Articulating Information Needs in XML Query Languages

JAAP KAMPS, MAARTEN MARX, MAARTEN DE RIJKE
and
BÖRKUR SIGURBJÖRNSSON
University of Amsterdam

Document-centric XML is a mixture of text and structure. With the increased availability of document-centric XML documents comes a need for query facilities in which both structural constraints and constraints on the content of the documents can be expressed. How does the expressiveness of languages for querying XML documents help users to express their information needs? We address this question from both an experimental and a theoretical point of view. Our experimental analysis compares a structure-ignorant with a structure-aware retrieval approach using the test suite of the INEX XML retrieval evaluation initiative. Theoretically, we create two mathematical models of users' knowledge of a set of documents and define query languages which exactly fit these models. One of these languages corresponds to an XML version of fielded search, the other to the INEX query language.

Our main experimental findings are: First, while structure is used in varying degrees of complexity, two thirds of the queries can be expressed in a fielded-search like format which does not use the hierarchical structure of the documents. Second, three quarters of the queries use constraints on the context of the elements to be returned; these contextual constraints cannot be captured by ordinary keyword queries. Third, structure is used as a search hint, and not as a strict requirement, when judged against the underlying information need. Fourth, the use of structure in queries functions as a precision enhancing device.

Categories and Subject Descriptors: H.2 [**Database Management**]: H.2.3 Languages—*Query Languages*; H.3 [**Information Storage and Retrieval**]: H.3.1 Content Analysis and Indexing; H.3.3 Information Search and Retrieval; H.3.4 Systems and Software; H.3.7 Digital Libraries

General Terms: Measurement, Performance, Experimentation

Additional Key Words and Phrases: Full-text XML querying, XPath, XML Retrieval

---

1. INTRODUCTION

There is an ever growing availability of semi-structured information, on the Web and in digital libraries. Increasingly, users, both expert and non-expert, have access to text documents, equipped with additional semantic information through XML-

---

Authors' addresses: J. Kamps, Archives and Information Studies, Faculty of Humanities, University of Amsterdam, Turfdraagsterpad 9, 1012 XT Amsterdam, the Netherlands. M. Marx, M. de Rijke, S. Sigurbjörnsson, ISLA, Faculty of Science, University of Amsterdam, Kruislaan 403, 1089 SJ Amsterdam, the Netherlands. Email: {`kamps,marx,mdr,borkur`}`@science.uva.nl`.

markup. Based on their content, XML documents may be 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 [May 1999]. Document-centric documents are loosely structured documents (often text) marked-up with XML, with electronic journals in XML providing important examples. 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 hierarchical nature of the XML markup calls for methods from the database field for querying structure, and the textual nature of the documents calls for approaches from the field of IR (cf. [Vianu 2001, 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 computer (XPath processor), based on the pattern appearing in the XML document, using an *exact match* approach. It is clearly defined which 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.

At INEX, the INitiative for the Evaluation of XML Retrieval [INEX 2006], the focus is on a *combined* approach to XML retrieval, featuring aspects of exact match and best match retrieval. 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 the user to (1) express information needs with a mixture of content and structure requirements; and (2) use best match querying of document-centric XML. Although the `about` function has the same syntax as `contains`, its semantics is not strictly defined but left to relevance judgments by human assessors. But how to interpret the structural part of these hybrid content-and-structure (CAS) queries? At INEX 2002 and 2003, structural constraints on the target element—the tag name of the XML element returned to the user—were strictly enforced. A more direct IR approach, adopted at INEX 2004 and 2005, is to view the whole query as an inexact statement of the underlying information need. In this case, there is no distinction with standard keyword queries in terms of the ground truth used to evaluate retrieval effectiveness. A user may decide either to articulate her information need using a keyword query, or to use a hybrid CAS query. Which one will be more effective for retrieving XML elements satisfying her information need?

This brings us to the main research problem of this paper: how does the expressiveness of languages for querying XML documents help users to articulate their information needs? Intuitively, CAS queries are more expressive and this should lead to more effective retrieval. In practice, however, experimental results are mixed at best. To a large extent, this paper is motivated by our own frustration with earlier experiments that gave diverging evidence for methods that take the structural constraints of a CAS query seriously, with some successes for shallow heuristics but without leading to a thorough understanding of the task. Hence, in this paper, we opt for a broader, more reflective approach, in which we (i) analyze, in great detail, a set of actual CAS queries and the corresponding relevance judgments; (ii) conduct a set of comparative retrieval experiments; and (iii) relate the expressiveness of the CAS queries to a theoretical model based on the user's knowledge of the document structure. Specifically, we address the following questions:

(1) How do users exploit the additional expressive power of structural constraints in their queries? What queries do users formulate? What is the meaning of these queries?
(2) What is the effect on retrieval performance of adding structural constraints to queries?
(3) What is the appropriate query language for XML retrieval?

We will answer the first two questions by an analysis of the INEX data. For the third, such an analysis has been carried out in [O'Keefe and Trotman 2004], resulting in a proposal for a query language based on their findings. We give a mathematical model of users' knowledge of an XML collection and link this to the appropriate expressive power of query languages. Our main results are:

(1) Structural constraints are mainly used as search hints, not as strict requirements.
    —The hierarchical nature of the documents is used in one third of the examined queries.
    —Three quarters of the queries put constraints on the context of the element to retrieve.
(2) Adding structural constraints has a positive effect on early precision and a negative effect on overall recall.
(3) Towards an answer to the third question we provide
    —A typology of different uses of content and structure queries.
    —Intuitive mathematical models of users' knowledge of a set of XML documents and the formulation of query languages which exactly fit this knowledge.

The rest of this paper is organized as follows. Section 2 describes the INEX dataset, the topic format and the query language. In Section 3 we discuss the retrieval task at INEX and analyze the queries used. In Section 4 we report on experiments comparing the retrieval effectiveness of structured queries versus ordinary queries. Section 5 contains a typology of different content and structure queries. In Section 6 we describe content-oriented flavors of XPath and provides semantic characterizations of their expressive power. We conclude in Section 7.

2. THE DATA: INEX 2003, 2004, AND 2005

This section describes the data used: the INEX document collection, the topic format and the query language for the content and structure queries.

2.1 The INEX XML Document Collection

The queries we study are run against the XML collection that is made available by the INitiative for the Evaluation of XML Retrieval [INEX 2006]. It contains over 12,000 articles from 21 IEEE Computer Society journals, marked up with XML tags. Most of the markup refers to layout instructions (like in a LaTeX document). Several additional "semantic tags" are used as well (like ⟨au⟩ to indicate names of authors). The DTD of the INEX XML document collection is rather complex. There are 192 different content types, including 11 different tag names for representing paragraphs; about 170 tag names are actually used in the collection, including articles ⟨article⟩, sections ⟨sec⟩, author names ⟨au⟩, affiliations ⟨aff⟩, etc. On average an article contains 1,532 elements and the average element depth is 6.9.

The INEX setup is such that we should think of the INEX document collection as a forest of articles. These are XML documents whose roots have the tag name `article`. Because the actual storage of the documents may be different, most queries start with the prefix `//article`.[1] This is only an artifact of the representation and we will treat the tag name `article` as referring to the root of a document.

2.2 The INEX Topic Format

At INEX, two types of topic are used: Content-Only (CO) topics and Content-And-Structure (CAS) topics. All topics contain the same three fields as traditional IR topics [Harman 1993]: title, description and narrative. The title is the actual query submitted to the retrieval system. The description and narrative describe the information need in natural language. The described information need is used to judge the relevancy of the retrieved answers to the queries. The difference between the CO and CAS topics lies in the topic title. In the case of CO topics, the title describes the information need as a small list of keywords. In the case of CAS topics, the title describes the information need using XPath 1.0 extended with the `about` function discussed below. At INEX 2003, full XPath was allowed, and at INEX 2004 and 2005 a restricted version of XPath was used [Sigurbjörnsson and Trotman 2003; Trotman and Sigurbjörnsson 2005]. In this paper we analyze the title part of the CAS topics, which we simply call *queries* from now on.

2.3 The NEXI Query Language

The specific instructions for topic development at INEX 2004 [Sigurbjörnsson et al. 2004c] stated that CAS queries

—should use only descendant axis (i.e., `//`),

—should use only boolean `and` and `or` in filter expressions,

—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) [Trotman and Sigurbjörnsson 2005].

The `about` function is the IR counterpart of the familiar XPath `contains` function. Recall that if `P` is an XPath expression denoting a set of nodes, the query `contains(P,'phrase')` returns true when evaluated at a node n if there exists a node m reachable from n via the path `P` and the text value of m contains `phrase`.

---

[1] For example, only three CAS queries out of the 34 used at INEX 2004 do not start with this prefix; however, these queries are prefixed with either ⟨sec⟩ or ⟨abs⟩ tags that only occur in the context of an ⟨article⟩ tag anyway.

Because of its strict, boolean character, `contains` is not suitable for expressing the kind of information needs we meet at INEX. The semantics of `about(P,'phrase')` is intentionally not specified formally in the INEX guidelines. As an example, consider the following query:

> Find sections explaining the vector space model.

