# Score Region Algebra: Building a Transparent XML-IR Database

Vojkan Mihajlović Henk Ernst Blok Djoerd Hiemstra Peter M.G. Apers
CTIT, University of Twente
P.O. Box 217, 7500AE Enschede, The Netherlands
{v.mihajlovic, h.e.blok, d.hiemstra, p.m.g.apers}@utwente.nl

## ABSTRACT

A unified database framework that will enable better comprehension of ranked XML retrieval is still a challenge in the XML database field. We propose a logical algebra, named score region algebra, that enables transparent specification of information retrieval (IR) models for XML databases. The transparency is achieved by a possibility to instantiate various retrieval models, using abstract score functions within algebra operators, while logical query plan and operator definitions remain unchanged. Our algebra operators model three important aspects of XML retrieval: element relevance score computation, element score propagation, and element score combination. To illustrate the usefulness of our algebra we instantiate four different, well known IR scoring models, and combine them with different score propagation and combination functions. We implemented the algebra operators in a prototype system on top of a low-level database kernel. The evaluation of the system is performed on a collection of IEEE articles in XML format provided by INEX. We argue that state of the art XML IR models can be transparently implemented using our score region algebra framework on top of any low-level physical database engine or existing RDBMS, allowing a more systematic investigation of retrieval model behavior.

**Categories and Subject Descriptors:** H.2.3 [Languages]: Query languages; H.3.3 [Information search and retrieval]: Retrieval models;

**General Terms:** Performance, Design, Experimentation, Verification, Theory.

**Keywords:** information retrieval, databases, structured documents, XML, region algebra.

## 1. INTRODUCTION

XML was initially developed as a standard for storing, carrying and exchanging data. With the rapid growth of data stored in XML format, ranked information retrieval (IR) from XML collections becomes a vital requirement. Several systems have been developed in recent years that address this requirement.

### 1.1 From flat file to XML IR

An important class of XML IR systems is based on traditional (flat file) information retrieval methods (e.g., [21, 23, 28]) that represent a document as a "bag of words" [22] and where in most cases inverted file structures provide the basis for implementing a retrieval system. Although these systems are faster and simpler than any DBMS, they have important drawbacks when XML IR is being considered. Out of many properties of traditional IR systems, we discuss only the following four:

- traditional IR systems lack the notion of *data independence* [12]: any change in what constitutes a document, or any change in document structure or used retrieval model, will lead to system developers needing to change major parts of the retrieval system
- traditional IR systems are developed for a simple query language, consisting in most cases of a set of terms
- most traditional IR systems are retrieval model specific, i.e., the retrieval model (or a small class of similar retrieval models) is hard-coded in the system
- traditional IR systems are developed for a specific storage structure, in most cases inverted files.

Although the concept of data independence was not a too big hurdle for the development of flat-file IR [19, 36], we argue that it is important for the development of XML IR. Unlike in the vast majority of flat file IR systems, where *documents* were the only units in which the user would search for information or which user would obtain as answer from the system, in XML IR the main focus is on nested XML *elements*. Furthermore, the notion of document is blurred as the whole collection of XML documents can be considered as one huge XML document with an artificial root element.

In XML IR, the user can specify not only his information need, but also where to search for information. This leads to the introduction of the specification of search elements in query languages used for the ranked retrieval on XML. Most of the XML IR query languages use the W3C query languages (XQuery [3] or XPath [8]) as a starting point and extend them with IR-like search expressions. Typical examples are full-text search extension of XQuery [1], Narrowed Extended XPath (NEXI) [33], and an extension of XQL, one of the predecessors of XPath, named XIRQL [13].

Furthermore, hierarchical organization of XML documents enables the distinction between *search elements* and *answer* *elements* in XML IR, where both element types are not predefined as in flat file IR (documents). Search elements are elements where the user searches for specific information, while answer elements are elements that the user would like to obtain as an answer to a query. We illustrate an information retrieval search over XML documents for the user information need: "I would like to find sections addressing language models in an article that has an abstract discussing information retrieval or a probabilistic database". This information need can be expressed in NEXI [33] (which uses a subset of XPath and extends XPath with an *about* function for ranked retrieval and is adopted as an official query language in the INitiative for the Evaluation of XML Retrieval (INEX) [15]) as:

```
//article[about(.//abs, information retrieval)
  or about(.//abs, probabilistic database)]
  //sec[about(., language model)]
```

The example above introduces several additional query capabilities that are not recognized in flat file retrieval:

- The search for information can be performed in arbitrary part of the XML document (or collection of XML documents) denoted with tags, termed search elements. In our example, search elements are 'article', 'abs' and 'sec'.
- The search element is not necessarily an answer element at the same time. Only 'sec' is a search and answer element at the same time in the example query (denoted with '.').
- The user can perform searches in different XML elements and later combine them to get the answer element. In our example information search is performed in 'abs' elements using the terms "information retrieval" and "probabilistic database" which are combined in an *OR* expression.

The central part of any retrieval system is the retrieval model used. Although many approaches exist for XML information retrieval, most of the XML IR approaches are based on flat file tf.idf-like approaches [15]. For XML IR, the retrieval model has to incorporate additional aspects. We identify three aspects that a flexible XML retrieval system should provide to support XML-IR queries like the one above:

- element relevance score computation,
- element score combination, and
- element score propagation.

We use the term element *score* to denote the value that describes the estimated relevance of an element.

