volume LNCS(3493), pages 4–5, 2006.

[7] M. P. Consens, F. Rizzolo, and A. A. Vaisman. AxPRE Summaries: Exploring the (Semi-)Structure of XML Web Collections. In *ICDE 2008*, pages 1519–1521. IEEE, 2008.

[8] A. Doucet, L. Aunimo, M. Lehtonen, and R. Petit. Accurate retrieval of XML document fragments using EXTIRP. In *INEX 2003*, 2004.

[9] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *WWW 2001*, pages 613–622, 2001.

[10] N. Fuhr and K. Großjohann. XIRQL: A query language for information retrieval in XML documents. In *SIGIR 2001*, pages 172–180, 2001.

[11] L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram. XRANK: Ranked Keyword Search over XML documents. In *SIGMOD 2003*, pages 16–27, 2003.

[12] V. Hristidis, Y. Papakonstantinou, and A. Balmin. Keyword proximity search on XML graphs. *ICDE 2003*, pages 367–395, 2003.

[13] D. Jenkinson and A. Trotman. Focused access to XML documents. In *INEX 2007*, 2008.

[14] G. Kazai. Choosing an Ideal Recall-Base for the Evaluation of the Focused Task: Sensitivity Analysis of the XCG Evaluation Measures. In *INEX 2006*, pages 35–44, 2007.

[15] G. Kazai and M. Lalmas. Extended cumulated gain measures for the evaluation of content-oriented XML retrieval. *ACM Trans. Inf. Syst.*, 24(4):503–542, 2006.

[16] G. Kazai, M. Lalmas, and T. Rölleke. Focussed structured document retrieval. In *SPIRE 2002*, pages 241–247. Springer-Verlag, 2002.

[17] G. Kazai, B. Piwowarski, and S. Robertson. Effort-precision and gain-recall based on a probabilistic navigation model. In *ICTIR 2007*, 2007.

[18] B. Piwowarski and G. Dupret. Expected Precision-Recall with User Modelling (EPRUM). In *SIGIR 2006*, pages 260–267. ACM Press, 2006.

[19] B. Piwowarski and P. Gallinari. Expected ratio of relevant units: A measure for structured document information retrieval. In *INEX 2003*, pages 158–166, April 2004.

[20] B. Piwowarski, P. Gallinari, and G. Dupret. Precision recall with user modeling (PRUM): Application to structured information retrieval. *ACM Trans. Inf. Syst.*, 25(1):1, 2007.

[21] B. Piwowarski, A. Trotman, and M. Lalmas. Sound and Complete Relevance Assessment for XML Retrieval. *ACM Trans. Inf. Syst.*, 2008 (To Appear).

[22] V. Raghavan, P. Bollmann, and G. Jung. A critical investigation of recall and precision as measures of retrieval system performance. *ACM Trans. Inf. Syst.*, 7(3):205–229, 1989.

[23] S. M. Ross. *Introduction to Probability Models.* Academic Press, New York, 8th edition, 2003.

[24] A. Theobald and G. Weikum. The XXL search engine: ranked retrieval of XML data using indexes and ontologies. In *SIGMOD 2002*. ACM, 2002.

[25] A. Trotman. Wanted: Element retrieval users. In *INEX 2005*, volume LNCS(3493), pages 58–64, 2006.