In the NEXI query language this is naturally stated as

> `//sec[about(.,'vector space model')].`

This query uses `//sec` to ask for sections. The query restricts those sections by the `about(.,'vector space model')` function in the filter expression. The dot indicates that the query should return only those sections which are about `'vector space model'`. The latter part is to be interpreted as saying that the section is *relevant* to the information need expressed by the phrase `'vector space model'`. In the spirit of IR, the ultimate decision of relevancy 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 the 'SMART system' to be relevant to the information need expressed above. The next information need extends the example (in fact, this is CAS topic 151 from INEX 2004):

> In articles discussing web searching find sections explaining the vector space model.
>
> `//article[about(.,'web search engine')]//sec[about(.,'vector space model')].`

The resulting query now has two content-based restrictions. A first restriction is on the *requested elements* (i.e., the XML elements returned to the user), which targets sections explaining the vector space model just as the earlier query above. A second restriction is on the *context* surrounding the requested elements (i.e., on particular elements outside the requested element): the article should be about web search engines. The two restrictions are linked by a structural constraint, which here simply states that the section is indeed part of the article.

In this paper, we will not assume that the structural parts of a NEXI query—neither the tag names, nor the way they are nested—are strictly enforced. Rather, we are interested in how the NEXI queries written by users relate to the perceived relevance of the retrieved elements.

2.4 INEX CAS Queries

There has been a task using structured queries at every edition of INEX so far. However, we do not consider the queries from the INEX 2002 CAS task, since a very different and ambiguous query language was used [Fuhr et al. 2003].

As mentioned previously, in the INEX 2003 CAS task full XPath queries were allowed, with the `about` function replacing the `contains` predicate. There is a straightforward mapping from the INEX 2003 CAS queries into the NEXI format [Trotman and Sigurbjörnsson 2005]. The main change is to replace child steps ("`/`"), which are no longer allowed in NEXI, with corresponding descendant ("`//`") steps. We use the resulting set of 30 CAS queries (version 1.4.7) with query numbers 61–90 [Fuhr et al. 2004].

Element	2003		2004		Total	
sec *(section)*	10	(33.3%)	16	(47.1%)	26	(40.6%)
article	12	(40.0%)	5	(14.7%)	17	(26.6%)
p *(paragraph)*	1	(3.3%)	4	(11.8%)	5	(7.8%)
* *(wildcard)*	2	(6.7%)	2	(5.9%)	4	(6.3%)
abs *(abstract)*	2	(6.7%)	2	(5.9%)	4	(6.3%)
bb *(bibliography entry)*	1	(3.3%)	1	(2.9%)	2	(3.1%)
vt *(vita)*	1	(3.3%)	1	(2.9%)	2	(3.1%)
bdy *(body)*	–	–	1	(2.9%)	1	(1.6%)
bib *(bibliography)*	–	–	1	(2.9%)	1	(1.6%)
fig *(figure)*	–	–	1	(2.9%)	1	(1.6%)
fm *(front matter)*	1	(3.3%)	–	–	1	(1.6%)

Table I. Frequency of requested elements in the 30 CAS queries of INEX 2003 and 34 CAS queries of INEX 2004.

The NEXI query language was officially introduced at INEX 2004. We use the set of 34 CAS queries (version 2004-7) with query numbers 127–147, and 149–161; for details, see [Fuhr et al. 2005].

From INEX 2005, we use both the set of CO+S topics having an optional CAS title field (hence the name CO plus structure) and the set of CAS topics [Fuhr et al. 2006]. There are 40 CO+S topics (version 2005-003), numbered 202–241. We focus on the 28 topics with a CAS title field, numbered 202–205, 207–208, 210–212, 216, 219–220, 222–226, 228–234, 236, and 238–240. There are 17 CAS topics (version 2005-003), numbered 244, 247, 250, 253, 256, 257, 258, 260, 261, 264, 265, 269, 270, 275, 280, 284, and 288.

3. THE MEANING OF CONTENT-AND-STRUCTURE

In this section we start to answer our first research question from the introduction. We examine how users express their information needs in the NEXI query language. Given that information needs are notoriously hard to investigate, and that we do not have access to a real user-base of an operational system, we look for evidence that will, at least, approximate users and information needs. At INEX, all participants are involved in topic creation and assessment, giving us access to NEXI queries formulated by a large group of people, together with their relevance judgments. We proceed in two steps. First, we discuss the CAS queries for 2003 and 2004. Because of the different assessment procedure and task set-up, we discuss the INEX 2005 data separately, although the findings for 2005 closely mirror the analysis of the INEX 2003 and 2004 data.

We find that the elements requested in queries should be viewed as retrieval hints, not as strict requirements on the results: over half of the relevant elements has another tag name than the one specified in the query.

3.1 What is Asked For and What is Returned

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 as answers. Table I lists which kind of elements were requested in the 64 CAS queries studied across the two years. We see that sections (sec) and articles (article) are the most popular elements asked for. Interestingly, article targets were the most popular requested element in 2003, but lost their appeal in 2004—much in the spirit of XML element retrieval.

We view a CAS query as a means to locate relevant information rather than an end in itself. At INEX, the requested element is not strictly enforced, but merely regarded as a retrieval hint [Kazai et al. 2004, p. 237]:

> 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 behind the VCAS sub-task is to allow the evaluation of XML retrieval systems that aim to implement approaches, 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.

Hence, the CAS query is treated just like ordinary CO queries, as an imprecise statement of an information need. The narrative describing the underlying information need is authoritative for the relevance assessments. In the example query above,

> `//article[about(.,'web search engine')]//sec[about(.,'vector space model')]`,

the narrative field reads:

> I'm writing a thesis about matching methods used in web search engines and web agents. For this purpose I'm looking for information about the vector space model. Relevant sections discuss the vector space model, preferably at length. The sections must be in articles that are about some aspect of web search engines or agents.

Looking at the relevance judgments based on this narrative will reveal, for example, how we should interpret the `//sec` in the CAS query. Does this mean literally `<sec>` and nothing else? If this is essential to the information need of the topic creator, the relevance judgments based on the narrative of the topic of request will reflect this. But perhaps it merely means something section-ish, like a section, subsection, or paragraph? Or is it a hint that the sought information is likely to occur in a section? Or something else?[2]

Elements Judged Relevant. We use version 2.5 of the assessments for INEX 2003, in the Vague CAS or VCAS version that is not post-filtered for requested elements. There are judgments for all 30 queries numbered 61–90. In INEX, assessors make relevance judgments on a graded, two-dimensional scale. To be able to look at the distribution of relevant elements we use a quantization that results in boolean

---

[2] For readers interested in the particular example topic: As it turns out, of the elements that are judged relevant by the topic author, two tags stand out: 36% is a section (or subsection) but 41% a paragraph.

Element	2003		2004		Total	
p+	370	(23.45%)	854	(31.41%)	1224	(28.48%)
sec+	580	(36.78%)	262	(9.64%)	842	(19.60%)
vt	41	(2.60%)	747	(27.47%)	788	(18.34%)
article	188	(11.92%)	73	(2.68%)	261	(6.08%)
bb	94	(5.96%)	104	(3.82%)	198	(4.61%)
bdy	145	(9.19%)	36	(1.32%)	181	(4.21%)
au	0	(0.00%)	110	(4.05%)	110	(2.56%)
fnm	0	(0.00%)	104	(3.82%)	104	(2.42%)
st	14	(0.89%)	90	(3.31%)	104	(2.42%)
fig	8	(0.51%)	53	(1.95%)	61	(1.42%)
abs	27	(1.71%)	13	(0.48%)	40	(0.93%)
it	2	(0.13%)	37	(1.36%)	39	(0.91%)
ref	0	(0.00%)	34	(1.25%)	34	(0.79%)
scp	0	(0.00%)	32	(1.18%)	32	(0.74%)
atl	5	(0.32%)	23	(0.85%)	28	(0.65%)
app	17	(1.08%)	9	(0.33%)	26	(0.61%)
fm	14	(0.89%)	11	(0.40%)	25	(0.58%)
li	14	(0.89%)	9	(0.33%)	23	(0.54%)
bm	11	(0.70%)	9	(0.33%)	20	(0.47%)
list	12	(0.76%)	2	(0.07%)	14	(0.33%)
item	11	(0.70%)	1	(0.04%)	12	(0.28%)
b	1	(0.06%)	10	(0.37%)	11	(0.26%)

Table II. Frequency of elements judged relevant for all assessed CAS queries at INEX 2003 and 2004. We only show tag names that occur at least 10 times over both years.

relevance judgments. Specifically, we focus on elements rated as highly exhaustive and highly specific—also called strict or (3,3) assessments. For the two queries numbered 61 and 73, there are no elements judged as highly exhaustive and highly specific. We also use version 3.0 of the assessments for INEX 2004, containing judgments for the 26 queries numbered 127–137, 139–145, 149–153, and 155–157. For the four queries numbered 133, 140, 143, and 144, there are no elements judged as highly exhaustive and highly specific.

Table II lists the frequencies of element-types judged relevant for the remaining CAS queries. We collapse equivalent tags for sections and paragraphs, as defined in [Sigurbjörnsson et al. 2004c], and we use sec+ and p+ to denote the equivalence classes of sections and paragraphs, respectively. We see that the most popular elements are paragraphs (p+) and sections (sec+).

