Understanding Content-and-Structure

Jaap Kamps[1,2] Maarten Marx[2] Maarten de Rijke[2] Börkur Sigurbjörnsson[2]

[1] Archives and Information Studies, University of Amsterdam, Amsterdam, The Netherlands
[2] Informatics Institute, University of Amsterdam, Amsterdam, The Netherlands
{kamps,marx,mdr,borkur}@science.uva.nl

ABSTRACT

Document-centric XML is a mixture of text and structure. With the increased availability of document-centric XML content comes a need for query facilities in which both structural constraints and constraints on the content of the documents can be expressed. This has generated considerable interest in both the IR and DB communities, and has lead to the launch of evaluation efforts tailored for XML documents. One of the driving and long-standing research questions here is: How does the increased expressiveness of languages for querying XML documents help users to better, and more effectively, express their information needs? And closely related to this: How should we evaluate systems that enable users to express their information needs using both content and structural constraints?

In this paper we address these research questions. Our analysis follows two lines: What requirements can *in principle* be expressed in query languages for document-centric XML documents? And: How do users *actually* use such languages? For the former, we provide mathematical characterizations of two query languages, one for users with next to no knowledge of the document structure (ignorant users), and one for users that have some, but not complete, knowledge of the document structure (semi-ignorant users). To address the latter issue, we examine the topics formulated in the second query language as part of the 2004 edition of the INEX XML retrieval initiative. Our main findings are as follows: First, while structure is used in varying degrees of complexity, over half of the queries can be expressed in the very restrictive ignorant user language. Second, structure is used as a search hint, and not a search requirement, when judged against the underlying information need. Third, the use of structure in queries functions as a precision device. Fourth, the underlying retrieval task of content-and-structure querying is no different from the ordinary natural language query retrieval task. From those findings we derive a number of recommendations for the evaluation of systems that cater for content-and-structure queries.

1. INTRODUCTION

Increasingly, users, both expert and non-expert, have access to text documents, equipped with some semantic hints through XML-markup. How can we query such data? We could adopt a standard IR approach: perform best match querying using plain text queries. But this would not allow users to specify constraints on the document structure. Alternatively, we could query the documents using a database approach: perform exact-match using XPath queries. But here, effective query formulation is non-trivial and recall is often too low.

Within the INitiative for the Evaluation of XML Retrieval (INEX) [14], the two approaches are combined. Free text search functionality is added to XPath, in the form of a new `about` function. With the same (standard) syntax as the standard `contains` function, the `about` function has two main features; it allows us to (1) express information needs with a mixture of content and structure requirements; and (2) use best-match querying of document-centric XML.

How do users exploit the expressive power offered by such languages? User-oriented studies from INEX have shown that full XPath is too complex for querying document-centric XML documents, even for experts. Moreover, it is unrealistic to assume that casual users have full knowledge of the structure of the documents they want to query. We discuss several XPath fragments (extended with `about`) that are simpler and, we believe, more effective for querying document-centric XML for users.

Our aim in this paper is not to complement the proposed languages with an algebra and implementations—such issues are being addressed elsewhere, see e.g., [1, 9, 10, 23]. Instead, our main aim is to understand how the increased expressiveness of query languages tailor-made for querying XML documents can help users to better, and more effectively, express their information needs. We pursue this aim from two perspectives, subjecting these new query languages to a number of sanity checks: we need to understand their *expressive power*, and we need to assess the types of *information needs* they are meant to address. To deal with the former issue we relate the query languages to logical languages, for which expressiveness results are well-known. To deal with the latter issue we analyze a set of user queries formulated in the INEX query language, and the sets of elements judged relevant by these users, both made available through INEX. Our main findings are as follows: First, while structure is used in varying degrees of complexity, over half of the queries can be expressed in the very restrictive *ignorant user* language. Second, structure is used as a search hint, and not a search requirement, when judged against the underlying information need. Third, the use of structure in queries functions as a precision device. Fourth, the underlying retrieval task of content-and-structure querying is no different from the ordinary natural language query retrieval task. Building on our user-oriented findings, we address the second aim of this paper: understanding how we should evaluate systems that cater for content-and-structure queries; we derive recommendations concerning topics, metrics, and assessments.

In Section 2 we provide background on querying document-centric XML. In Section 3, we discuss content-oriented flavors of XPath and provide semantic characterizations of their expressive power. Section 4 describes the content-and-structure language used at INEX 2004, and analyzes the resulting queries and assessments. In Section 5, we describe how structure helps users improve the quality of retrieval results. In Section 6 we change tack and derive implications of our user-oriented findings for the evaluation of content-and-structure retrieval engines. We conclude in Section 7.

2. QUERYING XML

XML can be used to mark up content in various ways. Based on the content, XML documents are often categorized into two groups: *data-centric* and *document-centric*. The former contain highly structured data marked up with XML tags, an example being geographic data in XML [20]. Document-centric documents are loosely structured documents (often text) marked-up with XML, with electronic journals in XML providing important examples. For our experiments we use the document-centric XML collection that comes with the INEX test suite [14]. It contains over 12,000 articles from 21 IEEE Computer Society journals, marked up with XML tags. On average an article contains 1532 elements and the average element depth is 6.9. About 170 tag names are used, such as articles ⟨article⟩, sections ⟨sec⟩, author names ⟨au⟩, affiliations ⟨aff⟩, etc.

