Proceedings of the Joint Workshops on XML, IR, and DB

Ricardo Baeza-Yates
Yoelle Maarek
Thomas Roelleke
Arjen P. de Vries

Proceedings of the Joint Workshops on XML, IR, and DB

Ricardo Baeza-Yates – rbaeza@dcc.uchile.cl
Yoelle Maarek – yoelle@il.ibm.com
Thomas Roelleke – thor@dcs.qmul.ac.uk
Arjen P. de Vries – arjen@acm.org

July 29, 2004

Proceedings of the *3rd XML and Information Retrieval Workshop* and the *1st Workshop on the Integration of Information Retrieval and Databases (WIRD'04)*, Sheffield, United Kingdom, July 29, 2004. Workshops held at the 27th Annual International ACM SIGIR Conference.

©2004 by the individual authors/owners.

Contents

3rd XML and Information Retrieval Workshop

Preface

This workshop is the third in the series of XML and Information Retrieval workshops that were held in at SIGIR'2000 (Athens, Greece, see SIGIR Forum Fall 2000 issue at `http://www.acm.org/sigir/forum/S2000/XML_report.pdf`) and SIGIR'2002 (Tampere, Finland, see SIGIR Forum Fall 2002 issue at `http://www.acm.org/sigir/forum/F2002/maarek.pdf`). It complements the INEX (Initiative for the Evaluation of XML Retrieval) meetings that have been organized for the last two years, by providing researchers a useful forum for discussing (before implementing) and evaluating their models at INEX in the second half of the year.

We pursue the exchange of ideas between researchers who are now active in this IR sub-field and attract new interested researchers, on issues related to the application of IR methods to XML data for querying, retrieval, navigating, etc. We have gone a long way since the first edition in 2000, when XML was entirely dominated by the DB community, which have their own venues for XML database issues. Nevertheless, there is still room for more efforts in this field, in particular from the IR point of view.

This year, we received seven submissions from where four were selected. We also decided to work closely with the First Workshop on DB-IR integration, as XML is an important part of that problem, by having a joint invited talk and a common program.

Ricardo Baeza-Yates, CWR/DCC, Univ. of Chile
Yoelle Maarek, IBM Haifa, Israel

Organisers and PC-chairs

Ricardo Baeza-Yates, University of Chile, Chile
Yoelle S. Maarek, IBM Haifa Research Lab, Israel

Programme Committee

Mariano Consens, University of Toronto, Canada
Michael Gertz, UC at Davis, USA
Torsten Grabs, Microsoft Corporation, USA
David Hawking, CSIRO, Australia
Joemon Jose, Glasgow University, UK
Mounia Lalmas, Queen Mary University of London, UK
Yosi Mass, IBM Haifa Research Lab, Israel
Prabhakar Raghavan, Verity Inc., USA
Dan Suciu, University of Washington, USA

Extending the XML Fragment Model to Support Querying over Annotated text

Andrei Z Broder*, David Carmel**, Yoelle S. Maarek**, Matan Mandelbrod**, Yosi Mass**

*IBM T.J. Watson Research Center
Yorktown Heights, NY, USA
**IBM Research Lab in Haifa
Haifa 31905, ISRAEL

ABSTRACT

The XML Fragment model offers a convenient formalism for querying XML collections "by example", that is, by formulating the query as a piece of XML that expresses the user's needs. This allows relevant results to be returned either as full documents or as XML Fragments, using a simple extension of the vector space model for ranking.

In this work, we investigate extending this model to text analytics applications where semantic tags (e.g., names, entities, relations, etc.) are automatically generated to annotate the underlying text. Each type of tag can easily be represented as an XML element, but the spans of these tags often cross over each other, which, of course, is not allowed by the XML DOM structure. We discuss how our original XML Fragments model can be extended to query annotated documents with possibly overlapping annotations and illustrate our approach with examples of queries over annotated documents generated in the context of IBM's Unstructured Information Management Architecture (UIMA) framework.

Keywords

XML Search & Retrieval, XML Fragments, Text Annotators, UIMA

1. INTRODUCTION

The XML fragment approach, [4, 5] proposes a model for expressing queries over XML collections in the same form as the documents that compose the collection. For the sake of clarity, we remind here a few basic principles of the XML Fragment model before discussing the extension we propose in this work.

The XML Fragment model allows users to express their information needs in an approximate manner, thus following the classical information retrieval view. XML Fragments are defined as a "well balanced region of an XML document". Examples of XML Fragment queries are given in Figure 1. Note that the last example is an extreme case of a well-balanced fragment that does not contain any tag.

```
1. <book> search </book>
2. <book> <title> searching</title></book>
3. <year>1973</year> <author>knuth</author>
4. combinatorial algorithms
```

Figure 1: Examples of XML Fragments

Let us now take as example the XML document shown in Figure 2. The document will be returned as a result for all the queries listed in Figure 1.

```
<book>
   <volume value = 4/>
   <year>1973</year>
   <title>Combinatorial Algorithms</title>
   <author>Donald E. Knuth</author>
   <chapter number=7>
     <title>Combinatorial Searching</title>
   …</chapter>
   <chapter number=68
     <title>Recursion</title>
   …</chapter>
</book>
```

Figure 2: Examples of searchable XML documents

Relevance of results is measured using an extension of the classical cosine similarity measure where vectors of "terms in context" are compared rather than simple vector of terms. The exact semantics of XML Fragments as queries is detailed in [5], while ranking mechanisms are described and evaluated in [4].

In this paper we investigate a departure from the XML Fragment model that allows us to query the annotated documents generated by text analytics applications, and semantic analyzers. These applications process unstructured documents and extract semantic information that is added as "annotations" to the original text. A typical semantic annotation decorates a string of text and is characterized by a label (possibly with attributes) and a scope over a continuous portion of text (begin position/end position). These semantic annotations can be of various sorts; classical examples include proper name identification, location identification (to tag names of places, city, countries, etc.), and entity relationships.

Figure 3 gives an example of relationships that can be used for answering questions such as posed in TREC's question answering track. A possible question in that track, might be "Where is the Bahai Temple?", or "Where is Haifa?" Annotations such as illustrated in that figure, and other annotations that would tag "The Bahai Temple" as a monument, "Haifa" as a city, and "Israel" as a country, are typical elements of advanced text analytics applications.

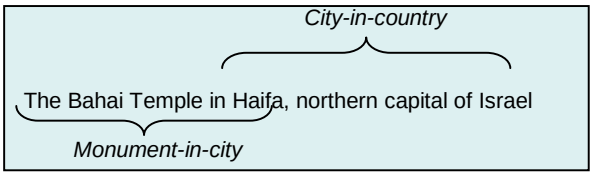

Figure 3: example of annotated document

These relationships might be generated by different analyzers or annotators, each specializing on a specific semantic aspect, but decorating the same piece of text. Each annotator generates a distinct view of the same piece of text, which can be represented as a well balanced XML piece as seen below

<Monument-in-city> The Bahai Temple in Haifa, </Monument-in-city> northern capital of Israel, is a pilgrimage destination of the Bahai faith.

and

The Bahai Temple in <City-in-country>Haifa, northern capital of Israel</City-in-country> is a pilgrimage destination of the Bahai faith.

However, if all these views are to be merged into a unique piece of annotated text we obtain the following piece of text represented in Figure 4 that is not valid XML

<Monument-in-city> The Bahai Temple in <City-in-country> Haifa,</Monument-in-city> northern capital of Israel</City-in-country> is a pilgrimage destination of the Bahai faith.

Figure 4: Invalid XML

An alternative was of representing annotations over text is to use spans of the form *label(begin position, end position),* where label is the name of the span (e.g. *monument)* and *being position* and *end position* are position counts based on simple tokens. Thus in our example if "the" is the 1st token, "Bahai", the 2nd , "Temple" the 3rd, etc., the two annotations in our example would be expressed as:

- Monument-in-the-city[1,5]
- City-in-country[5,9]

These two annotations cross over, (both span over the token "Haifa" at location 5), which is key problem if we decide to use XML as their representation format.

Note that the idea behind the sentence in Figure 3 might be expressed in many ways, yielding different intersections of the spans, for instance: "The Bahai Temple is located in northern Israel, in Haifa" (City-in-country contained inside Monument-in-city) or "Notable sights in Israel include the Bahai Temple in Haifa." (City-in-country contains Monument-in-city).

Similarly, on the query side, annotated documents which embed overlapping spans cannot be expressed as XML documents. A query over annotated documents could require a piece of text to appear under several annotations and as a consequence would not be easily expressed as XML Fragments. An example of a user's need might be to discover monuments that are located in Haifa, Israel. A simple way to express this need, that would match the document in Figure 3, would be to use the following non-well-formed piece of XML:

<Monument-in-city>
<City-in-country> Haifa,</Monument-in-city>
Israel</City-in-country>

Of course, beside this not being valid XML, this formulation would not match the alternate expressions discussed above.

We investigate here how the XML fragment model can be modified to deal with overlapping spans. Section 2 proposes an extension of the XML Fragment model and discusses some related work. Section 3 addresses implementation issues. Section 4 details the actual needs of advanced text analytics applications, and real-life customer requirements that motivate this work. We conclude by summarizing the contribution of this still preliminary work and identifying future directions.

2. ADDING ANNOTATIONS SUPPORT TO XML FRAGMENTS

Let us first formalize our representation of annotated documents. An annotated document is identified by its original text and by the set of annotations that decorate that text. We mark an annotated document D by D = (T, A), where T is the original Text and A is the set of annotations over that text. Each annotation is identified by *label[start, end]* where *label* is the annotation label (similar to a element name in XML) and *[start,end])* are the offsets of the first token and the last token in the text that it spans[1].

We distinguish between an annotation and an annotation instance, in a similar way that XML distinguishes between an element and an element instance. Thus, an annotation <*s*> denotes an arbitrary span and *xs = s[start, end]* denotes an instance of <*s*> that spans over the text between locations start and end.

Note that xs is not a unique identifier and there can be multiple annotations with the same label *s* and the same *start* and *end* locations.

We have formally defined in [4] how queries are satisfied by components. We define here components for annotated documents so as to be able to reuse the query satisfaction concept.

Definition 1 (component) – Let D = (T, A) be an annotated document and <s> an annotation label. Let xs be an instance of <*s*>, xs = *s[start,end]*. We define the component of *xs*, D_{xs} = $D(T_{xs},A_{xs})$ where T_{xs} is the piece of text within D that is spanned between the *start* and *end* positions and A_{xs} is a set of annotations consisting of the subset of the annotations of D whose spanned texts are fully contained between the *start* and *end* positions of *xs*. The annotations offsets are adjusted relative to T_{xs}.

Let us consider the example given in Figure 3, and add to it an additional annotation, tagging Haifa at location five as a city, e.g., *City[5,5]*. The annotation label <City-in-country> induces one instance *xCity-in-country = City-in-country[5,9*], and the component of this instance $D_{xCity\text{-}in\text{-}country}$ = $D(T_{xCity\text{-}in\text{-}country},A_{xCity\text{-}in\text{-}country})$ where $T_{xCity\text{-}in\text{-}country}$ is the text "Haifa,

[1] We assume the text is tokenized and each token in the text gets a unique sequential number indicating its location in the text.

northern capital of Israel" and $A_{xCity\text{-}in\text{-}country}$ is the set of annotations spanned over $T_{xCity\text{-}in\text{-}country}$, $A_{xCity\text{-}in\text{-}country}$ = {*City-in-country[5,9], City[5,5]*}.

In the remaining of this section we introduce with two new operators (intersection and concatenation) that allow to query annotated documents with XML Fragments.

2.1 Intersection of Annotations

As discussed above, each separate annotation can be represented as a valid XML. Invalid XML is obtained only by merging annotations that cross over the same piece of text. In this work, we focus on supporting XML Fragment queries over annotated documents that might not be representable as valid XML documents. In other words, we support queries that impose a combination of structural constraints on the same piece of text.

To enable such queries we introduce a new XML fragment operator, '*' in order to combine annotations. Assume that we are looking for a piece of text *t* that appears under 2 annotations *<s1>* and *<s2>*, i.e. as part of the text that is spanned by both annotations. We propose then to express our query as *<s1*s2>t</s1*s2>*. The intersection operator is of course not limited to two annotations and can intersect as many annotations as needed. In the following we define the intersection operator more formally.

Definition 2 (intersection) – Let D = (T,A) be an annotated document and let $xs_1 = s_1[start_1, end_1]$, $xs_2 = s_2[start_2, end_2]$ be two annotations in A. We define $start = max(start_1, start_2)$ and $end = min(end_1, end_2)$. If $start \leq end$ we define a new annotation $xs_1 * xs_2 = s_1 * s_2\ [start, end]$ and $D_{xs1*xs2}$ as the component induced by that annotation.

For example, the intersection annotation *Monument-in-city*City-in-country [5,5]* induces one component instance $D_{Mounument\text{-}in\text{-}city*City\text{-}in\text{-}country}$ with Text = "Haifa" and an Annotations set *A= {City[5,5]}*. With this definition of $D_{xs1*xs2}$ we can use the base procedure introduced in [4] to check whether a Query Q is satisfied by a document component D_{xs}. Note that the intersection operator is insensitive to order, i.e., both queries *<s1*s2>t</s1*s2>* and *<s2*s1>t</s2*s1>* are satisfied by exactly the same components.

2.2 Concatenation of Annotations

The second operator over annotations we introduce here is the concatenation of consecutive annotations to represent a consecutive piece of text spanned by both annotations. We enable a new XML fragment operator '+' in order to concatenate annotations. The concatenated annotation is the full text spanned by both annotations. Since valid annotations span over consecutive text with no holes, only annotations that have at least one common token can be concatenated, otherwise, the concatenation fails and return the empty annotation.

Definition 3 (concatenation) – Let D = (T, A) be an annotated document, let $xs_1 = s_1[start_1, end_1]$, $xs_2 = s_2[start_2, end_2]$ be two annotations in A and assume without loss of generality that $start_1 \leq start_2$. If $start_2 \leq end_1$ we define $end = max(end_1, end_2)$ and a new annotation $xs_1 + xs_2 = s_1 + s_2\ [start_1, end]$. We define $D_{xs1+xs2}$ as the component induced by that annotation.

For example, the concatenation of *Monument-in-city[1,5]* and *City-in-country[5,9]* induces a component $D_{Monument\text{-}in\text{-}city+City\text{-}in\text{-}country}$ which is identical to the document in Figure 3.

According to this definition, the concatenation operator is also insensitive to the annotations order. The only requirement, induced from the definition of annotation validity, is that the two annotations cross over each other, i.e., that they have a non-empty intersection. Otherwise the operator fails and returns the empty annotation.

Note that the key motivation for using operators over annotations is that while they do not comply with any valid DTD, they nevertheless permit to express valid XML Fragments. Indeed, by definition, XML Fragments do not need to respect any specific DTD or schema; they only need to be "well balanced" pieces of XML and the same XML parsing utilities can be used to process queries as described in the next section.

2.3 Related Work

A similar framework to the one described in this work is *region algebra* [6, 10, 14] that reasons about arbitrary regions of textual data. Region algebras model the text as a sequential stream of tokens and a region is defined by a starting position and an ending position. A set of operators are defined over regions; e.g. the *containing* operator returns true if one region contains the second. The *followed_by* operator validates whether one region follows the second. It has been shown that Boolean operators (AND, OR, NOT) can be fully expressed by region algebra operators. In addition, region algebra is frequently used for XML retrieval [14]. However, region algebra only handles nested regions [10] and cannot express the overlapping annotations we consider in this work.

3. IMPLEMENTATION

The XML Fragment model has been fully implemented in the Juru Search engine as described in the JuruXML [12, 13] search engine. We give here a brief description of the implementation while full details can be found in the above references.

The indexing process generates an inverted index in which not only terms but also XML tags are stored, together with their associated postings list, including frequency information and location information represented as offsets of term and tag occurrences.

A key advantage of using valid XML rather than ad-hoc spans is that this allows us to use a standard XML parser to analyze XML fragments-based queries. The only limitation is that the new operators (*, +) being not valid chars in an XML tags, we have to conduct a trivial macro expansion and replace those operators with a valid distinguished string. (e.g. *<s1*s2>* is replaced by *<s1.IntersectionOf.s2>*), before sending queries to a standard XML parser such as Xerces [15]. We also add an encapsulating *<root> </root>* tag in order to transform well-balanced XML Fragment queries into well formed XML. An internal tree structure is maintained, which holds the information generated by processing tags together with the hierarchical representation of the query produces by the parser. The query is then run against the index, and results ranked by relevance are returned using an XML fragment specific ranking mechanism, described in [4]. Finally, we check that all query constrains are satisfied by traversing the query

tree and by using the terms' offsets, as kept in the inverted index, to check for containment.

For example, for the query given below, and the document in Figure 3, we extract from the inverted index the location of the content word *Haifa* to be 5 and the span of <*Monument-in-city* * *City-in-country*> to be [5, 5]. It is then easy to check that the query constraints are satisfied by that document.

```
<Monument-in-city * City-in-country>
  Haifa
</Monument-in-city*City-in-country>
```

4. APPLICATIONS

The motivating application for this work is IBM's Unstructured Information Management Architecture (UIMA) [7, 8] a software architecture for supporting applications that integrate search and analytics over a combination of structured and unstructured information. UIMA is based on document-level analysis performed by component processing elements named Text Analysis Engines (TAEs). Examples of TAEs include language translators, document summarizers, named-entity detectors, and relationship extractors, such as "HoldsDuring", "Located", and so on. Each TAE specializes in discovering specific concepts (or "semantic entities") implicit in the document text.

The document analysis is based on a data structure, named the Common Analysis Structure (CAS) [9], which contains the original document (the subject of analysis) and associated metadata in the form of annotation with respect the original text. The CAS is passed through an application-specified sequence of TAEs. Each TAE in the sequence considers the input CAS, potentially infers and adds additional annotations, and outputs an updated CAS. Annotations in the CAS are maintained separately from the document itself, and they often over-lap.

The first major application of UIMA is BioTeKS [11], a system for text analytics for life science. BioTeKS integrates research technologies from multiple IBM Research labs using the UIMA platform. A main goal of UIMA is to support "semantic search" – the capability to find documents based on the semantic content discovered by the TAEs. To this end, UIMA specifies search engine indexing and query interfaces: the indexing interface supports the indexing of tokens as well as the indexing of annotations and in particular cross-over annotations; correspondingly, the query interface supports queries that may be predicated on nested or over-lapping structures of annotations and tokens in addition to Boolean combinations of tokens and annotations.

We present below some queries that take advantage of the intersection tag, using the documents given in Figure 5. For each example, we provide the description of the information need and its expression as an XML Fragment. For the sake of clarity, the examples are oversimplified. However in practice, the need to be able to specify intersection of complex annotations is quite common.

Doc1:
Dr. Orgo proved that Ibuprofen reduces the effectiveness of Aspirin which however, according to Orgo, enhances the antipyretic action of Acetaminophen…

Doc2:
Reuters reported that Bin Laden was in Pakistan in March.

Figure 5: Examples of a source documents

The entity relationship extractor annotates the text with annotations that span over text as shown in figure below:

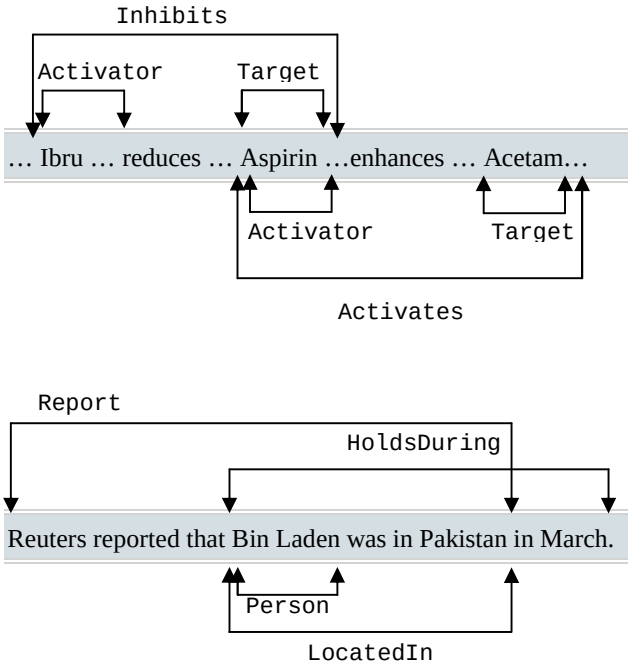

Figure 6: Annotations as spans over Doc1 and Doc2

We give below a few examples of XML Fragment based queries, Q1 is a simple example, while Q2, Q3, Q4 deal illustrate the intersection and concatenation operators.

Q1 – Simple query

Description of need: Aspirin as Activator

Query:

```
<Activator>Aspirin</Activator>
```

Result – Doc1 is returned.

Q2 – Intersection of spans

Description: Aspirin appearing under both Inhibits and Activates annotations.

Query:

```
<Inhibits*Activates>Aspirin
</Inhibits*Activates>
```

Result – Doc 1 is returned. According to Definition 2, one component instance is induced by the intersection tag, namely $D_{Inhibits[1,9]*Activates[9,15]}$ with Text = 'Aspirin' and with Annotations = *{Activator[1,1]}*. Since the text 'Aspirin' appears under this new component, the Query is satisfied by the document.

Q3 – Containment of spans

Description: Aspirin appearing under the annotation 'Activator' which is contained in the intersection of 'Inhibits' and 'Activates'.

Query:

```
<Inhibits*Activates>
   <Activator>Aspirin</Activator>
</Inhibits*Activates>
```

Result – Doc1 is returned. As described in the previous example, the annotation <Activator> appears in the document component $D_{Inhibits[1,9]*Activates[9,15]}$ and hence the Query is satisfied by our document.

Q4 – Concatenation of spans

Description: Find reports from Reuters about events in Pakistan in March, that is, find simultaneous occurrences 'Reuters', 'Pakistan', and 'March' contained in the concatenation span of 'Report' and 'HoldsDuring'.

Query:

```
< Report+HoldsDuring >
   +Pakistan +March +Reuters
</Report+HoldsDuring >
```

Result – Doc2 is returned since all mandatory content words are contained in the document component Dxs1+xs2, with *xs1 = Report[1,8]* and xs2 = *HoldsDuring[4,10]*.

5. CONCLUSION

This work describes a proposal for extending the XML Fragment model to support cross-over text annotations. This was motivated by the actual requirements of the IBM text analytics research community that has adopted a modular approach to building applications, based on IBM's Unstructured Information Management Architecture (UIMA). The core of this approach consists of independently constructed Text Analysis Engines (TAEs) that produce multiple annotations over the same underlying text documents by associating semantic labels to spans (sequences of consecutive tokens) in the document.