Requested versus Relevant Elements. Next, we investigate how often the element that is judged relevant actually has the tag name specified by the query. Consider Table III; the rows show the tag names of requested elements as stated in the query and the columns show the tag names of elements judged relevant. E.g., if we look at the assessments of all topics requesting sections (sec), we see that 38.6% of the relevant elements are sections (sec+), 25.9% are paragraphs (p+), and 5.2% are articles. If we look at the diagonals of the tables, we see that the assessors frequently felt that their information needs were also satisfied by elements not respecting the target constraints.[3] Still, in most cases the elements satisfying the

---

[3] The surprising numbers for the article topics are due to strange assessments of one of the article topics in 2004, which is most likely due to a misinterpretation of the assessment guidelines.

2003	article	sec+	p+	abs	vt	*other*
article (10)	**24.4%**	26.0%	19.8%	0.2%	0.8%	10.2%
sec (10)	7.3%	**50.1%**	27.2%	1.8%	–	4.9%
p (1)	6.5%	18.5%	**50.0%**	5.4%	–	9.8%
abs (2)	7.5%	47.3%	22.6%	**8.6%**	–	6.5%
vt (1)	–	–	–	–	**97.4%**	2.6%

2004	article	sec+	p+	abs	vt	*other*
article (2)	**10.8%**	1.3%	1.6%	–	–	82.3%
sec (10)	3.3%	**27.7%**	24.7%	0.9%	0.4%	43.0%
p (4)	4.0%	26.0%	**48.0%**	–	–	22.0%
abs (2)	16.0%	–	24.0%	**24.0%**	–	36.0%
vt (1)	–	–	44.0%	–	**52.0%**	4.0%

Total	article	sec+	p+	abs	vt	*other*
article (12)	**18.5%**	15.2%	11.8%	0.1%	0.5%	42.2%
sec (20)	5.2%	**38.6%**	25.9%	1.4%	0.2%	16.6%
p (5)	5.6%	21.1%	**49.3%**	3.5%	–	9.9%
abs (4)	9.3%	37.3%	22.8%	**11.9%**	–	8.5%
vt (2)	–	–	42.9%	–	**53.1%**	4.0%

Table III. Frequency of relevant elements (columns) for queries asking for elements with tag name (rows). The number of aggregated queries is indicated between brackets.

Element	CO+S		CAS		Total	
sec *(section)*	14	(50.0%)	12	(70.6%)	26	(57.8%)
article	6	(21.4%)	2	(11.8%)	8	(17.8%)
* *(wildcard)*	7	(25.0%)	1	(5.9%)	8	(17.8%)
p *(paragraph)*	–	–	2	(11.8%)	2	(4.4%)
bdy *(body)*	1	(3.6%)	–	–	1	(2.2%)

Table IV. Frequency of requested elements in the 28 CO+S queries and the 17 CAS queries of INEX 2005.

target constraints are the largest category. We conclude that the element names as requested in the query can indeed only be considered as a retrieval *hint*, and not as a strict constraint on the output of a query. While not strictly enforced, however, there seems to be a preference for XML elements of the type requested. E.g., if users ask for sections, they are more likely to judge sections as relevant than any other kind of tag.

3.2 CO+S and CAS at INEX 2005

We now repeat the analysis that we just carried, but now for the INEX 2005 CAS queries instead of the INEX 2003 and 2004 queries. As mentioned previously, at INEX 2005 there was both an optional CAS query for the CO+S task, as well as a separate CAS task [Fuhr et al. 2006]. For the CAS task, we use the judgments based on the narrative field (which corresponds to the VVCAS subtask where all structural constraints are interpreted as vague). All CAS queries at INEX 2005 are in the NEXI query language; Table IV shows the requested elements. The resulting distribution is very similar to what we observed in earlier years.

There where some radical changes in the assessment procedure, resulting in qrels that contain information on the fraction of text highlighted by the assessor as

	CO+S		VVCAS		Total	
p+	404	(43.6%)	558	(57.3%)	962	(50.6%)
sec+	176	(19.0%)	278	(28.5%)	454	(23.9%)
it	72	(7.8%)	1	(0.1%)	73	(3.8%)
item	54	(5.8%)	12	(1.2%)	66	(3.5%)
tt	31	(3.3%)	0	(0.0%)	31	(1.6%)
fig	20	(2.2%)	6	(0.6%)	26	(1.4%)
ariel	23	(2.5%)	0	(0.0%)	23	(1.2%)
li	5	(0.5%)	18	(1.8%)	23	(1.2%)
abs	7	(0.8%)	15	(1.5%)	22	(1.2%)
list	2	(0.2%)	20	(2.1%)	22	(1.2%)
st	13	(1.4%)	7	(0.7%)	20	(1.1%)
ref	18	(1.9%)	0	(0.0%)	18	(0.9%)
art	13	(1.4%)	3	(0.3%)	16	(0.8%)
fgc	6	(0.6%)	6	(0.6%)	12	(0.6%)
url	12	(1.3%)	0	(0.0%)	12	(0.6%)
b	11	(1.2%)	0	(0.0%)	11	(0.6%)
la	1	(0.1%)	10	(1.0%)	11	(0.6%)
label	11	(1.2%)	0	(0.0%)	11	(0.6%)
lit	10	(1.1%)	1	(0.1%)	11	(0.6%)

Table V. Frequency of elements judged relevant for the all assessed CO+S and CAS queries at INEX 2005. We show only tag names that occur at least 10 times in total.

relevant, as well as exhaustiveness judgments on a slightly modified scale. We follow the strict quantization, and treat as relevant those elements that are completely highlighted, and are highly exhaustive [Kazai and Lalmas 2006]. We use version 7 of the INEX 2005 ad hoc assessments for CO+S and CAS. There are judgments for 19 CO+S topics 202, 203, 205, 207, 208, 210, 212, 216, 219, 222, 223, 228–230, 232–234, 236, and 239; for the two topics 205 and 228 there are no strictly relevant elements. There are also judgments for 10 VVCAS topics, numbered 253, 256, 257, 260, 261, 264, 265, 270, 275, and 284; only for topic 257 there is no strictly relevant element.

Table V shows the distribution of elements judged relevant for any of the 17 CO+S queries and 9 CAS queries. The distribution is somewhat different from earlier years, with almost three-quarters of the relevant elements being paragraphs or sections. This concentration may be a result from deriving the specificity judgments from the highlighted text. For example, it is not very intuitive to highlight an article element in its entirety.

In Table VI, we contrast the requested elements with the elements judged relevant. Here we see the effect of focusing on sections and especially paragraphs. For paragraphs, the requested and relevant element types are a close match. For sections as request element, sections are the second most frequent element type already after paragraphs. For article requests, an article element is hardly ever judged relevant. Again, the rationale for this is likely related to the unnaturalness of highlighting a large chunk of text—such as a complete article—in its entirety.

4. THE EFFECT OF STRUCTURE ON RETRIEVAL EFFECTIVENESS

In this section we answer the second question from the introduction: What is the effect on retrieval performance of adding structural constraints to queries? The

CO+S	article	sec+	p+	abs	vt	*other*
article (6)	–	18.4%	70.4%	1.0%	–	10.6%
sec (9)	–	**32.7%**	38.8%	2.8%	–	25.7%
* (2)	–	45.1%	23.3%	–	–	31.6%

VVCAS	article	sec+	p+	abs	vt	*other*
article (2)	**1.3%**	35.3%	52.3%	1.8%	–	9.3%
sec (5)	–	**13.9%**	74.5%	0.1%	–	11.5%
p (1)	–	10.5%	**71.1%**	2.6%	–	15.8%
* (1)	–	54.5%	45.5%	–	–	–

Table VI. Frequency of relevant elements (columns) for queries asking for elements with tag name (rows). The number of aggregated queries is indicated between brackets.

experimental evidence given in this section indicates that structural constraints function as a precision enhancing device: useful for promoting the precision of initially retrieved documents, possibly reducing fall-out but also reducing recall.

When querying a collection of structured documents, users can express their information needs in a more precise way using a hybrid content-and-structure query—one that combines natural language with structural elements. The hypothesis is that such a more precise statement of the information need will lead to improved retrieval effectiveness compared to traditional keyword queries. We test this hypothesis using the following experimental analysis. From the CAS topics we create three sets of queries: (1) the original NEXI queries, (2) a NEXI query where the only structural constraint is on the target, and (3) an ordinary keyword search queries consisting of all keywords in a NEXI query. For example, let's look at the *structured query*

```
//article[about(.//abs, sorting)]//sec[about(., heap sort)].
```

We turn it into a *target-only query* by merging all the `about` constraints in a single about function:

```
//article//sec[about(., sorting heap sort)].
```

We remove all structural constraints and turn it into a *content-only query* by replacing the target constraint with a `*`:

```
//*[about(., sorting heap sort)].
```

We create three runs using the exact same set-up, one for each set of queries. The only difference between the runs is the used query, making the results directly comparable on equal grounds. We compare the results using several standard IR measures.[4]

4.1 Experimental Setup

We base our experimental evidence on the INEX 2003 and 2004 CAS content and structure task, in combination with the vague CAS qrels [Fuhr et al. 2005]. To allow for a direct comparison with the earlier analysis of queries and judgments, we treat

---

[4] A similar experiment based on official runs submitted to INEX 2004 is reported in [Kamps et al. 2005].

only highly specific and highly exhaustive elements as relevant (i.e., the so-called strict assessments). The strict quantization caters for systems that attempt to retrieve very high quality results, both in terms of exhaustivity and specificity. Over the two years (2003 and 2004), there are 50 topics with at least one relevant element according to the strict assessments. The mean number of strict assessments per topic is 85.9 and the median is 28.5. We evaluate our system using two evaluation programs: `trec_eval` and EvalJ. We do not penalize overlap to allow for direct comparison with the earlier analysis of the whole sets of queries and judgments (where pruning the set of relevant elements would be unnatural). This caters for systems that estimate the relevance of arbitrary elements, as input for a particular interface, or for further processing. As a case in point, in related work we have seen evidence that overlap can be useful if handled appropriately by the result presentation interface [Kamps et al. 2006].