Whereas emerging standards for querying XML, such as XPath and XQuery, can be very effective for querying data-centric XML, another approach seems to be needed for querying document-centric XML. The latter task is a natural meeting point of two disciplines: the XML nature of the documents calls for methods from the database field for querying structure, and the textual nature of the documents calls for approaches from the field of information retrieval (IR) (cf. [31, Section 5]). It is interesting to contrast the two subtasks. As to querying structure, XML query languages such as XPath have a definite semantics. Judging whether an element satisfies an XPath query can be done by a machine (XPath processor), based on the pattern appearing in the XML document, using an *exact match* approach. It is clearly defined which nodes or elements match a given query. An XPath processor will return precisely these elements with no inherent ranking of results. In contrast, for querying text IR uses free text queries. These can be keywords or full sentences describing an information need. An IR system uses a *best match* approach: it attempts to rank results by their topical relevance to the user's query.

As pointed out above, several studies are dedicated to understanding the formal and/or computational properties of hybrid content and structure query languages. Our focus is different: on query languages as a means for users to express their information needs more precisely (as opposed to queries with no structural constraints).

3. THE EXPRESSIVE POWER OF CONTENT ORIENTED XPATH

To query document-centric XML documents we need a hybrid query language, in which content and structural requirements can be expressed and mixed. At INEX, an XPath-like query language has been proposed for this purpose. The syntax of the language looks like XPath, but does not have the same strict semantics. It can be seen as an *extension* of a *subset* of XPath.

In this section, we will first motivate why XPath needs to be *restricted* and examine some fragments of XPath (Section 3.1). We will then motivate why those fragments need to be *extended* with the `about` function (Section 3.2).

3.1 Restricting XPath

Experience from INEX has shown that people—in this case, academics familiar with query languages—have great difficulties in using (the navigational part of) XPath to formulate queries that combine content and structural aspects [24]. The restriction to navigational XPath was originally motivated by the fact that it is a widely used technology, whence it was assumed that it would be easily learnable. This assumption proved to be wrong.

Based on the extensive data described in [24], we argue that the cause of users' difficulties in writing content-and-structure queries can be traced back to a combination of two related items: (1) Users have no, or at best incomplete, knowledge of the structure of documents, i.e., of the DTD.[1] (2) Users have problems handling the expressive power of full XPath. In particular, the fact that the same query can be expressed in several fundamentally different ways proved problematic for users. These observations give rise to two constraints on XPath fragments: it should be possible to formulate information needs even with limited knowledge of the DTD, and the expressive power should be restricted.

A user's knowledge about a set of documents can be naturally formalized in terms of an *indiscernibility relation* over the elements selected by an XPath query: a binary relation that identifies elements in a document. What does such a relation have to do with query languages? We say that a language is *safe* or *well-designed* if indiscernible elements cannot be distinguished by an expression in the query language. This design criterion will help us single out natural XPath fragments. In fact, the fragments discussed below have a perfect fit with two user profiles formalized by an indiscernibility relation: not only are they safe, they are also complete in the sense that every first-order definable set of indiscernible elements can be defined in the language.

Below, we define two user profiles, both capturing users with limited knowledge of the DTD. First, we consider, what we call, *ignorant users* who only know the tag names. Second, we consider *semi-ignorant users*, who know the tag names and have some clue about the hierarchal structure of the elements, without knowing the full details. For both profiles we will design fragments that are *safe* for the sketched user profiles; we interpret this as saying that the chance that a user makes a semantic mistake when describing her information need in terms of XPath is minimal. For clarity, in this subsection we only consider the *navigational* part of XPath. The next subsection deals with the `about` function.

Ignorant Users. Users formulating queries at INEX did not have a clear idea of the DTD of the collection [24]. Typically, they browsed the documents and picked up some

[1]The DTD of the INEX XML document collection was extremely complex. There were 192 different content types, including 11 different tag names for representing paragraphs.

knowledge about the available tags in this manner. This can be viewed as an XML version of fielded search. For users who know (a subset of) the tag names, but do not (want to) know the structure of the documents, we create an XPath fragment which exactly fits their knowledge. Specifically, our ignorant user is able to ask questions like: "Give me sections about weather forecasting where an author is affiliated in California". In a hybrid XPath-like language this could be written as:

```
//sec[about(.,'weather forecasting') and
                              //aff[about(.,'California')]
```

More generally, the user can express her information need as a conjunction of two boolean formulas: one restricting the element of interest, and the other restricting the surrounding document. The following syntax, which we call *non-structure aware XPath* allows this. A query is of the form `//::tag[P]`, where `tag` is either the wild card `*` or a tag name, and `P` is a predicate created using '`and`,' '`or`,' and '`not`' from location paths of the form `//::tag`. Note that when $//::\mathtt{t}$ is used in a filter it means "there exists a descendant of the root with tag `t`". I.e., `//::t` simply says that somewhere in the document there is a `t` element.