This approach has many admirable properties: it allows the independent development of components, the construction of complex TAEs from simple ones, and improvement in the overall semantic understanding. However, from the point of view of search technology, the use of cross-over spans in queries and in documents means that standard XML search won't work for UIMA documents.

To finesse this problem we have introduced two new tag operators that express conditions related to overlapping spans in queries (namely, the intersection and the union of cross-over spans) as valid XML. This allows us to easily extend the semantics we previously defined for XML fragments, and even, after simple processing, to keep our XML utilities, such as a standard parser and our search run-time.

Our short-term goal is to verify with the audience of the workshop whether these cross-over spans are as common as we believe and whether similar needs arose outside the context of IBM's UIMA. We would also like to verify whether our tag operators are acceptable by other practitioners in the field of XML retrieval. Should the answers be positive, our long-term goal is to promote these proposals to various standards making bodies, such as W3C XML Query. We want to do this not only for standardization purposes, but also to increase the awareness of the needs of the IR community, needs that are rather different from those of the publishing and DB communities that currently control most XML official bodies.

6. ACKNOWLEDGMENT

We would like to thank David Ferrucci for his help with UIMA infrastructure and annotated examples.

7. REFERENCES

[1] A. Berglund, S. Boag, D. Chamberlin, M.F. Fernandez, M. Kay, J. Robie and J. Simeon, "XML Path Language (XPath) 2.0 *W3C Working Draft*, 12 Nov 2003.

[2] S. Boag, D. Chamberlin, M.F. Fernandez, D. Florescu, J. Robie and J. Simeon, "XQuery 1.0: An XML Query Language", *W3C Working Draft*, 12 Nov 2003

[3] A. Broder, "A taxonomy of Web search*"*, *SIGIR Forum*, 36:2, Fall 2002.

[4] A. Z. Broder, Y. Maarek, M. Mandelbrod and Y. Mass, "Using XML to Query XML, From Theory to Practice". In Proceedings of RIAO'04, Avignon, France, Apr. 2004,

[5] D. Carmel, Y. Maarek, M. Mandelbrod, Y. Mass, A. Soffer, "Searching XML Documents via XML Fragments"*,* In *Proceedings of SIGIR'2003*, Toronto, Canada, Aug. 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", Computer Journal 38:43{56}, 1995.

[7] D. Ferrucci and A. Lally. "UIMA: An Architectural Approach to Unstructured Information Processing in the Corporate Research Environment." To appear in *Natural Language Engineering*, 2004.

[8] D. Ferrucci and A. Lally. "UIMA by Example." To appear in *IBM Systems Journal*, 2004.

[9] T. Goetz and O. Suhre, "Design and Implementation of the Common Analysis System." To appear in *IBM Systems Journal* , 2004.

[10] J. Jaakkola and P. Kilpelainen. Nested text-region algebra. Technical report, University of Helsinki, 1999.

[11] R. Mack et al., "BioTeKS: Text Analytics for Life Sciences using the UIMA". To appear in *IBM Systems Journal* , 2004.

[12] Y. Mass, M. Mandelbrod, E. Amitay, D. Carmel, Y. Maarek and A. Soffer, "JuruXML, an XML Retrieval System at INEX'02", In Proceedings of INEX'02, Schloss Dagstuhl, Germany, Dec 2002.

[13] Y. Mass and M. Mandelbrod, "Retrieving the Most relevant XML Components", In Proceedings of INEX'03, Schloss Dagstuhl, Germany, Dec 2003.

[14] Vojkan Mihajlovic, Djoerd Hiemstra and Peter Apers, ``On Region Algebras, XML Databases, and Information Retrieval", In Proceedings of the 4th DIR., 2003.

[15] Xerces, XML parsers in Java and C++. http://xml.apache.org

Document Structure Matching for Heterogeneous Corpora

Ludovic DENOYER Guillaume WISNIEWSKI Patrick GALLINARI

Université Paris 6
LIP6
8 rue du capitaine Scott
75015 PARIS – France

Ludovic.denoyer@lip6.fr wisniewski@poleia.lip6.fr Patrick.gallinari@lip6.fr

ABSTRACT

Querying heterogeneous XML document collections is an open problem. This will require building some sort of correspondence between the DTD of the different sources. We consider here the problem of matching the structure of XML documents from different sources. We introduce for that a stochastic structured document model and describe preliminary experiments performed on the INEX collection.

Keywords

1. INTRODUCTION

For the 2002 and 2003 editions, the Initiative for the Evaluation of XML retrieval (INEX) has been based on a homogenous document collection of IEEE scientific articles. This XML collection corresponds to a single DTD which is the union of the different journals DTDs. In most practical applications of XML retrieval, collections will be heterogeneous with documents coming from different sources with different DTDs. Handling heterogeneous collections is then a new challenge for XML IR which is the topic of a new explorative track at INEX 2004.

Heterogeneity has been considered for a long time in IR but only for plain text documents. In the database community, querying heterogeneous sources has also been a major concern for many years in different application domains like data integration, data warehousing, web services, etc. This has been explored more recently for semi-structured data and XML databases. Schema matching, i.e. identifying the semantic correspondences between elements of two schemas – or DTDs for XML - has been identified as a key operation for this problem. This is the first step for constructing complete mappings between two representations. Automatic or semi automatic schema matching has recently motivated a growing number of academic works [1]. Generally, it is supposed that different sources are available, each with a known schema. Different schema element characteristics are usually considered for the matching process: tag names in XML, data instance characteristics (special symbols, number of characters, etc), data type, metadata, etc. The match operator is often learned from labeled examples using basic machine learning techniques. Test sets for evaluation are usually relational tables or semi-structured web data. Such sources usually have a poor textual content and meaningful tag names.

Some of these ideas from schema matching could be helpful for XML IR. The context is however different and specific IR solutions should be developed. For example, for XML documents, tag names often have a weak semantic and few if any associated descriptive metadata, different tags may denote similar objects (e.g. the "item" tags in INEX), most important in many cases the DTD need not to be known. The document tree could be rather large and the relative importance of structural and content information is different than for databases. In many cases, the structure information is poor. Although some structural query elements (e.g. authors, dates) should be considered strictly, many others should be considered as vague constraints

Solutions to be developed for querying heterogeneous XML collections will also have to establish a correspondence between the structural elements of the different documents. The work described here is a preliminary attempt towards this direction.

We focus on the matching of document structures for IR on XML collections. Our guess is that, like for schema matching in databases, this is an essential operation for developing IR systems for heterogeneous XML collections. As an example, suppose one has developed an IR engine for some specific collection. This IR engine will use a mediated DTD and/or predefined tag names corresponding to this collection. One possible solution for querying new documents in the same domain is to transform these

documents into this mediated format. This is the point of view we adopt here.

We propose a stochastic structured document model for this matching task. This model will be trained using samples from a collection expressed in a mediated DTD. After training, it is able to take as input a new document, and to output a structured representation of this document, were the document elements have been tagged according to the mediated format. A characteristic of this model is that it takes into account both the structure and the content of the collection documents.

The matching problem is introduced in section 2. The model itself is described in section 3, preliminary experiments on the INEX collection are presented in section 4.

2. DOCUMENT STRUCTURE MATCHING

The general matching problem we want to deal with is the following : given a mediated DTD which is used for querying a collection of XML documents, how to transform new documents in this mediated DTD so that they could be queried using the same search engine ? This transformation should respect the semantic of the document elements.

We will make here the following hypothesis:

- Document collections considered for a matching task are about the same domain and have roughly similar elements. For example, we could consider different collections of scientific papers about computer science. The papers could have different logical structures but it makes sense to map them onto a general "scientific paper" schema. This is justified since matching completely heterogeneous collections hardly make sense in practice.
- The documents may come from different sources and the DTD may not be known. We will not use any source DTD information for the mapping. This is different from the database context where the schema is most often mandatory.

The document structure matching problem considered here is a simplified instance of this general problem. We want to associate each new document with the mediated format without changing the document tree structure. This transformation will associate each document tag with a tag of the mediated DTD, it will not merge nor cut document elements.

We will suppose that there exists a set of documents already expressed in the mediated DTD. When the latter is a specific collection DTD, this set will be the collection itself. Otherwise, it is supposed that a basic transformation has been manually defined for transforming an initial set of documents. For example for the INEX collection, different tags have been defined as equivalent.

The existence of such a training corpus is necessary when the transformation has to take into account the elements content. This is different from many databases applications where only the structural information (schema) is considered.

In the proposed approach, the DTD of new documents need not to be known. The matching will be inferred directly from the structural and content information of the document itself.

3. DOCUMENT MATCHING MODEL

The proposed approach relies on learning a stochastic model of the documents on the training collection. Both a stochastic grammar of the document structure and a model of the content information associated to each tag in a given context will be learned.

When a new document is processed, it is transformed into the most probable structured document representation according to the stochastic model.

The effect of this transformation will be to associate each document element with a tag name from the mediated DTD. This association is not a simple tag → tag correspondence, but depends on the structural and content contexts of the element. The stochastic model is able to learn simultaneously the transformations for different initial DTDs. When processing a new document, the most probable transformation will then be chosen.

3.1 STRUCTURED DOCUMENT

We consider a structured (XML) document as a tree. Each tree node n corresponds to a structural entity of the document and contains two types of information:

- A tag information (t_n) from a discrete set of tags T defined by the mediated DTD e.g. (*part, section, title*,…).
- A content information (c_n), we restrict ourselves here to textual content so that c_n is a sequence of words from a vocabulary V .

Figure 1 gives an example of such a structured document.

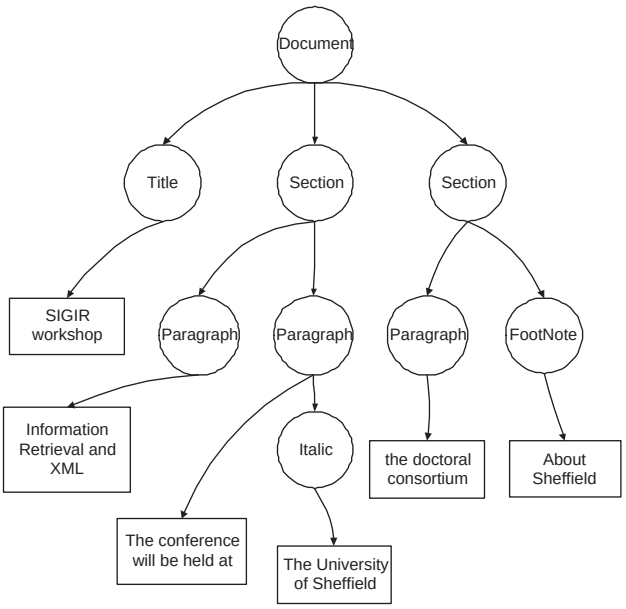

Figure 1: A tree representation for a structured document. Circle nodes correspond to structural information (tags) and square nodes to textual information.

3.2 A GENERATIVE MODEL FOR STRUCTURED DOCUMENTS

The structured document model takes into account both the content information and the structural information present in the documents.

Let d be a structured document, $(n_1,...,n_{/d/})$ its set of nodes with $/d/$ the number of nodes. Each n_i is a pair (t_{n_i}, c_{n_i}) with t_{n_i} its tag and c_{n_i} its textual information if any. Let $pa(n_i)$ denotes the parent of n_i in the tree and $children(n_i)$ the ordered list of n_i children. θ will denote the document model parameters.

The probability that model θ has generated document *d* is:

$$\begin{aligned} P(d/\theta) &= P(n_1,...,n_{/d/}/\theta) \\ &= P\left((t_{n_1},c_{n_1}),...,(t_{n_{/d/}},c_{n_{/d/}})/\theta\right) \\ &= P(t_{n_1},...,t_{n_{/d/}}/\theta)P(c_{n_1},...,c_{n_{/d/}}/t_{n_1},...,t_{n_{/d/}},\theta) \end{aligned} \quad (1)$$

The structural probability $P(t_{n_1},...,t_{n_{/d/}}/\theta)$ describes how the model generates the different tags of *d*. **The content probability** $P(c_{n_1},...,c_{n_{/d/}}/t_{n_1},...,t_{n_{/d/}},\theta)$ describes how the model generates the textual information for each node.

Direct estimation of *P(d/ θ)* is unfeasible for large documents. We will then make simplifying hypothesis in order to compute this probability.

3.2.1 STRUCTURAL PROBABILITY

We will use a formalism which is very similar to probabilistic tree-languages [7]. Let $childrentag(n_i)$ denotes the ordered list of tags corresponding to $children(n_i)$. In order to simplify the estimation of $P(t_{n_1},...,t_{n_{/d/}}/\theta)$, we make a first order independence hypothesis: the set of variables $childrentag(n_i)$ will be conditionally independent from all other tag variables given their parent:

$$\begin{aligned} &P(t_{n_1},...,t_{n_{/d/}}/\theta) \\ &= \prod_{n_i}\left(P(childrentag(n_i)/t_{n_i},\theta)\prod_{n\in children(n_i)}P(t_n/childrentag(n_i),\theta)\right) \\ &= \prod_{n_i}P(childrentag(n_i)/t_{n_i},\theta) \end{aligned}$$

The last equality follows from $P(t_n/childrentag(n_i),\theta)=1$.

This dependence relation between the tag variables for the document from Figure 1 could be modeled by a tree like belief network as shown in Figure 2.

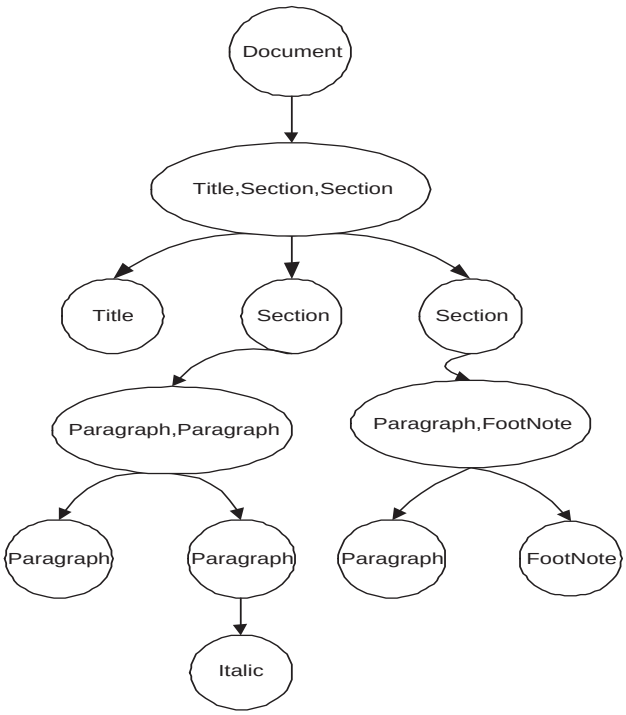

Figure 2: The belief network corresponding to the structural probability computation

The probability $P(childrentag(n_i)/t_{n_i},\theta)$ corresponds to the probability of seeing a particular sequence of tags inside a node with tag t_i. This will be estimated from the

training document collection. These probabilities for all parent-children configurations correspond to the probability of all instances of the DTD constraints which could be observed on the document collection. In this sense, they could be considered as a stochastic DTD inferred from the document collection. However, this model does not use any formal collection DTD when it is present.

3.2.2 CONTENT PROBABILITY

We make the hypothesis that the content of a node depends only on the tag of the node:

$$P(c_{n_1},...,c_{n_{/d/}} / t_{n_1},...,t_{n_{/d/}},\theta) = \prod_{n_i} P(c_{n_i} / t_{n_i},\theta)$$

Let $c_{n_i} = (w_{n_i}^1,...,w_{n_i}^{/n_i/})$ be the sequence of words in n_i, we further make an independence assumption between the term occurrences. The content model then becomes a Naïve Bayes model [8] conditioned on the node tag:

$$P(c_{n_1},...,c_{n_{/d/}} / t_{n_1},...,t_{n_{/d/}},\theta) = \prod_{n_i} \prod_{k=1}^{k=/n_i/} P(w_{n_i}^k / t_{n_i},\theta)$$

3.2.3 FINAL PROBABILITY

With these simplifying assumptions, the document probability (1) becomes:

$$P(d/\theta) = \prod_{n_i} \left(P(childrentag(n_i)/t_{n_i},\theta) \prod_{k=1}^{k=/n_i/} P(w_{n_i}^k / t_{n_i},\theta) \right) \quad (2)$$

The corresponding Bayesian network for the document of Figure 1 is shown in Figure 3.

3.3 LEARNING

In order to learn the model, we have to estimate:

- $P(t_1,...,t_m / t,\theta)$ for each tag t and each possible sequence of t children tags $t_1,...,t_m$
- $P(w/t,\theta)$ for each word w and each tag t

over the training set D of structured documents.

For that we will maximize the log-likelihood of the structured document collection given the document model θ:

$$L_D = \sum_D \left[\sum_{n_i} \left(\log P(childrentag(n_i)/t_{n_i},\theta) \sum_{k=1}^{/n_i/} \log P(w_{n_i}^k / t_{n_i},\theta) \right) \right]$$

This amounts at solving the equation $\nabla_\theta L = 0$ under equality constraint which ensures that probabilities sum to one. This system has an analytical solution. We used here the following robust estimators derived from this solution:

- $P(t_1,...,t_m / t,\theta) = \dfrac{N^t_{t_1,...,t_m}}{N^t} + \varepsilon$
- $P(w/t,\theta) = \dfrac{N^t_w + 1}{N^t + /V/}$

where $N^t_{t_1,...,t_m}$ is the number of nodes with tag t and with children $t_1,...,t_m$ in D, N^t is the number of nodes with tag t in D, N^t_w is the number of times word w is appearing in a node with tag t.

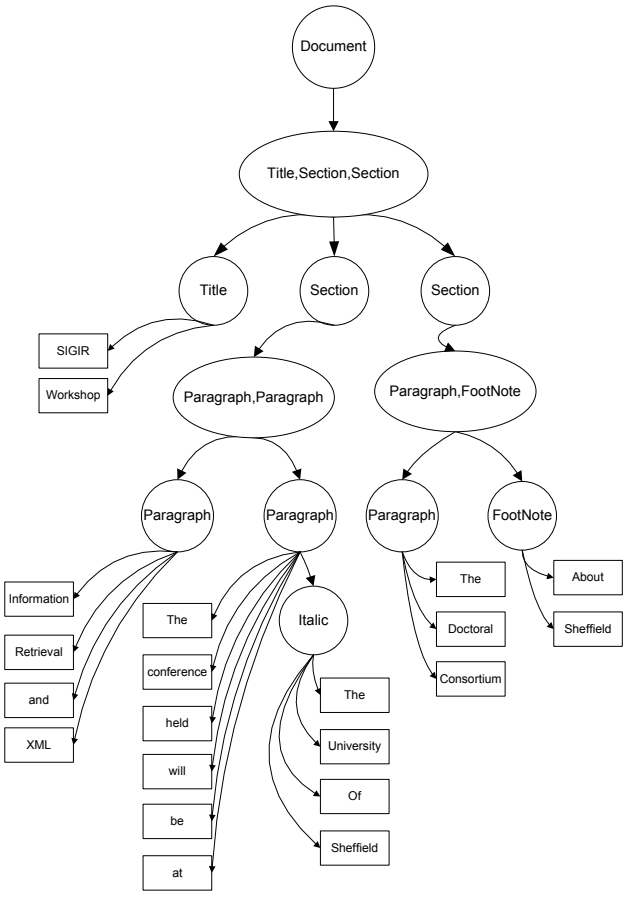

Figure 3: The final belief network corresponding to the structured document model

3.4 INFERENCE

Once θ is learned, a new document will be transformed into a structured document in the mediated DTD. For that we will have to find the most probable structure according to our model, i.e. to solve:

$$t_d = \arg\max_{t_{n_1},...t_{n_{/d/}}} P(d/\theta)$$
$$= \arg\max_{t_{n_1},...t_{n_{/d/}}} P(t_{n_1},...t_{n_{/d/}}/\theta) P(c_{n_1},...c_{n_{/d/}}/t_{n_1},...t_{n_{/d/}},\theta) \quad (3)$$

Solving this equation is similar to the decoding step in Hidden Markov Models, except that we deal here with trees instead of sequences. The best tagged tree is obtained via a dynamic programming algorithm (not described here).

The complexity of this decoding step is $O(/d/+/T/+/R/)$ where /d/ is the number of nodes of the document, /T/ is the number of possible tag labels and $/R/ = card(N^t_{t_1,...,t_m} \neq 0)$ is the number of "derivation rules" learned. It is thus linear wrt $/d/, /T/, /R/$.

4. EXPERIMENTS

4.1 INEX CORPUS

We performed preliminary experiments using the INEX collection of documents. This is a collection of XML articles from 20 different journals and proceedings of the IEEE Computer Society. It is composed of about 12,000 documents which represent more than 7,000,000 XML elements. Documents have been preprocessed using a stop-list and the Porter stemmer. All the words appearing in more than 20 documents were kept (about 27,000 words in all). The corpus was split in two, one part corresponding to the articles issued from the *"IEEE transaction ..."* journals (4233 articles) and the other part corresponding to articles from other IEEE journals (7874 articles). Training was performed on the second part and testing on "*IEEE transaction ...*" articles. This appeared as a natural split for the matching task since the two collections have different structure. The articles form the *"IEEE transaction ..."* journals are not formatted like the articles of the other journals. Training allows learning the structure of the first collection. "*IEEE transaction*" articles were then formatted according to this learned structure. Note that only the content and tree structure information of the document to be formatted are used with this model, the tag names of the document in the test set are not used.

In a first series of experiments, we kept the 139 INEX tags for training. This provides an evaluation of our method for a complex matching task. In a second series, we used a much smaller number of tags keeping only the 5 tags *headings, paragraph, section, list, misc*. The idea here is that only a small number of tag elements were used for querying XML documents in past INEX. We therefore wanted to evaluate the model for such a transformation.

4.2 EVALUATION MEASURE

In order to evaluate our model on the test collection, we used two measures:

- recall over the nodes. This is the percentage of nodes correctly labeled by the model i.e.: the model has retrieved the right tag for the node.
- percentage of documents from the test corpus with more than x% correctly labeled nodes for different values x.

For each series, we performed experiments by using:

- only the structural information: $t_{d=} \arg\max_{t_{n_1},...,t_{n_{/d/}}} P(t_{n_1},...,t_{n_{/d/}}/\theta)$. This is called the **Structure model**
- only the content information: $t_d = \arg\max_{t_{n_1},...,t_{n_{/d/}}} P(c_{n_1},...,c_{n_{/d/}}/t_{n_1},...,t_{n_{/d/}},\theta)$. This is called the **Content model**
- both content and structural information (eq.2). This is the **Full model**

4.3 RESULTS

Recall for the different experiments are shown in Table 1. The full model is able to label correctly about 86 % of the nodes for the 5 tags setting and 65 % when using 139 tags.