We create three runs using the queries discussed above: one based on the *content-only query*, one on the *target-only* query, and one based on the *structured query*. The runs differ in the amount of structure, ranging from no structured constraints used to all structured constraints used.

4.2 Processing Content-Oriented XPath

We process the queries using the three step strategy proposed for processing content-oriented XPath queries in [Sigurbjörnsson et al. 2004b]:

(1) *Decomposition.* First, the NEXI query is decomposed into a sequence of pairs of the form (`location path`, `content description`), one for each `about` function. In the case of the *heap sorting* example above, this yields:

```
(//article//abs, 'sorting')   (//article//sec, 'heap sort')
```

(2) *Retrieval.* For each (`location path`, `content description`) pair, we score XML elements satisfying the *location path* using a language model retrieval approach. For the *heap sorting example*, this results in two different result sets for each of the `about` function in the query.

(3) *Mixture.* Now we put things together. For each element satisfying the target constraints, we consider other elements satisfying the tree pattern of the query. In case of the *heap sorting* example, this would only consider the corresponding abstract (⟨abs⟩) elements for a particular section element. We take the maximal scoring element for each of the `about` functions. The resulting score for the element satisfying the target constraints is simply the sum of the scores of the `about` functions in the query. For the *heap sorting* example, the final score of a section would be the sum of the section's score and the corresponding abstract's score.

We refer to [Sigurbjörnsson et al. 2004b] for more details on the approach. In principle, we use the same approach for all three versions of our queries. But, of course, in the case of *content-only* and *target-only* queries the decomposition and mixture steps are trivial, since for those queries there is only one about function.

	Content-only	Target-only		Structured	
MAP	0.0988	0.0724	(−26.7%)	0.0835	(−15.5%)
MAep	0.0992	0.0724	(−27.0%)	0.0835	(−15.8%)

Table VII. Effectiveness of our runs in terms of mean average precision (MAP using `trec_eval`) and mean average effort-precision (MAep using EvalJ).

4.3 Retrieval Model

For the *Retrieve* step, we use a multinomial language model [Hiemstra 2001]. We assume query terms to be independent, and rank elements according to:

$$P(e|q) \propto P(e) \cdot \prod_{i=1}^{k} P(t_i|e), \tag{1}$$

where q is a query made out of the terms $t_1, \ldots, t_k$. We estimate the element language model by taking a linear interpolation of two language models:

$$P(t_i|e) = \lambda \cdot P_{mle}(t_i|e) + (1 - \lambda) \cdot P_{mle}(t_i), \tag{2}$$

where $P_{mle}(\cdot|e)$ is a language model for element e; and $P_{mle}(\cdot)$ is a language model of the collection. The parameter λ is an interpolation factor (smoothing parameter). Finally, we assign a prior probability to an element e relative to its length in the following manner:

$$P(e) \propto |e|^{\beta} \tag{3}$$

where $|e|$ is the size of an element e. For a more detailed description of our retrieval approach we refer to [Sigurbjörnsson et al. 2004a]. In all our experiments we use the value 0.15 for the smoothing parameter λ. We use different values for the length prior depending of whether we are ranking target elements or context elements. We set $\beta = 1.5$ when we rank target elements, and set $\beta = 0.0$ when we rank context elements.

4.4 Results

Mean Average Precision. We first consider the results in terms of mean average precision (MAP) and mean average effort-precision (MAep). Table VII shows the respective scores. The content-only run is clearly superior. This is, indeed, a disappointing result because the poorer scoring XPath-oriented runs use a more articulate query. However, the difference is not significant, neither in terms of MAP nor in terms of MAep. To obtain a better understanding we zoom in on the performance at different recall levels. Figure 1 shows the interpolated precision at the eleven standard recall levels. We see an interesting phenomenon. While the content-only run clearly outperforms the XPath-oriented runs on higher recall levels, the XPath-oriented runs outperforms the content-oriented run on lower recall levels.

Early Precision. We zoom in further and look explicitly at the performance on the initially retrieved elements. Table VIII shows the mean precision and cumulative gain at ranks 5, 10, 20, and 30. Here, we see a complete reversal from the picture in Table VII: now, the XPath-oriented runs are superior. For P@5 both XPath-oriented runs are significantly better than the content-only run (t-test, $p < 0.05$).

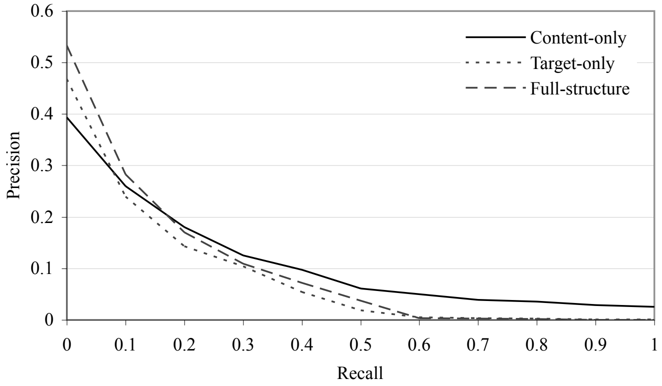


Fig. 1. Interpolated precision at standard recall levels (using `trec_eval`).

Precision	Content-only	Target-only		Full-structure	
P@5	0.2000	0.2840	(+42.0%)	0.3265	(+63.3%)
P@10	0.1820	0.2460	(+35.2%)	0.2531	(+39.1%)
P@20	0.1700	0.1880	(+10.6%)	0.1796	(+5.6%)
P@30	0.1527	0.1653	(+8.3%)	0.1531	(+0.3%)
nxCG@5	0.2220	0.2840	(+27.9%)	0.3265	(+47.1%)
nxCG@10	0.2217	0.2536	(+14.4%)	0.2633	(+18.8%)
nxCG@20	0.2358	0.2122	(−10.0%)	0.2025	(−14.1%)
nxCG@30	0.2391	0.1978	(−17.3%)	0.1842	(−23.0%)

Table VIII. Mean precision (`trec_eval`) and cumulative gain (EvalJ) at rank 5, 10, 20, and 30.

	Content-only	Target-only		Structured	
MRR	0.3491	0.4403	(+26.1%)	0.5085	(+45.7%)

Table IX. Mean reciprocal rank scores (`trec_eval`).

For P@10 the run using full structure significantly outperforms the content-only run (t-test, $p < 0.05$). We zoom in even further and look solely at the first relevant element retrieved. Table IX shows the mean reciprocal rank (MRR) of the first found relevant element. The outcome confirms the early precision results: the XPath-oriented runs are superior to the content-oriented run. In terms of mean reciprocal rank, the run using full structure is significantly better than the content-only run (t-test, $p < 0.05$).

Conclusion. Our results show that structured queries do *not* lead to improved mean average precision scores; in fact, we see a substantial, albeit not significant, drop in mean average precision. However, this can be attributed completely to poor scoring at higher recall levels. If we zoom in on the initially retrieved elements, or on the first found relevant element, the outcome is reversed: structured queries lead to significantly better early precision scores. The experimental evidence indicates that structural constraints function as a precision enhancing device: useful for promoting the precision of initially retrieved documents, possibly reducing fall-out but also reducing recall.

These results are consistent with experiments which we ran as part of INEX 2005, using the INEX 2005 CO+S queries and various EvalJ measures [Sigurbjörnsson et al. 2006]. We have also looked at generalized evaluation measures. The results in terms of early precision depend on the used quantization function. We see the same behavior as shown in Table VIII for the default generalized measure using the quantization "sog2" (which prioritizes specificity over exhaustivity). However, for a quantization as "gen" the content-only run is also superior at early ranks. The results in term of mean average effort-precision, as shown in Table VII, hold for both these generalized quantizations: the content-only query outperforms the structured queries.

5. EXPRESSING INFORMATION NEEDS WITH CONTENT-AND-STRUCTURE

We have now seen that searchers use the additional expressive power of structural constraints that is offered by the NEXI query language as search hints, not as strict requirements (Section 3). Also, the usage of NEXI's structural constraints has a positive effect on early precision and a negative effect on overall recall (Section 4). This brings us back to the first research question from the introduction: what, then, are the typical sort of Content-and-Structure queries that users formulate in the NEXI query language?

In this section we zoom in on the way structure is used in queries. On the one hand, we find that three quarters of the queries have constraints on the context surrounding the requested elements, hence could not have been phrased as an ordinary free text query. On the other hand, structure is not exploited that much: two thirds of the queries do not use the hierarchical structure of the documents. They simply require that certain keywords occur in elements with a certain tag name.