We turn to a semantic characterization of this fragment. In social network theory [32] several indiscernibility relations have been proposed, including the useful and robust notion of *bisimulation* (a.k.a. 'regular equivalence'). We need the following special "structurally unaware" version.

Definition 1. Let D, D' be documents and B a non-empty binary relation between the elements of D and D'. We call B a *structure unaware bisimulation* if, whenever xBy holds for two elements x, y in D, then

1. x and y have the same tag name;
2. if there exists an $x' \in D$, then there exists a $y' \in D'$ such that $x'By'$; and
3. conversely for $y' \in D'$.

Let $\phi(x)$ be a first-order formula (in one free variable) in a suitable vocabulary; $\phi(x)$ is *invariant under bisimulations* whenever the following holds: for any a, b and bisimulation B, if $\phi(a)$ and aBb hold, then $\phi(b)$ holds as well.

A few comments. First, since we are usually comparing elements within a single document, our notion of indiscernibility relation is an *auto*-bisimulation, where D and D' in Definition 1 are the same document. Secondly, in the usual definition of bisimulation, the clauses in items 2 and 3 above are conditioned on x' (and y') being "structurally" related to x (and y, respectively); but our ignorant user is not aware of the structure, hence we omitted these conditions.

Theorem 2. *1. Elements that are related by a structure unaware bisimulation cannot be distinguished by a non-structure aware XPath expression.*

2. Every first-order formula that is invariant under structure unaware bisimulations is definable by a non-structure aware XPath expression.

We can conclude that this language fits perfectly to the sketched user profile: the first part of the theorem states that it is *safe*, the second that it is *complete.*

Semi-Ignorant Users.

For semi-ignorant users, we will define two equivalent XPath fragments. One coincides with the fragment proposed in [24] and is supported by the query working group at INEX 2003 [29]. We will show that these fragments have a meaningful semantic characterization. The fact that these fragments fits a common user profile is strong evidence for its naturalness.

Semi-ignorant users have some ideas about the hierarchical structure of the documents. E.g., they know that paragraphs are below sections but, as pointed out in [24], they need not know that there *can* be elements in between. For this reason, [24] proposes Positive Descendant XPath: the fragment of XPath in which only the descendant axis may be used and the booleans in the predicates are restricted to "`and`" and "`or`".

We sketch two possible ways in which semi-ignorant users might pose queries. Suppose a user is interested in 'bisimulation' theorems which appear in sections about 'XPath.' He knows about the theorem tag ⟨`theorem`⟩ and the section tag ⟨`sec`⟩; he also knows that theorems can be nested somewhere inside sections. This user might ask:

```
//sec[about(.,'XPath')]//theorem[about(.,          (1)
                                    'bisimulation')]
```

Another user might formulate the same need as:

```
//theorem[about(.,'bisimulation') and              (2)
              ancestor::sec[about(.,'XPath')]
```

The two users seem to engage in different mental processes when formulating their queries. The first thinks top-down: zoom in on a relevant section and then specify what sort of information should be retrieved from that section. The second approaches the problem bottom-up: determine a segment of interest and then think about sections that might contain the segment. The authors of this paper disagree on which scenario is more natural. Both scenarios can be captured in an XPath fragment, and we will show that the two fragments are equivalent.

To admit formulation (2) above, we need to allow both descendant and ancestor relations. We provide O'Keefe and Trotman's fragment [24] with a double characterization: a semantic one in terms of simulations, and a syntactic one, as a fragment of a well-known language in computer science, the temporal logic CTL. First, we need some definitions.

Definition 3. *Positive Temporal XPath* consists of queries of the form //`tag`[`P`], where `P` is in the following restriction of navigational XPath:

- the only axis relations are `descendant` and `ancestor`;
- only boolean `and` and `or` can be used in filters.

As none of the above two XPath fragments contains negations, bisimulation is too strong a notion [18]. As a general fact, positive fragments correspond to simulations, which are bisimulations from which one of the directions is dropped. We use $<$ to denote the descendant relation between elements; i.e., $x < y$ means that y is a descendant of x.

Definition 4. Let D, D' be documents and B a non-empty binary relation between the elements of D and D'. We call B a *temporal simulation* if, whenever xBy, then

1. x and y have the same tag names;

2. if there exists an $x' \in D$ such that $x < x'$, then there exists a $y' \in D'$ such that $y < y'$ and $x'By'$; and

3. similarly when $x' < x$.

Temporal simulations correspond to users that know the element hierarchy: note that both elements below *and* above have to be simulated. The next theorem is an analogue of Theorem 2 for Positive Descendant XPath: it is both safe and complete for semi-ignorant users.

Theorem 5. *Let X be a set of nodes. The following are equivalent on trees.*

1. *X is definable by a first-order formula in one free variable in the signature with $<$ and unary predicates which is preserved under temporal simulations.*

2. *X is definable as the answer set of a Positive Descendant XPath formula.*

3. *X is definable as the answer set of a Positive Temporal XPath query.*

The proof uses ideas from modal logic [3, Theorem 2.78] together with ideas from [2, Theorem 3.2]. We conjecture that the language in item 3 of Theorem 5 is exponentially more succinct than the language in item 2.