The first two aspects are inherited from the flat file IR model, although the second aspect can also denote the combination of scores for different search elements (see Section 2.1). The third one, sometimes called "augmentation" is XML specific and is recognized in the work of Fuhr and Großjohann [13] and Grabs and Shek [18], where the authors specified weighting factors for upwards propagation of scores in an XML tree representation. However, we think that the concept of downwards propagation should also be considered as, e.g., in our example we have to propagate relevance scores from the 'article' element to the 'sec' answer element.

## 1.2 Towards the database approach

Besides numerous query languages, different physical implementations for ranked retrieval on XML have been proposed, ranging from modified inverted file structures [17, 28] to full database implementations [11, 16]. For each indexing structure chosen for physical implementation, special algorithms need to be derived to enable fast execution of XML IR primitives. Whatever the primitives are, these algorithms will be dependent on XML storage structure.

Therefore, besides the adaptation of traditional IR systems, we can identify the other class of XML IR systems based on relational database technology. XML-IR database systems use well established database operators and thirty years' experience in relational database management systems (RDBMS). RDBMS can be either enriched with an IR-like front end (loosely-coupled) [29] or tightly-coupled with IR search primitives [11, 13, 14], as recognized in [35]. Although relational systems are prevalent in the manipulation of documents structured as relations, they have difficulty handling nested structures such as XML. This is especially the case with handling containment relations (i.e., containment joins [24]), which are one of the basic operations used in information retrieval. That is why most of XML-IR database systems have so far been loosely-coupled systems. Recently, numerous enhancements have been proposed for handling containment relations, such as the staircase join [20] and the multipredicate merge join [37], proving that relational technology can handle efficiently queries over XML data, including the containment queries.

The main characteristic of the database approach is a strong separation between conceptual, logical and physical levels [34]. By using different data models at each of those levels, *data abstraction* is provided. For XML-IR systems, following this separation in levels gives another, additional advantage over flat file IR systems: by choosing the appropriate level of abstraction for each database level, the development of scoring techniques, handling structural information, is simplified, and kept *transparent* for the rest of the system design, making it flexible with respect to the query language and physical implementation used. Furthermore, the reasoning that can be made at 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 at the physical level we are able to generate different logical query plans, speeding up the execution and lowering the memory requirements for query execution at the physical level.

Therefore, we identify the logical level as the central level that should provide the transparency considering XML-IR database systems. Unlike in the approach of Amer-Yahia et al. [1], we want to integrate relevance score computations within the algebra. In [1] authors aimed to support a full-text search extension to XQuery (based on full-text query specification [6]), assuming that the scoring method is retrieval model implementation dependent, abstracting in that way from the problems of the XML IR database integration and of element score propagation and combination. Our approach is closer to the approach of Fuhr et al. [13, 14], where the authors developed an XML IR path-based algebra and an XML IR query language named XIRQL. Unlike Fuhr et al., we base our algebra on containment relations among XML elements and not on the paths to XML elements, following the region algebra approaches [2, 5, 9, 31].

The basic idea behind region algebra approaches is the representation of text documents as a set of *regions* (sometimes termed extents [5]), where each region is defined by its start and end positions. The aim of the region algebra approaches is modeling search in (semi-)structured documents using containment and set operators, The earliest region algebra approaches were a PAT system presented in [31], and the work of Burkowski [5] and Clarke et al. [9]. These approaches were later extended to support new operators, such as positional inclusion and direct inclusion, by Navarro and Baeza-Yates [2] and Consens and Milo [10].

To model three basic XML IR aspects, namely element relevance score computation, element score propagation, and element score combination, at the logical level of a database, we extended the original region algebra approaches with scoring operators and termed this new algebra *Score Region Algebra (SRA)*. The overall goal of the score region algebra is to transparently model different aspects of query formulation and search and answer element specification, and to support different retrieval models with different parameter specification applied to XML.

### 1.3 Outline

This paper is organized as follows. In the next section, we specify score region algebra used to define a framework for flexible and transparent XML ranked retrieval and illustrate how different retrieval models can be instantiated in score region algebra based on identified retrieval aspects. We present the experimental setup, including a description of our prototype system, and the evaluation results in Section 3. The paper is concluded with a short discussion and directions for future research.

## 2. TRANSPARENT LOGICAL ALGEBRA

In Section 2.1 we specify in more detail the three key aspects of XML IR. Next, in Section 2.2 we present four retrieval models that we use. In Section 2.3 we define our score region algebra (SRA) for use at the logical level of databases to enable transparent specification of retrieval models.

### 2.1 Three aspects of XML IR

In XML IR query processing, three key aspects are identified: element relevance score computation, element score combination, and element score propagation. To discuss these aspects, we first identify the three basic entities in a typical NEXI query expression. Recall our example query from Section 1:

```
//article[about(.//abs, information retrieval)
  or about(.//abs, probabilistic database)]
  //sec[about(., language model)]
```

**terms** In our example query, we can identify six different query terms: ‘information’, ‘retrieval’, ‘probabilistic’, ‘database’, ‘language’, and ‘model’.

**answer elements** We can distinguish three different structural constraints[1] (i.e., element or tag name specifications): ‘article’, ‘abs’, and ‘sec’. According to the NEXI specification, the answer element is the last element that has an *about* predicate specified, i.e., in our example the ‘sec’ element.

**search elements** All other elements which are not answer elements are called search elements. Within the search elements, we distinguish two different kinds: the lowest element inside an about, i.e., the last element in the *about* path expression, and other elements. There exists one special case where the lowest search element inside an about is ‘.’, that refers to the search element that directly precede the about clause. Thus, it can happen that the search element is the answer element at the same time (e.g., the ’sec’ element in our example query).

