Score Region Algebra: A Framework for Structured IR

Vojkan Mihajlović
CTIT, University of Twente
P.O. Box 217, 7500AE Enschede, The Netherlands
v.mihajlovic@utwente.nl

ABSTRACT

We address the problem of developing a flexible framework for information retrieval (IR) in structured documents, such as XML. The framework is able to support a wide range of structured IR queries, transparent instantiations of different retrieval models, and different physical implementations. It is based on so-called *score region algebra (SRA)* that can express the following four essential ranked retrieval aspects for structured IR: term and element selection, element relevance score computation, element score propagation, and element score combination. Our preliminary research shows that different instantiations of each aspect, as well as different combinations of these instantiations, yield significantly different results. Our goal is to better understand structured IR by studying these aspects alone and their combination in the framework of SRA, and to use this knowledge to improve our structured IR system.

Categories and Subject Descriptors

H.2 [**Database Management**]: Logical Design—*data models*; H.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*retrieval models, search process*; I.1 [**Symbolic and Algebraic Manipulation**]: Languages and Systems—*special-purpose algebraic systems*

Keywords

Database systems, information retrieval, XML, logical algebra, retrieval models, XML ranked retrieval.

1. MOTIVATION FOR THE RESEARCH

Structured documents are gaining popularity through standards for storing, carrying, exchanging, and presentation of data, such as SGML, XML, and HTML. With the rapid growth of data stored in structured format, especially XML, information retrieval (IR) on structured collections, becomes an important issue [1, 9, 10, 12, 20, 21]. The main goal of our research is the development of a framework for structured IR that can support different IR query languages for structured documents, *transparent* specification of different retrieval models, and different physical implementations. The framework should be flexible to enable rich query specification, ranging from a set of terms to a combination of searches performed in different parts of a structured collection, specified using search terms and (context) search elements. Its transparency should be reflected in its ability to keep the same framework while supporting different retrieval models. Finally, it should support distinct physical storage structures and access algorithms.

1.1 Traditional vs. structured IR systems

The basic task of traditional information retrieval systems is to perform search based on a user query, consisting in most cases of a set of query terms. Using a search method (model or approach), it should provide the user with a (ranked) list of answers satisfying his information need. Traditional IR systems represent a document as a "bag of words", where inverted file structures provide the basis for implementing a retrieval system. Although this leads to a simple and efficient implementation, these systems lack the notion of *data independence* [8]: any change in what constitutes a document, or any change in document structure, will lead to system developers changing major parts of the system.

In structured IR, the user has the opportunity to specify where in the document he would like to search for information or which part of the document he would like to see as an answer. This leads to a set of complex IR aspects that are not addressed in traditional IR systems. We argue that, to develop a flexible and transparent structured IR system, we need to support four basic *structured IR aspects*: element and term selection, element relevance score computation, element score combination, and element score propagation.

In our work we focus on XML as it is most frequently used structural format nowadays. We illustrate these three aspects on a query example expressed in NEXI query language [25], where the user searches XML collection for section elements ('sec') that contain paragraphs ('p') describing a perfect bouquet:

```
//sec[about(.//p, perfect bouquet)]
```

Even though the NEXI query example is relatively simple, it is expressive enough to illustrate the main differences between traditional IR and structured IR. If we analyze the example query we can distinguish between elements where users search for information ('p'), named *search elements*, elements which should be presented to the user as an an-

SIGIR'04 Doctoral Consortium,

swer to a query ('sec'), named *answer elements*, and *search terms* ('perfect' and 'bouquet'). Therefore, the first task of structured IR is to *select* the elements and terms based on query specification. The next task would be to determine the *relevance score* for the element in the *about* clause ('p') with respect to each of query terms ('perfect' and 'bouquet'). Then the relevance scores for a 'p' element with respect to both terms should be *combined* to produce the resulting score. Finally, the scores from the search element 'p' should be *propagated* to the answer element 'sec'. In Section 4 we elaborate more on these aspects and below we give arguments why it is more appropriate to use a database approach for information retrieval in structured documents.