3.2 Extending XPath

Now that we have looked at restrictions of the navigational part of XPath to "manageable" fragments, we look at extensions with the `about` function. Although `about` has the same syntax as the XPath function `contains`, their semantics are radically different. Because of its strict, boolean character, `contains` is not suitable for text rich documents. The semantics of `about` is meant to be very liberal. Consider the element `<aff>'Stanford University'</aff>`. A human assessor will likely decide that `about(.//aff,'California')` returns true if that element is below the node of evaluation; but an XPath processor equipped only with `contains` would have difficulties trying to do the same. As a more elaborate example, look at the following query (against a collection containing several articles):

> Find articles where the author is affiliated in California. From those articles return sections about weather forecasting systems.

In a hybrid syntax, mixing content and structure, this would be something like

```
//article[about(.//au//aff,'California')]//sec[
            about(.,'weather forecasting systems')]
```

This query has two content-based restrictions, linked by a structural constraint. The semantics of this query is not strict. In the spirit of information retrieval, the ultimate decision of relevance is in the hands of a human assessor, who may bring lots of context and world knowledge to her judgment. E.g., a human assessor is likely to judge a section about 'storm prediction systems' to be relevant to the information need expressed above.

4. EXPRESSING INFORMATION NEEDS WITH CONTENT-AND-STRUCTURE

Now that we have seen what properties can *in principle* be expressed by ignorant and semi-ignorant users in their respective hybrid query languages, we take a closer look at what users *actually* express in the semi-ignorant query language used at INEX 2004. We will see that many of the queries submitted can actually be expressed in the ignorant sublanguage. But let's not run ahead of ourselves.

4.1 The INEX Query Format

At INEX, two types of topics are used. Content-Only (CO) topics and Content-And-Structure (CAS) topics. All topics contain the same three fields as traditional IR topics [6, 11]: title, description and narrative. The description and narrative describe the information need in natural language. The difference between the CO and CAS topics lies in the topic title. In the case of the CO topics, the title describes the information need using a small list of keywords. In the case of CAS topics, the title describes the information need using (a flavor of) XPath extended with the `about` function. At INEX 2003, full XPath was allowed, and at INEX 2004 descendant positive XPath (i.e., the restricted fragment for semi-ignorant users) is used [29, 30]. Below we analyze the title part of the CAS topics.

4.2 INEX 2004 Queries

The specific instructions for topic development at INEX 2004 [28] stated that CAS queries

- should use only descendant axis (i.e., `//`),
- should use only boolean `and` and `or`,
- should contain at least one `about` statement, and
- the rightmost filter should be an `about` statement.

The resulting language is called NEXI (Narrowed Extended XPath I) [30]. We consider the set of 34 CAS topics (version 2004-7) with topic numbers 127–147, and 149–161.

4.2.1 Knowledge of the document structure

Because of the restrictions just listed, the NEXI language is a proper subset of the *semi-ignorant user* language discussed in Section 3. We can break down the 34 NEXI topics based on the two types of users identified in Subsection 3.1.

Ignorant Users. They know only (some of) the tag names in the collection, but are ignorant of the structure of the documents. In total there are 11 topics that reflect this type of user.[2] The topic numbers are 128, 134, 136, 141–143, 145, 151, 152, 159, and 160.

Semi-Ignorant Users. These users have some idea of the hierarchical structure of the documents. I.e., they know (some of) the tag names in the collection, and (some of) the legitimate nesting of tags. There are 23 topics that reflect this type of user. The topic numbers are: 127, 129–133, 135, 137–139, 140, 144, 146, 147, 149–150, 153–158, 161.

[2]We ignore the restriction to only returning XML elements within articles [19]. I.e., most queries start with `//article` to reflect this constraint. Only three queries do not start with this prefix, however these queries are prefixed with either $\langle$sec$\rangle$ or $\langle$abs$\rangle$ tags that only occur in the context of an $\langle$article$\rangle$ tag anyway.

Element	Frequency	Percentage
sec	16	47.06%
article	5	14.71%
p	4	11.76%
*	2	5.88%
abs	2	5.88%
bb	1	2.94%
bdy	1	2.94%
bib	1	2.94%
fig	1	2.94%
vt	1	2.94%

Table 1: Frequency of requested elements in the 34 CAS topics of INEX 2004.

Even with the explicit instructions to construct content-and-structure queries, no less than one-third of the resulting queries can be expressed in the very restricted *ignorant user language* introduced in Section 3.

4.2.2 *Requested elements*

One of the main advantages of using CAS queries is that they allow the user to specify the types of elements that should be returned. Table 1 lists the elements resulting from the granularity constraints in the 34 CAS topics.