In our example query, the search elements are: ‘article’, ‘abs’, and ‘sec’. The lowest search elements inside abouts are ‘abs’ and ‘.’, i.e., ‘sec’.

#### 2.1.1 Element (relevance) score computation

The first task in XML ranked retrieval is to produce the relevance score for all nodes in the XML collection matching the lowest search element in each about. Following the NEXI specification, we consider each term in isolation per lowest search element in the respective about clauses. In this section, we describe how to compute a score per search element-term pair. The combination of these scores is discussed in the next section.

In the example query, relevance scores have to be determined for the ‘abs’ elements with respect to the four terms: ‘information’, ‘retrieval’, ‘probabilistic’, and ‘database’. For the ‘sec’ element we have to compute scores for two terms: ‘language’ and ‘model’. By employing a retrieval formula that specifies the relevance of an XML element given a query term (see, e.g., Equation 1 in Section 2.2), we can compute the ‘abs’ and ‘sec’ element relevance scores per term.

#### 2.1.2 Element score combination

As about clauses can have more than one query term typically and as scores are computed on a per term basis, those scores have to be combined on a per lowest search element basis. Depending on the preferred behavior, this can be seen as an OR or an AND combination. In our example query, the first two *about* clauses contain two terms each. We can interpret this either as ‘abs’ should be about ‘information’ AND ‘retrieval’ or ‘abs’ should be about ‘information’ OR ‘retrieval’. It is up to the implementer of a specific model to make a choice, unless the user explicitly specified AND or OR in the NEXI expression. Additionally, NEXI allows for explicit specification of AND and OR combination of *about* path expressions, as can be seen in our example query.

In a NEXI path expression that expresses an ancestor-descendant relationship, in case of consecutive search elements, the ancestor search element can have multiple matching descendant search elements. Propagating scores between ancestors and descendants is called score propagation. This aspect is discussed in the next section.

#### 2.1.3 Element score propagation

Although it might seem unnecessary in our example as we have used it until now, the necessity of the propagation to the common ancestor element can be seen in the case of the following, different NEXI query:

[1]Here we assume a strict interpretation of structural constraints, although element names can be considered as a hint for the retrieval system. See, e.g., vague content-and-structure (VCAS) queries in INEX [15].

```
//article[about(.//abs, information retrieval) and
  about(.//kwd, probabilistic database)]
```

To perform an AND-like combination of 'abs' (abstract) and 'kwd' (keyword) elements in this case, we need to propagate the scores to the common ancestor 'article' element.

We can define the element score propagation as the translation of scores to the ancestor or descendant elements where these scores can be combined based on the type of combination explicitly specified in NEXI. We can distinguish between two types of score propagation: upwards and downwards score propagation. For our original example query, the score should be propagated upwards from 'abs' to 'article' elements that matches the upwards score propagation. This scenario happens if the NEXI predicate has logically combined multiple *abouts*, as in our NEXI query example from Section 1:

```
//article[about(.//abs, information retrieval) or
  (.//abs, probabilistic database)]
```

The second scenario is when the *about* clause contains at least one element selection. In the example with two element selections in the *about*:

```
//article[about(.//sec//p, information retrieval)]
```

scores need to be propagated from 'p' (paragraph) elements to 'sec' (section) elements and then to 'article' elements.

In the case of downwards propagation scores should be propagated from search elements to the contained search or answer elements. In our example in Section 1, the scores are propagated from the search elements 'article' to the 'sec' elements which are answer elements. If the 'sec' element is not the answer element, i.e., if we have `//sec//p[about(., language model)]` instead of `//sec[about(., language model)]` in our query example, the scores should be further propagated downwards from 'sec' to 'p' elements.

## 2.2 Retrieval models

In this section we describe four state of the art retrieval models to test the transparency of our approach: statistical language models [22], where we use two types, language models with and without smoothing, the Okapi (INQUERY) model [7, 30], the tf.idf model [32], and the Garden Point XML (GPX) model [16][2].

In these approaches, the relevance score of a document is based on the fact that documents that contain more occurrences of a term, i.e., have higher *term frequency*, are more important to the user. Additionally, to incorporate the significance of a term for ranked retrieval, these models also include background statistics. Background statistics are based on a number of terms in the whole collection, i.e., *collection frequency*, or number of documents containing a term, i.e., *document frequency.*

**Language model** In the language model approach (with smoothing), the relevance score of the document ($doc$) given the query terms ($tm_i$, $i = 1, 2, ..., n$, $tm_i \in q$) can be computed as:

$$S(doc|q) = \prod_{i=1}^{n} (\lambda \frac{tc(tm_i, doc)}{len(doc)} + (1-\lambda) \frac{tc(tm_i, col)}{len(col)}) \quad (1)$$

where $n$ is the number of terms ($tm_i$) in the query ($q$), $tc(tm_i, doc)$ denote the number of occurrences of a term $tm_i$ in the document $doc$, $len(doc)$ is the length of the document (i.e., the number of terms it contains), and $\lambda$ is a smoothing parameter (ranging from 0 to 1) that specifies the relative influence of the foreground and background statistics in the final document ranking score computation. The language model approach without smoothing is just a special case of language modeling approach where $\lambda = 1$.