1.2 The database approach

The main characteristic of the database approach is a strong separation between conceptual, logical, and physical levels [26]. For structured IR systems, following this separation in levels gives another, additional advantage: by choosing the appropriate level of abstraction for the logical level, the development of scoring techniques handling structural information is simplified, and kept transparent for the rest of the system design. Therefore, having a logical level with the algebra defined in it would provide the right level of abstraction considering different IR query languages (e.g., XQuery with full-text search extensions [4], NEXI [25], XIRQL [9]) formed on the conceptual level, and the storage structure chosen on the physical level (e.g., [15]). In such a way, the needed *data independence* on logical level is provided.

Also an important reason for defining a logical algebra is to enable expression of structured IR queries, i.e., score computation and ranking of context elements. The algebra should provide a specific level of *IR understanding*, based on the retrieval model used, instantiated using four basic structured IR aspects in logical algebra. Furthermore, the reasoning that can be done on the logical level can be useful for query rewriting and *optimization*. Using knowledge about the size of the operands and the cost for the execution of different operators on the physical level, we are able to generate different logical query plans achieving faster execution times and lower memory requirements when executed on physical level.

Concluding, in our research we would like to give an answer to the question on how we can develop a flexible and transparent structured IR system system that will support all four structured IR aspects. We argue that the answer is in the algebra on logical level of a database system, that we named Score Region Algebra (SRA). After introducing the related work on region algebras and on IR and databases in the next section, we explain how logical algebra can be used to define scoring operators in Section 3. In Section 4 we elaborate more on four XML IR aspects modeled in SRA and give some issues on SRA operator properties and the evaluation of our approach.

2. RELATED WORK

Related work is divided in two parts: (i) region algebra approaches for search in structured documents that we use as a base for our logical algebra and (ii) the most illustrative database approaches to XML IR with which we compare our approach.

2.1 Structured Documents Search

Region algebras were introduced as an answer to the problem of modeling and searching in (semi-)structured documents. The basic idea behind the region algebra approaches is the representation of text documents as a set of *regions*, where each region is defined by its start and end positions. The elementary region algebra operators define containment relation among regions and set operations (union, intersection, and difference). The application of the idea of text regions to XML documents is straightforward. Each XML document can be seen as a sequence of tokens, i.e., start tags, end tags, and terms, where tokens can then be grouped if necessary (begin and end tag) and represented as regions (see [17] for more details).

The earliest region algebra approaches used for text search were the PAT system presented in [23], followed by work of Burkowski [3] and Clarke et al. [6] in the area of textual databases. They were later extended to support additional operators, like positional inclusion (containment relation with the specified relative position of a contained region) and direct inclusion (parent-child relationship), by Navarro and Baeza-Yates [2] and Consens and Milo [7]. The usefulness of region algebra for structured document search is proven in the work of Jaakkola and Kilpelainen [13] and Miller [19]. Recently, Masuda et al. [16] extended the definition of algebra operators to support ranked retrieval. However, their approach can not be considered as fully algebraic since the ranking is not defined in the scope of algebraic operators but rather as a side effect of the application of algebraic operators (similar to the approach presented in [3]).

2.2 XML IR in Database Systems

Amer-Yahia et al. developed a full-text search extension to XQuery [1]. The goal of the extension is to support highly expressive XML full-text query specification [4]. The extension introduces a "full match" model that supports scoring based on full-text predicates. However, as the model does not allow the combination of scores for search in distinct XML elements (since the result of each element search is cast back to the XQuery data model from the full match model), the concepts of element score propagation and combination are not completely supported. Moreover, the authors assume that the model used for ranked retrieval is implementation dependent, abstracting in that way from problems of the XML IR database integration, and score propagation and combination aspects of structured retrieval.

Fuhr et al. [9, 10] developed a path-based XML IR algebra and query language named XIRQL. The algebra integrates concepts from logic-based probabilistic IR models and databases. It supports term weighting and vague predicates (vagueness in the retrieved elements). Following the logic-based model, the algebra does not allow distinct specifications of the score propagation and combination. Another property of the model is that the potential answer elements are predefined in the system and the score can only be computed for these elements.

3. SRA FRAMEWORK

The basic idea behind our logical algebra is to enable transparent specification of retrieval models and to integrate XML database and information retrieval approaches, as discussed in [1, 8]. Unlike many approaches for ranked

Table 1: Score region algebra operators.