5.1 A Typology of Content-and-Structure Queries

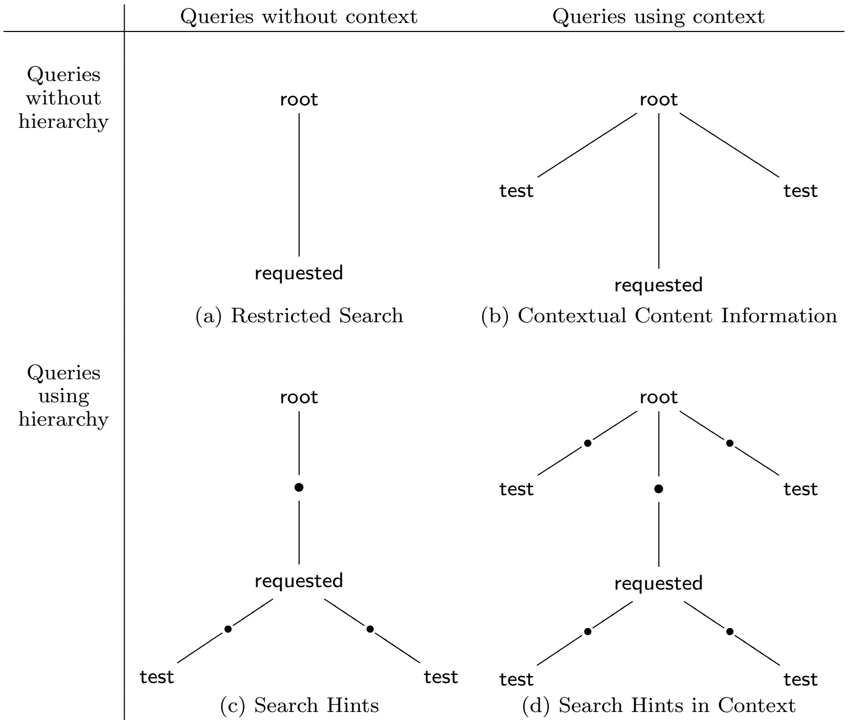
To see how users use structure in their queries, we break down the set of queries by increasing complexity. We use the two following dimensions.

(1) Hierarchy: whether the query uses hierarchical information about the documents.

(2) Context: whether the query puts content constraints on text occurring outside of the element to be returned.

The first dimension, *Hierarchy*, corresponds to the unique tree structure of an XML document. Standard fielded search allows for restricting search to particular fields, think of a library catalogue (OPAC) where fields like "author" or "title" can be used to restrict search. The hierarchical structure of XML allows for contextual selections, think of distinguishing author elements in the bibliography from the author element in the front matter of an article.

The second dimension, *Context*, corresponds to a unique property of structured queries that cannot be captured by ordinary keyword queries. A CAS query can put constraints on particular elements occurring in the context of the elements to be returned. That is, they may make content restrictions on text that is not returned to the user. For example, a user may want to retrieve sections while the query also refers to the article's abstract, which is on a disjoint path in the article's XML tree.



(a) Restricted Search (b) Contextual Content Information
(c) Search Hints (d) Search Hints in Context

Fig. 2. Four categories of queries.

The two dimensions result in four categories which are graphically depicted in Figure 2. The resulting categories are:

(a) Restricted Search. This category has queries in which structure is only used as a constraint on the returned elements. The query is an ordinary content-only query, but the search is restricted to particular XML elements. A typical example of such a query is to restrict the search to sections:

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

In general, such queries have the form `//tag[P]` where `P` is a positive boolean combination of functions `about(.,'xxx')`.

(b) Contextual Content Information. This category is similar to the *Restricted Search* category, but additionally we may put content restrictions on the environment in which the requested element occurs. A typical example looks like:

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

This query asks for sections about `yyy` in documents which contain an abstract (`abs`) about `xxx`. In general, such queries have the form `//tag[P]`, where `P` is a positive boolean combination of functions `about(.,'xxx')` and `about(//tag,'xxx')`. Note that `about(//abs, 'xxx')` expresses that somewhere below the root of the document there is an abstract (`abs`) which is about `xxx`.

(c) Search Hints. This category is again similar to the *Restricted Search* category, but additionally we may put content restrictions on sub-elements of the requested element, and we may use the hierarchical nature of the documents. These

extra restrictions can be viewed as search hints or retrieval cues to the system. A typical example is a query which asks for sections about `xxx` containing a theorem about `yyy`:

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

The general form of such queries is `path[P]`, where `P` is a positive boolean combination of functions `about(.,'xxx')` and `about(.path ,'xxx')`, and `path` is a location path sequence of the form `//tag`$_1$`//...//tag`$_n$.

(d) Search Hints in Context. This category combines the *Search Hints* with the *Contextual Context Information* categories. An example is a query which asks for sections about `xxx` containing a theorem about `yyy`, in documents which contain an abstract (`abs`) about `zzz`:

```
//sec[about(.,'xxx') and about(.//thm,'yyy') and
about(//abs,'zzz')].
```

The general form of queries of this category is `path[P]` where `P` is a positive boolean combination of `about(.,'xxx')`, `about(.path,'xxx')` and `about(path ,'xxx')`, and `path` is a location path sequence of the form `//tag`$_1$`//...//tag`$_n$.

XML Fragments. To help situate the query categories just introduced, we recall the XML fragments proposed in [Carmel et al. 2002; Carmel et al. 2003] as a simple alternative to XPath for content and structure queries. XML Fragments are queries that are structured like the wanted documents. For example, consider a query like CAS topic 131 from INEX 2004:

```
//article[about(.//au,"Jiawei Han") AND about(.//abs,"data mining")];
```

it is translated to the following XML Fragment query

```
<article>
   <au>"Jiawei Han"</au>
   <abs>"data mining"</abs>
</article>.
```

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 for marking the requested element is introduced [Carmel et al. 2003]. Our approach differs from XML fragments in our focus on the descendant axis instead of the child axis, and our distinction between users having varying degrees of knowledge about valid tag nesting. E.g., *Contextual Content Information* can only be correctly specified in XML fragments using additional knowledge of the DTD.

5.2 How Structure is Used

Returning to the CAS queries of INEX 2003 and 2004, we provide a classification in terms of our four categories in Table X. We based this classification not on the actual syntactic shape of the queries, but on the fact whether they could equivalently be expressed in the query format of the category. The *Contextual Content Information* category is the most popular with 41%, followed by the *Search Hints in Context* category with 36%. No less than 55% of the 64 CAS queries does *not* use the hierarchical structure of the documents (categories *Restricted Search* and *Search Hints* combined). However, we also see that no less than 77% of the queries

	Fraction			
Category	2003	2004	Total	Query numbers
(a) Restricted search	13%	15%	14%	78, 79, 84, 86, 127, 136, 142, 143, 152
(b) Contextual content information	33%	47%	41%	61, 62, 63, 64, 68, 73, 74, 75, 77, 90, 128, 129, 130, 131, 132,134, 135, 137, 138,141, 144, 145, 151, 158, 159, 160
(c) Search hints	13%	6%	9%	67, 69, 80, 83, 147, 153
(d) Search hints in context	40%	32%	36%	65, 66, 70, 71, 72, 76, 81, 82, 85, 87, 88, 89 133, 139, 140, 146, 149, 150, 154, 155, 156, 157, 161

Table X. Classification of the INEX 2003 and 2004 CAS queries.

	Fraction			
Category	CO+S	CAS	Total	Query numbers
(a) Restricted search	32%	12%	24%	203, 207, 208, 210, 212, 219, 230, 231, 236, 257, 270
(b) Contextual content information	50%	71%	58%	202, 204, 220, 222, 223, 224, 225, 226, 228, 229, 232, 233, 234, 238, 244, 247, 253, 256, 258, 261, 264, 269, 275, 280, 284, 288
(c) Search hints	11%	12%	11%	205, 211, 216, 250, 260
(d) Search hints in context	7%	6%	7%	239, 240, 265

Table XI. Classification of the INEX 2005 CO+S and CAS topics

use content constraints on particular elements occurring in the context of the elements to be returned (categories *Contextual Content Information* and *Search Hints in Context* combined).

We repeat the classification over the four query categories for the INEX 2005 queries in Table XI. We see that almost one-third of the CO+S queries are of the *Restricted Search* category; the CAS queries are more complex. Over all INEX 2005 CAS queries, no less than 82% of the 45 queries does *not* use the hierarchical structure of the documents (*Restricted Search* and *Search Hints*). On the other hand, 65% of the queries does constrain the content of elements outside the requested element (*Contextual Content Information* and *Search Hints in Context*).

As to the first research question in the introduction (How do users exploit the additional expressive power of structural constraints in their queries?), we have two main findings. On the one hand, we see that two thirds of the CAS queries do *not* use the hierarchical structure of the documents. Or, equivalently, the hierarchical nature of the documents is used in one third of the queries we examined. Specifically, this is the case for 66% of all 109 CAS queries. On the other hand, we also see that almost three quarters of the queries use content constraints on particular elements occurring in the context of the elements to be returned. This is the case for 72% of all 109 CAS queries. These contextual constraints cannot be captured by ordinary keyword queries.