Structure information by itself allows us labeling correctly respectively 73 % and 49 % of the tags. For the content model, we show two values which are respectively the recall over all nodes (Content with all nodes) and the recall over all nodes with a textual content. In the first case, the nodes with no content (about 25% of the nodes) will be considered as errors. The first value measures the usefulness of the content model for labeling the INEX corpus, while the second one gives an idea about its usefulness for a corpus with textual content at each node.

	Content with all nodes	*Content without empty nodes*	*Structure*	*Structure and Content*
Number of nodes	≈5,000,000	≈3,500,000	≈5,000,000	≈5,000,000
5 tags	58%	81%	72.90%	86.50%
139 tags	27.80%	38.70%	49.70%	65.30 %

Table 1: Recall for the structure, content and full models for the two experiments

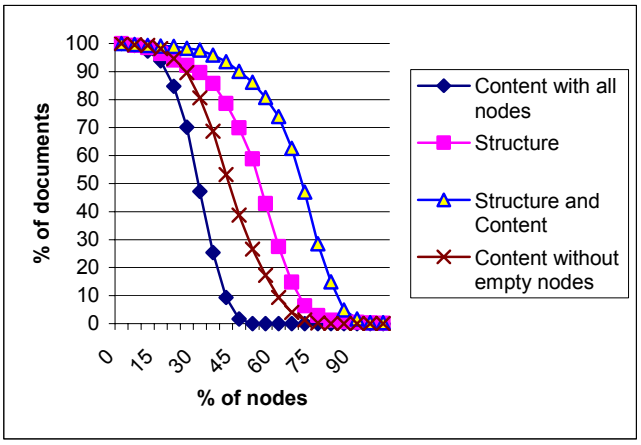

Figure 4: Percentage of documents with more than % nodes correctly tagged (139 tags). For example, for the structure model, about 50% of the test documents have about 80% of their nodes correctly labeled.

The curves in Figures 4 and 5 give the ability of the models for tagging correctly full documents. For the 139 tags setting, the full model has correctly tagged almost 90% of the nodes for about 40 % of the documents. This is clearly not sufficient for a fully "automatic" labeling. The task is however difficult here since the number of tags is large. In such a situation, the stochastic model could be a component of a more complex labeling system. For the 5 tags experiments, this model has been able to correctly tag about 80 % document with an accuracy of about 85 %. This level of performance could be acceptable for many IR applications.

Both measures show that structure and content information are complementary and that considering them together increases significantly the performances.

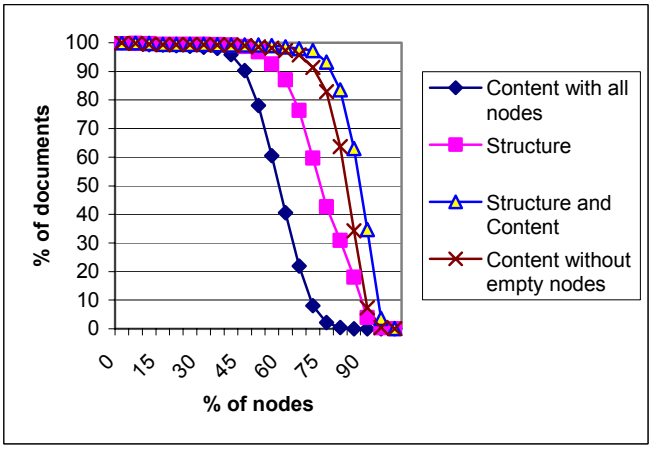

Figure 5: percentage of documents with more than % nodes correctly tagged (5 tags).

5. RELATED WORK

XML document matching as defined here shares similarities with schema matching in databases. Many schema matching approaches only consider schemas and make use of handcrafted rules for matching the elements of different schemas. Automatic methods have been developed for relieving this expensive process. [1] provides a survey and a typology of recent approaches to automatic schema matching.

The work closest to ours is probably that of Doan et al. [2]. They also considered transforming new DTDs into a mediated one using a 1-1 tag correspondence. They use a set of basic similarity measures and classifiers each operating on different schema element characteristics. These classifiers provide local scores which are linearly combined to give a global score for each possible tag. The final decision corresponds to the mediated tag with the highest score. Combining the different scores is a key idea in their approach. One of their classifiers operates on the element textual content. Up to our knowledge, this is the only system which takes into account this information for schema matching. They also make use of some basic relational information for scoring an element by considering the co-occurrence of parent-child tags. Compared to this approach, our model proposes a generative model of structured documents and focuses on structure and content which are the more important elements for XML IR.

Madhavan et al. [6] use many of the ideas developed in [2] for the more general problem of learning associations betweens pairs of schema. They also compute different local scores which are then combined for the final decision. Other relevant but less related work is for example [3, 4, 5].

6. FUTURE WORK

This is preliminary work, the model still needs enhancements, more analyses and tests on other collections have still to be performed. The model could be enriched by using additional information (e.g. tag names, etc). However the proposed model has already shown very encouraging results on a medium size XML collection.

7. ACKNOWLEDGEMENTS

This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

8. REFERENCES

[1] Rahm E. and Bernstein P., 2001, A survey of approaches to automatic schema matching, *The VLDB Journal, 10*, 4, 334-350

[2] Doan A., Domingos P., and Halevy A., 2003, Learning to Match the Schemas of Data Sources: A Multistrategy Approach. *Machine Learning 50*, 3, 279-301

[3] Li W. and Clifton C., 2000, SEMINT: A tool for identifying attribute correspondences in heterogeneous databases using neural networks, *Data & Knowledge Engineering 33*, 1, 49-84

[4] Berlin J. and Motro A., Autoplex, 2001: Automated Discovery of Contents for Virtual Databases. *Proceedings of COOPIS-01, Sixth IFCIS*, 108-122.

[5] Melnik S., Garcia-Molina H., and Rahm E., 2000, Similarity Flooding: AVersatile Graph Matching Algorithm and its Application to Schema Matching, *18th International Conference on Data Engineering*

[6] Madhavan J., Bernstein P., Chen K., Halevy A., 20003, and Shenoy P., Corpus-based Schema Matching, *Workshop on Information Integration on the Web at IJCAI* 2003.

[7] Carrasco R. and Rico-Juan J., 2002, A similarity between probabilistic tree languages: application to XML document families. *Pattern Recognition, 36*,9, 2197-2199

[8] Lewis D., Naive bayes at forty, 1998, The independence assumption in information retrieval, *European Conference on Machine Learning*

The impact of leaf nodes relevance values evaluation in a propagation method for XML retrieval

Karen Sauvagnat
IRIT
118 route de Narbonne
31062 Toulouse Cedex 4
+33 5 61 55 68 99
sauvagna@irit.fr

Mohand Boughanem
IRIT
118 route de Narbonne
31062 Toulouse Cedex 4
+33 5 61 55 74 16
bougha@irit.fr

ABSTRACT
In this paper, we describe an IR approach to XML retrieval using content and structure conditions based queries. The XFIRM model we propose is based on a complete query language, derived from Xpath, and on a relevance values propagation method. We propose here to evaluate the importance of the weighting formula used to assign relevance values to leaf nodes prior to propagation. Our experiments are evaluated thanks to the INEX evaluation initiative. Results show a relatively high precision, which, depending on the query context, can be increased thanks to a combination of IR models.

Categories and Subject Descriptors
H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval models – *retrieval models*.

General Terms
Algorithms, Experimentation

Keywords
XML retrieval, XFIRM, relevance propagation

1. INTRODUCTION
One of the main advantages of the XML format is its ability to combine structured and un-structured (i.e. text) data. As a consequence, the granularity of the information processed by Information Retrieval Systems (IRS) dealing with XML documents is not necessarily the whole documents : indeed, the structure of XML documents allows IRS to retrieve information units (i.e. nodes of XML documents).The main challenge in XML retrieval is to thus retrieve the most exhaustive[1] and specific[2] information units [7] answering a given query. Approaches dealing with this challenge can be divided into two main sub-groups [3]. On the one hand, the data-oriented approaches use XML documents to exchange structured data (as for example whole databases). The database community was the first to propose solutions for the XML retrieval issue, using data-oriented approaches. Unfortunately, the suggested approaches typically expect binary answers to very specific queries. On the other hand, the document-oriented approaches consider that tags, inserted by documents creators, describe the documents logical structure. The IR community has adapted traditional IR approaches to address the user information needs in XML collection. The increased interest of the community has become apparent within the 2 past INEX evaluation initiatives [4] [5].

Fuhr et al. [6] proposed an augmentation method for processing XML documents. In this approach, standard term weighting formulas are used for indexing so called "index nodes" of the document. Index nodes are not necessarily leaf nodes, because this structure is considered to be too fine-grained. For computing inner nodes indexing weights, the weights from the most specific index nodes are propagated towards the inner nodes. During the propagation, the weights are down-weighted by multiplying them with a so-called augmentation factor. This way, more specific elements are preferred during retrieval.

The approach we describe in this paper is also based on an augmentation method. However, in our approach, all leaf nodes are indexed. Indeed, we think that even the smallest leaf node can contain relevant information (it can be a title or sub-title node for example). Moreover, the distance separating nodes in the document tree is a parameter used when propagating the relevance values in the tree. The goal of this paper is to evaluate the importance of leaf nodes relevance value calculation in the overall performance of the system. In section 2, we quickly present the XFIRM *(XML Flexible Information Retrieval Model*) model and the associated query language. Section 3 evaluates our approach via experiments carried out on the INEX collection.

2. THE XFIRM MODEL

2.1 The XFIRM Query language
The XFIRM model is based on a complete query language, that allows queries to be expressed with simple keyword terms and/or with structural conditions [9].

In its more complex form, the language allows hierarchical conditions on document structure to be expressed and the user can specify which type of element needs to be returned (thanks to the *te: (target element)* operator):

Some examples of XFIRM queries can be found below:

(i) // te: p [weather forecasting systems]

(ii) // article[security] // te: sec ["facial recognition"]

(iii) // te: article [Petri net] //sec [formal definition] AND sec [algorithm efficiency]

(iv) // te: article [] // sec [search engines]

[1] An information unit is exhaustive to a query if it contains all the required information

[2] An information unit is specific to a query if all its content concerns the query

This respectively means that (i) the user wants a paragraph about *weather forecasting systems*, (ii) a section about *facial recognition* in an article about *security* , (iii) an article about *Petri net* containing a section giving a *formal definition* and another section talking about *algorithm efficiency,* and *(iv)* an article containing a section about *search engines*.

When expressing the possible content conditions, the user can use simple keyword terms (or phrases), possibly preceded by + or - (which means that the term should or should not be in the results). Terms can also be connected by Boolean operators.

2.2 Query processing

The approach we propose to deal with queries containing content and structure conditions, is based on relevance values propagation. The query evaluation in XFIRM is carried out as explained below.

2.2.1 Query decomposition

Each XFIRM query can be decomposed in sub-queries SQ_i as follows:

$$Q = // SQ_1 // SQ_2 //.../te : SQ_j //.../SQ_n \quad (1)$$

Where the *te:* operator indicates which element is the target element. Each sub-query SQ_i can then be re-decomposed into elementary sub-queries $ESQ_{i,j}$, possibly linked by boolean operators. They are of the form:

$$ESQ_{i,j} = tg\ [q] \quad (2)$$

Where *tg* is a tag name and $q = \{t_1, ...t_n\}$ is a set of keywords, i.e. a content condition.

For example, the XFIRM query: *// te: article [search engines] // sec [Internet growth] AND sec [Google]*, can be decomposed into sub-queries and elementary sub-queries as follows:

SQ_1 = *article [search engines]*

SQ_2 = *sec [Internet growth] AND sec [Google]*

$ESQ_{1,1}$ = *article [search engines]*

$ESQ_{2,1}$ = *sec [Internet growth]*

$ESQ_{3,1}$ = *sec [Google]*

2.2.2 Evaluating the relevance of leaf nodes

The first step in query processing is to calculate the relevance values of each leaf node *ln* according to the content conditions (if they exist). Let $q=\{t_1,...,t_n\}$ be a content condition. Relevance values are evaluated using $RSV_m(q,nf)$ (Retrieval Status Value), where *m* is an IR model.

$$RSV_m(q, \ln) = f_{j=1,..,n}\ (q_j, w_j^{(ln)}) \quad (3)$$

where:

- q_j is the weight of the term t_j in the content condition *q*
- $w_j^{(ln)}$ is the weight of the term t_j in the leaf node *ln*
- *f* is a function that allows to aggregate the weights of the terms t_j for the leaf node *ln.*

2.2.3 Structure processing

2.2.3.1 Elementary sub-queries $ESQ_{i,j}$ processing

The result set $R_{i,j}$ of $ESQ_{i,j}$ is a set of pairs *(node, relevance*) defined as follows:

$$R_{i,j} = \{ (n, r_n) / n \in \{construct(tg)\} \text{ and } r_n = F_k\ (RSV_m(q, nf_k), dist(n,nf_k)) \} \quad (4)$$

Where : - r_n is the relevance value of the node *n*

- the *construct(tg)* function allows the set of all nodes having *tg* as tag name to be created

- the $F_k\ (RSV_m(q, nf_k), dist(n,nf_k))$ function allows relevance values of the leaf nodes nf_k being descendants of the node *n* to be propagated and aggregated, in order to calculate the relevance value of *n*. The distance $dist(n,nf_k)$ which separates the node *n* from the leaf nodes nf_k in the document tree (i.e. the number of arcs that are necessary to join *n* and nf_k) is used as a parameter during the propagation.

2.2.3.2 Subqueries SQ_i processing

Once each $ESQ_{i,j}$ has been processed, subqueries SQ_i are rebuilt using the commutative operators $\oplus_{AND}$ et $\oplus_{OR}$ defined below :

Definition 1 : Let $N = \{ (n, r_n) \}$ and $M = (m, r_m) \}$ *be* two sets of pairs (node, relevance)

$N \oplus_{AND} M = \{ (l, r_l) /$ *l is the nearest common ancestor of m and n or l=m (respectively n) if m (resp .n) is ancestor of n (resp. m) ,* $\forall$ *m, n being in the same document and* $r_l = aggreg_{AND}(r_n , r_m, dist(l,n), dist(l,m))\}$ (5)

$N \oplus_{OR} M = \{ (l, r_l) / l=n \in N$ *or* $l=m \in M$ *and* $r_l = r_n$ *or* $r_m \}$ (6)

Where $aggreg_{AND}(r_n , r_m , dist(l,n),dist(l,m)) = r_l$ defines the way relevance values r_n and r_m of nodes *n* and *m* are aggregated in order to form a new relevance r_l.

2.2.3.3 Queries processing

The result set of sub-queries SQ_i are then used to process the whole queries. As defined above, in each query, a target element is specified.

$$Q = // SQ_1 // SQ_2 //.../te : SQ_j //.../SQ_n$$

The aim of processing the whole query is to propagate the relevance values of SQ_i sub-queries to the nodes belonging to the result set of sub-query SQ_j (which defines the target element). For this purpose, non-commutative operators ∇ and Δ are defined below :

Definition 2 : Let $R_i = \{(n, r_n)\}$ and $R_{i+1} = \{\{m, r_m)\}$ be two sets of pairs (node, relevance)

$R_i \nabla R_{i+1} = \{(n, r_n) / n \in R_i$ *is ancestor of* $m \in R_{i+1}$ *and* $r_n = prop\_agg(r_n , r_m, dist(m,n))\}$ (7)

$R_i \Delta R_{i+1} = \{(n, r_n) / n \in$ *is descendant of* $m \in R_{i+1}$ *and* $r_n = prop\_agg(r_m , r_n, dist(m,n))\}$ (8)

Where $prop\_\_agg(r_n , r_m, dist(n,m)) \rightarrow r_n$ allows to aggregate relevance weights r_m of the node *m* and r_n of the node *n* according to the distance that separates the 2 nodes, in order to obtain the new relevance weight r_n of node *n* .

The result set R of a query Q is then defined as follows:

$$R = R_j \nabla (R_{j+1} \nabla (R_{j+2} \nabla \ldots)) \quad (9)$$

$$R = R_j \Delta (R_{j-1} \Delta (R_{j-2} \Delta \ldots)) \quad (10)$$

Indeed, this is equivalent to propagating relevance values of results set $R_{j+1}, \ldots, R_n$ and $R_1, \ldots, R_n$ respectively upwards and downwards in the document tree.

3. Experiments and results

Our experiments have been evaluated using the INEX evaluation initiative [4], on the 2003 SCAS (Strict Content and Structure) task. The INEX collection, 21 IEEE Computer Society journals from 1995-2002, consists of 12 135 documents with extensive XML-markup. Participants to INEX SCAS task have to perform CAS (Content and Structure) queries, which contain explicit references to the XML structure, and restrict the context of interest and/or the context of certain search concepts.

The INEX metric for evaluation is based on the traditional recall and precision measures. To obtain recall/precision figures, the two dimensions of relevance (exhaustivity and specificity) need to be quantized onto a single relevance value. Quantization functions for two user standpoints were used: (i) a "strict" quantization to evaluate whether a given retrieval approach is capable of retrieving highly exhaustive and highly specific document components, (ii) a "generalised" quantization has been used in order to credit document components according to their degree of relevance.

3.1 Leaf nodes relevance evaluation

In order to evaluate the impact of the model used for leaf nodes relevance values evaluation, we tested the following functions :

- $$RSV(q, \ln) = \sum_{i=1}^{n} w_i^q * w_i^{\ln} \quad \text{with} \quad (11)$$

$$w_i^q = tf_i^q * ief_i \text{ And } w_i^{\ln} = tf_i^{\ln} * ief_i$$

Where :

- tf_i is the term frequency in the query q or in the leaf node ln

- ief_i is the inverse element frequency of term i, i.e. $log\ (N/n+1)+1$, where n is the number of leaf nodes containing i and N is the total number of leaf nodes.

- $$RSV\ (q, \ln) = \sum_{i=1}^{n} tf_i^q * \frac{tf_i^{\ln} * (h_1 + h_2 * ief_i)}{h_3 + h_4 * \frac{l}{\Delta l} + h_5 * tf_i^{\ln}} \quad (12)$$

Where

- h_1, h_2, h_3, h_4 and h_5 are constant parameters, respectively set to 0.2, 0.8, 0.2, 0.7 and 1 for our experiments

- l is the length of the leaf node ln

- Δl is the average leaf node length

The second function is inspired by OKAPI [8] and SMART term weighting and is used in the full-text retrieval search engine Mercure [1]. The values of *h1, h2, h3, h4* and *h5* parameters have been set thanks to experiments on TREC collections [2].

3.2 Implementation issues

The transformation of INEX CAS queries to XFIRM queries was fairly easy.

During $ESQ_{i,j}$ processing, the most relevant leaf nodes are found, and for each of these leaf nodes, XFIRM looks for ancestors. In order to have a correct system response time, the propagation is stopped when 1500 "correct" ancestors have been found (i.e. ancestors having a correct tag name). When an INEX topic contains a condition on the article publication date, this condition is not translated in the XFIRM language, as propagation with a very common term (like a year) is too long. To solve this issue, queries are processed by XFIRM without this condition, and results are then filtered on the article publication date.

Moreover, a Dictionary index is used to find equivalent tags, according to INEX guidelines.

Finally, we use the following propagation functions while processing queries:

$$F_k(RSV\ (q, nf_k), dist\ (n, nf_k)) = \sum_{k=1..n} \alpha^{dist\ (n, nf_k)} * RSV\ (n, nf_k) \quad (13)$$

Where $\alpha \in]0..1]$ is a parameter allowing to adjust the importance of the distance between nodes in the different functions.
For our experiments, α is set to 0.9

$$aggreg_{AND}\ (r_n, r_m, dist\ (l, n), dist\ (l, m)) = \frac{r_n}{dist\ (l, n)} + \frac{r_m}{dist\ (l, m)} \quad (14)$$

$$prop\_agg(r_n, r_m, dist(m, n)) = \frac{r_n + r_m}{dist(m, n)} \quad (15)$$

3.3 Runs

We tested 4 runs, described below:

1. **xfirm.T.tf-ief** : function (11) is used. Only the title field of INEX CAS queries is used for query indexing.
2. **xfirm.TK.tf-ief** : function (11) is used. Title and Keywords fields of INEX CAS queries are used for query indexing.
3. **xfirm.T.Mercure** : function (12) is used. Only the title field of INEX CAS queries is used for query indexing.
4. **xfirm.TK.Mercure:** function (12) is used. Title and Keywords fields of INEX CAS queries are used for query indexing.

In addition, these runs were compared to the best run we performed last year in the INEX SCAS task with a "fetch and browse" method: **Mercure2.pos_cas_ti** [10].

3.4 Analysis of the results

For all 5 runs, we issued 30 queries, expressed with the XFIRM language. The 5 runs resulted in 10 average precision (for strict and generalized quantization) that are shown in Table 1. The associated recall-precision curves for strict quantization are plotted in Figure 1.

Table 1: Average precision for the 5 runs

	Average precision (strict)	Average precision (generalized)
Xfirm.T.tf-ief	0,2675	0,2276
Xfirm.TK.tf-ief	**0,2898**	**0,2300**
Xfirm.T.Mercure	0,2448	0,2032
Xfirm.TK.Mercure	0,2467	0,2154
Mercure2. pos_cas_ti	0,1620	0,1637

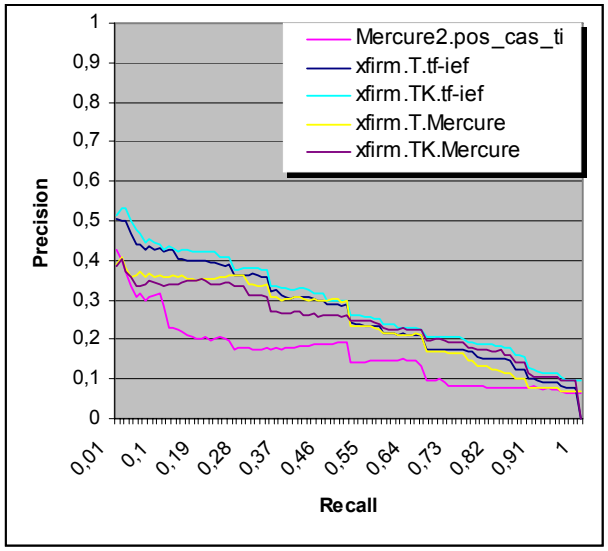

Figure 1: Recall/precision curves for strict quantization

The first point to notice is the relatively high precision for all runs. Almost all of them would have been ranked in the top ten of official INEX evaluation, with *Xfirm.T.tf-ief* and *Xfirm.TK.tf-ief* ranked third for strict quantization (Table2).