Operator	Operator definition
$\sigma_{\text{n=name, t=type}}(R)$	$\{r \mid r \in R \wedge n = name \wedge t = type\}$
$R_1 \sqsupset R_2$	$\{r_1 \mid r_1 \in R_1 \wedge \exists r_2 \in R_2 \wedge r_2 \prec r_1\}$
$R_1 \sqsubset R_2$	$\{r_1 \mid r_1 \in R_1 \wedge \exists r_2 \in R_2 \wedge r_1 \prec r_2\}$
$R_1 \sqsupset_p R_2$	$\{r \mid r_1 \in R_1 \wedge (s,e,n,t) := (s_1,e_1,n_1,t_1) \wedge t_1 = node \wedge t_2 = term \wedge p := f_{\sqsupset}(r_1,R_2)\}$
$R_1 \not\sqsupset_p R_2$	$\{r \mid r_1 \in R_1 \wedge (s,e,n,t) := (s_1,e_1,n_1,t_1) \wedge t_1 = node \wedge t_2 = term \wedge p := f_{\not\sqsupset}(r_1,R_2)\}$
$R_1 \blacktriangleright R_2$	$\{r \mid r_1 \in R_1 \wedge (s,e,n,t) := (s_1,e_1,n_1,t_1) \wedge t_1 = node \wedge t_2 = node \wedge p := f_{\blacktriangleright}(r_1,R_2)\}$
$R_1 \blacktriangleleft R_2$	$\{r \mid r_1 \in R_1 \wedge (s,e,n,t) := (s_1,e_1,n_1,t_1) \wedge t_1 = node \wedge t_2 = node \wedge p := f_{\blacktriangleleft}(r_1,R_2)\}$
$R_1 \sqcap_p R_2$	$\{r \mid r_1 \in R_1 \wedge r_2 \in R_2 \wedge (s_1,e_1,n_1,t_1) = (s_2,e_2,n_2,t_2) \wedge (s,e,n,t) := (s_1,e_1,n_1,t_1) \wedge p := p_1 \otimes p_2\}$
$R_1 \sqcup_p R_2$	$\{r \mid r_1 \in R_1 \wedge r_2 \in R_2 \wedge ((s,e,n,t) := (s_1,e_1,n_1,t_1) \vee (s,e,n,t) := (s_2,e_2,n_2,t_2)) \wedge p := p_1 \oplus p_2\}$

retrieval in structured documents, the algebra we define assumes that ranking is a part of the algebra and not a side effect of performing some operations on regions (like in [3, 16]) or a separate IR module (see, e.g., [21]). Therefore, we follow the approach taken by Fuhr et al. [9, 10], although we base our algebra on the containment model rather than path model, and do not make any restrictions on the definition of the retrieval model. By defining the algebra in such a way, we have the opportunity to utilize the optimization methods not just for basic region algebra operators, but for the ranking region algebra operators as well.

Score region algebra (SRA) enables transparent formal specification of the retrieval model and provides an opportunity to use and compare different retrieval models within the same framework without the need to change the SRA model. This is possible because the instantiation of the retrieval model is algebra-independent. The only aspect of the algebra that changes with the introduction of distinct retrieval models are operator properties. The SRA data model consists of regions (R) identified by five attributes, namely, region start (s), region end (e), region name (n), region type (t), and region score (s).

In Table 1 we defined the basic SRA operators used for XML retrieval, where $r_i \prec r_j \Leftrightarrow s_j < s_i \leq e_i < e_j$. The first three SRA operators copy the score of regions in the first operand and either select regions based on their name and type attributes (σ) or based on their containment relation using operators containing ($\sqsupset$) and contained by ($\sqsubset$). The other operators in Table 1 include score manipulation. In SRA, element relevance score computation is done using operators $\sqsupset_p$ and $\not\sqsupset_p$. The score propagation to the element that is either contained or contains search elements is defined with operators $\blacktriangleleft$ and $\blacktriangleright$, respectively. The scores can be combined via operators $\sqcap_p$ for "and" and $\sqcup_p$ for "or" logical combination. As can be seen in the operator definition, the instantiation of the retrieval functions ($f_{\sqsupset}$, $f_{\not\sqsupset}$, $f_{\blacktriangleleft}$, and $f_{\blacktriangleright}$) and abstract combination operators ($\otimes$ and $\oplus$) that defines scoring mechanism is left open, as the framework of SRA enables different instantiation of retrieval models.