6. QUERY LANGUAGES FOR CONTENT AND STRUCTURE QUERIES

We have now seen that searchers often do not use all of the additional expressive power of structural constraints that is offered by the NEXI query language. A natural question suggests itself at this point: is the NEXI query language the most appropriate way of providing these features?

The NEXI query language is an extension of a subset of XPath (see Section 2.3). The motivation for restricting XPath is that users find it hard to state their information need in XPath and tend to make semantic mistakes in their query formulations. In this section we analyze why users make such mistakes and build a corresponding user profile. Then we show that the NEXI query language is a perfect fit for this user profile: on the one hand, users cannot make the found semantic mistakes because the language is restricted (the language is *safe*); on the other, they can express every information need belonging to this user profile (the language is *complete*).

6.1 Less Power is Better

At INEX, the focus is on retrieving sets of elements from document-centric XML documents using information about the content of the elements and their location in the documents. For this reason, it was decided to restrict the query language to the navigational part of XPath 1.0; in [Gottlob et al. 2002] this language is defined as Core XPath. The only objects which are manipulated in this language are sets of nodes (i.e., there are no arithmetical or string operations). Besides these restrictions, the full power of location paths is supported (except for namespace and attribute axis), including filter expressions being closed under the boolean operators. At INEX 2003, Core XPath expanded with the `about` function was used as a query language. The results were disappointing: many queries did not match the information need as described in the narrative and description part; often, the information need was much broader than the XPath expression [O'Keefe and Trotman 2004]. A typical mistake was the use of / (child axis) where // (descendant) was intended. These semantic mistakes can likely be attributed to the fact that users have no, or at best incomplete, knowledge of the structure of documents, that is, of the DTD. To reduce the chance of making such semantic mistakes, O'Keefe and Trotman [2004] argued that apart from the descendant axis no other axis relations should be used in queries. This recommendation was implemented in the INEX 2004 NEXI query language (described in Section 2.3). In this section, we provide a theoretical basis for this recommendation by giving a mathematical model of a user's knowledge of a document collection and by relating the expressive power of the NEXI query language to this model.

6.2 Modeling Users' Knowledge of a Document Collection

How can we give a mathematically precise and yet intuitive model of a user's knowledge of a document collection? The starting point is that we want to model users with incomplete knowledge of the structure of documents. For such users, certain structural changes made to a document will not be discernible: the user considers the two documents to be the same. For instance, most INEX users will not distinguish the two documents in Figure 3 solely based on the tag structure. The idea is that the less knowledge a user has, the more structural differences will remain unnoticed, hence the more documents will be considered the same. For a user, two indiscernible documents are the same, and a query should return the same answers from both documents. But there are XPath queries which return different answers

```
<root>
   <paragraph>
      Here is some text.
      Here is some more text.
   </paragraph>
</root>
```

```
<root>
   <paragraph>
      Here is some text.
   </paragraph>
   <paragraph>
      Here is some more text.
   </paragraph>
</root>
```

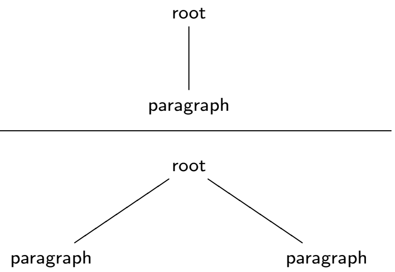


Fig. 3. Two XML documents and their corresponding structure trees.

on the documents in Figure 3 (for instance, `//paragraph[1]` which returns the *first* paragraph in document order). This is the reason for considering weaker fragments of XPath, those for which indiscernible documents yield identical answers.

To summarize, a user's knowledge about a set of XML documents can be formalized in terms of an *indiscernibility relation* between documents. Given such an indiscernibility relation, we are looking for a corresponding query language. Now there are two competing forces at work on the desired language: *safety*, which reduces expressive power, and *completeness*, which asks for as much expressivity as possible. In a safe query language, users cannot write queries which return different answers from documents which these users consider to be the same. A safe language is designed to avoid making semantic mistakes by forbidding the user to pose such queries. We will shortly see that the NEXI language is an example of a safe and complete query language.

6.3 User Profiles

Below, we define two user profiles in terms of indiscernibility relations, both capturing users with limited knowledge of the DTD of the document collection. First, we consider what we call *structure-unaware users* who only know the tag names. Second, we consider *hierarchy aware users*, who know the tag names and have some clue about the hierarchical structure of the elements, without knowing the full details. For both profiles, we design fragments that are safe and complete. The analysis here covers only the structure of the documents and abstracts away from the content. So we remove the `about` function from the query language and concentrate solely on its navigational aspects.

6.3.1 *Structure-Unaware Users.* Users formulating queries at INEX did not have a clear idea of the DTD of the collection [O'Keefe and Trotman 2004]. Typically, they browsed the documents and picked up some knowledge about the available tags in this manner. Their queries can be viewed as an XML version of fielded search. Recall that standard fielded search allows for restricting search to particular fields, think of a library catalogue (OPAC) where fields like author or title can be used to restrict search. For users who know (a subset of) the tag names, but do not (want to) know the structure of the documents, an XPath fragment which exactly fits their knowledge can be created. The typical queries of a structure-unaware

```
<root>
   <section>
     <theorem>
     </theorem>
   </section>
</root>
```

```
<root>
   <section>
   </section>
   <theorem>
   </theorem>
</root>
```

Fig. 4. Example of two indiscernible documents for structure-unaware users.

user are the *Restricted Search* and *Contextual Content Information* queries from Section 5. Figure 4 shows an example of documents that are indiscernible for structure-unaware users. For this user profile, a query like *Give me theorems below sections* would not be safe, because it would return different answers from both documents. In the query language fitting the structure unaware user profile, a user can only express safe queries like *Give me theorems* (e.g., `//theorem`).

The following syntax, which we call *structure unaware XPath*, allows us to pose these queries. 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 `self::tag` and queries of the form `//tag[P]`. Note that when `//tag[P]` is used in a filter it means "there exists a descendant of the root labeled `tag` at which the predicate `P` evaluates to true." I.e., `//tag[P]` simply says that somewhere in the document there is a `tag` element making `P` true. `self::tag` expresses that the current node is labeled by `tag`.

We turn to a semantic characterization of this fragment. In social network theory [Wasserman and Faust 1994] several indiscernibility relations have been proposed, including the useful and robust notion of 'regular equivalence.' This notion is more commonly known as *bisimulation*, an equivalent notion introduced by modal logicians [van Benthem 1983]. We need the following special "structurally unaware" version.

Definition 6.1. Let D, D' be documents and B a binary relation between the elements of D and D' connecting the roots. We call B a *structure unaware bisimulation* if,

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

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 all documents D, D', elements d, d', and bisimulations $B \subseteq D \times D'$, if dBd', then $\phi(d)$ is true if and only if $\phi(d')$ is true.

A few comments. First, the relation which connects the roots and the paragraph elements in the two trees in Figure 3 is a structure unaware bisimulation. Also, there exists such a bisimulation between the document trees in Figure 7. Secondly, first-order formulas in one free variable can be seen as an alternative stronger query language than XPath (for the relative expressive power of the two cf., [Marx and de Rijke 2005]). Thirdly, in the usual definition of bisimulation, the clauses in items 2 and 3 are more complicated (as in item 2 in Definition 6.3 below), and say that the structure of D should be preserved in D'; but our imagined user is not aware of the structure, hence we omitted these conditions. In effect, two document trees can be related by a bisimulation if there is no tag name l which labels an element in one tree but not in the other.

THEOREM 6.2. *(Safety). Let D, D' be documents, B a structure unaware bisimulation, and P a structure unaware XPath expression. Then $X \subseteq D$ is the answer set of P on D if and only if $\{d' \in D' \mid \exists d \in X : dBd'\}$ is the answer set of P on D'.*

(Completeness). For every first-order formula that is invariant under structure unaware bisimulations there exists an equivalent structure unaware XPath expression.

We can conclude that structure unaware XPath is a perfect fit for the user profile sketched: the first part of the theorem states that it is *safe*, the second that it is *complete*.

Before we give a formal proof of Theorem 6.2 we provide the intuition for the (easy) safety part. (This is formally proved by an induction on the structure of the query). Consider the query `//section[//abstract]`. Suppose that it returns an element d on document D, and that B is a bisimulation between D and D', connecting d and d'. Safety says that the query should then also return d' when evaluated on D'. We can prove that as follows. The label of d is `section`. Because dBd' holds, the label of d' is also `section`. Because the predicate `//abstract` returns true at d, there must be an element $c \in D$ labeled `abstract`. By the bisimulation condition, then, there is a $c' \in D'$ such that cBc'. But then c' is also labeled `abstract`. Thus `//abstract` also returns true at d' and d' is returned as an answer to `//section[//abstract]`.

PROOF. Theorem 6.2 is a reformulation of Van Benthem's characterization theorem for the modal logic of the universal modality [Blackburn et al. 2001, Theorem 2.68]. The language of this logic is propositional with an extra unary operator $\Diamond$. This language is interpreted on sets (of worlds) W, equipped with a valuation of the propositional variables. Each formula denotes a subset of W. The Boolean connectives are interpreted by their corresponding set theoretic operations. The modal formula $\Diamond\phi$ denotes the empty set if ϕ denotes the empty set, and W otherwise.