Table 2: Ranking of official INEX submissions and of our runs for strict quantization

rank	Avg precision	Organization	Run ID
1	0.3182	U. of Amsterdam	UamsI03-SCAS-MixedScore
2	0.2987	U. of Amsterdam	UamsI03-SCAS-ElementScore
	0.2898		***Xfirm.TK.tf-ief***
	0.2675		***Xfirm.T.tf-ief***
3	0.2601	Queensland University of Technology	CASQuery_1

The propagation method increases in a very significant way the results we obtained with a fetch and browse method (run *Mercure2.pos_cas_ti*).

The use of title and keywords fields of INEX topics increases the average precision of both runs *Xfirm.TK.tf-ief* and *xfirm.TK.Mercure* comparing to the runs *.Xfirm.T.tf-ief* and *Xfirm.T.Mercure*, even if there is a loss of precision for some particular queries.

The IR model used for evaluating leaf nodes relevance values seems to have a high impact. The simplest formula (11) which does not take into account the leaf nodes length obtains the best results. However, if we examine the results for each query, formula (12) is more efficient when the target element of the CAS query is a whole article. If we combine the two runs *xfirm.T.tf-ief* and *Xfirm.T.Mercure*, using *Xfirm.T.Mercure* only when the target element is an article, we obtain almost 10,54 % of improvement for strict quantization, which seems to signify that the choice of weighting formula for leaf nodes relevance value calculation might depend on the query context.

Our methods need to be carried out on other topics/collections to confirm these performances. We view this work as a starting point for finding an appropriate formula for leaf nodes relevance value evaluation. Moreover, an evaluation of propagation functions has to be done.

4. REFERENCES

[1] Boughanem, M., Chrisment, C., Soule-Dupuy, C. : *Query modification based on relevance back-propagation in ad-hoc environment.* Information Processing and Management, 35 (1999).

[2] Boughanem, M. , Chrisment, C., Tmar, M.: *Mercure and MercureFiltre applied for Web and Filtering tasks at TREC-10*. In : TREC-10, Gaithersburg, Maryland (USA), Nov. 2001.

[3] Fuhr, N., Grossjohann, K. "*XIRQL: A query Language for Information Retrieval in XML Documents*". In Proc. ACM SIGIR, New Orleans, USA, 2001.

[4] Fuhr, N., Malik, S., Lalmas, M : *Overview of the Initiative for the Evaluation of XML Retrieval (INEX) 2003*. In Proceedings of INEX 2003 Workshop, December 2003.

[5] Gövert, N., Kazai, G. : *Overview of the Initiative for the Evaluation of XML Retrieval (INEX) 2002*. In Proceedings of INEX 2002 Workshop, December 2002.

[6] Gövert Norbert, Abolhassani, M., Fuhr, N., Grossjohann, K. : *Content-oriented XML retrieval with HyREX*. In Proceedings of the first INEX Workshop, December 2002.

[7] Lalmas, M., "*Dempster-Shafer theory of evidence applied to structured documents: modeling uncertainty*". In Proc. ACM-SIGIR, Philadelphia, 1997.

[8] Robertson, S.E., Walker, S., Hancock-Beaulieu, M.M. : *Okapi at TREC 3*. In Proceedings of the 3rd Text Retrieval conference (TREC 3), NIST, 1994, Virginia, USA.

[9] Sauvagnat, K., Boughanem, M. : *Le langage de requête XFIRM pour les documents XML*. Actes du Congré Inforsid 2004, Biarritz, France.

[10] Sauvagnat, K., Hubert, G., Boughanem, M., Mothe, J. *: IRIT at INEX 03*. In Proceedings of INEX 2003 Workshop, December 2003

Filtering in XML Retrieval: a Prospective Analysis

Huyen-Trang Vu
LIP6, Université Paris 6
8, rue du capitaine Scott
75015 Paris, France
vu@poleia.lip6.fr

Benjamin Piwowarski
LIP6, Université Paris 6
8, rue du capitaine Scott
75015 Paris, France
bpiwowar@poleia.lip6.fr

Patrick Gallinari
LIP6, Université Paris 6
8, rue du capitaine Scott
75015 Paris, France
gallinar@poleia.lip6.fr

ABSTRACT

In XML retrieval paradigm, elements inside a document can be returned as answers to a user request. Since the information in an element is more specific than in a whole document, this might reduce the user effort in finding relevant information. However, as XML documents are composed of nested elements, many of which being possibly relevant to the user information need, retrieval systems must take care of the overlap issue before showing answers to the user. In this paper, we investigate how to disallow overlapping results by considering it as a filtering problem.

1. INTRODUCTION

With the widespread use of the eXtensible Markup Language (XML), representation formats for structured document like e.g. DocBook have been adopted as standards. New tools are being developed in order to manage, store and retrieve information from XML documents. In the information retrieval (IR) community, XML Retrieval has been considered as an important extension to standard IR: two SIGIR workshops were held on this subject [3, 1] and INEX (INitiative for the Evaluation of XML Retrieval) is now in its third year.

With XML documents, retrieval systems should identify the most specific elements that are relevant to a query, and not (whole) documents. As in passage retrieval, the underlying idea is to reduce the user effort needed to reach relevant information. User satisfaction not only depends on the ratio of irrelevant information but also on the amount of time (s)he has to spend reading redundant information. In flat document IR, this is related to the problem of redundancy and novelty detection where a document can contain information that has already been seen by the user. In the XML framework, redundancy arises from the logical organisation of the documents themselves: nested elements (e.g. a paragraph and its enclosing section) can be returned as relevant elements. The user will then be presented twice with the same information (the content of the paragraph).

An analysis of INEX'03 participants' submissions showed that an average of 28 % elements in the ranked list are duplicate, i.e. are either an ancestor or a descendant of an element ranked higher in the list, with 18 % in the latter (see fig. 1). In this paper, we examine different ways to avoid this type of redundancy. We consider the overlap issue as a filtering problem applied to a rank list of XML elements. It means that filtering is not performed by the retrieval engine itself but operates as a post-processing step on the retrieved list.

The article is organised as follows. In section 2, we describe related work. We give a formal description of the problem in section 3, and present our proposed approaches in section 4. In section 5, we describe some experiments to conduct.

2. RELATED WORK

Let us first examine how redundancy has been handled in different IR settings like passage retrieval and hypertext IR.

"Flat document" IR aims at retrieving documents that fulfil a user information need. It has been argued that it would be useful to present to the user a compact list in which documents with similar content have been removed [2, 21, 19]. This idea of novelty/redundancy detection has been extended to the case of passage retrieval even at sentence level [17]. Since novelty detection is difficult to handle while computing the relevance score of a document, all approaches to that task assume that the input is a set of relevant documents returned by an IR system. From that point of view, two different approaches have been used in the literature:

Adaptive filtering in which documents are examined one after another in some order and a binary decision (either keep it or remove it) is made for each document [21, 17].

Re-ordering where a new score is computed for each document by taking into account its relation with those already seen in the list [2, 19]

In order to detect novelty/redundancy, different techniques have been proposed, ranging from simple ones (based on word occurrences) to more elaborate ones (modelling content by language models).

In hypertext IR, it is interesting to retrieve documents that are *entry points* to relevant ones [7, 11]. Simple techniques have been proposed for finding BEP (Best Entry Point) inside relevant documents. They are similar to those used in adaptive filtering. A simple one consists in removing a document whenever there is an incoming link from another document ranked higher in the list [7]. For hierarchically (or tree) structured web sites, Lalmas and Moutogianni [11] propose to compare the score of a document to that of its children (linked documents) scores and a binary decision is then taken.

In the context of structured document retrieval task, Chiaramella et al. [4] introduce the notions of specificity and exhaustivity of an element with respect to a given query. An element is specific if it only discusses topics of the user query. It is exhaustive if it covers all aspects of the query. They also require that answers are consistent meaning that returned elements must be completely distinct one from the other. Searching for the most specific and most exhaustive elements is thus a recursive process. Though this is an interesting idea, their approach is purely logical and difficult to use with real systems that are not based on logic framework.

XML retrieval systems have used simple strategies to tackle the overlapping issue. Some of them depend on the score value of the document element, whereas others do not.

Cui et al. [6] use a simple criterion to select non-nesting elements before estimating their relevance score. First of all, a candidate set is created from elements containing at least one query term. If in the candidate set, there are two elements with a containment relation, the ancestor will be removed. This means that the deepest element in the document tree is preferred by this selection method. Extensions of hypertext filtering were proposed by Lalmas et al. [12, 10]. They try to identify Best Entry Points (BEP) in documents with different rules that take into account the ratio of relevant children over all children of an element, the number of relevant consecutive siblings or the average relevance value of each layer in the document tree.

Crestani et al. [5] re-score an element through expected utility (EU) values of two possible decisions: retrieve (EU^+) or not (EU^-) the element. They suggest that new score for an element can then be either $\mathtt{EU}^+$, $\mathtt{EU}^+ - \mathtt{EU}^-$ or $\frac{\mathtt{EU}^+}{\mathtt{EU}^-}$. Preliminary experiments on a small collection have shown an improvement in term of mean average precision. However, the utility values used for the computation of EU are empirical and the metrics they used are not fully adapted to XML retrieval evaluation.

In INEX'03, participants proposed different methods to overcome the overlapping problem. [13] suggested rules for merging local retrieved results while [14, 16] used simple filtering over the ranked lists. [18] selected elements by a coverage criterion computed from the number of query terms in each element. Evaluating the impact of these different strategies is difficult. There are two main reasons for that. Firstly, the metrics used to evaluate XML retrieval systems reflect quite indirectly the overlap problem. Secondly, it is difficult to make the distinction between the impact of the retrieval system itself and that of the filtering module.

Empirical evidence in XML IR has already shown that filtering is an important problem. It has not been systematically studied yet and only rather heuristic approaches have been tested. In the remainder of the article, we present a number of ideas that could be used in order to study more precisely the XML filtering problem.

3. PROBLEM FORMULATION

3.1 Problem Description

We aim to handle a set of XML elements. Each element has an associated score (RSV). The RSV of an element should measure the relevance of the element with respect to a given query in the context of XML retrieval.

The output of the filtering step is a ranked list. This list must be *consistent*, i.e. should contain as few redundant elements as possible. It should also be *optimal*, i.e. highly relevant elements should have the highest scores.

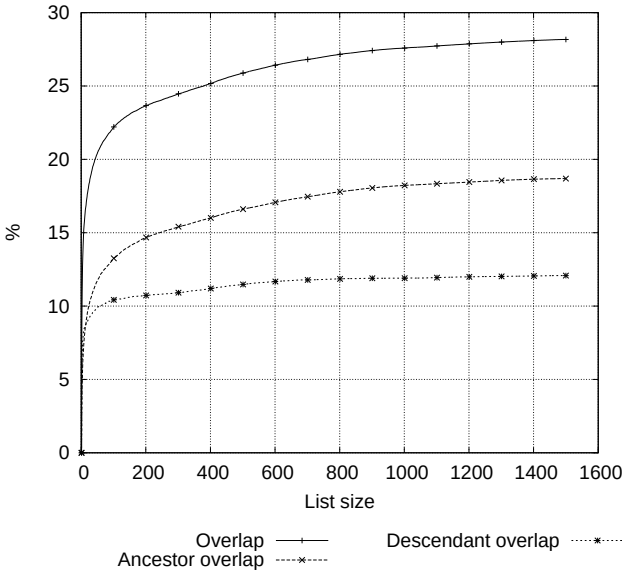

Figure 1: Overlapping statistics are averaged over all INEX'03 submitted lists. The curve ancestor (resp. descendant) gives the % of elements in the list for which at least one ancestor (resp. descendant) is ranked higher in the list. The last curve (overlap) gives the % of elements for which at least either an ancestor or a descendant is ranked higher in the list.

3.2 Problem Analysis

In standard IR, retrieved documents are ranked by decreasing order of relevance score (RSV) for a given query. The implicit assumption is that the documents are *independent* one from each other. This ranking principle does not hold in cases where a large number of potentially relevant answers are highly redundant [2]. Moreover, in XML retrieval, redundancy means physically duplicative information as elements are nested. In the remainder of this section, we describe which properties an XML element should fulfil in order to be relevant and less redundant to an user.

Assumptions

We assume that the final answers are ranked *linearly* since in our opinion, despite the development of advanced visualisation tools, users still prefer ranked lists. We furthermore suppose that each retrieved element is a *meaningful* information unit, which can be consulted alone without need of context information. The assumption that the user *reads carefully* each answer is reasonable because of the above assumption of informative unit. A consequence of this assumption is that an element and its descendants should not be both retrieved.

As in standard IR, the user might stop after consulting a certain number of answers. Highly relevant elements should therefore be ranked higher in the list: this is related to the *optimality* criterion described below.

Desiderata for answers

Answers are required to be *consistent*: users may be discouraged from viewing more elements and may feel lost if they are directed to too much redundant information.

The filtered list should be *optimal*: highly relevant elements should be ranked first. This point is important since filtering may remove some nested elements with a higher RSV. For example, consider the case of a section with a RSV of 0.8 and its paragraph with a RSV of 0.6. If the section is filtered out because it is not specific enough, the user might not see the paragraph as it has a smaller score. Since this paragraph "represents" its section, it should be ranked higher as the section was removed.

While it is easy to check if an element is enclosed into another, it is not trivial to determine an optimal policy. It should be a compromise between the RSV, the structural relationships with other elements and user preferences.

Despite the importance of these properties for a user, we argue that the filtering process must be *simple* enough, that is, it must not take too much time.

3.3 Filtering process

A filtering process can be decomposed into three distinct phases: (1) initial re-scoring, (2) filtering and (3) final re-scoring. Phase (1) will allow to include some information about the relations between document elements. Phase (2) is a purely logical process: a binary decision (either keep or remove) is taken for each element. Phase 2 ensures the list is *consistent*. The purpose of the last phase (3) is to ensure the optimality of the list. Existing attempts to filtering either performed phase (1) or (2). None did both. The optimality problem (phase 3) has not been considered.

4. PROPOSAL

In this section, we present in detail our proposal by following the three phases described in the previous section.

4.1 Re-scoring

RSV scores are usually computed independently for the different elements. It might be useful to integrate some more information related to the nested relationships between elements, i.e. to score an element with respect to its context.

4.1.1 Utility theory

The idea is to use the EU as in [5] whenever the underlying model outputs RSV that are probabilities. With respect to [5], we plan to perform further experiments in order to fully investigate this approach: in particular, we plan to use machine learning techniques to learn the utility values for the fact of showing an element (or not), knowing that the element is relevant (or not) and that its parent is relevant (or not). We denote this approach **RS1**.

One of our retrieval model is based on Bayesian Networks [16]. In this model, an element can obtain three distinct states: not relevant, too big (exhaustive but not really specific) or exact (exhaustive and specific). For each element, we thus have two independent values (as the three values sum to 1). Furthermore, the extra information brought by the "too big" state could be used. We plan to use the EU methodology with the probabilities delivered by the Bayesian Network model (**RS2**).

4.1.2 Novelty score

The general idea is to move elements one by one from a given set (the list of retrieved elements) to another (the set of re-scored elements). Each selected element maximises a criterion, which is a balance between relevance and novelty: this value will become new score for that element.

Formally, let us consider the sets E_t of n elements to re-score and the set E'_t of re-scored elements at a given time t. Note that at the beginning of the process ($t = 0$) $E'_0 = \emptyset$ and that at the end of the process ($t = n$) $E_n = \emptyset$.

Like other works on novelty detection, we assume the *relevance* of an element to a given query is independent of its *novelty* with respect to a given set of elements. This means that the novelty of an element with respect to a set of elements is independent of query. At a given time t, the new value $RSV_t^{(1)}$ of an element e in E_t is defined as [2]:

$$RSV_t^{(1)}(e, q, E'_t) = \lambda RSV(e, q) + (1 - \lambda) Nov(e, E'_t) \quad (1)$$

where $\lambda \in [0, 1]$ is a parameter, $RSV(e, q)$ is the score of the retrieval system for element e with respect to the query q, $Nov(e, E_t)$ is the novelty of element e with respect to a set of elements E_t. For simplicity, we can define

$$Nov(e, E'_t) = \min_{e' \in E'_t} Nov(e, e') \quad (2)$$

where $Nov(e, e')$ is the novelty of an element e with respect to another element e'. This value can be approximated through the size of elements in an appropriate unit (number of words, number of characters, etc.)

$$Nov(e, e') = f\left(\frac{length(e)}{length(e')}\right) \quad (3)$$

The element e^* with the highest value $RSV_t^{(1)}$ will be selected: $E'_{t+1} = E'_t \cup \{e^*\}$ and $E_{t+1} = E_t \setminus \{e^*\}$. The new score of the element e^* is $RSV_t^{(1)}(e^*)$. The process is then iterated until the set E_t is empty (**RS3**).

Zhai et al. pointed out that the linear combination is only meaningful when similarity measures $RSV()$ and $Nov()$ are in the same scale [20]. It does not hold in some cases. We

therefore can use another criterion presented in [20] (**RS4**), the so-called *cost-based* measure:

$$RSV_t^{(2)}(e) = RSV(e,q)(\rho + Nov(e, E'_t))$$

where $\rho \geq 0$ is a parameter.

4.2 Filtering

In this section, we propose some filtering methods. We distinguish two kinds of filtering. The first one is related to adaptive filtering, whereas the other is based on a recursive processing of the document tree. In the remainder of this section, we are interested in both strict (no overlap) and tolerant (some overlap) filtering.

4.2.1 List filtering

The first kind of filtering methods examines elements in the list one by one , beginning with those with the highest RSV. Let us denote $E = (e_1, ..., e_n)$ the ordered list of n elements (either by decreasing initial score values delivered by the retrieval system or by re-scoring values) and FL_{i-1} the filtered list after $(i-1)$ iterations of this process. The latter list can thus contain up to i elements.

We will also denote by $anc(e)$ (resp. $desc(e)$) the set of ancestors (resp. descendants) of an element e in the document tree.

The rules we plan to experiment are as follows:

Rule F1 (no descendant). With this rule, an element is removed whenever its ancestor is already in the filtered list. This is related to the assumption we made about the user behaviour: when (s)he consults the list, (s)he reads the elements entirely. (S)he has already been reading the content of the element if its ancestor is in the list. Formally, if $anc(e_i) \in FL_{i-1}$ then the element is not added to the list.

Rule F2 (no ancestor). This rule is symmetric to rule F1: an element will not be presented to the user if one of its descendant is already in the list. The argument is that if the relevant information in the element in consideration belongs to the descendant that has been seen, it is useless to show to the user the duplicate information even if this is relevant. For example, if a section contains only one paragraph and if the paragraph has been presented to the user, it is redundant to show her/him the section. Formally, if $desc(e_i) \in FL_{i-1}$, the element is not added to the list. While this rule can be useful to remove redundant information, it seems too harsh when the element already in the list is really small compared to the element which is filtered: consider for example the case of a snippet in italics and its enclosing section.

Rule F3 (selective ancestors) We want to relax rule F2, in order to allow redundancy in some cases by estimating the possibility that e_i contains novelty with respect to elements of FL_{i-1}. The novelty value of e_i with respect to FL_{i-1} is defined in eq. (2). If $Nov(e_i, FL_{i-1}) < Nov_{Threshold}$ then the element is removed. $Nov_{Threshold}$ is a parameter which can be fixed empirically or learnt from data.

4.2.2 Filtering on the document tree

In the previous section, we have shown how it is possible to reduce some overlap using an approach related to adaptive filtering. It is also possible to process each document in turn, and to select a set of elements within that document in a way that ensures no overlap in the final list. We describe such approaches in this section.

For each node, starting from the root of the document tree, we take a binary decision: either we keep this element (and we do not consider its children) or we return at least one of its descendants by applying the same rule to each of its children. This algorithm guarantees the non-overlapping of selected elements as the recursive process is stopped whenever we return an element.

The criterion used to take the binary decision can be simple. For each element e in the tree, e is not added in the list:

Rule F4 if the ratio of relevant elements among its children is superior to a given threshold, or

Rule F5 if the average score of relevant elements is superior to a given threshold.

These criteria are simple enough to experiment. However, it is possible to do better. Our idea is to use an existing metric and try to optimise the list of returned elements according to this measure. We choose the Expected Ratio of Relevant Units, which is a generalisation of standard recall [15] in the context of XML retrieval. We thus denote GR this measure. The probability that an element is consulted by the user is defined to be proportional to the score of the element in consideration:

$$P(e \text{ is seen}) \propto RSV(e,q)$$

This probability is used while computing the value of the measure GR. We can then compute (1) the expected value of GR if only the parent were in the list and compare it to (2) the expected value of GR if only the children were in the list. If the latter is superior to the former, the element is discarded and the recursive process is iterated over its children (**F6**).

4.3 Final scoring

For the final score, we can apply two different strategies. The first one is to keep the score (**FS1**), the other one is to get the maximum RSV of elements which were removed (**FS2**).

5. PROPOSED EXPERIMENTS

Since we focus on processing the result from a retrieval model, the results are not independent of the retrieval system performance. In order to examine the respective capabilities of filtering methods to know which filtering rules should be applied to a given system, we need to know exactly the retrieval system behaviour, which is usually unrealistic. We therefore choose indirect approaches by either analysing the effect of the filtering strategies over representative runs in the INEX'03 collection or by generating a hypothetical retrieval system for which we can control the behaviour.

Note that rules F1 and F2 described in the last section were already used in some official submissions in INEX'03 [14, 16]. However, according to all official measures used in INEX'03, these runs are poor, none of them are in the top 10 while their initial versions without filtering are considered good. The first phase of our experiment aims at understanding this in detail: if their poor performance is due to an unsuitable application of filtering strategies or due to evaluation metrics.

Furthermore, in order to investigate our different filtering propositions without technical dependence on a particular retrieval system, we aim at creating a hypothetical retrieval system which calculates relevance scores based on the assessments of the INEX collection. In this case, it is necessary to build such a hypothetical system because in $RS1$ we need probability values while these are not always supported by some retrieval models, and our filtering strategies might reduce the number of answers, which make any relative comparison between systems not very meaningful. The INEX scale [9] used to assess elements with respect to a given query has two dimensions: exhaustivity and specificity. We will need to transform assessments into scores in real values. Later, in order to simulate the behaviour of a search engine, we will add some noise to this score function. As this noise should be related to real noise, we will compute for a sample of retrieval systems the distribution of scores with respect to a given assessment. The deterministic transformation of an assessment a into a RSV will then be followed by a random sampling with respect to the probability distribution computed for this assessment.