We can now express the NEXI example query in SRA, where C denotes the set of all regions in the collection:

$$\sigma_{\text{t=node, n='sec'}}(C) \blacktriangleright ((\sigma_{\text{t=node, n='p'}}(C) \sqsupset_p \sigma_{\text{t=term, n='perfect'}}(C)) \sqcap_p (\sigma_{\text{t=node, n='p'}}(C) \sqsupset_p \sigma_{\text{t=term, n='bouquet'}}(C)))$$

How we can specify these operators to support transparent instantiation of retrieval models is explained in the following section.

4. RESEARCH QUESTIONS

In this section we give an overview of the intended studies we would like to perform. They are based on the structured IR aspects identified in Section 1: element and term selection, element score computation, element score propagation, and element score combination, and on SRA operators and operator properties. Our research so far indicates that the effectivness of the structured IR systems highly depend on the instantiation of these aspects in SRA [15, 17, 18] and demands more elaborate studies. The section is concluded with the specification of the research evaluation framework.

4.1 Element and term selection

As we illustrated in Section 1, in the retrieval process we first have to perform selection on search and answer elements and search terms in the query with all the elements and terms in the collection. In SRA this aspect is modeled with selection operator (σ) and containment operators ($\sqsupset$ and $\sqsubset$), where containment operators are used to model successive element selection. For example, `sec//p` is modeled as $\sigma_{\text{t=node, n='p'}}(C) \sqsubset \sigma_{\text{t=node, n='sec'}}(C)$.

Terms Besides the strict selection of terms as defined in the selection operator (e.g., $\sigma_{\text{t=term, n='perfect'}}(C)$) the selection criterion could be more "vague". For example, we could search for elements that have the same root (stem) or similar semantic meaning (synonyms) in query and collection (similar to [1]). Furthermore, the term selection operator can be extended to support "vague selection" that assigns higher scores to terms exactly matching query terms than to terms that are synonyms or have the same stem.

Term modifiers Some IR systems enable users to say if some of the terms in the search query are more important (denoted with, e.g., '+') than the others, or enable the specification of term importance (weight) in the form of a number ranging from 0 to 1 assigned to the query term, whose value depicts the importance of a term to the user. For example, in [18] we used scale operator to increase the score of regions containing '+' terms, but other solutions are possible.

Search and answer elements Due to structured document heterogeneity (e.g., different XML Schemas or DTDs), different documents may have different element names for the same concept and single document can have more than one structured description [20]. Furthermore, the user might not be certain where he would like to search for information or might not know the element names in the collection. In both cases, he should be able to give some hints to the system where he would like to perform the search and what he would like to see as an answer. As a consequence, the structured IR system needs something like synonyms for tag names, although the "synonym search" process should recognize the structural elements (i.e., tag names) that denote similar concepts in structured documents instantiated from different structured schemas (e.g., 'section' and 'sec'). In such a way, the user will be able to query heterogeneous

collections with different tag notations by specifying a single query, or query a single collection without knowing its structure. The equivalence tags introduced in INEX [24] and context resemblance measures introduced by Carmel et al. [5] are first steps towards this kind of "structural synonyms". Thus, the SRA operators for (strict) search and answer elements specification, modeled with $\sigma_{\text{t=node}}$, $\sqsupset$, and $\sqsubset$, should be extended to support vague matching.

4.2 Element relevance score computation

Element relevance score computation is used to define how well a specific element is described by a query term. It can be considered as an adaptation of the document relevance score computation in the flat file retrieval where relevance score is computed for a term in an arbitrary XML element instead of a predefined document. In SRA, element relevance score computation is defined using the $\sqsupset_p$ operator for containment and the $\not\sqsupset_p$ operator for non-containment relation ('$-$' in front of a term in the query). Thus, in the example query we specify the relevance score computation for 'p' element and term 'perfect' as $\sigma_{\text{t=element, n='p'}} \sqsupset_p \sigma_{\text{t=term, n='perfect'}}$.