With this interpretation of the modality $\Diamond$, the modal language is just a syntactic variant of the predicates of structure unaware XPath. Consider the following translations:

$$\begin{array}{ll}
 & (\texttt{self} :: \texttt{p})^b \;=\; p \\
p^f \quad\;\; = \texttt{self} :: \texttt{p} & (\texttt{self} :: *)^b \;=\; \top \\
(\cdot)^f \text{ commutes with the booleans} & (\cdot)^b \text{ commutes with the booleans} \\
(\Diamond\phi)^f \;=\; // * [\phi^f] & (//\texttt{tag}[\texttt{P}])^b \;=\; \Diamond(\texttt{tag} \wedge \texttt{P}^b) \\
 & (// * [\texttt{P}])^b \;\;=\; \Diamond P^b
\end{array}$$

By a straightforward induction we can prove that for each model,

(1) the denotation of ϕ is X if and only if X is the answer set of `//*[`ϕ^f`]`;

(2) X is the answer set of `//p[P]` if and only if the denotation of $\texttt{p} \wedge \texttt{P}^b$ is X.

```
<root>
   <section>
     <subsection>
     </subsection>
     <paragraph>
     </paragraph>
   </section>
</root>
```

```
<root>
   <section>
     <subsection>
       <paragraph>
       </paragraph>
     </subsection>
   </section>
</root>
```

Fig. 5. Example of two documents that are indiscernible for hierarchy aware users with respect to the section/paragraph nesting.

Item (1) is used to prove completeness. Let $F(x)$ be a first order formula that is invariant under structure unaware bisimulations. Then by Van Benthem's theorem, there exists a modal formula ϕ such that for every model, for each element d, $F(d)$ holds if and only if d is in the denotation of ϕ. But then by (1), `//*[`ϕ^f`]` is the XPath expression equivalent to $F(x)$. With item (2) we prove safety. Let B be a bisimulation between D and D', and let $d \in D$. By definition, there must be a d' such that dBd'. By the safety part of Van Benthem's theorem, d and d' make the same modal formulas true. But then by (2), d is in the answer set of any XPath expression `//p[P]` if and only if d' is. □

6.3.2 *Hierarchy Aware Users.* Hierarchy aware users have some clue about the hierarchical structure of the documents. E.g., they know that paragraphs are below sections, but need not know that there may be elements in between [O'Keefe and Trotman 2004]. Figure 5 shows an example of documents that are indiscernible for hierarchy aware users with respect to the section/paragraph nesting. For this user profile, a query like *Give me paragraphs directly below sections* would not be safe, because it would return the different answers from both documents. In the query language fitting the structure unaware user profile, a user can only express safe queries like *Give paragraphs below sections* (e.g., `//section//paragraph`). For this reason, O'Keefe and Trotman [2004] proposed Positive Descendant XPath: the fragment of XPath in which only the descendant and self axis relations may be used and the booleans in the predicates are restricted to "`and`" and "`or`". Note that all types of queries discussed in Section 3 can be formulated in this fragment.

As this XPath fragment does not contain negation, bisimulation is too strong a notion [Kurtonina and de Rijke 1999]. 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 6.3. Let D, D' be documents and B a binary relation between the elements of D and D' connecting the roots. We call B a *vertical simulation* from D to D' if, for all $x \in D$, for all $x' \in D'$, whenever xBx' holds, then

1. x and x' have the same tag names;
2. for all $y \in D$ such that $x < y$, there exists a $y' \in D'$ such that $x' < y'$ and yBy'; and
3. similarly when $y < x$.

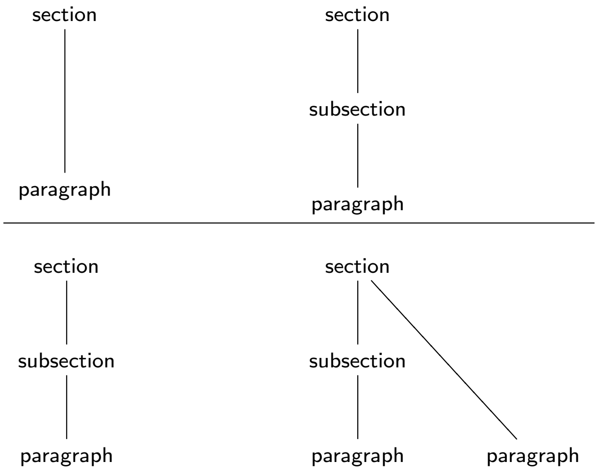


Fig. 6. Two examples of simulations, but not bisimulations.

Let $\phi(x)$ be a first-order formula (in one free variable) in a suitable vocabulary; $\phi(x)$ is *preserved under vertical simulations* whenever the following holds: for all documents D, D', elements d, d', and vertical simulations $B \subseteq D \times D'$, if dBd', then $\phi(d)$ implies $\phi(d')$

Vertical simulations capture users that know the element hierarchy: note that both elements below *and* above have to be simulated.

Figure 6 contains two examples, in which we have simulations from the document on the left-hand side to the document right-hand side, but not conversely. In the example at the top we cannot simulate the subsection under the section. In the example at the bottom we cannot simulate the paragraph without a subsection ancestor. The next theorem is an analogue of Theorem 6.2 for Positive Descendant XPath: it is both safe and complete for hierarchy aware users.

THEOREM 6.4. *(Safety). For each positive descendant XPath query P, if B is a vertical simulation from D to D' connecting d and d', and P returns d on D, then P also returns d' on D'.*

(Completeness). Let $\phi(x)$ be a first order formula which is preserved under vertical simulations. Then there exists a union of positive descendant XPath formulas which on every document returns exactly those elements d for which $\phi(d)$ holds.

PROOF. In the proof of safety we see that all clauses in the definition of a simulation are needed. We prove it by a double induction on the structure of the query.

CLAIM 1. *Let B be a simulation from D to D' such that dBd'. Then for any positive descendant XPath predicate `P`, if `P` is true at d, then it is true at d'.*

PROOF. By induction on the structure of `P`. If `P=self::tag`, then the claim holds because dBd' implies that d and d' have the same label. Boolean combinations are taken care of by the inductive hypothesis. If `P=.//tag[Q]` and `P` holds at d, then there exists an e such that $d < e$ and e's label is `tag` and `Q` is true at e. But then there exists an e' such that $d' < e'$ and eBe'. By inductive hypothesis then, the label of e' is `tag` and `Q` is true at e'. Thus `.//tag[Q]` is true at d'.

If `P=//tag[Q]` we use the fact that the roots are connected by the simulation and apply the previous argument. ☐

CLAIM 2. *Let B be a simulation from D to D' such that dBd'. Then for any positive descendant XPath query* $//\mathtt{t}_1\mathtt{[P}_1\mathtt{]}//\ldots//\mathtt{t}_n\mathtt{[P}_n\mathtt{]}$*, if it returns d on D, then it returns d' on D'.*

PROOF. By induction on the number of `//`. For the base case, the query is of the form $//\mathtt{t}_1\mathtt{[P}_1\mathtt{]}$ and we can use Claim 1. Thus suppose the query is of the form $//\mathtt{t}_1\mathtt{[P}_1\mathtt{]}//\ldots//\mathtt{t}_n\mathtt{[P}_n\mathtt{]}//\mathtt{t}_{n+1}\mathtt{[P}_{n+1}\mathtt{]}$ and it returns d on D. Then there is a $c \in D$ such that $c < d$ and $//\mathtt{t}_1\mathtt{[P}_1\mathtt{]}//\ldots//\mathtt{t}_n\mathtt{[P}_n\mathtt{]}$ returns c on D. By definition of the simulation, there must be a $c' \in D'$ such that $c' < d'$ and cBc'. By inductive hypothesis then $//\mathtt{t}_1\mathtt{[P}_1\mathtt{]}//\ldots//\mathtt{t}_n\mathtt{[P}_n\mathtt{]}$ returns c' on D'. Now d's label is $\mathtt{t}_{n+1}$ and it makes $\mathtt{P}_{n+1}$ true. By Claim 1, dBd' implies that the same holds for d'. But then $//\mathtt{t}_1\mathtt{[P}_1\mathtt{]}//\ldots//\mathtt{t}_n\mathtt{[P}_n\mathtt{]}//\mathtt{t}_{n+1}\mathtt{[P}_{n+1}\mathtt{]}$ returns d' on D. ☐

This concludes the proof for safety. The proof for completeness uses ideas from modal logic [Blackburn et al. 2001, Theorem 2.78] together with ideas from [Benedikt et al. 2003, Theorem 3.2]. It essentially involves two steps. First, one shows that the following two query languages can define exactly the same sets of elements:

(a) unions of positive descendant XPath formulas;

(b) formulas of the form `//*[P]` where `P` is a positive ancestor and descendant XPath formula.

The formalism under (b) is a syntactic variant of positive temporal logic, very much like in the proof of Theorem 6.2. The second step in the proof is now easy: the appropriate version of Van Benthem's theorem now provides the completeness result.