4.3 The INEX 2004 Assessments

At INEX, relevance is assessed on the basis of the narrative describing the underlying information need. As Kazai et al. [16, p.237] put it:

> CAS queries are topic statements, which contain explicit references to the XML structure, and explicitly specify the contexts of the user's interest (e.g. target elements) and/or the contexts of certain search concepts (e.g. containment conditions). [...] Although users may think they have a clear idea of the structural properties of the collection, there are likely to be aspects to which they are unaware. The idea [...] is to allow the evaluation of XML retrieval systems [...] where not only the content conditions within a user query are treated with uncertainty but also the expressed structural conditions. [...] The path specifications should therefore be considered hints as to where to look.

In the spirit of textual IR, the granularity constraint is not strictly enforced, but merely regarded as a retrieval hint. Hence, it is of interest to look at the tag-names of elements that are judged relevant for the respective topics.

4.3.1 *Elements judged relevant*

We use version 3.0 of the assessments, containing judgments for the 26 topics numbered 127–137, 139–145, 149–153, and 155–157. Moreover, we focus on elements rated as highly exhaustive and highly specific—also called strict or (3,3) assessments. For the 4 topics numbered 133, 140, 143, and 144, there are no elements judged as highly exhaustive and highly specific. Table 2 lists the frequencies of element types judged relevant for the remaining 22 CAS topics. We collapse the tag equivalences for sections and paragraphs, as defined in [28].

Element	Frequency	Percentage
p+	854	31.41%
vt	747	27.47%
sec+	262	9.64%
au	110	4.05%
bb	104	3.82%
fnm	104	3.82%
st	90	3.31%
article	73	2.68%
fig	53	1.95%
it	37	1.36%
bdy	36	1.32%
ref	34	1.25%
scp	32	1.18%
atl	23	0.85%
abs	13	0.48%
fm	11	0.40%
b	10	0.37%

Table 2: Frequency of elements judged relevant for all assess CAS topics at INEX 2004. We only show tag names occurring at least 10 times.

	article	sec+	p+	abs	vt
article (2)	10.8%	1.3%	1.6%	–	–
sec (10)	3.3%	27.7%	24.7%	0.9%	0.4%
p (4)	4.0%	26.0%	48.0%	–	–
abs (2)	16.0%	–	24.0%	24.0%	–
vt (1)	–	–	44.0%	–	52.0%

Table 3: Frequency of relevant elements (columns) for topics with particular granularity constraint (rows). The number of aggregated queries is indicated between brackets.

4.3.2 *Cross product*

We also investigate how often the element that is judged relevant actually has the tagname specified by the granularity constraint. Consider Table 3; the rows show the tag-names of elements resulting from the granularity constraints, and the columns show the tag-names of elements judged relevant.[3] It is clear from the table that the granularity constraint can indeed be considered as a retrieval *hint*. Although it is far from strictly enforced, there seems to be a preference for the type of XML elements satisfying it.

4.4 How Structure is Used

To further our understanding of the role of structure in content-and-structure topics, we break down the set of topics by increasing complexity. We define four categories:

Restricted Search. This category has topics in which structure is only used as a granularity constraint. The topic is an ordinary content-only topic, where the search is restricted to particular XML elements. There is a filter on the target element having no nested path constraint. A typical example of such a topic is to restrict the search to sections, this may look like:

```
//sec[about(., ''xxx'')].
```

[3]Note, especially for granularity constraints abs and vt, that we do not distinguish between paragraphs appearing within or outside the element matching the granularity constraint.

This category has 5 topics: 127, 136, 142, 143, and 152.

Contextual Content Information. This category is similar to the *Restricted Search* category, but now there may be a content restriction on the environment in which the requested element occurs. A typical example of such a topic is one asking for sections from articles with a content restriction on the abstract, this may look like:

```
//article[about(.//abs, ‘‘xxx’’)]//sec[about(., ‘‘yyy’’)].
```

The category contains 16 topics: 128, 129, 130, 131, 132, 134, 135, 137, 138, 141, 144, 145, 151, 158, 159, and 160.

Search Hints. This category contains topics with a complex filter in which a nested path occurs, but the element targeted by the nested paths resides inside the requested element. I.e., the user provides a particular retrieval cue to the system. An example of such a topic may be, when interested in sections on a topic, to tell the system to look for certain terms to appear in a theorem environment, this may look like:

```
//sec[about(., ‘‘xxx’’) and about(.//thm, ‘‘yyy’’)].
```

There are 2 topics in this category, numbered 147, and 153.

Search Hints in Context. The fourth and last category deals with topics with a nested path that targets elements that are disjoint from the requested element. Here, the user is really exploiting her knowledge of the structure of the documents, and conditions the retrieval of elements on the content found along other paths in the document tree. I.e., the condition is evaluated against parts of the text that are not being returned to the user as a result. E.g., one might be looking for sections, in papers authored by someone. This may look like:

```
//article[about(.//fm//au, ‘‘xxx’’)]//bdy//sec[about(.,
    ‘‘yyy’’)].
```

There are 11 topics in this category, numbered 133, 139, 140, 146, 149, 150, 154, 155, 156, 157, and 161.