Terms In flat file IR systems, the relevance score of a document is determined by counting the number of occurences of a term in a document (term frequency), possibly with additional statistical information like collection frequency or inverse document frequency of a term. In [18] a basic statistical language model (LM) is used to implement the relevance score computation for the region (element) r_1 (with element name n_1) with respect to the term regions in R_2 that consists of regions that all have the same region name attribute n_2.

$$f_{\sqsupset}^{LM}(r_1, R_2) = p_1 \cdot (\lambda \frac{\sum_{r_2 \in R_2 | r_2 \prec r_1} p_2}{size(r_1)} + (1-\lambda) \frac{|R_2|}{size(Root)}) \quad (1)$$

Here $|R|$ is the number of regions in the region set R, $size(r)$ is the size of the region r (i.e., number of terms in a region), *Root* is the XML collection root, and λ is the smoothing parameter ($0 \leq \lambda \leq 1$).

We can instantiate different retrieval models for element score computation. For example, we give the instantiation of function $f_{\sqsupset}(r_1, R_2)$ for Okapi model [22].

$$f_{\sqsupset}^{\text{Okapi}}(r_1, R_2) = p_1 \cdot$$
$$ln \frac{|\{r \in C | n = n_1\}| - |\{r \in C | n = n_1 \wedge \exists r_2 \in R_2 \wedge r_2 \prec r_1\}| + 0.5}{|\{r \in C | n = n_1 \wedge \exists r_2 \in R_2 \wedge r_2 \prec r_1\}| + 0.5}$$
$$\cdot \frac{(k_1+1) \cdot \sum_{r_2 \in R_2 | r_2 \prec r_1} p_2}{k_1((1-b) + b \frac{size(r_1)}{avg\_size(n_1)}) + \sum_{r_2 \in R_2 | r_2 \prec r_1} p_2} \quad (2)$$

Here C is the set of all regions in the collection, $avg\_size(n)$ is the average size of regions with region attribute name n in the collection, and k_1 and b are parameters in the Okapi model. Similarly, we can instantiate other models, such as tf.idf-based models (see e.g., [5]).

If we assume that $f_{\sqsupset}(r_1, R_2) = p_1 \cdot g(r_1, R_2)$, and function g is normalized, we can define score for $\not\sqsupset_p$ operator as, e.g., $f_{\not\sqsupset}(r_1, R_2) = p_1 \cdot (1 - g(r_1, R_2))$. Which element relevance score computation model is the best for XML IR is one of the main questions in our research.

Phrases So far most of the extensions of the basic retrieval models with phrase modeling were not so successful in flat file as well as in XML IR systems (see, e.g., [14, 18]). It is an open question if it can be improved. The element relevance score computation for phrases in our model is defined similar as for single terms, where the region R_2 would be the region that contains the phrase (for details see [18]).

4.3 Element score combination

Similar to flat file retrieval systems, structured retrieval systems may search for different terms in the same element which can be logically combined, using conjunction or disjunction (i.e., "and" and "or"). Moreover, information search in different elements can be also combined via logical expressions (see NEXI query examples in [11]). To model this we identified the element score combination aspect. In score region algebra, score combination is expressed using $\sqcap_p$ for "and" (see example SRA expression) and $\sqcup_p$ for "or" combination. The abstract score combination functions in these operators could be defined as a sum or max ($\oplus$) and product or min ($\otimes$) as in [18]. Other instantiation of abstract operators are also possible, e.g., $p_1 \oplus p_2 = 1 - (1 - p_1) \cdot (1 - p_2)$.

4.4 Element score propagation

Unlike in flat file IR systems, in structured retrieval systems search elements are not necessarily answer elements at the same time. This can be seen in our example NEXI query, where the search element is 'p' and the answer element is 'sec'. Consequently, it is important to specify the score propagation aspect, as recognized by Fuhr et al. [9] and Grabs and Shek [12]. In the example SRA expression the upwards score propagation, i.e., score propagation to the containing region, is specified using $\blacktriangleright$ operator. Furthermore, if the user has more precise knowledge of the collection, he can define more advanced queries which combine multiple search elements and answer element, like:

```
//sec[about(., flower)]//p[about(., perfect bouquet)]
```

For such queries we must specify how scores are propagated to the answer elements ('p') from the search element ('sec'). This downwards propagation, i.e., propagation to the contained region, is specified in SRA using $\blacktriangleleft$ operator. For example, in [15] for $\blacktriangleright$ operator different implementations are used, like min, max, average, weighted sum, and it is shown that it makes a difference in the retrieval results.