**Okapi** The Okapi (INQUERY) retrieval model is based on the BM25 algorithm [30]:

$$S(doc|q) = \sum_{i=1}^{n} (ln \frac{N - dc(tm_i) + 0.5}{dc(tm_i) + 0.5} \cdot$$

$$\frac{(k_1 + 1) \cdot tc(tm_i, doc)}{k_1((1-b) + b\frac{len(doc)}{avdl}) + tc(tm_i, doc)} \cdot \frac{(k_3 + 1) \cdot tc(tm_i, q)}{k_3 + tc(tm_i, q)}) \quad (2)$$

where $N$ is the total number of documents in the collection, $dc(tm_i)$ is the number of documents in the collection that contain term $tm_i$, $avdl$ is the average document length, $tc(tm_i, q)$ is the number of terms $tm_i$ in the query $q$, and $k_1$ (between 1.0 and 2.0), $k_3$ (between 0 and 1000), and $b$ ($\approx 0.75$) are constants.

In the INQUERY system, several functions are implemented to combine the scores of single term relevance score computations, such as sum, multiplication, probabilistic interpretation (see [7]).

**tf.idf** For the tf.idf approach, we used the basic tf.idf formula specified in [32]:

$$S(doc|q) = \sum_{i=1}^{n} tc(tm_i, doc) \cdot ln \frac{N}{dc(tm_i, doc)} \quad (3)$$

The parameters of the formula are the same as in Equations 1 and 2.

**GPX** Finally, the basic formula in the GPX approach defines the relevance score of the document with respect to the query terms as[3]:

$$S(doc|q) = A^{n-1} \sum_{i=1}^{n} \frac{tc(tm_i, el)}{tc(tm_i, col)} \quad (4)$$

where A is the parameter with a value between 3 and 10.

In the next section, after introducing score region algebra, we explain how these IR models can be applied to SRA.

## 2.3 Score Region Algebra

The application of the idea of text regions to XML documents is straightforward. Each XML document can be seen as a sequence of tokens, e.g., start tags, end tags, terms, etc., where each token can be indexed to model the XML tree structure (see, e.g. [26]), represented as a set of text regions. To be able to represent XML properly, the definition of a region in score region algebra is richer than in previous region algebra approaches. In the specification of our SRA data model we distinguish between different node types in XML documents in order to provide a uniform platform for defining the region algebra operators. Furthermore, we enrich the original model with a region score attribute and introduce a number of operators for score manipulation. The logical data model of SRA is based on *region sets*, where each region is defined below.

DEFINITION 1. *The SRA data model is defined on the domain $R$ which represents a set of region tuples. Region tuple $r$ ($r \in R$), $r = (s, e, n, t, p)$, is defined by these five*

[2]Although this is not a well known retrieval model we have chosen it as it is among the most effective ones presented at INEX 2004 workshop: http://inex.is.informatik.uni-duisburg.de:2004.

[3]Note that the original formula defines score computation for leaf XML elements ($el$) instead of documents [16].

**Table 1: Score region algebra operators.**

| Operator | Operator definition |
|---|---|
| $\sigma_{n=name,t=type}(R)$ | $\{r \mid r \in R \wedge n = name \wedge t = type\}$ |
| $\sigma_{\diamond num}(R_1)$ | $\{r_1 \mid r_1 \in R_1 \wedge \exists r_2 \in C \wedge t_2 = term \wedge r_2 \prec r_1 \wedge n_2 \diamond num\}$, where $\diamond \in \{=,<,>,\leq,\geq\}$ |
| $R_1 \sqsupset R_2$ | $\{r_1 \mid r_1 \in R_1 \wedge \exists r_2 \in R_2 \wedge r_2 \prec r_1\}$ |
| $R_1 \sqsubset R_2$ | $\{r_1 \mid r_1 \in R_1 \wedge \exists r_2 \in R_2 \wedge r_1 \prec r_2\}$ |
| $R_1 \sqsupset_p R_2$ | $\{r \mid r_1 \in R_1 \wedge (s,e,n,t) := (s_1,e_1,n_1,t_1) \wedge t_1 = node \wedge t_2 = term \wedge p := f_{\sqsupset}(r_1,R_2)\}$ |
| $R_1 \blacktriangleright R_2$ | $\{r \mid r_1 \in R_1 \wedge (s,e,n,t) := (s_1,e_1,n_1,t_1) \wedge t_1 = node \wedge t_2 = node \wedge p := f_{\blacktriangleright}(r_1,R_2)\}$ |
| $R_1 \blacktriangleleft R_2$ | $\{r \mid r_1 \in R_1 \wedge (s,e,n,t) := (s_1,e_1,n_1,t_1) \wedge t_1 = node \wedge t_2 = node \wedge p := f_{\blacktriangleleft}(r_1,R_2)\}$ |
| $R_1 \sqcap_p R_2$ | $\{r \mid r_1 \in R_1 \wedge r_2 \in R_2 \wedge (s_1,e_1,n_1,t_1) = (s_2,e_2,n_2,t_2) \wedge (s,e,n,t) := (s_1,e_1,n_1,t_1) \wedge p := p_1 \otimes p_2\}$ |
| $R_1 \sqcup_p R_2$ | $\{r \mid r_1 \in R_1 \wedge r_2 \in R_2 \wedge ((s,e,n,t) := (s_1,e_1,n_1,t_1) \vee (s,e,n,t) := (s_2,e_2,n_2,t_2)) \wedge p := p_1 \oplus p_2\}$ |