That language (b) is at least as strong as language (a) is rather easy and shown in [Marx and de Rijke 2005]. The main step in the proof of the other direction is to show that unions of positive descendant XPath formulas are closed under intersections in the sense of [Benedikt et al. 2003]. This can be done using the technique from the proof of their Theorem 3.2. ☐

Descendant or Descendant-or-self?. Positive Descendant XPath has great syntactic appeal because the only operator is //. It is a natural fragment because it corresponds exactly to the hierarchy aware users. Still, one could argue that it is too expressive for these users. Consider the two document trees in Figure 7. There are no vertical simulations between these two. But, according to the data and the arguments in [O'Keefe and Trotman 2004], INEX users consider them to be the same. We can easily adjust our notion of simulation to cater for this: instead of simulating the descendant relation $<$, only simulate the descendant-or-self relation $\leq$. Then, these two documents even vertically bisimulate. Unfortunately, there is

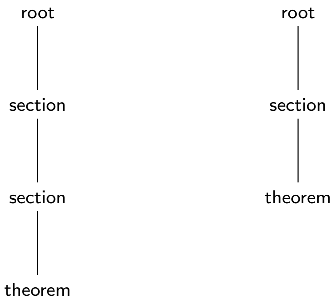


Fig. 7. Document trees that do not bisimulate.

no appealing abbreviated syntax for the corresponding query language ("Positive Descendant-or-Self XPath").

7. DISCUSSION AND CONCLUSIONS

Our findings are based on an unconditional IR approach to XML retrieval. That is, we view queries as inexact statements of an underlying information need, and the ground truth for evaluation is based on the usefulness of the retrieved elements with respect to the information need rather than on a literal match with the query. This approach seems a close fit to searching document-centric XML on the web, where expert and non-expert users with varying degrees of knowledge of the DTD may still want to exploit particular markup to focus their search. In many other scenarios, think of searching data-centric XML, other approaches may be more natural. Although we looked at a prototypical specimen of document-centric XML, full-text scientific articles in predominantly lay-out markup, there would be obvious value in repeating the type of analysis in this paper for other XML collections.

Our study provides a range of evidence to support the view that the structure in queries functions as a precision device for XML retrieval: it is a search hint rather than a search requirement. Vague structural matching has a long history. The pioneering work on XIRQL had vague structural matching as one of its key points [Fuhr and Großjohann 2001; 2004]. Also in XML fragments [Carmel et al. 2002; Carmel et al. 2003], documents that are a partial match to the structure can still be retrieved. The CAS task at INEX has gradually embraced vague structural matching, and taken it further to a pure IR approach in which there is, from the point of view of evaluation, no difference between keyword topics and structured topics. A useful overview of the various CAS tasks is provided in [Trotman and Lalmas 2006]; their conclusions strongly support our pure IR approach.

We can identify a number of important lessons for future work in information retrieval from document-centric XML collections. Simply combining powerful XML query languages with IR-style retrieval and ranking of results does not work. The addition of structure to queries is not a simple recipe for improving results. This is in line with earlier work: the use of structure in queries has been studied extensively; 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 but rarely anything substantial [Fagan 1987]. As retrieval models became more advanced, the usage of query operators was questioned. E.g., Mitra et al. [1997] 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 [2003] combine term-proximity heuristics with an Okapi model, obtaining 3%–8% improvements for Precision@5, 10, 20, with hardly observable impact on the MAP scores.

For XML retrieval we draw the following conclusions. First, as observed in [O'Keefe and Trotman 2004], less expressivity is better in that it reduces the chance of making semantic mistakes. We have shown that the proposed NEXI query language [O'Keefe and Trotman 2004] is not ad hoc, but has a precise mathematical characterization in terms of an intuitive user profile. Second, users tend not to use hierarchical structure in their queries. Two thirds of the queries can be expressed in the very restrictive structure unaware XPath fragment. This language allows searchers to express fielded queries in which the user can provide both the field names and what they should contain (more precisely, what they should be about). Third, three quarters of the queries use constraints on the context of the elements to be returned; these contextual constraints cannot be captured by ordinary keyword queries. Fourth, we found that structure is used as a search hint, and not as a strict search requirement, when judged against the underlying information need. As a consequence, we hypothesized that the use of structure in queries functions as a precision enhancing device. To test this hypothesis we conducted a set of experiments. The outcomes confirm that structured queries function as a precision enhancing device: useful for promoting the precision of initially retrieved documents, possibly reducing fall-out but also reducing recall. Structured queries can be a powerful tool, catering for the typical web searcher who is interested solely in the precision of the first handful of results—importantly, the INEX Interactive Track revealed that users rarely look beyond the first handful of returned elements [Tombros et al. 2005].

ACKNOWLEDGMENTS

We thank the participants of the Topic Format Working Groups at INEX 2002, 2003, and 2004. We gratefully acknowledge the reviewers for their comments that helped shape this paper.

This research was supported 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, 612.066.513, 612.069.006, 640.001.501, and 640.002.501.

REFERENCES

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

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

Carmel, D., Maarek, Y. S., Mandelbrod, M., Mass, Y., and Soffer, A. 2003. Searching XML documents via XML fragments. In *Proc. SIGIR*. 151–158.

Carmel, D., Maarek, Y. S., Mass, Y., Efraty, N., and Landau, G. M. 2002. An extension of the vector space model for querying XML documents via XML fragments. In *Proceedings SIGIR 2002 Workshop on XML and Information Retrieval*. 14–25.

Fagan, J. 1987. Experiments in automatic phrase indexing for document retrieval: A comparison of syntactic and non-syntactic methods. Tech. rep., Cornell University.

Fuhr, N., Gövert, N., Kazai, G., and Lalmas, M., Eds. 2003. *Proceedings of the First Workshop of the INitiative for the Evaluation of XML Retrieval (INEX 2002)*. ERCIM.

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

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

Fuhr, N., Lalmas, M., and Malik, S., Eds. 2004. *INEX 2003 Workshop Proceedings*.

Fuhr, N., Lalmas, M., Malik, S., and Kazai, G., Eds. 2006. *Advances in XML Information Retrieval and Evaluation: Fourth Workshop of the INitiative for the Evaluation of XML Retrieval (INEX 2005)*. Lecture Notes in Computer Science, vol. 3977. Springer-Verlag.

Fuhr, N., Lalmas, M., Malik, S., and Szlávik, S., Eds. 2005. *Advances in XML Information Retrieval: Third Workshop of the INitiative for the Evaluation of XML Retrieval (INEX 2004)*. Lecture Notes in Computer Science, vol. 3493. Springer.

Gottlob, G., Koch, C., and Pichler, R. 2002. Efficient algorithms for processing XPath queries. In *VLDB'02*.

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

Hiemstra, D. 2001. Using language models for information retrieval. Ph.D. thesis, University of Twente.

INEX 2006. INitiative for the Evaluation of XML Retrieval. `http://inex.is.informatik.uni-duisburg.de/`.

Kamps, J., de Rijke, M., and Sigurbjörnsson, B. 2006. What do users think of an XML element retrieval system? See Fuhr et al. [2006].

Kamps, J., Marx, M., de Rijke, M., and Sigurbjörnsson, B. 2005. Structured queries in XML retrieval. In *Proc. CIKM 2005*. ACM Press, 2–11.

Kazai, G. and Lalmas, M. 2006. INEX 2005 evaluation measures. See Fuhr et al. [2006].

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

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

Marx, M. and de Rijke, M. 2005. Semantic Characterizations of Navigational XPath. *ACM SIGMOD Record 34,* 2, 41–46.

May, W. 1999. Information extraction and integration with Florid: The Mondial case study. Tech. rep., Universität Freiburg, Institut für Informatik.

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

O'Keefe, R. A. and Trotman, A. 2004. The simplest query language that could possibly work. See Fuhr et al. [2004], 167–174.

Rasolofo, Y. and Savoy, J. 2003. Term proximity scoring for keyword-based retrieval systems. In *Proc. ECIR 2003*. 207–218.

Sigurbjörnsson, B., Kamps, J., and de Rijke, M. 2004a. An element-based approach to XML retrieval. See Fuhr et al. [2004], 19–26.

Sigurbjörnsson, B., Kamps, J., and de Rijke, M. 2004b. Processing content-oriented XPath queries. In *Proc. CIKM 2004*. ACM Press, 371–380.

Sigurbjörnsson, B., Kamps, J., and de Rijke, M. 2006. The effect of structured queries and selective indexing on XML retrieval. See Fuhr et al. [2006].

Sigurbjörnsson, B., Larsen, B., Lalmas, M., and Maalik, S. 2004c. INEX04 guidelines for topic development. In *INEX 2004 Workshop Pre-Proceedings*, N. Fuhr, M. Lalmas, S. Malik, and Z. Szlávik, Eds. 219–236.

Sigurbjörnsson, B. and Trotman, A. 2003. Queries, INEX 2003 working group report. See Fuhr et al. [2003].

Tombros, A., Larsen, B., and Malik, S. 2005. The interactive track at INEX 2004. See Fuhr et al. [2005], 410–423.

Trotman, A. and Lalmas, M. 2006. The interpretation of CAS. See Fuhr et al. [2006].

Trotman, A. and Sigurbjörnsson, B. 2005. Narrowed Extended XPath I (NEXI). See Fuhr et al. [2005], 16–40.

van Benthem, J. 1983. *Modal Logic and Classical Logic.* Bibliopolis, Napoli.

Vianu, V. 2001. A Web odyssey: from Codd to XML. In *Proc. PODS*. ACM Press, 1–15.

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

...