The previous suggestions (except RS2 which is specific to one of our retrieval model) are independent of any retrieval method and in some cases, they can be used together without conflict such as F1 and F2 (or F3). We want to examine such combinations to find out suitable processing method for each type of retrieval models.

In our proposal, there are some parameters such as utility values in 4.1.1 or thresholds in 4.2.1. We want to check their effects to the final results by comparing the stability of final results when fixing these parameters empirically and when training them from the data.

Quantitative evaluations are very important for confident conclusion over experimental results. For re-scoring methods, we can use statistic indices such as Spearman's rank correlation to check the significantly statistical difference between pairs of ranked lists: the original returned by the retrieval model and that after re-scoring, each of them with respect to that created from the assessment set (after transforming assessments into scores in real values).

For filtering strategies, it is more of a challenge because the number of elements has changed, we can not use recall-based measures over the same reference set for the list before and after filtering. The filtering effect will therefore be evaluated either with new reference set or by other indications, for example the precision value, the stability with respect to some parameters such as number of topics needed or size of test collection.

6. CONCLUSION

In this article, we have presented our planned work on XML filtering. We first motivated the need of XML filtering to reduce the user effort to access relevant information. Data analysis on the INEX'03 runs have shown that a high percentage of elements are redundant. We thus aim at removing redundant elements from the retrieved list before presenting results to the user.

Based on assumptions on the user behaviour, we described what are our requirements for XML filtering. Filtering is considered as a three phases process: re-scoring, filtering and final scoring. For each of these phases, we proposed different techniques which we plan to test. The experiments will first investigate the effect of the different filtering components in either available or an "ideal" environment, to clearly put in evidence the effect of the different filtering strategies.

7. REFERENCES

[1] R. Baeza-Yates, N. Fuhr, and Y. S. Maarek, editors. *ACM SIGIR 2002 Workshop on XML*, Aug. 2002.

[2] J. Carbonell and J. Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual Int. ACM SIGIR conference on Research and development in Information Retrieval*, pages 335–336. ACM Press, 1998.

[3] D. Carmel, Y. Maarek, and A. Soffer, editors. *ACM SIGIR 2000 Workshop on XML*, July 2000.

[4] Y. Chiaramella, P. Mulhem, and F. Fourel. A Model for Multimedia Information Retrieval. Technical report, IMAG, Grenoble, France, July 1996.

[5] F. Crestani, L. M. de Campos, J. M. Fernández-Luna, and J. F. Huete. Ranking Structured Documents Using Utility Theory in the Bayesian Network Retrieval Model. In M. A. Nascimento, E. S. de Moura, and A. L. Oliveira, editors, *SPIRE 2003*, volume 2857 of *Lecture Notes in Computer Science*, pages 168 – 182, Brazil, Oct. 2003. Springer-Verlag Heidelberg.

[6] H. Cui, J.-R. Wen, and J.-R. Chua. Hierarchical Indexing and Flexible Element Retrieval for Structured Document. In *Advances in Information Retrieval, 25th European Conference on IR Research, ECIR 2003, Pisa, Italy, April 14-16, 2003, Proceedings*, volume 2633 of *Lecture Notes in Computer Science*, pages 73–87, Pisa, Italy, Apr. 2003. Springer.

[7] M. E. Frisse. Searching for information in a hypertext medical handbook. *Communications of the ACM*, 31(7):880–886, 1988.

[8] N. Fuhr, M. Lalmas, and S. Malik, editors. *INEX 2003 Workshop Proceedings*, Dagstuhl, Germany, Mar. 2004.

[9] N. Fuhr, S. Malik, and M. Lalmas. Overview of the INitiative for the Evaluation of XML Retrieval (INEX) 2003. In Fuhr et al. [8].

[10] G. Kazai, M. Lalmas, and T. Rölleke. Focussed Structured Document Retrieval. In *9th Int. Symposium on String Processing and Information Retrieval SPIRE'02*, Lisbon, Portugal, Sept. 2002.

[11] M. Lalmas and E. Moutogianni. A Dempster-Shafer indexing for the focussed retrieval of a hierarchically structured document space: Implementation and experiments on a web museum collection. In *6th RIAO Conference, Content-Based Multimedia Information Access*, Paris, France, Apr. 2000.

[12] M. Lalmas and J. Reid. Automatic Identification of Best Entry Points for Focussed Structured Document Retrieval. In *CIKM Conference on Information and Knowledge Management*, New Orleans, Louisiana, USA, Nov. 2003. Poster.

[13] Y. Mass and M. Mandelbrod. Retrieving the most relevant XML Components. In Fuhr et al. [8].

[14] P. Ogilvie and J. Callan. Using Language Models for flat text queries in XML Retrieval. In Fuhr et al. [8].

[15] B. Piwowarski and P. Gallinari. Expected Ratio of Relevant Units: A Measure for Structured Information Retrieval. In Fuhr et al. [8].

[16] B. Piwowarski, H.-T. Vu, and P. Gallinari. Bayesian Networks and INEX'03. In Fuhr et al. [8].

[17] I. Soboroff and D. Harman. Overview of the TREC 2003 Novelty Track. In *Notebook Proceedings of TREC 2003*, 2003.

[18] A. Trotman and R. A. O'Keefe. Identifying and Ranking Relevant Document Elements. In Fuhr et al. [8].

[19] C. Zhai. *Risk Minimization and Language Modeling in Text Retrieval*. PhD thesis, Carnegie Mellon University, July 2002.

[20] C. Zhai, W. W. Cohen, and J. Lafferty. Beyond Independent Relevance: Methods and Evaluation Metrics for Subtopic Retrieval. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*. ACM, ACM Press, New York, NY, USA, 2003.

[21] Y. Zhang, J. P. Callan, and T. Minka. Novelty and redundancy detection in Adaptive Filtering. In *Proceedings of the 25th annual Int. ACM SIGIR conference on Research and development in information retrieval*, pages 81–88. ACM Press, 2002.

Workshop on the Integration of Information Retrieval and Databases (WIRD'04)

Preface

The first Workshop on the Integration of Information Retrieval and Databases (WIRD) explores the research area where data retrieval meets information retrieval. The purpose of this workshop is to bring together researchers of the database (DB) and information retrieval (IR) fields, facilitating exchange on the progress in developing and applying integrated IR+DB approaches (or, naturally, DB+IR solutions). Applications are numerous, including data warehousing, (semantic) web retrieval, heterogeneous and semi-structured data collections, semantically rich information systems, fact retrieval (query answering), free-text queries in relational and object-relational databases, etc.

The main goal of research on DB+IR technology is to improve the process of constructing advanced information systems. The 'database approach' allows more efficient and effective implementations, focusing on a higher level of data abstraction than the rather physical data structures and programming techniques used in IR. For achieving these potential benefits, DB+IR technology needs to satisfy on one hand the scalability and ranking requirements of IR-like applications, and on the other hand the data abstraction, manipulation, access and querying requirements of DB-like applications.

Three out of four submitted papers have been accepted. The first paper concentrates on a high level of abstraction for describing advanced retrieval systems. The second paper argues that XML-IR systems should be implemented following a traditional database architecture. The third paper makes a case for applying structured data in query expansion in a more traditional information retrieval setting. Summarising, we are pleased to see that the workshop has attracted papers covering perspectives on data abstraction and DB usage for IR.

We would like to thank the SIGIR reviewers who selected the workshop to be original and important. Further, we would like to thank the PC members for their high-quality reviews being excellent feedback to the authors. The result is an attractive programme that shall bring together the reseachers to shape the future of search applications.

Let the WIRD workshop be a forum for exchange on future DB+IR technology!

Thomas Roelleke, Queen Mary University London, London, UK
Arjen P. de Vries, CWI, Amsterdam, The Netherlands

Organisers and PC-chairs

Thomas Roelleke, Queen Mary University London, London, UK
Arjen P. de Vries, CWI, Amsterdam, The Netherlands

Programme Committee

Surajit Chaudhuri, Microsoft Research, Redmond, USA
Ingo Frommholz, Fraunhofer IPSI, Germany
Norbert Fuhr, University of Duisburg-Essen, Germany
Torsten Grabs, Microsoft, Redmond, USA
Djoerd Hiemstra, University of Twente, The Netherlands
Udo Kruschwitz, University of Essex, UK
Apostel (Paul) Natsev, IBM TJ Watson Research Cente, USA
Jayavel Shanmugasundaram, Cornell University, USA
Dan Suciu, University of Washington, Seattle, USA
Michael Taylor, Microsoft Research, Cambridge, UK

Annotation-based Document Retrieval with Four-Valued Probabilistic Datalog

Ingo Frommholz
frommholz@ipsi.fhg.de

Ulrich Thiel
thiel@ipsi.fhg.de

Thomas Kamps
kamps@ipsi.fhg.de

Fraunhofer IPSI
Integrated Publication and Information Systems Institute
Dolivostr. 15
D-64293 Darmstadt, Germany

ABSTRACT

The COLLATE system (collaboratory for annotation, indexing and retrieval of digitized historical archive material) provides film researchers with a collaborative environment in which historic documents about European films can be analysed, interpreted and discussed, using nested annotations and discourse structure relations among them. Annotations are metadata, and annotation threads form a hypertext containing positive and negative links, constituting a certain kind of context exploitable for document retrieval. In this paper, we discuss a solution for using annotations for information retrieval. To exploit annotation threads which consist of nested annotations and typed links between them, an annotation-based retrieval approach should have to cope with negative and contradictory statements. The nested annotation retrieval approach (NARA) is an approach addressing these issues. Based on this, we present NARAlog, an implementation using four-valued probabilistic datalog (FVPD), able to perform an in-depth analysis of annotation threads and to deal with contradictory statements.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Retrieval Models*

General Terms

Algorithms, Design

Keywords

Annotations, four-valued probabilistic logics, datalog, context-based retrieval, formal model

WIRD'04, the first Workshop on the Integration of Information Retrieval and Databases (WIRD'04), Sheffield, United Kingdom
© 2004 the author/owner

1. INTRODUCTION

Annotations are a means of supporting several tasks performed in Digital Libraries. They can assist the creation of new documents when new ideas and thoughts are discussed by means of annotations. Through annotations, users can interpret the document material at hand. Annotations support the effective use of documents by providing additional information which can make the content more intellectually accessible. Shared and public annotations can be used as building blocks for collaboration.

In this paper, we discuss how annotations can be exploited for information retrieval. Annotations made on documents can be seen as a special kind of *document context* containing additional information about the document. Using annotations for information retrieval means taking this context into account. From a syntactic point of view, annotations are a special kind of metadata [3]. They can appear in a simple form as direct comments on documents to more advanced forms like nested annotations with typed links connected to certain parts of documents. They can be textual, referential or graphical [2] and form a certain kind of hypertext (the so-called *annotation thread*) together with the document they are referring to. Annotations can contain content about content as well as additional content. They implicitly contain certain dialogue acts which might be made explicit by defining an appropriate annotation model. Nested annotations together with specific link types can be applied to model scientific discussions [6]. These discussions can contain additional content as well as possibly contradictory statements about the content of documents or annotations. Methods exploiting the annotation context for document retrieval should be able to perform an in-depth analysis of annotation threads in order to cope with the phenomena described above.

In the remainder of the paper, we will first briefly introduce the annotation model developed within the COLLATE system, which incorporates many features which can be found in recent annotation systems, such as nested annotations and typed links. We will then introduce our idea of a nested annotation retrieval approach (NARA) which, as the name implies, is capable of dealing with nested annotations as well as negative and contradictory statements. After that,

an example implementation of NARA based on four-valued probabilistic datalog (FVPD) [10] is introduced. Related work will be discussed and conclusions will be given.

2. THE COLLATE ANNOTATION MODEL

The COLLATE[1] system focuses on historic film documentation, dealing with documents about films of the 20s and 30s of the last century. Such documents can be, for example, censorship decisions, newspaper articles, etc. They are digitised and stored in the system repository. COLLATE supports the work between film scientists in different locations by establishing a collaboration cycle [9]: users can react to other user's contribution, and so the cycle continues. Users have the option of manually assigning keywords to the digitised documents as well as cataloguing them according to a pre-defined schema. One of the central concepts of COLLATE is to support document interpretation by enabling scientific discussion about documents through annotation threads.

Annotation threads consist of the annotated document (or a part of it) as root and nested annotations connected to the root. The links between the nodes of an annotation thread (documents and textual annotations) are typed with so-called *discourse structure relations*. In COLLATE, we defined the following relations: *elaboration* (giving additional information), *analogy* (describing similarities), *difference* (describing contrasts), *cause* (stating a cause for specific circumstances), *background information* (e.g., information about the background of an author), *interpretation* (of statements), *support argument* and *counterargument* (support or attack other arguments). Figure 1 shows an example of two discourse structure relations. The incorporation of these relations is discussed in more detail in [6]. Modeling annotation threads in this way gives us explicit information about the pragmatics of statements (through link types); this is important for the definition of suitable retrieval methods, as we will see later. An annotation thread in our case is a directed acyclic graph and forms a hypertext according to the definition in [1].

3. NESTED ANNOTATION RETRIEVAL APPROACH (NARA)

Since annotations contain valuable additional information about the document they are referring to, we will now briefly discuss how such information can be used for information retrieval. Due to their strong connection to documents, annotations can be seen as a kind of metadata. On the other hand, annotation threads build a rather complex hypertext [3], so advanced methods have to be applied.

Consider the example of an annotation thread as it is shown in Figure 1. The first annotation a_1 contains an interpretation of the content of d. A film scientist states that there might be other reasons for censoring the film d is talking about than described in d. a_1 extends the content of d, making it potentially interesting for users seeking for political censorship. But if we also take a_2 into account, we find a counterargument to a_1; another film scientist disagrees with the first one's opinion, so we have two contradictory statements. Ignoring a_2 or the link type between a_1 and a_2 would

[1] `http://www.collate.de/`

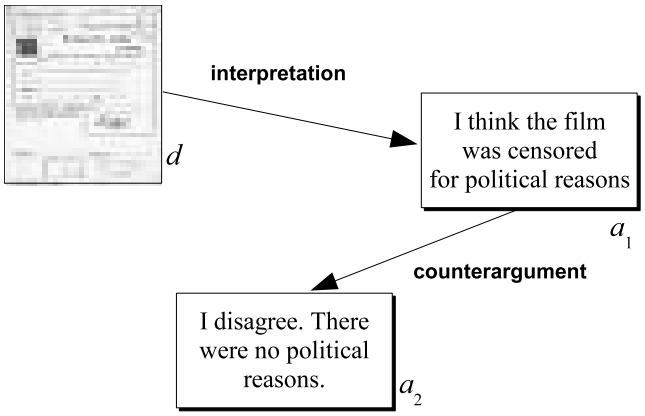

Figure 1: An example annotation thread

not be appropriate; whereas a_1 potentially raises the relevance of d when looking for political censorship, a_2 would lower it again. An in-depth analysis of the annotation thread has to be performed to cope with this situation accordingly, in essence with contradictory statements and nested annotations. This is addressed by our nested annotation retrieval approach (NARA).

Before discussing NARA, we provide a definition of an annotation thread.

Definition 1. Let A be the set of annotations and D be the set of documents. An *annotation thread* $\mathcal{A} = (N, E)$ is a directed acyclic graph with only one single root node $d \in D$. $N \subset D \cup A$ is the set of nodes in $\mathcal{A}$ representing documents and annotations, respectively; it is $n \in A$ if n is not the root node of $\mathcal{A}$. Let L be the set of link types; $E \subset N \times N \times L$ is the set of edges in $\mathcal{A}$ labeled with a link type $l \in L$.

In our framework, a query q consists of a set of query terms given by the user. We now outline our nested annotation retrieval approach which is a two-phase approach performing an in-depth analysis of annotation threads w.r.t. the query, using the content of annotations and the graph structure of the annotation threads.

1. *Initial content-based retrieval*: In this phase, the content-based retrieval status values (RSV) of each node in the annotation thread w.r.t. the query q are calculated. This is done using a retrieval function

$$r_{content} : N^D \times Q^D \longrightarrow \mathbb{R}$$

which maps a node description $n^D \in N^D$ of a document or annotation and a query description $q^D \in Q^D$ onto a real number. Depending on the underlying retrieval and indexing model, nodes and queries can be described as, e.g., vectors consisting of the weights of the terms contained in the document, annotation, or query, respectively. In the logic-based approach presented later, documents, annotations and queries are described using probabilistic facts.

Having calculated the RSVs for each node, we gain information to which degree these nodes are relevant w.r.t. the query q. This information is used for the analysis of the annotation thread, which is performed in the 2nd phase.

2. *Annotation-based re-weighting*: In this phase, the analysis of the annotation thread is performed in order to take the annotation context into account by biasing the initial retrieval status value $r_{content}(n^D, q^D)$ for each node d. The calculation of the context-based retrieval status value $r_{nara}(n, q)$ of a node n w.r.t. q can be described algorithmically as follows:

```
1: r_nara(n,q) = r_content(n^D, q^D)
2: if n is not a leaf in A then
3:    for all n' with (n,n',l) ∈ E do
4:       Calculate r_nara(n',q)
5:       r_nara(n,q) = f(r_nara(n,q), r_nara(n',q), l)
6:    end for
7: end if
```

The function f should stay undefined here. It biases the current weight $r_{nara}(n, q)$ using the context-based RSV of a successor node n' and the link type between n and n'. Implementations of NARA have to define f accordingly, especially in order to cope with inconsistent knowledge.

4. LOGIC-BASED IMPLEMENTATION OF NARA

In this section we will introduce NARAlog, which is an implementation of NARA using four-valued probabilistic datalog.

4.1 FVPD

Four-valued probabilistic datalog is an extension of probabilistic datalog. Similar to Prolog, its syntax consists of variables, constants, predicates and Horn clauses. Probabilities can be assigned to facts. Semantically, FVPD uses four different truth values (besides the classical ones, *true* and *false*) [10, 22]. These additional values are *unknown* and *inconsistent*. Basically, FVPD deals with an open world assumption (OWA). In contrast to a closed world assumption (CWA), the absence of an atom $p(a)$ does not imply $\neg p(a)$, but would assign the truth value *unknown* to $p(a)$ instead of *false*. In general, using four-valued logics and OWA, a model may contain a positive atom, a negative atom, neither of them or both. As an example, let the model $M = \{p(a), \neg p(a), p(b), \neg p(c)\}$ and the Herbrand base be $\{p(a), p(b), p(c), p(d)\}$. Then the following truth values would be assigned to the elements of the Herbrand base:

$p(a)$	*inconsistent*
$p(b)$	*true*
$p(c)$	*false*
$p(d)$	*unknown*

FVPD was originally developed for the retrieval of hypermedia documents and it integrates concepts coming from information retrieval (like uncertain inference) and deductive databases. In hypermedia documents, a similar situation might occur as we might have it with annotations: different nodes of a hypermedia document can contain contradictory statements. This can be handled with four-valued probabilistic logics, as the simple example in Figure 2 shows. Here, we see a hypermedia document d composed of two

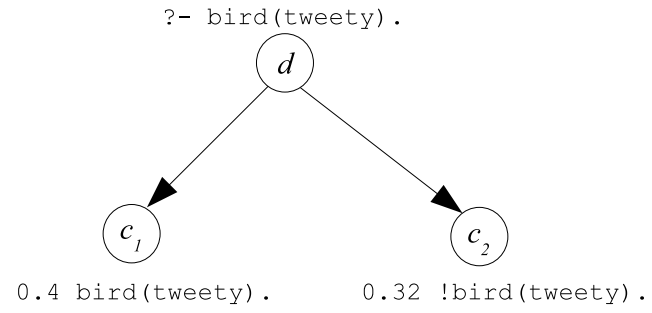

Figure 2: Hypermedia document with contradictory statements

chapters, c_1 and c_2. Each chapter constitutes its own context. Now consider the query `?- bird(tweety)`[2]. Both chapters c_1 and c_2 contain facts relevant to the query: c_1 provides evidence that `bird(tweety)` is true with the probability 0.4, and c_2 provides evidence that `bird(tweety)` is false with the probability 0.32. So we find contradictory information in both subparts of d (which is not contradictory in the subparts c_1 and c_2 themselves). To calculate the probability that `bird(tweety)` is true in the context of d, the positive evidence coming from c_1 (0.4) is combined with the non-negative evidence from c_2 $(1 - 0.32)$, resulting in $0.4 \cdot (1 - 0.32) = 0.272$. This process is called *knowledge augmentation* and it is discussed in more detail in [10, 22]. With HySpirit[3] there exists an implementation of FVPD [10].

4.2 NARAlog

The process of knowledge augmentation dealing with contradictory statements coming from subparts of hypermedia documents is similar to the process of dealing with such knowledge in an annotation thread. In hypermedia documents, we consider subparts (like c_1 and c_2 in the example described above) in order to calculate the final probability for a fact to be true in a higher context (like d). With NARA as described in Section 3, we consider direct annotations in the re-weighting phase. This makes FVPD an interesting framework for a logic-based implementation of NARA, which we call NARAlog.

NARAlog is based on the view of information retrieval as uncertain inference, as proposed in [23]. The retrieval function thus estimates the probability $P(d \rightarrow q)$ that a document d implies a query q. We state that an open world assumption is more suitable for our approach than a closed world assumption since we have to deal with positive and negative evidence in annotations which should be handled independently.

[2]Queries are formulated as goal clauses in FVPD
[3]`http://qmir.dcs.qmul.ac.uk/hyspirit.html`

NARAlog underlies some assumptions on the annotation collection. We assume that we have an annotation thread similar to the one used in COLLATE, and that the link types can be categorised in positive and negative ones w.r.t. their effect on the retrieval weight of the link source (see the discussion below). Link types are explicitly given or derived automatically with some uncertainty (methods determining link types are not an issue here). Annotations are atomic in that their pragmatics are consistent with the according link type, e.g. an annotation being a counterargument would not contain an additional interpretation. We assume that in this case users would create two corresponding annotations.

4.2.1 Components

Several components are needed in NARAlog, which are content-based retrieval weights of both documents and annotations, document and link types, positive and negative links and access probabilities. These components will be discussed now.

4.2.1.1 Content-based Indexing and Retrieval

In NARA, the results of the first phase are content-based retrieval weights determined by a retrieval function $r_{content}$, which is based on document and query descriptions usually derived by an indexing process. NARAlog indexes documents and annotations as probabilistic facts, using the predicates `term` and `termspace`. `term` is a binary predicate describing term weights w.r.t. documents. As an example, the probabilistic facts

```
0.2 term(d1, "political").
0.3 term(d1, "censorship").
```

would mean that the document `d1` contains the terms "political" and "censorship" and their weights are 0.2 and 0.3, respectively. These weights might be, e.g., normalised term frequencies. The facts

```
0.3 termspace("political").
0.5 termspace("censorship").
```

provide another weight to a term which is independent of the documents but unique for the whole termspace. This value might be, e.g., the inverse document frequency.