*attributes: region start attribute - s, region end attribute - e , region name attribute - n, region type attribute - t, and region score attribute - p. Region start and end attributes must satisfy ordering constraints ($e_i \geq s_i$). If $\prec$ denotes the equivalence $r_i \prec r_j \Leftrightarrow s_j < s_i \leq e_i < e_j$, we can state that for two arbitrary regions in SRA it is either $r_i \prec r_j$, $r_i \equiv r_j$, or $r_j \prec r_i$. Furthermore, each region in the SRA data model is unique.*

The semantics of region start and region end attributes are the same as in other region algebra approaches: they denote the bounds of a region. The region name attribute is used to denote node names, content words, element attribute names, element attribute values, etc. To distinguish between different name "roles" in XML we used the region type attribute. We use *node* for the element node in XML, *text* for the text node, *term* for the term present in a text node, etc. Finally, the region score information item is used to specify the relevance score of a region with respect to a given query.

The aim of SRA is to supports ranked retrieval as a part of the algebra, and not as a side-effect, which distinguishes it from other region algebra proposals that include ranked retrieval (e.g., [5]). The basic SRA operators are defined in Table 1. In the specification of region algebra operators we use $R_i$ ($i = 1, 2, ...$) to denote the region sets, their corresponding non-capitals to denote regions in these region sets ($r_i$), and corresponding indexed non-capitals to denote region attributes ($s_i, e_i, n_i, t_i, p_i$). With $C$ we denote the set of all regions in the collection and with *Root* we denote the (artificial) root element for the whole collection.

The operators in SRA take one or two region sets as operands and produce a region set as result. The first four operators enable Boolean selection of regions based on their attributes or containment relations. The selection operator, $\sigma$, has two variants. The first one ($\sigma_{n=name,t=type}(R)$) specifies the selection based on name and type attributes[4]. The second selection operator selects regions that contain a term region in which content (casted to a number) is equal, greater or equal, less or equal, greater or less than the number specified (*num*). The last two operators select regions based on their containment relations, i.e., regions that contain other regions ($\sqsupset$), or regions that are contained in other regions ($\sqsubset$).

The other five operators specify score manipulation among regions. To enable the instantiation of different retrieval models, they are defined using three abstract scoring functions: $f_{\sqsupset}$, $f_{\blacktriangleright}$, and $f_{\blacktriangleleft}$, and two abstract operators: $\otimes$ and $\oplus$. These abstract functions and abstract operators model three aspects of XML IR. They are specified based on auxiliary functions that count the number of regions in the region set $R$, denoted with $|R|$, compute the size of the region $r$: $size(r) = e - s - 1$, and compute the average size of the regions with the region name $n$ in the collection, denoted with $avg\_size(n)$.

[4]Leaving the selection criterion for one of the attributes unspecified corresponds to a wild-card, i.e., $\sigma_{t=node}$ will select all regions that have an XML element node type, regardless of their name attribute.

### 2.3.1 Element (relevance) score computation

Operator $\sqsupset_p$ models element relevance score computation, i.e., the concept that the search elements (regions in the first operand) should contain the term (region). Therefore, the function $f_{\sqsupset}(r_1, R_2)$, applied to a region $r_1$ and region set $R_2$, should result in the numeric value that specifies the relevance of the region (element) $r_1$ given the term regions in $R_2$ that it contains. Following the specification of four models in the previous section, we have four specifications of this abstract function. We assume that the default score value for element and term regions in all models is 1.0, except for the basic GPX model where element regions have the default score value of 0.0.

The language model ($LM$) can be instantiated based on Equation 1 and auxiliary functions as:

$$f_{\sqsupset}^{\mathrm{LM}}(r_1, R_2) = p_1 \cdot (\lambda \frac{\sum_{r_2 \in R_2 | r_2 \prec r_1} p_2}{size(r_1)} + (1-\lambda)\frac{|R_2|}{size(Root)}) \quad (5)$$

For a language model without smoothing, the relevance score computation function is specified using the same equations where $\lambda = 1$.

In the Okapi system we simplify the Equation 2 by removing the third fraction in the sum as it is based on a size of the query and is not supported in other models. The complex function $f_{\sqsupset}$ is specified as:

$$f_{\sqsupset}^{\mathrm{Okapi}}(r_1, R_2) = p_1 \cdot$$
$$ln \frac{|\{r \in C | n = n_1\}| - |\{r \in C | n = n_1 \wedge \exists r_2 \in R_2 \wedge r_2 \prec r_1\}| + 0.5}{|\{r \in C | n = n_1 \wedge \exists r_2 \in R_2 \wedge r_2 \prec r_1\}| + 0.5}$$
$$\cdot \frac{(k_1 + 1) \cdot \sum_{r_2 \in R_2 | r_2 \prec r_1} p_2}{k_1((1-b) + b\frac{size(r_1)}{avg\_size(n_1)}) + \sum_{r_2 \in R_2 | r_2 \prec r_1} p_2} \quad (6)$$

For the tf.idf approach we have :

$$f_{\sqsupset}^{\mathrm{tf.idf}}(r_1, R_2) = p_1 \cdot \sum_{r_2 \in R_2 | r_2 \prec r_1} p_2$$
$$\cdot\, ln \frac{|\{r \in C | n = n_1\}|}{|\{r \in C | n = n_1 \wedge \exists r_2 \in R_2 \wedge r_2 \prec r_1\}|} \quad (7)$$

The element relevance score computation in GPX model is specified based on Equation 4 as:

$$f_{\sqsupset}^{\mathrm{GPX}}(r_1, R_2) = p_1 \; op \; \frac{\sum_{r_2 \in R_2 | r_2 \prec r_1} p_2}{|R_2|} \quad (8)$$

In our experiments we use two variants of the GPX model, one where *op* is implemented as '+' with the default element region score 0, referred to as basic model (denoted with '*' in Table 2), and the other where *op* is implemented as '·' with the default element region score 1 (see Section 3.2).