4.5 Operator properties

By analyzing our NEXI query example, it can easily be deduced that the given SRA expression is not the only expression that can be delivered from the NEXI query. We can express the same information need in SRA as:

$$\sigma_{t=node,n='sec'}(C) \blacktriangleright ((\sigma_{t=node,n='p'}(C)$$
$$\sqsupset_p \ \sigma_{t=term,n='perfect'}(C)) \sqsupset_p \sigma_{t=term,n='bouquet'}(C))$$

where "and" combination is replaced with successive element relevance score computation operators. The question is will this SRA expression give the same results as the original one? If we follow the definition of operators in [18], where $f_{\sqsupset}(r_1, R_2) = p_1 \cdot g((s_1, e_1, n_1, t_1), R_2)$ and $\otimes$ is modeled as a product, and the score for region r_1 is 1, this is true. Thus, for the expression $(R_1 \sqsupset_p R_2) \sqcap_p (R_1 \sqsupset_p R_3)$ and $(R_1 \sqsupset_p R_2) \sqsupset_p R_2$ region scores will be equal:

$$(p_1 \cdot f_{\sqsupset}(r_1, R_2)) \cdot (p_1 \cdot f_{\sqsupset}(r_1, R_3) = (p_1 \cdot f_{\sqsupset}(r_1, R_3)) \cdot f_{\sqsupset}(r_1, R_2)$$

Similarly, if we define $\sqcap_p$ and $\sqcup_p$ as product and sum, the operator $\sqcap_p$ distributes over the operator $\sqcup_p$, since '$\cdot$' distributes over '$+$'. Vice versa is not the case. Additionally,

there are similar conditional properties of score operators which can be of interest for the optimization and which depend on the retrieval model specification (see [17]).

The properties of scoring operators in SRA can be studied from two perspectives. One is to test their validity in theory for different implementations of scoring operators (as we partially did in [17]) and the other is to test the consistency of the scoring results with different query plans considering also the issues of numeric stability problems in the implementation. The overall goal of using operator properties is to enable query optimization without the significant effect on system effectiveness.

4.6 Evaluation of the system

Apart from the validity of the SRA operator properties, which can be tested in theory (e.g., [17]) and by using arbitrary XML collections, we will use the framework provided by the INitiative for the Evaluation of XML Retrieval (INEX)[1] to test the effectiveness of our XML IR approach. INEX offers an XML collection of IEEE scientific articles with a size of more than 500MB, consisting of various research fields from the years 1995 to 2002, and metrics used for the evaluation of results. Each year participants are asked to create topics for the content-only (CO) task, i.e., search without imposing structural constraints, and for the content-and-structure (CAS) task, i.e., search that considers both, structure and content. Additionally participants are asked to perform two dimensional relevance assessments based on element exhaustivity and specificity. Starting from the last year heterogenous collection search is also supported in INEX.

5. CONCLUSION

A modern structured IR system should allow for a wide range of user queries. To satisfy the user information need, the structured IR system must be flexible and provide the best term and element matching, element relevance score computation, element score combination, and element score propagation mechanisms. We argue that the score region algebra is the right framework for developing a transparent structured IR system for testing different instantiations of the structured IR aspects and discovering the strong and weak points of structured retrieval models, as ones developed for XML IR, regardless of the query language used and the physical implementation.

6. REFERENCES

[1] S. Amer-Yahia, C. Botev, and J. Shanmugasundaram. TeXQuery: A Full-Text Search Extension to XQuery. In *Proceedings of the 13th WWW Conference*, 2004.

[2] R. Baeza-Yates and G. Navarro. Proximal Nodes: A Model to Query Document Databases by Content and Structure. In *Proceeding of ACM TOIS*, 1997.

[3] F.J. Burkowski. Retrieval Activities in a Database Consisting of Heterogeneous Collections of Structured Texts. In *Proceedings of the 15th ACM SIGIR Conference*, 1992.

[4] S. Buxton and M. Rys. XQuery and XPath Full-Text Requirements. Technical report, W3C, 2003.

[5] D. Carmel, Y.S. Marek, M. Mandelbrod, Y. Mass, and A. Soffer. Searching XML Documents via XML Fragments. In *Proceedings of the 26th ACM SIGIR Conference*, 2003.