Query terms are described as probabilistic facts in a similar way. Suppose the query is "political reasons", we would create the following facts:

```
qterm("political").
qterm("reasons").
```

(if no probabilistic weight is given, 1 is assumed).

Based on these facts, a content-based retrieval function implementing the initial NARA phase can be described as the following rule:

```
r_content(N) :- qterm(T) & termspace(T) &
                term(N,T).
```

Because of the query term "political", this rule would yield $1 \cdot 0.3 \cdot 0.2 = 0.06$ for `d1`.

4.2.1.2 Document and Link Types

Nodes in an annotation thread have to be classified whether they are documents or annotations. This can be done using the facts

```
document(d1).
annotation(a1).
annotation(a2).
```

which mean that `d1` is a document and `a1` and `a2` are annotations.

Furthermore, we have to model links of a certain type. This can be done, e.g., by creating appropriate binary predicates representing links, as can be seen in the following example.

```
interpretation(d1,a1).
counterargument(a1,a2).
```

These facts represent the links as they are found in Figure 1. In a system like COLLATE where the relation types are explicitly given, we can assign the probability 1 to each link predicate if the corresponding link appears between two nodes. However, other systems might not offer explicit link types, but have to automatically derive them from, e.g., the content of the link's source or destination node. This means such link types are estimated with a degree of uncertainty which can manifest itself in probabilistic weights less than 1.

4.2.1.3 Positive and Negative Links

Link types should be categorised w.r.t. the effect their destination nodes have on the calculation of the retrieval weight of the source node. Consider the example in Figure 1, annotation a_1, being an interpretation, would raise the context-based retrieval weight of d, but never lower it. On the other hand, the effect of the counterargument relation between a_1 and a_2 would mean that the context-based weight of a_1 is lowered according to the content-based weight of a_2 w.r.t. the query, which in turn would decrease the overall retrieval weight of document d. It should be noted that if we had a counterargument a_3 to a_2, this would lower the context-based weight of a_2 and in turn raise the weight of a_1.

The categorisation of link types into positive and negative ones can be done by creating appropriate rules like

```
pos_link(X,Y) :- interpretation(X,Y).
pos_link(X,Y) :- elaboration(X,Y).
neg_link(X,Y) :- counterargument(X,Y).
```

4.2.1.4 Access Probability

In order to make NARAlog customisable w.r.t. user preferences, the model should take into account the probability that an annotation is actually accessed and considered. The

access probability can depend on attributes such as the author of an annotation; users might prefer reading one author's annotations while neglecting another author's. If no such information is given, a fixed value for the access probabilities can be assumed. Access probabilities can be modeled using the `acc` predicate:

```
0.8 acc(d1,a1).
0.8 acc(a1,a2).
```

4.2.2 Context-based Retrieval Function

Having introduced all the required components of NARAlog, we will now discuss the context-based retrieval function r_{nara}. This function consists of certain rules and implements the re-weighting phase of NARA. As mentioned above, positive and negative evidence contained in annotations and links should be considered independently, which motivates an open world assumption. A logic-based approach should therefore collect positive and negative evidence and combine it accordingly, which is done with knowledge augmentation in FVPD.

Positive evidence is contained in a node n itself, so the context-based retrieval weight n in the annotation thread is first of all determined by its content-based weight, which can be expressed like this:

```
r_nara(N) :- r_content(N).
```

Positive evidence is contained in successor nodes n' of n if there exists a positive link between n and n'. In this case, the positive value of n should be increased to the degree of the context-based weight of n' and the probability that n' is actually visited:

```
r_nara(N) :- pos_link(N,N') & acc(N,N') &
             r_nara(N').
```

Negative evidence is contained in successor nodes n' of n if there exists a negative link between n and n'. The negative value of n should be increased accordingly:

```
!r_nara(N) :- neg_link(N,N') & acc(N,N') &
              r_nara(N').
```

4.2.3 Example NARAlog Program

Figure 3 shows an example of a NARAlog program. Lines 1-6 show the results of the indexing process of documents and annotations; as discussed above, both are described as probabilistic facts using the `term` and `termspace` relations. HySpirit offers the means to load such facts directly from relational databases. The nodes of the annotation thread are categorised into documents and annotations in the lines 8-10, links are modeled in lines 12 and 13. Line 15 and 16 contain access probabilities, in this example 0.8. The facts between lines 1 and 16 are static and could all be stored in the index database.

Lines 18 to 20 contain rules which categorise the link types w.r.t. their effect on the context based retrieval weight into positive and negative links. The query is reflected in lines 22 and 23. The initial phase is implemented as a rule considering the indexed content of documents and annotations (line 26). The re-weighting phase is reflected in lines 29-31, where values for positive and negative evidence are collected. `r_nara(N)` implements the re-weighting phase since it is recursively executed for every successor node. The ranking process itself is started in line 33. By inserting `document(D)` we exclude annotations from being retrieved.

4.2.4 Calculation

When traversing the annotation thread in the re-weighting phase, NARAlog collects positive and negative evidence w.r.t. the query. But since the rules in lines 30 and 31 only contain positive facts, we have to show how NARAlog (or more precisely: the underlying HySpirit system) deals with negative facts. We will show this by discussing an example run. For this, reconsider the example annotation thread in Figure 1. The probability $P(a_2 \rightarrow q)$ that a_2 implies the query q is given by its content-based value calculated in line 26 (since there are no more successors of this annotation in our example) and is 0.2456. The calculation of $P(a_1 \rightarrow q)$ is performed by including the evidence found in a_2; since there is a negative link between a_1 and a_2, we have negative evidence here. The corresponding probability is computed by combining the link type, the access probability and $P(a_2 \rightarrow q)$, so that we gain $1 \cdot 0.8 \cdot 0.2456 = 0.19648$ for $P(!\texttt{r\_nara(a1)})$. The only positive evidence for `r_nara(a1)` comes from the content-based retrieval weight `r_content(a1)` which is 0.2345. Both positive and negative evidence are combined resulting in $P(a_1 \rightarrow q) = P(\texttt{r\_nara(a1)}) \cdot (1 - P(!\texttt{r\_nara(a1)})) = 0.235 \cdot 0.80352 = 0.1888272$. The value for $P(d_1 \rightarrow q)$, which is what we are looking for, is 0.185019.

5. RELATED WORK

5.1 Annotations and Annotation Systems

Marshall *et al.* [16, 17, 18] provide results of empirical investigations of annotations. These considerations provide useful insights into how to incorporate annotations in a digital library. Several dimensions of annotations are identified, like formal vs. informal, explicit vs. tacit, personal vs. global annotations, etc [17]. Phelps and Wilensky discuss several properties of digital annotations which are realised in their multivalent annotation framework [21]. Annotations are seen as the basis for collaborative work.

Ovsiannikov *et al.* provide an overview of annotation systems and identify four main aspect of annotation usage: to remember, think, clarify and share [20]. Concepts of annotation technologies are derived based on these considerations. Agosti and Ferro create a conceptual model of annotations based on two dimensions: the meaning of annotations (e.g., comprehension, interpretation, cooperation and revision) and the sign of an annotation (textual, graphical or referential). They present an architecture of an annotation service in OpenDLib [2]. [3] contains an examination of annotations from a syntactic, semantic and pragmatic view. Some options on annotation-based information retrieval are discussed there. Furuta *et al.* present Walden's Path, a system to create new hypertext paths with annotations [11].

There are several other annotation systems and prototypes.

```
0.2 term(d1, "political").       0.3 term(d1, "censorship").
0.5 term(a1, "political").       0.5 term(a1, "reasons").
0.4 term(a2, "political").       0.6 term(a2, "reasons").

0.2 termspace("political").      0.5 termspace("censorship").
0.3 termspace("reason").

document(d1).
annotation(a1).
annotation(a2).

interpretation(d1,a1).
counterargument(a1,a2).

0.8 acc(d1,a1).
0.8 acc(a1,a2).

pos_link(X,Y) :- interpretation(X,Y).
pos_link(X,Y) :- elaboration(X,Y).
neg_link(X,Y) :- counterargument(X,Y).

qterm("political").
qterm("reasons").

# Initial content-based phase
r_content(N) :- qterm(T) & termspace(T) & term(N,T).

# Annotation-based re-weighting phase
r_nara(N) :- r_content(N).
r_nara(N) :- pos_link(N,N') & acc(N,N') & r_nara(N').
!r_nara(N) :- neg_link(N,N') & acc(N,N') & r_nara(N').

?- document(D) & r_nara(D).
```

Figure 3: Example NARAlog program

Yawas [7] is a Web annotation tool with which the user can annotate the document content or provide information, e.g., about the document type. Annotea [13] is another Web annotation tool which can deal with nested annotations and several annotation types (realised as typed links between annotations). DEBORA [19] is a digital library for Renaissance books. Annotations are used to share information and to create virtual books. In our COLLATE prototype, nested annotations are used to enable scientific discussions [9]. Some commercial systems like Word and Acrobat provide means for document annotation.

Although most of the work presented in this subsection did not focus on information retrieval, they give a valuable insight into the nature of annotations and how to model them, which in turn can be used to develop appropriate retrieval functions.

5.2 Hypertext Information Retrieval and Categorisation

Since annotations connected to their annotated resources can be seen as a specific hypertext [3], it is worth investigating common hypertext information retrieval (HIR) approaches w.r.t. their applicability for annotation-based information retrieval. [4] gives a good overview of some research in HIR. An interesting numeric approach is presented by Frei and Stieger [8] using spreading activation. Similar to NARA, there is an initialisation phase (where content-based RSVs are calculated), and a navigation phase where the hypertext structure is traversed to bias the initial RSV. The approach can cope with link types, but the possible incorporation of negative links does not suffice since, as described in Section 4.2.1.3, two consecutive negative links might again have a positive effect on the contextual RSV, which is not covered by the algorithms presented in [8].

As mentioned earlier in this paper, Fuhr and Rölleke use four-valued logics in order to retrieve complex objects [22, 10]. Their work is the foundation of the NARAlog approach presented here.

The idea of exploiting what others said about a document is formulated as well in [5]. In their approach of categorisation by context Attardi *et al.* use anchor texts and the surroundings of a hyperlink as additional information about the document. This information can be seen as annotations of a document since it often contains summaries, interpretations or reformulations of document content. Being tailored

to the World Wide Web, the approach presented in [5] does not use any link types and does not consider any positive or negative links.

5.3 Annotation-based Information Retrieval

It is noticable that there has not been much work presented so far in information retrieval using textual annotations. Some annotation systems provide simple full-text search mechanisms on annotations [20], but do not support annotation-based document search. After finding appropriate annotations, a user still has to browse to the according document. Results from document and annotation search are not combined. Yawas [7] offers some means to use annotations for document search, e.g. by enabling users to search for a specific document type considering annotations. The approach does not consider nested nor negative annotations.

An interesting approach using annotation-generated queries with relevance feedback is introduced by Golovchinsky *et al.* [12]. Here, annotations are markings given by users who judge certain parts of a document as being important when emphasising them. Evaluations show that using these kinds of annotations provides better retrieval effectiveness than classic relevance feedback. Being based on relevance judgements and annotations as markings rather than content, this approach is totally different from what we propose here.

6. FUTURE WORK

Experiments have yet to conclusively prove the impact and significance of using annotations as we proposed in this paper on retrieval effectiveness. We will perform experiments using the COLLATE collection, which at the moment consists of 6980 documents and 1994 annotations. Possible baselines for this would be neglecting annotations and the structure of an annotation thread. Besides effectiveness experiments have also to address the efficiency of our approach; possible solutions of making our approach more efficient would be to prune the annotation thread at a certain depth. Another important issue is to find other appropriate test collections containing more documents and annotations as in the COLLATE repository. Following the idea proposed in [5], annotations extracted from hypertext test corpora might gain a suitable collection for evalutation purposes.

Further research is needed to determine suitable weights for parameters like the access probability $P(\texttt{acc})$ described in Section 4.2.1.4. We think this probability is an important parameter to cover user's interests, but some suitable values have to be found in case this information is not available. One simple approach would be to assign a global value for all access probabilities.

In COLLATE we let users explicitly state which kind of annotation they create, so we know about the link type. It is required that some kind of atomicity is given. As an example, an annotation like "there were no political but economical reasons" would not be valid in our framework, since besides a counterargument ("there were no political reasons") it also contains an interpretation ("there were economical reasons"). The annotation author would have to create two annotations to formulate her point. Our model, being suitable for an experimental system like COLLATE, is not convenient w.r.t. usability. So it is desirable to automatically recognise link types like discourse relations between annotations or even parts of annotations, at least w.r.t. their effect (negative or positive). The work by Marcu and Echihabi [15] where they use Naive Bayes classifiers for determining discourse relation seems to be an interesting starting point for calculating probabilities for link types like the ones discussed in Section 4.2.1.2.

7. CONCLUSION

In this paper we have discussed NARA, which is an approach to perform information retrieval using nested annotation. Such nested annotation together with typed links between them build an annotation thread. NARA biases content-based retrieval status values by analysing the annotation thread. For this, NARA has to deal with negative and contradictory statements.

With NARAlog we have presented an implementation of NARA based on four-valued probabilistic datalog. Using four-valued logics and an open world assumption, NARAlog is capable of coping with nested annotation as well as negative and contradictory ones. The components of NARAlog, including content- and context-based retrieval functions, were discussed.

8. REFERENCES

[1] M. Agosti. An Overview of Hypertext. In Agosti and Smeaton [4], pages 27–47.

[2] M. Agosti and N. Ferro. Annotations: Enriching a Digital Library. In Koch and Sølvberg [14], pages 88–100.

[3] M. Agosti, N. Ferro, I. Frommholz, and U. Thiel. Annotations in digital libraries and collaboratories – facets, models and usage. In R. Heery and L. Lyon, editors, *Proc. 8th European Conference on Research and Advanced Technology for Digital Libraries (ECDL)*, 2004. To appear.

[4] M. Agosti and A. Smeaton, editors. *Information Retrieval and Hypertext*. Kluwer Academic Publishers, Norwell (MA), USA, 1996.

[5] G. Attardi, A. Gullí, and F. Sebastiani. Automatic Web page categorization by link and context analysis. In C. Hutchison and G. Lanzarone, editors, *Proceedings of THAI-99, 1st European Symposium on Telematics, Hypermedia and Artificial Intelligence*, pages 105–119, Varese, IT, 1999.

[6] H. Brocks, A. Stein, U. Thiel, I. Frommholz, and A. Dirsch-Weigand. How to incorporate collaborative discourse in cultural digital libraries. In *Proceedings of the ECAI 2002 Workshop on Semantic Authoring, Annotation & Knowledge Markup (SAAKM02)*, Lyon, France, July 2002.

[7] L. Denoue and L. Vignollet. An annotation tool for web browsers and its applications to information retrieval. In *Proceedings of RIAO 2000, Paris, April 2000*, April 2000.

[8] H. P. Frei and D. Stieger. The use of semantic links in hypertext information retrieval. *Information*

Processing and Management: an International Journal, 31(1):1–13, 1995.

[9] I. Frommholz, H. Brocks, U. Thiel, E. Neuhold, L. Iannone, G. Semeraro, M. Berardi, and M. Ceci. Document-centered collaboration for scholars in the humanities - the COLLATE system. In Koch and Sølvberg [14], pages 434–445.

[10] N. Fuhr and T. Rölleke. HySpirit – a probabilistic inference engine for hypermedia retrieval in large databases. In H.-J. Schek, F. Saltor, I. Ramos, and G. Alonso, editors, *Proceedings of the 6th International Conference on Extending Database Technology (EDBT), Valencia, Spain*, Lecture Notes in Computer Science, pages 24–38, Heidelberg et al., 1998. Springer.

[11] R. Furuta, F. M. Shipman, C. C. Marshall, D. Brenner, and H. Hsieh. Hypertext paths and the world-wide web: Experiences with Walden's Path. In *Hypertext '97: the Eighth ACM Conference on Hypertext*, pages 167–176. ACM Inc., UK, 1997.

[12] G. Golovchinsky, M. N. Price, and B. N. Schilit. From reading to retrieval: Freeform ink annotations as queries. In F. Gey, M. Hearst, and R. Tong, editors, *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 19–25, New York, 1999. ACM Press.

[13] J. Kahan, M. Koivunen, E. Prud'Hommeaux, and R. Swick. Annotea: An open rdf infrastructure for shared web annotations. In *Proceedings of the WWW10 International Conference*, Hong Kong, May 2001.

[14] T. Koch and I. T. Sølvberg, editors. *Proc. 7th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2003)*. Lecture Notes in Computer Science (LNCS) 2769, Springer, Heidelberg, Germany, 2003.

[15] D. Marcu and A. Echihabi. An unsupervised approach to recognizing discourse relations. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 368–375, July 2002.

[16] C. C. Marshall. Annotation: from Paper Books to the Digital Library. In R. B. Allen and E. Rasmussen, editors, *Proc. 2nd ACM International Conference on Digital Libraries (DL 1997)*, pages 233–240. ACM Press, New York, USA, 1997.

[17] C. C. Marshall. Toward an Ecology of Hypertext Annotation. In R. Akscyn, editor, *Proc. 9th ACM Conference on Hypertext and Hypermedia (HT 1998): links, objects, time and space-structure in hypermedia systems*, pages 40–49. ACM Press, New York, USA, 1998.

[18] C. C. Marshall and A. J. B. Brush. From Personal to Shared Annotations. In L. Terveen and D. Wixon, editors, *Proc. Conference on Human Factors and Computing Systems (CHI 2002) – Extended Abstracts on Human Factors in Computer Systems*, pages 812–813. ACM Press, New York, USA, 2002.

[19] D. Nichols, D. Pemberton, S. Dalhoumi, O. Larouk, C. Belisle, and T. M.B. DEBORA: Developing an Interface to Support Collaboration in a Digital Library. In J. Borbinha and T. Baker, editors, *Proceedings of the 4th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2000)*, Lecture Notes in Computer Science, pages 239–248, Berlin et al., 2000. Springer.

[20] I. A. Ovsiannikov, M. A. Arbib, and T. H. McNeill. Annotation technology. *Int. J. Hum.-Comput. Stud.*, 50(4):329–362, 1999.

[21] T. A. Phelps and R. Wilensky. Multivalent Annotations. In C. Peters and C. Thanos, editors, *Proc. 1st European Conference on Research and Advanced Technology for Digital Libraries (ECDL 1997)*, pages 287–303. Lecture Notes in Computer Science (LNCS) 1324, Springer, Heidelberg, Germany, 1997.

[22] T. Rölleke and N. Fuhr. Retrieval of complex objects using a four-valued logic. In *Proceedings of the 19th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 206–214, New York, 1996. ACM.

[23] C. J. van Rijsbergen. A non-classical logic for information retrieval. *The Computer Journal*, 29(6):481–485, 1986.

An XML-IR-DB Sandwich: Is it Better With an Algebra in Between?

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

ABSTRACT

In this paper we address the problem of immediate translation of XPath+IR queries to relational database expressions and exert the benefits of using an intermediate algebra. Adding an intermediate algebra on the logical level of a database enables a level of abstraction from both query languages for IR in XML documents and the underlying relational storage. This paper proposes a region algebra that can be extended to support ranking operators in an elegant way while staying algebraic. Furthermore, region algebra operator properties provide a firm ground for query rewriting and optimization.

1. INTRODUCTION

Despite the numerous existing systems dealing with XML querying, the problem of expressing as well as executing Information Retrieval-like (IR-like) queries over XML databases is still an open issue [1]. An IR-like query (an example is given in Figure 2 in Section 2) does not specify hard conditions on XML elements, but queries the collection for elements 'about' a certain topic. For instance, an XML element that is relevant to a query for elements about "relational databases" might not contain the phrase "relational databases", or even both words "relational" and "databases". IR-like queries should result in a ranked list of XML elements, in decreasing order of some score value that the system assigns to each element. The score value has to reflect the probability (or degree) of relevance of the element to the IR-like query.

A promising approach to executing XPath and XQuery is the use of relational database technology [9, 17], which is easily extended to IR-like querying of XML [6, 12]. What would be the most effective way to support IR-like querying in XPath and XQuery using relational database technology? The semantics of XPath and XQuery give rules for navigation through XML structure, but not the rules that specify how score values for XML elements should propagate and relate to each other. Similarly, the semantics of relational algebra introduce rules for manipulating relational tables that describe XML data, but again the rules for score computation and propagation cannot be derived from the relations present in the relational database.

We follow a three level database approach for developing an XML-IR system, consisting of conceptual, logical, and physical level. The benefits of the usage of a three level database management system is that we are able to provide data independence between the relational representation on the physical level, the proposed algebra on logical level, and the query language used on the conceptual level, and provide a certain level of abstraction from the information retrieval model used for ranked retrieval.

The introduction of an intermediate level allows for the usage of algebraic properties for query rewriting and optimization. The optimization should be achieved not only for the regular XPath/XQuery queries but for the IR-like queries as well. In this paper we study the usefulness of algebraic properties for query optimization and for developing and understanding IR-like extensions. The algebra we propose is based on so-called region algebras [2, 4, 11, 13]. Region algebras are sufficiently simple to study algebraic properties in depth, and they are sufficiently powerful to express IR-like queries as those proposed in the NEXI query language used for the evaluation of XML retrieval in INEX [16]. NEXI stands for "narrowed extended XPath". It only uses the descendant axis step from XPath, and it extends XPath with a special about-function that provides IR-like search. The region algebra can be easily extended to support other XPath axis steps with additional parent information [14]. The basic idea behind the algebra is to support as much as possible for the full text search requirements [3] and it is driven by the wish to integrate XML databases and information retrieval as discussed in [1].

Unlike many approaches for ranked retrieval in XML, the algebra we define assumes that the ranking is a part of the algebra and not a side effect of performing some operations on regions (like in [13]) or a separate IR module (like in many IR approaches for XML retrieval). Therefore, we follow the approach taken by Fuhr et al. [7, 8], although we base our algebra on containment model rather than path model, and do not make any restrictions on the definition of 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

WIRD'04, the first Workshop on the Integration of Information Retrieval and Databases (WIRD'04), Sheffield, United Kingdom
© 2004 the author/owner

```
<article lang=''en'' date=''10/02/04''>
   <title>Region algebra</title>
   <bdy>
      <sec>
         <p>Structured documents ...</p>
         <p>Text search ...</p>
      </sec>
      ...
   </bdy>
   ...
</article>
```

Figure 1: Example XML document.

algebra operators as well. This allows for the introduction of more powerful optimization techniques concerned with speeding-up the execution of operations that compute score values for ranked retrieval.