Carmel et al. [4, 5] proposed XML fragments as another, simple alternative to XPath for content and structure queries. Using the intuitive query-by-example underlying XML Fragments, only the *Restricted Search* and *Search Hint* categories can be expressed. For capturing queries in the other categories, a syntactic device is introduced [4].

5. HOW DOES STRUCTURE HELP?

If users are aware of the structure of documents in a collection, they can query the collection by means of constraints on both the content and the structure of desired XML elements, giving rise to the following

> **CAS Hypothesis** Hybrid content-and-structure queries are more expressive than ordinary natural language queries, and this will lead to better retrieval performance.

While the CAS hypothesis has great intuitive appeal, it is also rather vague and underspecified. In what sense can the structural part help to improve retrieval performance?

The use of structure in queries has been studied extensively in the literature; prominent examples include booleans, proximity and phrase operators. In early publications, the usage of phrases and proximity operators—as well as a careful usage of boolean operators—showed improved retrieval results [7, 8, 12, 13, 17], but rarely anything substantial. As retrieval models became more advanced, the usage of query operators was questioned. E.g., Mitra et al. [22] conclude that when using a good ranking algorithm, phrases have no effect on high precision retrieval (and sometimes a negative effect due to topic drift). Rasolofo and Savoy [25] combine term-proximity scoring heuristics with an Okapi model, obtaining 3%–8% improvements for Precision@5/10/20, with hardly observable impact on the MAP scores. Mishne and de Rijke [21] found that even on top of a good basic ranking scheme for web retrieval, phrases and proximity terms may bring improvements in retrieval effectiveness, both for MAP and high precision measures.

Where does this leave us with our content-and-structure queries? First, it is interesting to analyze how the expressiveness of content-and-structure queries is put to use, and, in particular, in what sense this may lead to better retrieval performance. To address the issue, we return to the four topic categories from Section 4. For each category, one would expect the structural constraints to have a precision-enhancing effect, either by specifying or constraining the granularity of the elements being sought (as in the *Restricted Search* category), or by constraining the environment in which the results being sought appear (as in the *Contextual Content Information* and *Search Hints* categories, or by imposing “non-local” structural and content constraints (as with the *Search Hints in Context* category). Overall, then, our expectation is that structural aspects of the query improve precision. Although this may reduce fall-out, it will also reduce recall. So, it is not clear what we may expect for CAS queries in terms of mean average precision. If we expect CAS queries to function as a precision device, then we should put more emphasis on measures that reflect this.

There is some experimental evidence that confirms the precision enhancing nature of structural constraints. As explained above, at INEX 2004 topics were assessed while treating the structural constraints as hints. Sigurbjörnsson et al. [27] experimented with a content-only based approach [15] vs. a content-and-structure based approach [26]: whereas the CO based approach resulted in superior MAP, the strict CAS-based approach resulted in improved early precision scores (MRR, Precision@10, etc). These findings suggest that the retrieval task underlying content-and-structure querying is no different from the ordinary natural language query retrieval task: they may be used as different ways of articulating the same information need.

6. EVALUATION LESSONS

The user-oriented upshot of the previous sections is that users use CAS in different, and often shallow or restricted ways, most likely as a precision enhancing device. How do these findings inform us about the *evaluation* of systems that handle content-and-structure queries? We discuss several aspects. We will first describe at a high level how we expect users to interact with a retrieval engine that supports content-and-structure querying.

Let’s imagine the mental process the user goes through when interacting with a retrieval engine. First, she starts with an abstract, informal information need. Then she needs to articulate her information need more formally. Naturally she would first use natural language and formulate her information need as a short list of keywords. If our user has additional knowledge about the types of documents or elements that would satisfy her information need, in particular about the document structure, she may consider rendering her information need in a structured language. The resulting query is, presumably, a more precise description of the original information need. However, the underlying information need has not changed. If our user doesn't have such additional knowledge, she simply won't add structure to her query. Either way, she will take her formulation of the information need, structured or not, to the retrieval engine. Independent of whether the query has structure, the task of the retrieval engine is to answer the user's information need. In the end, the success of the search process depends solely on whether the user's information need was satisfied.

Against this background, our findings from earlier sections have some clear implications for the evaluation of content-and-structure querying. Let's consider the various stages of the evaluation process, starting with topic development process. The topic formulation process could start with writing a detailed natural language description of the information need. This will imitate the formation of a mental image of the information need. The next step would be to formulate the need as a list of keywords. Then, in case the user/topic creator has the appropriate knowledge, and the collection supports it, the user/topic creator can also formulate her information need using a mixture of content and structure requirements. What we end up with, then, is one set of topics, all of which have a natural language title. However, a subset of the topics will also have a content-and-structure title. Of course, all topics have a narrative which verbosely describes the information need.[4]

The assessments will not differ from traditional IR assessments. The narrative will be authoritative when judging results. For topics whose information need is expressed in different query formats (i.e., with and without structural constraints), we get one query assessed for free.

In the evaluation phase, the added value of having information needs both expressed in natural language and in a structured language allows us to directly (and only for appropriate information needs) measure whether structural constraints can indeed be used to enhance keyword based queries. Systematic experiments with multiple ways of expressing the same information need, would help make progress on research questions such as our *CAS Hypothesis* in Section 5."