### 2.3.2 Element score propagation

The operators ▶ and ◀ specify propagation of scores to the containing or contained elements, respectively. Thus, the functions $f_{\blacktriangleright}(r_1, R_2)$ and $f_{\blacktriangleleft}(r_1, R_2)$ specify whether the propagation is performed with or without normalization, if the score values of contained or containing elements are summed, averaged or maximized, etc.

For the basic language modeling approach with and without smoothing, as well as for the basic tf.idf and Okapi model, we use the same approach. We employ a weighted sum normalized by the size of regions in the first operand for modeling upwards element score propagation. The simple sum of scores is used for downwards element score propagation. This choice was made because these functions showed good results in our experiments [27].

$$f_{\blacktriangleright}^{\text{LM,tf.idf,Okapi}}(r_1, R_2) = p_1 \cdot \frac{\sum_{r_2 \in R_2 | r_2 \prec r_1} p_2 \cdot size(r_2)}{size(r_1)} \tag{9}$$

$$f_{\blacktriangleleft}^{\text{LM,tf.idf,Okapi}}(r_1, R_2) = p_1 \cdot \sum_{r_2 \in R_2 | r_2 \prec r_1} p_2 \tag{10}$$

However, for basic GPX model (denoted with '*' in Table 2) we employ different computations for element score propagation:

$$f_{\blacktriangleright}^{\text{GPX}}(r_1, R_2) = p_1 + \sum_{r_2 \in R_2 | r_1 \prec r_2} p_2 \tag{11}$$

$$f_{\blacktriangleleft}^{\text{GPX}}(r_1, R_2) = p_1 + \sum_{r_2 \in R_2 | r_2 \prec r_1} p_2 \tag{12}$$

We also tried Equation 9 and Equations 10 for additional GPX runs with default element region score 1, as well as sum instead of weighted sum for upwards score propagation in other models, defined similar to Equation 10 (see Section 3.2).

### 2.3.3 Element score combination

The abstract operator $\otimes$ specifies how scores are combined in an AND expression, denoted in SRA by $\sqcap_p$, while the operator $\oplus$ defines score combination in an OR expression, denoted in SRA with $\sqcup_p$. In our basic retrieval models for basic experimental series, we make different choices for each model. For the basic language model, following [22] and [27], we use the instantiation where $\otimes$ is implemented as a product and $\oplus$ is implemented as a sum. Based on 'fuzzy' specification of tf.idf model in [25], in our basic tf.idf model $\otimes$ is modeled as $min$ and $\oplus$ is modeled as $max$. Following [7], in the Okapi (INQUERY) model we define these two abstract operators as follows:

$$p_1 \otimes p_2 = p_1 \cdot p_2 \tag{13}$$

$$p_1 \oplus p_2 = 1 - (1 - p_1) \cdot (1 - p_2) \tag{14}$$

Due to a somewhat different specification of the GPX model with respect to other models (see Equations 4 and 8), we instantiate $\otimes$ as well as $\oplus$ as:

$$p_1 \otimes p_2 = p_1 \oplus p_2 = \begin{cases} p_1 + p_2 & \text{if } p_1 = 0 \vee p_2 = 0 \\ A \cdot (p_1 + p_2) & \text{otherwise} \end{cases} \tag{15}$$

Although this formula in combination with formula in Equation 8 results in a retrieval model that is slightly different than the one given in Equation 4, it follows the semantics of the model which is to boost the scores for regions that contain more query terms.

Furthermore, we experimented with different implementations of score combination functions for each of the score computation functions as can be seen in Section 3.2.

## 3. EXPERIMENTS

In this section, we describe our prototype system, the XML document collection, and the query set we used to demonstrate the functionality of our approach (Section 3.1). Next, in Section 3.2, we show and discuss the results of our experimental runs.

### 3.1 Setup

To demonstrate the usefulness of our approach, we have built a prototype system and evaluated it against a well-known document collection and query set. Following good practice in database systems design, we setup our prototype following a typical three-layered architecture [34]:

**Conceptual layer** This layer takes a NEXI query expression as input, puts it through a filter to standardize/sanitize[5] it, and produces an SRA expression to be fed into the next layer.

**Logical layer** This layer takes an SRA expression as input. Like the previous layer, it does some filtering and standardization and transforms it into an expression for input to the next layer. Due to a modular setup, changing retrieval models is straightforward: one just plugs in a different transformation module. Building a new module, to capture yet another retrieval model to experiment with, takes less than half an hour.

**Physical layer** For the physical implementation we use a low-level physical DB engine - MonetDB [4][6]. The score region algebra operators are implemented as a set of procedures in Monet Interpreter Language (MIL).

Not many XML document collections exist that also come with IR queries and user assessments to evaluate the retrieval effectiveness of a system. We use the most well known collection, provided by the INEX initiative. This collection consists of IEEE journal papers in XML format. Each year a new set of so-called topics is constructed by the participants of the INEX workshop series. Those topics contain a NEXI query expression and a textual description of the so-called information need of the user, i.e., an explanation of what kind of answers should be considered as good results for that query. Every participant runs the queries on their system.