The paper is organized as follows. In Section 2 we explain how relational technology is used to process NEXI queries. We give the translation of NEXI queries into relational algebra and discuss why we need an intermediate level. Section 3 introduces our region algebra and discuss region algebra operator properties. In Section 4, we illustrate how operators for ranked retrieval follow the properties of basic region algebra operators and discuss the opportunities for query optimization in our region algebra, extended for ranked retrieval. We conclude the paper with a discussion and our plans for future research.

2. XML AND RELATIONAL DATABASES

In this section we explain the formation of the XML data set and discuss some issues on the indexing of XML documents. The relational storage of such documents is also discussed, along with the relational algebra expressions for two NEXI query examples.

2.1 Representing XML in Relational Databases

Most of the database approaches to XML choose to index XML documents before storing them into relational tables. The rationale for this is the structural organization of XML documents and the benefits that can be achieved when querying such indexed relational representation of XML documents. For an illustration we refer to [10] where the authors used the pre-post and stretched pre-post indexing scheme for the relational storage of XML documents. In our approach we used a variant of the stretched pre-post indexing[1] scheme that also indexes each word in XML text nodes. Note that the indexing also produces the initial data set for the data model that we define in Section 3.

The data set creation, i.e., the formation of the initial data set from (plain text) XML documents can be explained through the usage of a two step indexing process[2]. The indexing process is explained using an example XML document given in Figure 1. In the first step each token in the XML document (denoted with D) is indexed regarding its relative position with respect to its beginning and its type: $I_1 : D \rightarrow X$. As a result we obtain a set of elements: $x \in X$, uniquely identified by their position in the XML document. Each element has the form of $x = \{position, token, token\_type\}$ as shown in Table 1.

Table 1: Intermediate index structure (X) obtained after initial indexing (I_1) of XML document depicted in Figure 1.

position	token	token_type
0	<article>	start tag
1	lang	attribute name
2	''en''	attribute value
3	date	attribute name
4	''10/02/04''	attribute value
5	<title>	start tag
6	region	term
7	algebra	term
8	</title>	end tag
9	<bdy>	start tag
10	<sec>	start tag
11		start tag
12	structured	term
13	documents	term
...	...	...
54		end tag
...	...	...
576	</sec>	end tag
...	...	...
9876	</bdy>	end tag
...	...	...
10034	</article>	end tag

The second step produces regions that we can consider as the initial data set. These regions are produced by pairing corresponding tokens that represent opening and closing tags, attribute names and values, etc., and by removing mark-up delimiters from the tokens: $I_2 : X \rightarrow R$. This will result in a data set like the one presented in Table 2. Thus, the initial data set construction can be defined as a composition of two indexing procedures: $I = I_1 \circ I_2$. Although the indexing is a two step process it can be implemented as a single walk through an XML document using the SAX parser and stack structures (see [10]).

Table 2: Data model for XML document presented in Figure 1 obtained after the composition of initial indexing (I_1) and final indexing (I_2).

start	end	name	type
0	10034	article	node
1	2	lang	attr_name
2	2	en	attr_value
3	4	date	attr_name
4	4	10/02/04	attr_value
5	8	title	node
6	8	-	text
6	6	region	term
7	7	algebra	term
9	9876	bdy	node
10	576	sec	node
11	54	p	node
12	53	-	text
12	12	structured	term
13	13	documents	term
...	...	...	...

An indexed XML document, however, is not stored in one relational table since this table will be huge and in most

[1]Note that the term indexing differs from the concept of indexing as defined in the traditional database systems. It denotes the method used for creation of the initial data set.

[2]Although XML documents are actually graphs we will simplify the XML structure and treat these entities as if they were organized as a hierarchical (tree-like) structure.

cases (on most platforms) hard to process. In many relational approaches to XML different fragmentations of this basic table are used. The fragmentation can be horizontal, based only on type of XML nodes (like in [10] and [12]), vertical based on a name and/or type of XML elements, e.g., [6], or based on paths to XML nodes in an XML tree structure (like in [15]). For illustrative purpose we use horizontal fragmentation of XML data as presented in [12]. Consequently, different relational tables are defined for the XML element nodes and attribute nodes and the word table is defined for the words in the XML text nodes. This is depicted in Table 3. In further discussion we will not consider the attribute table since it is not of the interest for the issues that we are discussing in this paper.

Table 3: Relational data model for storing XML document presented in Figure 1.

Node table $\mathcal{N}$

start	end	name	type
0	10034	article	node
5	8	title	node
6	8	-	text
9	9876	bdy	node
10	576	sec	node
11	54	p	node
12	53	-	text
...	...	...	...

Word table $\mathcal{W}$

start	name
6	region
7	algebra
12	structured
13	documents
...	...

Attribute table $\mathcal{A}$

start	owner	name	type
1	0	lang	name
2	0	en	value
3	0	date	name
4	0	10/02/04	value
...	...	...	...

2.2 From XML Queries to Relational Algebra

The two example queries given in Figure 2 will be used as our leading examples in the following sections. As query language we use NEXI query language which has officially been adopted for INEX 2004[3]. Its detailed description can be found in [16]. For now we consider that the *about* condition inside queries is strict (corresponds to a Boolean search, i.e., *about* behaves the same as XPath *contains* expression). Later on in this paper we elaborate more on the use of the *about* clause for ranking.

For the chosen storage model, composed of $\mathcal{N}$ and $\mathcal{W}$ (and $\mathcal{A}$), we can directly transform any NEXI expression into relational algebra expression. For NEXI example query 1 depicted in Figure 2 a possible relational algebra expressions could be specified as given in Figure 3. We disregard the type attribute in expressions for brevity.

Note that there is a frequent usage of a group of expressions consisting of join and projection operations that simulate the XPath descendant/ancestor step. This group of expressions actually represents the bottleneck for XPath query processing, since its naive execution is extremely slow. A number of techniques have been proposed to speed up the execution of XPath descendant and ancestor steps, such as multi-predicate merge join [17], staircase join [10], containment join [12], etc. Using such abstract join operators, denoted with $\bowtie_{\sqsupset}$ (for expression types R_7, R_8 and R_{10} in Figure 3) and $\bowtie_{\sqsubset}$ (for expression type R_6 in Figure 3), the query plan for NEXI query example 2 can be expressed as shown in Figure 4.

[3]http://inex.is.informatik.uni-duisburg.de:2004/.

Figure 3: Relational query plan for example query 1 given in Figure 2.

$$
\begin{aligned}
R_1 &= \sigma_{name=\text{“}article\text{“}}(\mathcal{N}) \\
R_2 &= \sigma_{name=\text{“}bdy\text{“}}(\mathcal{N}) \\
R_3 &= \sigma_{name=\text{“}sec\text{“}}(\mathcal{N}) \\
R_4 &= \sigma_{name=\text{“}structured\text{“}}(\mathcal{W}) \\
R_5 &= \sigma_{name=\text{“}documents\text{“}}(\mathcal{W}) \\
R_6 &= \pi_{start_2,end_2,name_2}(R_2 \bowtie_{start_2>start_1,end_2<end_1} R_1) \\
R_7 &= \pi_{start_3,end_3,name_3}(R_3 \bowtie_{start_3<start_4,end_3>end_4} R_4) \\
R_8 &= \pi_{start_3,end_3,name_3}(R_3 \bowtie_{start_3<start_5,end_3>end_5} R_5) \\
R_9 &= R_7 \cap R_8 \\
R_{10} &= \pi_{start_6,end_6,name_6}(R_6 \bowtie_{start_6<start_9,end_6>end_9} R_9)
\end{aligned}
$$

Figure 4: Relational query plan for example query 2 given in Figure 2.

$$
\begin{aligned}
R_1 &= \sigma_{name=\text{“}article\text{“}}(\mathcal{N}) \\
R_2 &= \sigma_{name=\text{“}bdy\text{“}}(\mathcal{N}) \\
R_3 &= \sigma_{name=\text{“}sec\text{“}}(\mathcal{N}) \\
R_4 &= \sigma_{name=\text{“}p\text{“}}(\mathcal{N}) \\
R_5 &= \sigma_{name=\text{“}region\text{“}}(\mathcal{W}) \\
R_6 &= \sigma_{name=\text{“}algebra\text{“}}(\mathcal{W}) \\
R_7 &= \sigma_{name=\text{“}XML\text{“}}(\mathcal{W}) \\
R_8 &= \sigma_{name=\text{“}information\text{“}}(\mathcal{W}) \\
R_9 &= \sigma_{name=\text{“}retrieval\text{“}}(\mathcal{W}) \\
R_{10} &= R_2 \bowtie_{\sqsubset} R_1 \\
R_{11} &= ((R_{10} \bowtie_{\sqsupset} R_5) \cap (R_{10} \bowtie_{\sqsupset} R_6)) \bowtie_{\sqsupset} (R_3 \bowtie_{\sqsupset} R_7) \\
R_{12} &= R_4 \bowtie_{\sqsubset} R_{11} \\
R_{13} &= (R_{12} \bowtie_{\sqsupset} R_8) \cap (R_{12} \bowtie_{\sqsupset} R_9)
\end{aligned}
$$

2.3 Do We Need an Algebra in Between?

There might be a number of reasons to define an algebra. First of all, as we saw in the previous section, to be able to express XPath+IR (NEXI) queries in relational databases we need new operators for efficient execution of XPath+IR subexpressions, such as descendant and ancestor steps, containment conditions, etc. The exact technique how we implement the subexpression is defined on the physical level, and it does not have to be unique, i.e., we can have multiple variants of relational expression for the same XPath+IR subexpressions. The execution times for distinct implementations differ regarding the relational storage of the XML data, parameters of the relational tables, and index structure used for the acceleration of relational expression execution in relational databases.

Another important issue concerning immediate translation of XPath+IR expressions into relational algebra is that the algebraic expressions are highly dependent on the relational model chosen for the storage of XML data. If we change the relational storage model, the relational algebra expressions for each query have to be rewritten according to the

Figure 2: NEXI queries.

Example NEXI query 1:

```
//article//bdy[about(.//sec, structured) and about(.//sec, documents)]
```

Example NEXI query 2:

```
//article//bdy[about(., region) and about(., algebra)][about(.//sec, XML)]//p[about(., information) and about(., retrieval)]
```

relational model. This is especially the case for XML, since usually huge relational tables, which have more than a million entries, are typically broken into a number of smaller ones using one of the fragmentation methods mentioned in section 2.1. Having a logical level with the algebra defined in it would provide the right level of abstraction considering different XPath+IR queries formed on the conceptual level, and the relational storage structure chosen on the physical level. In such a way we provide the needed *data independence* on logical level. 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 usage of main memory when executing on physical level.

A final but important reason for defining an algebra is to enable the expression of IR-like queries (*about* in NEXI), i.e., score computation and ranking of XML elements. Therefore, the algebra should provide a specific level of *IR understanding* that is based on the retrieval model used for score computation. The operators used for score computation should adhere to certain operator properties which can be used for query optimization based on the definition of score operators.

The exact way of how we use region algebra operator properties on the logical level, and how we extend the region algebra to support ranked retrieval is explained in the next two sections.

3. REGION ALGEBRA

For defining the intermediate logical level we have chosen the region algebra approach, because it is already well established in the area of structured document retrieval [2, 4, 11, 13], and because of the useful properties of region algebra operators as we discuss in the remainder of the paper.

With the specification of the region algebra data model we provide a uniform platform for defining region algebra operators. We discuss the basic XML region algebra data model which can be defined using four region attributes, based on the indexed data set described in the previous section (for more details see [12]).

DEFINITION 1. *The basic region algebra data model is defined on the domain R which represents a set of region tuples. A region tuple r ($r \in R$), $r = (s, e, n, t)$, is defined by these four attributes: region start attribute - s, region end attribute - e, region name attribute - n, and region type attribute - t. The region start and end attributes must satisfy ordering constraints ($e_i \geq s_i$).*

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 attributes are used to denote node names, content words, attribute names, attribute values, etc. To be able to distinguish different node types in XML the type information is needed.

Table 4: Basic score region algebra operators.

Operator	Operator definition
$\sigma_{n=name}(R)$	$\{r \mid r \in R \wedge n = name\}$
$R_1 \sqsupset R_2$	$\{r_1 \mid r_1 \in R_1 \wedge \exists r_2 \in R_2 \wedge s_1 < s_2 \wedge e_1 > e_2\}$
$R_1 \sqsubset R_2$	$\{r_1 \mid r_1 \in R_1 \wedge \exists r_2 \in R_2 \wedge s_1 > s_2 \wedge e_1 < e_2\}$
$R_1 \sqcap R_2$	$\{r \mid r \in R_1 \wedge r \in R_2\}$
$R_1 \sqcup R_2$	$\{r \mid r \in R_1 \vee r \in R_2\}$

Next, we define the basic region algebra operators. The definition of region algebra operators is based on the operators specified in the previous region algebra approaches, extended to support a specific XML structure. Table 4 defines the following five basic region algebra operators: selection (σ), containing ($\sqsupset$), contained by ($\sqsubset$), region set intersection ($\sqcap$), and region set union ($\sqcup$). 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)[4].

As can be noticed in Table 4, instead of an interval operator used in [2, 4, 11, 13] (usually denoted with $I(token)$ in region algebra approaches) which is actually not a real region algebra operator but rather specifies the indexing function applied to a specified token that returns the region set of the occurrences of *token* in a document, we introduced a selection operator (σ). The selection operator is a unary region algebra operator that operates on a region set and produces a region set as a result. It is defined to enable the selection of regions formed during the initial data set creation (explained in section 2.1), based on name (and type) region attributes.

Following the query examples given in Figure 2, we give the same query execution plans defined using the region algebra operators instead of the relational ones. The region algebra query plans for query examples 1 and 2 are given in Figure 5 and Figure 6. We use C to denote the initial data set of regions. We can note a great resemblance between the previous relational query plans and region algebra query plans. This exerts the simplicity of transforming region algebra expressions into relational expression. However, the change in the relational storage will result in the change of query plan on the physical level, while the query plan on the logical level will remain the same.

We can state here that in order to model XML properly, we

[4]In Table 4 we do not use the type information as it is not of great importance for this paper. Thus, the exact definition of selection operator should be $\sigma_{n=name,t=type}(R) = \{r \mid r \in R \wedge n = name \wedge t = type\}$, if type attribute would be used.

Figure 5: Region algebra query plan for example query 1 given in Figure 2.

$$\begin{aligned}
\text{ARTICLE} &= \sigma_{n=\text{“}article\text{“}}(C)\\
\text{BDY} &= \sigma_{n=\text{“}bdy\text{“}}(C)\\
\text{SEC} &= \sigma_{n=\text{“}sec\text{“}}(C)\\
\text{STRUCTURED} &= \sigma_{n=\text{“}structured\text{“}}(C)\\
\text{DOCUMENTS} &= \sigma_{n=\text{“}documents\text{“}}(C)
\end{aligned}$$

$$\begin{aligned}
R_1 &= (\text{SEC} \sqsupset \text{STRUCTURED}) \sqcap (\text{SEC} \sqsupset \text{DOCUMENTS})\\
R_2 &= (\text{BDY} \sqsubset \text{ARTICLE}) \sqsupset R_1
\end{aligned}$$

Figure 6: Region algebra query plan for example query 2 given in Figure 2.

$$\begin{aligned}
\text{ARTICLE} &= \sigma_{n=\text{“}article\text{“}}(C)\\
\text{BDY} &= \sigma_{n=\text{“}bdy\text{“}}(C)\\
\text{SEC} &= \sigma_{n=\text{“}sec\text{“}}(C)\\
\text{P} &= \sigma_{n=\text{“}p\text{“}}(C)\\
\text{REGION} &= \sigma_{n=\text{“}region\text{“}}(C)\\
\text{ALGEBRA} &= \sigma_{n=\text{“}algebra\text{“}}(C)\\
\text{XML} &= \sigma_{n=\text{“}XML\text{“}}(C)\\
\text{INFORMATION} &= \sigma_{n=\text{“}information\text{“}}(C)\\
\text{RETRIEVAL} &= \sigma_{n=\text{“}retrieval\text{“}}(C)
\end{aligned}$$

$$\begin{aligned}
R_1 &= \text{BDY} \sqsubset \text{ARTICLE}\\
R_2 &= ((R_1 \sqsupset \text{REGION}) \sqcap (R_1 \sqsupset \text{ALGEBRA})) \sqsupset (\text{SEC} \sqsupset \text{XML})\\
R_3 &= (\text{P} \sqsubset R_2)\\
R_4 &= (R_3 \sqsupset \text{INFORMATION}) \sqcap (R_3 \sqsupset \text{RETRIEVAL})
\end{aligned}$$

could enrich the definition of a region with the additional information of XML references, parent or level information, etc. For details on some extensions on region algebra approaches we refer to papers [12] and [14].

3.1 Region Algebra Operator Properties

In this section we discuss properties of algebraic operators and their use for query rewriting and optimization. Some of the properties are illustrated using the examples given in Figure 5 and Figure 6. Many properties are mentioned in papers about region algebra by Clarke et al. [4], and Jaakkola and Kilpelainen [11], but none of the papers discuss their usage. Our study on region algebra shows that there are only few operators that have the basic operator properties such as: identity, inverse, commutativity, associativity, and distributivity. However, there is a number of region algebra specific properties which can be considered as a special case of distributivity and associativity properties.

In general, we can distinguish two classes of binary region algebra operators. The first class consists of containment operators: $\sqsupset$, $\sqsubset$, while the second class consists of the standard set operators: $\sqcap$ and $\sqcup$. Only set operators have the identity element, which is the initial data set C for operator $\sqcap$ (property (1)), and the empty set $\emptyset$ for operator $\sqcup$ (property (2)). There is no inverse element for any of the operators. The set operators are commutative (properties (3) and (4)) and associative (properties (5) and (6)). Considering the distributivity property, only some combinations of operators follow it. The operators $\sqsupset$ and $\sqsubset$ distribute over the operator $\sqcup$ (properties (7) and (8)), while the operator $\sqcap$ distributes over the operator $\sqcup$ and vice versa (properties (9) and (10)).

Identity

$$R \sqcap C = C \sqcap R = R \tag{1}$$

$$R \sqcup \emptyset = \emptyset \sqcup R = R \tag{2}$$

Commutativity

$$R_1 \sqcap R_2 = R_2 \sqcap R_1 \tag{3}$$

$$R_1 \sqcup R_2 = R_2 \sqcup R_1 \tag{4}$$

Associativity

$$(R_1 \sqcap R_2) \sqcap R_3 = R_1 \sqcap (R_2 \sqcap R_3) \tag{5}$$

$$(R_1 \sqcup R_2) \sqcup R_3 = R_1 \sqcup (R_2 \sqcup R_3) \tag{6}$$

Distributivity

$$R_1 \sqsupset (R_2 \sqcup R_3) = (R_1 \sqsupset R_2) \sqcup (R_1 \sqsupset R_3) \tag{7}$$

$$R_1 \sqsubset (R_2 \sqcup R_3) = (R_1 \sqsubset R_2) \sqcup (R_1 \sqsubset R_3) \tag{8}$$

$$R_1 \sqcap (R_2 \sqcup R_3) = (R_1 \sqcap R_2) \sqcup (R_1 \sqcap R_3) \tag{9}$$

$$R_1 \sqcup (R_2 \sqcap R_3) = (R_1 \sqcup R_2) \sqcap (R_1 \sqcup R_3) \tag{10}$$

Special cases of associativity and distributivity

There are several interesting properties of the region algebra operators which can be useful for query rewriting and optimization on the logical level of a database. Here we mention the special case of containment operator associativity (property (11)), containment operator normalization (property (12)), and special case of set operator distributivity (property (13)). The first two properties are mentioned in papers [4] and [11]. We know of no publication on region algebras that mentions the third property.

For operators $op1 \in \{\sqsupset, \sqsubset\}$ and $op2 \in \{\sqsupset, \sqsubset\}$ properties (11) and (12) hold.

$$(R_1\ op1\ R_2)\ op2\ R_3 = (R_1\ op2\ R_3)\ op1\ R_2 \tag{11}$$

$$(R_1\ op1\ R_2)\ op2\ R_3 = (R_1\ op1\ R_2) \sqcap (R_1\ op2\ R_3) \tag{12}$$

For operators $op1 \in \{\sqcap, \sqcup\}$ and $op2 \in \{\sqsupset, \sqsubset\}$ property (13) is true.

$$(R_1\ op1\ R_2)\ op2\ R_3 = (R_1\ op2\ R_3)\ op1\ (R_2\ op2\ R_3) \tag{13}$$

To illustrate properties (11) and (12) we use the region algebra expression specified for the example query 1, given in Figure 5:

(BDY $\sqsubset$ ARTICLE) $\sqsupset$ ((SEC $\sqsupset$ STRUCTURED) $\sqcap$ (SEC $\sqsupset$ DOCUMENTS))

The expression may be read as follows: “Retrieve bdy-elements *contained-by* article-elements *containing* the *intersection* of sec-elements *containing* the term 'structured' and sec-elements *containing* the term 'documents'.” Using the property (11) we can rewrite this expression into:

(BDY $\sqsupset$ ((SEC $\sqsupset$ STRUCTURED) $\sqcap$ (SEC $\sqsupset$ DOCUMENTS))) $\sqsubset$ ARTICLE

Furthermore, using the property (12) this expression can be rewritten into:

(BDY $\sqsupset$ ((SEC $\sqsupset$ STRUCTURED) $\sqsupset$ DOCUMENTS)) $\sqsubset$ ARTICLE

or using again the property (11) to:

(BDY $\sqsupset$ ((SEC $\sqsupset$ DOCUMENTS) $\sqsupset$ STRUCTURED)) $\sqsubset$ ARTICLE

Using properties (11) and (12) we are able to choose the most appropriate query plan assuming that we have the information on which subexpressions are more selective. This reasoning can be applied for choosing which subexpressions will be more selective for SEC $\sqsupset$ DOCUMENTS or SEC $\sqsupset$

STRUCTURED, or similarly for BDY $\sqsupset$ ((SEC $\sqsupset$ DOCUMENTS) $\sqsupset$ STRUCTURED) or BDY $\sqsubset$ ARTICLE expressions. For example, since usually all regions from the BDY region set are contained in the ARTICLE region set, BDY $\sqsubset$ ARTICLE expression should be pushed up in the query plan as it is not a selective expression. Also the formulations of the query with the $\sqcap$ operator can be useful for parallel execution of two containment subqueries, if there exist an opportunity for such execution.