Turning to metrics now, as we pointed out, we expect structural hints to be a precision enhancing device. The evaluation should try to answer whether that is actually the case by comparing systems not solely based on MAP but also on initial/early precision metrics such as MRR, Precision@10, etc.

It is important to realize that a content-and-structure query depends on both the information need and the structure of the document collection. Whereas natural language queries usually depend solely on the information need, structured queries also crucially depend on knowledge of the types of elements that are relevant, i.e., what tags-names they have, or how these are nested. This implies that, in a sense, structure is never an inherent part of an information need itself; at most, it is part of a formal query that offers one (out of a many) way of expressing the information need. This does not mean that the notions of structure and information need are independent. The structured part of the query may capture a part of the information almost literally. As an example of a situation where a structured expression can be very helpful, consider a user who wants to look at vitae of machine learning students. The natural language expression of this information need may look like

```
vitae machine learning student.
```

In terms of the markup of the INEX collection, which has a special tag for vitae, and the NEXI query language, this query may be better expressed as

```
//vt[about(., machine learning student)].
```

Note, however, that even if we have multiple expressions of an information need, the need itself stays the same. Structural constraints do not alter the original information need: they merely express the need differently, more formally. And it is important to understand how useful or helpful they are, and to set up the appropriate experiments that will tell us exactly that.

[4]As an aside, this procedure should have a positive effect on the pool quality, as we can pool together retrieval results that are derived from essentially different representations of the same information need.

7. CONCLUSIONS

Document-centric XML is a mixture of text and structure. With the increased availability of document-centric XML content, we require query facilities in which both structural constraints and constraints on the free text of the documents can be expressed. This has generated considerable interest in the IR community, and has lead to the launch of evaluation efforts tailored for XML documents. One of the driving and long-standing research questions is: How does the increased expressiveness of query languages tailor-made for querying XML documents help users to better, and more effectively, express their information needs? And closely related to this: How should we evaluate systems that enable users to express their information needs using both content and structural constraints?

We addressed these research questions from two angles: what requirements can *in principle* be expressed in query languages for document-centric XML documents? And, how do users *actually* use such languages? For the former, we gave mathematical characterizations of two query languages, in terms of suitable variations on the notion of bisimulation. To address the latter, we provided a detailed examination of the topics formulated in the NEXI query language as part of the 2004 edition of the INEX XML retrieval initiative. Our main findings are as follows. First, while structure is used in varying degrees of complexity, over half of the queries can be expressed in the very restrictive *ignorant user* language. Second, structure is used as a search hint, and not a search requirement, when judged against the underlying information need. Third, the use of structure in queries functions as a precision device. Fourth, the underlying retrieval task of content-and-structure querying is no different from the ordinary natural language query retrieval task. From those findings we derive a number of recommendations for the evaluation of systems that cater for content-and-structure queries: First, if we expect content-and-structure queries to function as a precision device, we should also look at measures that reflect this. Second, if the underlying retrieval task is the same for content-only and content-and-structure topics, we could use a single topic set with title fields for both a natural language query as well as a structured query. Third, if we introduce an additional title field for a content-and-structure query, it should be optional, so that users will formulate a structured query only in case the underlying information need naturally gives rise to it.

8. ACKNOWLEDGMENTS

Thank you to all the participants of the Topic Format Working Groups at INEX 2002–2004. This research was supported, in part, by the Netherlands Organization for Scientific Research (NWO) under project numbers 017.001.-190, 220-80-001, 264-70-050, 354-20-005, 612-13-001, 612.-000.106, 612.000.207, 612.066.302, and 612.069.006.

REFERENCES

[1] S. Amer-Yahia, L. V. S. Lakshmanan, and S. Pandit. FleXPath: flexible structure and full-text querying for XML. In *SIGMOD'04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 83–94. ACM Press, 2004.

[2] M. Benedikt, W. Fan, and G. Kuper. Structural properties of XPath fragments. In *Proc. ICDT*, 2003.

[3] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic.* Cambridge University Press, 2001.

[4] D. Carmel, Y. S. Maarek, M. Mandelbrod, Y. Mass, and A. Soffer. Searching XML documents via XML fragments. In C. Clarke, G. Cormack, J. Callan, D. Hawking, and A. Smeaton, editors, *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 151–158. ACM Press, New York NY, USA, 2003.

[5] D. Carmel, Y. S. Maarek, Y. Mass, N. Efraty, and G. M. Landau. An extension of the vector space model for querying XML documents via XML fragments. In R. Baeza-Yates, N. Fuhr, and Y. S. Maarek, editors, *Proceedings SIGIR 2002 Workshop on XML and Information Retrieval*, pages 14–25, 2002.

[6] CLEF. Cross-Language Evaluation Forum, 2004. `http://www.clef-campaign.org`.

[7] W. Croft, H. Turtle, and D. Lewis. The use of phrases and structured queries in information retrieval. In *Proceedings of the 14th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 32–45. ACM Press, 1991. ISBN 0-89791-448-1.