[6] C.L.A. Clarke, G.V. Cormack, and F.J. Burkowski. An Algebra for Structured Text Search and a Framework for its Implementation. *The Computer Journal*, 38(1), 1995.

[7] M. Consens and T. Milo. Algebras for Querying Text Regions. In *Proceedings of the ACM PODS*, 1995.

[8] N. Fuhr. Models for integrated information retrieval and database systems. *IEEE data engineering bulletin*, 19(1), 1996.

[9] N. Fuhr and K. Großjohann. XIRQL: A query language for Information Retrieval in XML Documents. In *Proceedings of the 24th ACM SIGIR Conference*, 2001.

[10] N. Fuhr and K. Großjohann. XIRQL: An XML Query Language Based on Information Retrieval Concepts. *ACM TOIS*, 22(2):313–356, 2004.

[11] N. Fuhr, M. Lalmas, and S. Malik, editors. *Proceedings of the 2nd INEX Workshop*, ERCIM Publications, 2004.

[12] T. Grabs and H.-J. Shek. Generating Vector Spaces On-the-fly for Flexible XML Retrieval. In *Proceedings of the XML and IR Workshop – 25th ACM SIGIR Conference*, 2002.

[13] J. Jaakkola and P. Kilpelainen. Nested Text-Region Algebra. Technical Report C-1999-2, Department of Computer Science, University of Helsinki, 1999.

[14] M. Jiang, E. Jensen, S. Beitzel, and S. Argamon. Choosing the Right Bigrams for Information Retrieval. In *Proceeding of the Meeting of the International Federation of Classification Societies*, 2004.

[15] J. List, V. Mihajlović, A. de Vries, G. Ramirez, and D. Hiemstra. The TIJAH XML-IR System at INEX 2003. In *Proceedings of the 2nd INEX Workshop*, ERCIM Publications, 2004.

[16] K. Masuda, T. Ninomiya, Y. Miyao, T. Ohta, and J. Tsujii. A Robust Retrieval Engine for Proximal and Structural Search. In *Proceedings of HLT-NAACL*, 2003.

[17] V. Mihajlović, D. Hiemstra, H. E. Blok, and P. M. G. Apers. An XML-IR-DB Sandwich: Is it Better with an Algebra in Between? In *Proceedings of the SIGIR Workshop on Information Retrieval and Databases*, 2004.

[18] V. Mihajlović, G. Ramírez, A. P. de Vries, D. Hiemstra, and H. E. Blok. TIJAH at INEX 2004: Modeling Phrases and Relevance Feedback. In *Proceedings of the 3rd INEX Workshop, LNCS 3493, Springer*, 2005.

[19] R.C. Miller. *Light-Weight Structured Text Processing*. PhD thesis, Computer Science Department, Carnegie-Mellon University, 2002.

[20] P. Ogilvie. Retrieval Using Structure for Question Answering. In *The First TDM Workshop on XML Databases and Information Retrieval*, 2004.

[21] J. Pehcevski, J. A. Thom, and A-M. Vercoustre. RMIT INEX Experiments: XML Retrieval Using Lucy/eXist. In *Proceedings of the 2nd INEX Workshop*, ERCIM Publications, 2004.

[22] S. E. Robertson and S. Walker. Some Simple Effective Approximations to the 2-Poisson Model for Probabilistic Weighted Retrieval. In *Proceedings of 17th ACM SIGIR Conference*, 1994.

[23] A. Salminen and F.W. Tompa. PAT Expressions: An Algebra for Text Search. In *Proceedings of the 2nd COMPLEX Conference*, 1992.

[24] B. Sigurbjörnsson, B. Larsen, M. Lalmas, and S. Malik. INEX04 Guidelines for Topic Development. In *Proceedings of the 3rd INEX Workshop, LNCS 3493, Springer*, ERCIM Publications, 2005.

[25] A. Trotman and R. A. O'Keefe. The Simplest Query Language That Could Possibly Work. In *Proceedings of the 2nd INEX Workshop*, ERCIM Publications, 2004.

[26] D. Tsichritzis and A. Klug. The ANSI/X3/SPARC DBMS framework report of the study group on database management systems. *Information systems*, 3, 1978.

[1] http://inex.is.informatik.uni-duisburg.de:2005.