By pooling and manual reviewing by the participants the results are assessed, i.e., checked whether they are good answers or not. These assessments are then aggregated by the

[5]In our case the standardization consists of removing the '"' characters denoting phrases, '+' modifiers for query terms and terms with '-' modifiers, denoting important and unimportant terms. Note that our system can handle phrases and term modifiers [27] though we left it out as it goes beyond the scope of this paper.

[6]At the same time we developed the PostgreSQL implementation but we used the MonetDB implementation for our experiments since it was faster and less resource demanding.

Table 2: Experimental series.

| Series | Score computation ($f_{\sqsupset}$) | Propagation ($f_{\blacktriangleright}$) | Combination ($\otimes$) | Combination ($\oplus$) | mean average prec. |
|---|---|---|---|---|---|
| I | LM, $\lambda = 1$ | weighted sum | product | sum | **0.1261** |
| I$^a$ | LM, $\lambda = 1$ | sum | sum | sum | 0.1216 |
| II | LM, $\lambda = 0.5$ | weighted sum | product | sum | 0.2247 |
| II$^a$ | LM, $\lambda = 0.5$ | sum | product | sum | **0.2367** |
| II$^b$ | LM, $\lambda = 0.5$ | weighted sum | sum | sum | 0.1201 |
| III | Okapi, $k_1 = 1.5$, $b = 0.75$ | weighted sum | product | prob. sum | 0.1351 |
| III$^a$ | Okapi, $k_1 = 1.5$, $b = 0.75$ | sum | sum | sum | 0.2358 |
| III$^b$ | Okapi, $k_1 = 1.5$, $b = 0.75$ | weighted sum | sum | sum | **0.2578** |
| IV | tf.idf | weighted sum | min | max | 0.1425 |
| IV$^a$ | tf.idf | sum | product | prob. sum | **0.1594** |
| IV$^b$ | tf.idf | sum | sum | sum | 0.1561 |
| V | GPX*, $N = 5$ | sum* | exp. sum | exp. sum | **0.2782** |
| V$^a$ | GPX, $N = 5$ | sum | exp. sum | exp. sum | 0.2778 |
| V$^b$ | GPX, $N = 5$ | weighted sum | exp. sum | exp. sum | 0.2519 |

organization and made available to the participants. Using a tool provided by the organization, each participant than can compute how good their system is performing in terms of precision and recall.

We use the 30 topics and corresponding assessments of 2003 (see [33] and Appendix in [15]) to test our architecture for each of the models described in Section 2. In Section 3.2 we present the results of these experiments. We perform several experiments using the settings described above. First of all we ran the system for the basic models and later we perform experiments with varying score propagation and combination functions for these models, as described in Section 2.3.

## 3.2 Results

In this section we discuss the results of the experiments described above using the topics from INEX 2003. In Figure 1 we show the comparison of recall-precision graphs for our basic retrieval models (experimental series), aggregated over all 30 topics. The mean average precision for basic (I to V) and additional experimental series (denoted with $^a$ and $^b$) is given in the last column of Table 2. The mean average precision is actually the average precision aggregated over all topics. To produce recall-precision graphs and compute mean average precision we use the official INEX tool for the evaluation (see [15] for details). The highest mean average precision for each score computation model is given in bold.

As can be seen in the table the results depend a lot on the function used for relevance score computation. The best runs for language models with smoothing, Okapi, and GPX significantly outperform language models without smoothing and tf.idf. However, for language models with smoothing, Okapi, and GPX, mean average precision is quite different for different combinations of score propagation and combination functions. For example, the mean average precision decreases with almost 50% if we compare series II and II$^b$ for language models and increases for almost 100% in series III and III$^b$ for Okapi. On the other hand, no matter what kind of functions for score propagation and score combination we use the mean average precision for tf.idf models is approximately 0.15 (see series IV, IV$^a$, and IV$^b$).

In our experiments the best results are obtained by using the GPX model. However, the question is whether some of the other models models for score computation (Okapi or language models) with the right choice for the score combination and score propagation functions and the right value for parameters, $\lambda$, $k_1$, and $b$, can boost the mean average precision and outperform GPX. Additionally, the effectiveness of each model can be studied with respect to each topic in isolation, to determine which retrieval model is the most appropriate for each topic. This could help us to classify topics based on their features, such as number of query terms, existence of upwards or downwards score propagation, etc., and apply the best retrieval model for each topic type. We hope to answer these questions in our future research.