[8] J. Fagan. Experiments in automatic phrase indexing for document retrieval: A comparison of syntactic and non-syntactic methods. Technical report, Cornell University, 1987.

[9] N. Fuhr and K. Großjohann. XIRQL: A query language for information retrieval in XML documents. In D. H. Kraft, W. B. Croft, D. J. Harper, and J. Zobel, editors, *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 172–180. ACM Press, New York NY, USA, 2001.

[10] N. Fuhr and K. Großjohann. XIRQL: An XML query language based on information retrieval concepts. *ACM Transactions on Information Systems*, 22:313–356, 2004.

[11] D. Harman. Overview of the first Text REtrieval Conference (TREC-1). In *Proc. TREC-1*, 1993.

[12] D. Hawking and P. Thistlewaite. Relevance weighting using distance between term occurrences. Technical Report TR-CS-96-08, Department of Computer Science, Australian National University, 1996.

[13] D. Hull, G. Grefenstette, B. Schultze, E. Gaussier, H. Schutze, and J. O. Pedersen. Xerox TREC-5 Site Report: Routing, Filtering, NLP, and Spanish Tracks. In *Proceedings TREC-5*, pages 167–180, 1997.

[14] INEX. INitiative for the Evaluation of XML Retrieval, 2004. `http://inex.is.informatik.uni-duisburg.de:2004/`.

[15] J. Kamps, M. de Rijke, and B. Sigurbjörnsson. Length normalization in XML retrieval. In M. Sanderson, K. Järvelin, J. Allan, and P. Bruza, editors, *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 80–87. ACM Press, New York NY, USA, 2004.

[16] G. Kazai, M. Lalmas, and B. Piwowarski. INEX 2004 relevance assessment guide. In N. Fuhr, M. Lalmas, S. Malik, and Z. Szlávik, editors, *INEX 2004 Workshop Pre-Proceedings*, pages 241–248, 2004.

[17] E. Keen. Term position ranking: some new test results. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 66–76. ACM Press, 1992. ISBN 0-89791-523-2.

[18] N. Kurtonina and M. de Rijke. Expressiveness of concept expressions in first-order description logics. *Artificial Intelligence*, 107(2):303–333, 1999.

[19] M. Lalmas and G. Kazai. INEX 2004 retrieval task and result submission specification. In N. Fuhr, M. Lalmas, S. Malik, and Z. Szlávik, editors, *INEX 2004 Workshop Pre-Proceedings*, pages 237–240, 2004.

[20] W. May. Information extraction and integration with FLORID: The MONDIAL case study. Technical report, Universität Freiburg, Institut für Informatik, 1999.

[21] G. Mishne and M. de Rijke. Boosting web retrieval through query operations. In *Proceedings ECIR 2005*, 2005.

[22] M. Mitra, C. Buckley, A. Singhal, and C. Cardie. An analysis of statistical and syntactic phrases. In *Proceedings of RIAO-97*, 1997.

[23] G. Navarro and R. Baeza-Yates. A language for queries on structure and contents of textual databases. In E. A. Fox, P. Ingwersen, and R. Fidel, editors, *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 93–101. ACM Press, New York NY, USA, 1995.

[24] R. A. O'Keefe and A. Trotman. The simplest query language that could possibly work. In *Proceedings of the 2nd INEX Workshop*, 2004.

[25] Y. Rasolofo and J. Savoy. Term proximity scoring for keyword-based retrieval systems. In *Proceedings 25th European Conference on IR Research (ECIR 2003)*, pages 207–218, 2003.

[26] B. Sigurbjörnsson, J. Kamps, and M. de Rijke. Processing content-oriented XPath queries. In *Proceedings of the Thirteenth ACM Conference on Information and Knowledge Management (CIKM 2004)*, pages 371–380, 2004.

[27] B. Sigurbjörnsson, J. Kamps, and M. de Rijke. The University of Amsterdam at INEX 2004. In N. Fuhr, M. Lalmas, S. Malik, and Z. Szlávik, editors, *INEX 2004 Workshop Proceedings*, pages 104–109, 2004.

[28] B. Sigurbjörnsson, B. Larsen, M. Lalmas, and S. Maalik. INEX04 guidelines for topic development. In N. Fuhr, M. Lalmas, S. Malik, and Z. Szlávik, editors, *INEX 2004 Workshop Pre-Proceedings*, pages 212–218, 2004.

[29] B. Sigurbjörnsson and A. Trotman. Queries, INEX 2003 working group report. In *Proceedings of the 2nd INEX Workshop*, 2004.

[30] A. Trotman and B. Sigurbjörnsson. Narrowed Extended XPath I (NEXI). In N. Fuhr, M. Lalmas, S. Malik, and Z. Szlávik, editors, *INEX 2004 Workshop Pre-Proceedings*, pages 219–236, 2004.

[31] V. Vianu. A Web odyssey: from Codd to XML. In *Proc. PODS*, pages 1–15. ACM Press, 2001. ISBN 1-58113-361-8.

[32] S. Wasserman and K. Faust. *Social Network Analysis.* Cambridge University Press, 1994.