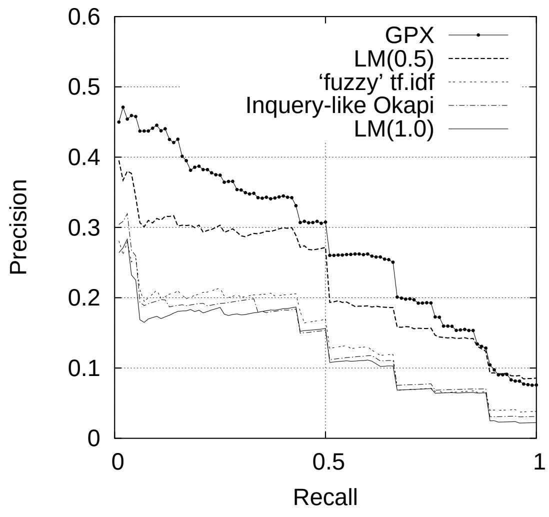

**Figure 1: Comparison of recall precision graphs for basic experimental series.**

## 4. CONCLUSIONS AND FUTURE WORK

Most XML-IR systems adapt and extend existing flat file IR systems to support the searching of structured XML documents. Since these approaches are retrieval model specific and depend on the physical implementation, it is difficult to adapt them to support different retrieval models. We believe that existing XML-IR database approaches have to be made *transparent* in order to satisfy complex user information needs expressed on top of XML collections. We argue that the right architectural level to achieve this transparency is the logical level. In that way the system is also flexible for different aspects of IR search over XML at the conceptual level and distinct implementations of storage schemes and access algorithms at the physical level. By developing a transparent score region algebra at the logical level of a flexible three-level database system we are able to support the application of state of the art IR models to ranked XML retrieval. Also, it provides us with a uniform framework where we can compare the effectiveness and study properties of different XML retrieval models as we have shown in this paper.

We are planning to further investigate the usefulness of the transparent logical algebra by applying different models to XML IR and to study the properties of score region algebra operators with respect to their consistency in ranking and their efficiency. We are also concerned with the modeling of term modifiers ('+' and '−'), explicit term and element weights, and phrases in score region algebra (see, e.g., [27]). Furthermore, we aim to better understand stemming and synonyms for terms, and vague treatment of search and answer elements throughout the instantiation of retrieval models in the SRA. Finally, we aim to investigate which combination of scoring functions in score region algebra operators is more appropriate for different topic types, such as topics with same search and answer elements, queries without upward or downward propagation, etc. This will guide us to a more effective and efficient, transparent XML-IR database system.

## 5. 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 *ACM TOIS 15 (4)*, volume 15, 1997.

[3] S. Boag, D. Chamberlin, M.F. Fernandez, D. Florescu, J. Robie, and J. Simeon. XQuery 1.0: An XML Query Language. Technical report, W3C, 2002.

[4] P. Boncz. *Monet: a Next Generation Database Kernel for Query Intensive Applications*. PhD thesis, CWI, 2002.

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

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

[7] J. P. Callan, W. B. Croft, and S. M. Harding. The INQUERY Retrieval System. In *Proceedings of the 3rd DEXA Conference*, 1992.

[8] J. Clark and S. DeRose. XML Path Language XPath Version 1.0. Technical report, W3C, 1999.

[9] 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.

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

[11] D. Florescu and I. Manolescu. Integrating Keyword Search into XML Query Processing. In *Proceedings of the 9th WWW Conference*, 2000.

[12] N. Fuhr. Models for Integrated Information Retrieval and Database Systems. *IEEE Data Engineering Bulletin*, 19(1), 1996.

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

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

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

[16] S. Geva. GPX - Gardens Point XML Information Retrieval at INEX 2004. In *Proceedings of the 3rd INEX Workshop, LNCS 3493, Springer*, 2005.

[17] N. Gövert, M. Abolhassani, N. Fuhr, and K. Großjohan. Content-oriented XML Retrieval with HyRex. In *Proceedings of the 1st INEX Workshop*, ERCIM Publications, 2003.

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

[19] D.A. Grossman and O. Frieder. *Information retrieval: algorithms and heuristics*. The Kluwer international series in engineering and computer science. Kluwer Academic, Boston, 1998. ISBN 0-7923-8271-4.

[20] T. Grust, S. Sakr, and J. Teubner. XQuery on SQL Hosts. In *Proceedings of the 30th VLDB Conference*, 2004.

[21] L. Guo, S. Feng, C. Botev, and J. Shanmugasundaram. XRANK: Ranked Keyword Search over XML Documents. In *Proceedings of the 26th ACM SIGMOD*, 2003.

[22] D. Hiemstra. *Using Language Models for Information Retrieval*. PhD thesis, University of Twente, Twente, The Netherlands, 2001.

[23] J. Kamps, M. de Rijke, and B. Sigurbjörnsson. Length normalization in XML retrieval. In *Proceedings of the 27th ACM SIGIR*, 2004.

[24] 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.

[25] Y. Mass and M. Mandelbrod. Component Ranking and Automatic Query Refinement for XML Retrieval. In *Proceedings of the 3rd INEX Workshop, LNCS 3493, Springer*, 2005.

[26] 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 (WIRD'04)*, 2004.

[27] 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.

[28] P. Ogilvie and J. Callan. Using Language Models for Flat Text Queries in XML Retrieval. In *Proceedings of the 2nd INEX Workshop*, ERCIM Publications, 2004.

[29] 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.

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

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

[32] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGrow-Hill, New York, NY, USA, 1st edition, 1983.

[33] 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.

[34] D. Tsichritzis and A. Klug. The ANSI/X3/SPARC DBMS Framework Report of the Study Group on Database Management Systems. *Information systems*, 3, 1978.

[35] S. R. Vasanthakumar, J. P. Callan, and W. Bruce Croft. Integrating INQUERY with an RDBMS to Support Text Retrieval. *IEEE Data Engineering Bulletin*, 19(1), 1996.

[36] A.P. de Vries, M.G.L.M. van Doorn, H.M. Blanken, and P.M.G. Apers. The miRRor MMDBMS Architecture. In *Proceedings of the 25th VLDB Conference*, 1999.

[37] C. Zhang, J. Naughton, D. DeWitt, Q. Luo, and G. Lohman. On Supporting Containment Queries in Relational Database Management Systems. In *Proceedings of the 20th ACM SIGMOD*, 2001.