Property (13) is explained on a part of the example query 2, denoted with R_2 in Figure 6. Using the expression $R_1 =$ BDY $\sqsubset$ ARTICLE the part of the query example 2 can be expressed in region algebra as follows:

$$((R_1 \sqsupset \text{REGION}) \sqcap (R_1 \sqsupset \text{ALGEBRA})) \sqsupset (\text{SEC} \sqsupset \text{XML})$$

Using property (13) for operators $\sqcap$ and $\sqsupset$, this expression can be rewritten into:

$$((R_1 \sqsupset \text{REGION}) \sqsupset (\text{SEC} \sqsupset \text{XML})) \sqcap ((R_1 \sqsupset \text{ALGEBRA}) \sqsupset (\text{SEC} \sqsupset \text{XML}))$$

We would obtain a similar region algebra expression for the `or` expression of the example NEXI query 1 in Figure 2 instead of the `and` expression, where operator $\sqcap$ will be replaced with the operator $\sqcup$. This will provide the opportunity for e.g., parallelization.

Furthermore, using the property (11) the next expression could be obtained from the previous one:

$$((R_1 \sqsupset (\text{SEC} \sqsupset \text{XML})) \sqsupset \text{REGION}) \sqcap ((R_1 \sqsupset (\text{SEC} \sqsupset \text{XML})) \sqsupset \text{ALGEBRA})$$

and after the usage of property (12) the final expression is:

$$((R_1 \sqsupset (\text{SEC} \sqsupset \text{XML})) \sqsupset \text{REGION}) \sqsupset \text{ALGEBRA}$$

Therefore, instead of six operands and five operators we have a reduction to five operands and four operators, where the selection of regions R_1 that contain sections that contain term XML is pushed down to the first subexpression (assuming it is highly selective).

A similar expression can be obtained for the `or` combination in the *about*, where the distributivity property (8) could be applied as the last step:

$$((R_1 \sqsupset (\text{SEC} \sqsupset \text{XML})) \sqsupset (\text{REGION} \sqcup \text{ALGEBRA})$$

4. UNDERSTANDING IR

In this section some issues about the impact of introducing relevance ranking (i.e., score computation) in region algebra are discussed.

4.1 Relevance Ranking in Region Algebra

Relevance ranking cannot be explicitely expressed in the native relational algebra. To store the score information additional attribute for each entry in relational tables must be introduced. It stores the ranking score values for particular XML regions during the query execution. Furthermore, a number of operators have to be defined in the relational algebra which combination should express the score computation, i.e. instead of a join operator in Figure 4 we would use a combination of relational score operators. However, the introduction of score operators in the region algebra is easier and more elegant than in the relational algebra since the score computation is done on the right level of abstraction (logical level) and without considering the issues of how these operators are implemented on the physical level, i.e., in the relational algebra.

We use the same example query expressions given in Figure 2, except that we treat the *about* clause as a vague constraint instead of the strict interpretation in previous sections. Thus, paths and terms in the about clause do not have to be strictly matched, and the vague match is defined by the retrieval model. To be able to express IR-like search in XML databases the region algebra can be extended to support ranked retrieval. For that purpose the basic region algebra data model is extended with an additional attribute called *score* (denoted with p to resemble the probabilistic value).

To enable the score computation and the region ranking based on computed scores new region algebra operators are introduced. For each binary region algebra operator defined in Table 4 the probabilistic counterpart is defined. To distinguish between basic region algebra operators and score region algebra operators we use the index $_p$ for score operators. The score operators are depicted in Table 5. Note that the operators $\sqsupset_p$ and $\sqsubset_p$ produce all regions from the first operand (R_1) as a result, except that the score value (p_3) is changed according to the operator definition. The definitions of other two operators ($\sqcap_p$ and $\sqcup_p$) are similar to the definitions of basic ones except that they include score manipulation.

In the definition of score operators we introduced two complex scoring functions: $f_{\sqsupset}$ and $f_{\sqsubset}$, as well as two abstract operators: $\otimes$ and $\oplus$, which define the retrieval model. By using such definition of operators we leave their exact implementation for the physical level (for more details on this issue see [12]). However, for the operator $\oplus$ we assume that there exist a default value for score (denoted with d), and in case when the region r_1 is not present in the region set R_2 the score is computed as $p3 = p1 \oplus d$ and in case when the region r_2 is not present in the region set R_1 the score is computed as $p3 = d \oplus p_2$.

The functions $f_{\sqsupset}(r, R)$ and $f_{\sqsubset}(r, R)$, applied to a region r_1 and region set R_2, result in the numeric value that takes into account the score values of region $r_2 \in R_2$ and the probabilistic value that reflects the structural relation between the region r_1 and the region set R_2. For containing operator usually many regions from the region set R_2 are contained in the region r_1 (e.g., sections inside the article element). Although for contained by operator there is a small chance that the region r_1 is contained by a set of regions present in R_2, it can happen that e.g., there are nested XML elements with the same name (e.g., section inside other sections), and therefore, one region can be contained in multiple regions with the same name.

Following the previous discussion we can define complex functions as follows:

$$f_{\sqsupset}(r, R) = p * \sum_{\bar{r} \in R \sqsubset R'} (g_{\sqsupset}(\bar{r}, r) * \bar{p})$$

$$f_{\sqsubset}(r, R) = p * \sum_{\bar{r} \in R' \sqsupset R} (g_{\sqsubset}(\bar{r}, r) * \bar{p})$$

We assume that R' is the region set containing a single region r, and $g_{\sqsupset}(\bar{r}, r)$ and $g_{\sqsubset}(\bar{r}, r)$ are abstract functions

Table 5: Region algebra operators for score manipulation.

Operator	Operator definition
$R_1 \sqsupset_p R_2$	$\{r \mid r_1 \in R_1 \wedge (s,e,n) := (s_1,e_1,n_1) \wedge p := p_1 * f_{\sqsupset}(r_1, R_2)\}$
$R_1 \sqsubset_p R_2$	$\{r \mid r_1 \in R_1 \wedge (s,e,n) := (s_1,e_1,n_1) \wedge p := p_1 * f_{\sqsubset}(r_1, R_2)\}$
$R_1 \sqcap_p R_2$	$\{r \mid r_1 \in R_1 \wedge r_2 \in R_2 \wedge (s_1,e_1,n_1) = (s_2,e_2,n_2) \wedge (s,e,n) := (s_1,e_1,n_1) \wedge p := p_1 \otimes p_2\}$
$R_1 \sqcup_p R_2$	$\{r \mid r_1 \in R_1 \wedge r_2 \in R_2 \wedge ((s,e,n) := (s_1,e_1,n_1) \vee (s,e,n) := (s_2,e_2,n_2)) \wedge p := p_1 \oplus p_2\}$

used to define the score propagation based on the structural relation between the region r and regions in the region set R. In the straightforward implementation functions $g_{\sqsupset}(\bar{r}, r)$ and $g_{\sqsubset}(\bar{r}, r)$ can be constant functions equal to e.g., 1. If we base the retrieval model on the term frequency, former function can be defined as $g_{\sqsupset}(\bar{r}, r) = \frac{size(\bar{r})}{size(r)}$. Similarly latter function can be defined as $g_{\sqsubset}(\bar{r}, r) = \frac{size(\bar{r})}{\sum_{\bar{r}} size(\bar{r})}$. Since the exact retrieval model is not the main issue in this paper we will not elaborate more on it.

The abstract operator $\otimes$ specifies how scores are combined in an `and` expression, while the operator $\oplus$ defines the score combination in an `or` expression inside the NEXI predicate. In this paper we take the simple approach where $\otimes$ is a product of two score values, while $\oplus$ is the sum of scores, as it shows good behavior for retrieval (see [12]).

To illustrate the elegance of expressing score computation in region algebra we show how we can express NEXI query 1 in score region algebra:

(BDY $\sqsupset_p$ ((SEC $\sqsupset_p$ STRUCTURED)$\sqcap_p$(SEC $\sqsupset_p$ DOCUMENTS))) $\sqsubset$ ARTICLE

which very much resembles the original query plan for example query 1 given in Figure 5.

4.2 Properties of Score Operators

Considering the properties of score operators we can exert that some of the properties follow ones defined for the region algebra without scores, some of them hold only if some conditions are satisfied (conditional properties which depend on the underlying retrieval model), and some of them are no longer valid.

Operator $\sqcap_p$ defines the Boolean-like AND combination of scores obtained for two regions with the same region bounds (i.e., s and e values). It preserves the identity and inverse element properties from the $\sqcap$ operator (property (1)), but only in case the default score value for all regions in the initial region set is the value which is the identity element for abstract operator $\otimes$, i.e., 1 for the multiplication.

$$R \sqcap_p C = C \sqcap_p R = R, \text{ i.e., } p * 1 = 1 * p = p, \forall r \in R \qquad (14)$$

Furthermore, the operator $\sqcap_p$ is commutative or associative (properties (3) and (5)) if the operator $\otimes$ is commutative or associative, respectively, which is the case for multiplication.

An extension of the set union operator is given by the $\sqcup_p$ operator. It defines the Boolean-like OR combination of scores for two regions. Similarly to $\sqcap_p$ operator, operator $\sqcup_p$ preserves the identity and inverse element properties from the $\sqcup$ operator (property (2)) but only in case the default value (d) taken for the $\sqcup_p$ operator is the value which is the identity element for the abstract operator $\oplus$, i.e., 0 for the summation in our case.

$$R \sqcup_p \emptyset = \emptyset \sqcup_p R = R, \text{ i.e., } p + 0 = 0 + p = p, \forall r \in R \qquad (15)$$

As in the $\sqcap_p$ operator case, commutativity and associativity properties depend on the definition of $\oplus$ operator. In other words, operator $\sqcup_p$ is commutative or associative (properties (4) and (6)) if the operator $\oplus$ is commutative or associative, respectively, which is true for the summation.

Following the reasoning above and the fact that each region can equally likely be the right answer to a user query, we will consider that the default value for region score in the initial data set C is 1, from now on, and that the default value for score d of a region not present in the region set for operator $\sqcup_p$ is 0.

Based on the definition of operators using the region frequency it can be proven that the operators $\sqsupset_p$ and $\sqsubset_p$ do not distribute over the operator $\sqcup_p$ in general case (properties (7) and (8)). However, the operator $\sqcap_p$ distributes over the operator $\sqcup_p$, since $*$ distributes over $+$ (property (9)). Vice versa is not the case (property (10)).

$$R_1 \sqcap_p (R_2 \sqcup_p R_3) = (R_1 \sqcap_p R_2) \sqcup_p (R_1 \sqcap_p R_3) \qquad (16)$$

There are some additional conditional properties of score operators which can be of interest for the optimization. If we assume that functions $f_{\sqsupset}(r, R)$ and $f_{\sqsubset}(r, R)$ are not dependant on the score value of a region r (i.e., $f_{\sqsupset}(r, R) = f_{\sqsupset}(s, t, n, R)$ and $f_{\sqsubset}(r, R) = f_{\sqsubset}(s, t, n, R)$ property (11) holds for $op1_p \in \{\sqsupset_p, \sqsubset_p\}$ and $op2_p \in \{\sqsupset_p, \sqsubset_p\}$.

$$(R_1\ op1_p\ R_2)\ op2_p\ R_3 = (R_1\ op2_p\ R_3)\ op1_p\ R_2 \qquad (17)$$

In other words the score for each region in the result region set, denoted with p, is computed as:

$$p = (p_1 * f(r_1, R_2)) * f(r_1, R_3) = (p_1 * f(r_1, R_3)) * f(r_1, R_2)$$

We use $f(r, R)$ to denote one of the functions $f_{\sqsupset}(r, R)$ or $f_{\sqsubset}(r, R)$.

Furthermore, if the score value for all regions in the first operand R_1 is equal to 1 (default value for all regions), and we assume that the regions in each operand, R_2 and R_3, have the same score value, denoted with p_2 and p_3, property (12) holds.

$$(R_1\ op1_p\ R_2)\ op2_p\ R_3 = (R_1\ op1_p\ R_2) \sqcap_p (R_1\ op2_p\ R_3) \qquad (18)$$

i.e., for every region in the result set we obtain score p:

$$\begin{aligned} p &= (1 * \sum_{\bar{r}} (g(\bar{r}, r_1)) * p_2) * \sum_{\bar{r}} (g(\bar{r}, r_1)) * p_3 \\ &= (1 * \sum_{\bar{r}} (g(\bar{r}, r_1)) * p_2) * (1 * \sum_{\bar{r}} (g(\bar{r}, r_1)) * p_3 \end{aligned}$$

where $g(\bar{r}, r)$ is used either for $g_{\sqsupset}(\bar{r}, r)$ or for $g_{\sqsubset}(\bar{r}, r)$ and $\bar{r} \in R \sqsupset R'$ or $\bar{r} \in R' \sqsubset R$ based on the type of operators $op1_p$ and $op2_p$.

If we consider the expression R_4 in the NEXI query 2 we can come up with two query plans shown below.

((P $\sqsubset_p$ R_2) $\sqsupset_p$ INFORMATION) $\sqcap_p$ ((P $\sqsubset_p$ R_2) $\sqsupset_p$ RETRIEVAL)

and ((P $\sqsupset_p$ INFORMATION) $\sqcap_p$ (P $\sqsupset_p$ RETRIEVAL)) $\sqsubset_p$ R_2

Although they are almost the same we could not apply property (18) to the first query plan since the scores of regions in $\text{P} \sqsubset_p R_2$ are not equal to 1 in general case. For the second query plan the score value for all regions in P is 1 and the property can be applied. Thus, at the end we can come up with the query plan as shown below:

$$((\text{P} \sqsupset_p \text{INFORMATION}) \sqsupset_p \text{RETRIEVAL}) \sqsubset_p R_2$$

A version of property (13) for score operators does not hold for $\sqcap_p$ score operator, but holds for $\sqcup_p$. For example next equation is not true.

$$(R_1 \sqcap_p R_2) \sqsupset_p R_3 = (R_1 \sqsupset_p R_3) \sqcap_p (R_2 \sqsupset_p R_3)$$

It will be true only if $f_{\sqsupset}(r_{1,2}, R_3) = 1$ which is not true in general case:
$(p_1 * p_2) * f_{\sqsupset}(r_{1,2}, R_3) \neq (p_1 * f_{\sqsupset}(r_{1,2}, R_3)) * (p_2 * f_{\sqsupset}(r_{1,2}, R_3))$
However, next equation is true for $op_p = \{\sqsupset_p, \sqsubset_p\}$.

$$(R_1 \sqcup_p R_2)\ op_p\ R_3 = (R_1\ op_p\ R_3) \sqcup_p (R_2\ op_p\ R_3) \qquad (19)$$

i.e.,
$(p_1 + p_2) * f_{op}(r_{1|2}, R_3) = (p_1 * f_{op}(r_{1|2}, R_3)) + (p_2 * f_{op}(r_{1|2}, R_3))$

5. CONCLUSIONS AND FUTURE WORK

In this paper we address the problem of translating and executing IR-like queries over XML documents stored in relational databases. We exert the usefulness of intermediate logical level, for which we chose region algebra. The region algebra provides a number of properties which can be used for *query optimization* on the logical level of a database. Furthermore, the region algebra can support score operators used for ranked retrieval as an integral part of the algebra, and not as a sideffect. The expressiveness considering ranked retrieval in our region algebra is far more sophisticated than in other region algebra approaches that support ranked retrieval, like [2] and [13].

An important property of the region algebra is that expressing query plans using the operators given in Table 4 and Table 5 preserves *data independence* between the conceptual, the logical, and the physical level of a database. Similarly, these operators partially enable the separation between the structural query processing and the underlying probabilistic model used for ranked retrieval: a design property termed content independence in [5].

We are planning to further investigate the usefulness of region algebra operator properties and to experimentally evaluate the benefits of intermediate logical level. Further study on the influence of the definition of score operators (score functions and abstract operators) on score operator properties is needed. Our future research is also concerned with the consequences of modifying or changing the retrieval model used, e.g., by adding background statistics (i.e., collection frequency, document frequency) or by adapting the model for phrase search, etc. Moreover, we will work on the theoretical foundations as a support of retrieval models used for handling scores in region algebra.

6. REFERENCES

[1] S. Amer-Yahia, C. Botev, and J. Shanmugasundaram. TeXQuery: A Full-Text Search Extension to XQuery. In *Proceedings of the 13th conference on World Wide Web*, pages 583–594, 2004.

[2] F.J. Burkowski. Retrieval Activities in a Database Consisting of Heterogeneous Collections of Structured Texts. In *Proceedings of the 15th ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 112–125, 1992.

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

[4] 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):43–56, 1995.

[5] A.P. de Vries. Content independence in multimedia databases. *Journal of the American Society for Information Science and Technology*, 52(11):954–960, September 2001.

[6] D. Florescu and I. Manolescu. Integrating Keyword Search into XML Query Processing. In *Proceedings of the 9th International World Wide Web Conference*, pages 67–76, 2000.

[7] N. Fuhr and K. Großjohann. XIRQL: A Query Language for Information Retrieval in XML Documents. In *Proceedings of the 24th ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 172–180, 2001.

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

[9] T. Grust. Accelerating XPath Location Steps. In *Proceedings of the 21st ACM SIGMOD International Conference on Management of Data*, pages 109–120, 2002.

[10] Torsten Grust and Maurice van Keulen. Tree Awareness for Relational DBMS Kernels: Staircase Join. In H. M. Blanken, T. Grabs, H.-J. Schek, R. Schenkel, and G. Weikum, editors, *Intelligent Search on XML*, volume 2818 of *Lecture Notes in Computer Science/Lecture Notes in Artificial Intelligence (LNCS/LNAI)*, pages 179–192. Springer-Verlag, Berlin, New York, etc., August 2003.

[11] J. Jaakkola and P. Kilpelainen. Using sgrep for Querying Structured Text Files. Technical Report C-1996-83, Department of Computer Science, University of Helsinki, 1996.

[12] 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 Initiative on the Evaluation of XML Retrieval (INEX 2003)*, ERCIM Workshop Proceedings, 2004.

[13] Katsuya Masuda. A Ranking Model of Proximal and Structural Text Retrieval Based on Region Algebra. In *Proceedings of the ACL-2003 Student Research Workshop*, pages 50–57, 2003.

[14] V. Mihajlović, D. Hiemstra, and P. Apers. On Region Algebras, XML Databases, and Information Retrieval. In *Proceedings of the 4th Dutch-Belgian Information Retrieval Workshop*, 2003.

[15] G. Ramirez and A. P. de Vries. Combinining Indexing Schemes to Accelerate Querying XML on Content and Structure. In *Proceedings of the first Twente Data Management Workshop (TDM'04),* to apear, 2004.

[16] A. Trotman and R. A. O'Keefe. The Simplest Query Language That Could Possibly Work. In N. Fuhr, M. Lalmas, and S. Malik, editors, *Proceedings of the Second Workshop of the INitiative for the Evaluation of XML retrieval (INEX)*, ERCIM Publications, 2004.

[17] C. Zhang, J. Naughton, D. DeWitt, Q. Luo, and G. Lohman. On Supporting Containment Queries in Relational Database Management Systems. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, pages 425–436, 2001.

Toward Entity Retrieval over Structured and Text Data

Mayssam Sayyadian, Azadeh Shakery, AnHai Doan, ChengXiang Zhai
Department of Computer Science, University of Illinois at Urbana-Champaign
{sayyadia, shakery, anhai, czhai}@cs.uiuc.edu

ABSTRACT

Many real-world applications increasingly involve both structured data and text. Hence, managing both in an efficient and integrated manner has received much attention from both the IR and database communities. To date, however, little research has been devoted to semantic issues in the integration of text and data. In this paper we introduced a problem in this realm: *entity retrieval*. Given data fragments that describe various aspects of a real-world entity, find all other data fragments as well as text documents that describe that same entity. As such, entity retrieval is a novel retrieval problem, which differs from both regular text retrieval and database search in that it explicitly requires matching information at the *semantic* level; matching syntactically as done in the current search engines and relational databases would be inherently non-optimal. We define entity retrieval and conduct a case study of retrieving information about a researcher from both the Web and a bibliographic database (DBLP). We propose several methods for exploiting the structured information in the database to improve entity retrieval over the text collection. Specifically, we present a query expansion mechanism based on extracted information from structured data. Experiment results show that selectively using more structured information to expand the text query improves entity retrieval performance on text. We conclude the paper with future research directions for entity retrieval.

Keywords

Entity Retrieval, Semantic Retrieval, Query Expansion, Information Retrieval, Disambiguating Search Results

1. INTRODUCTION

Traditionally, text and structured data have been managed in different ways, resulting in two related but separate fields: information retrieval (IR) and management of structured data, as exemplified by relational database systems.

However, the boundary between the two fields has become much blurred in the past few years. For example, database researchers have been actively studying how to incorporate text search facilities into relational database systems, while IR researchers have investigated exploiting structure over the data (e.g., XML structure) to perform more expressive keyword queries (see the related work section). This trend is motivated largely by a fundamental need to manage text and structured data in an integrated way, given the proliferation of applications that involve *both* kinds of data. In such applications, users seek all information that can help solving a problem, regardless of whether the information comes from text or structured data. Hence, it is important to develop techniques that enable the efficient discovery of information from both text and structured data.

This paper aims to develop such techniques. It focuses on *entity retrieval*, the problem of finding missing information about a *real-world entity* (e.g., person, course, or place) from both text and structured data, given some initial information about the entity. Examples include finding information about a researcher from the World-Wide Web and a set of bibliographic databases, given a paper written by that researcher, or finding information about a product of a business competitor, from online discussion groups and a purchase database.

As described, entity retrieval (or ER for short) plays a fundamental role in many information seeking contexts, but has received little attention. Furthermore, current IR and database techniques do not appear to be well suited to this problem. IR techniques has been optimized largely for retrieving documents on *general topics* (e.g., "data mining"). When applied to retrieving documents about a particular real-world entity, say person, they often return documents about multiple persons that are mixed together: a highly undesirable situation. Current relational database techniques also do not fare well on this problem, as they can only perform *syntactic matching* and thus often return information about multiple real-world entities.

It is also worth mentioning that the notion of relevance in text retrieval is *subjectively* defined; the criterion is in the user's mind. But in ER, the notion of relevance has an *objective* definition. For example, we may simply use a person's name as a query to search over text. But this only works when the name is not ambiguous. Similarly, querying a database with a person's name may return tuples that describe a different entity who happens to have the same name as our target entity. Thus to optimize ER results, we must infer entity-level semantics, that is, decide if two data fragments refer to the same real-world entity. This is a major reason on why ER is a challenging problem.

WIRD'04, the first Workshop on the Integration of Information Retrieval and Databases (WIRD'04), Sheffield, United Kingdom
© 2004 the author/owner

Clearly, we can study the ER problem given (1) only structured data, (2) only text data, and (3) a combination of text and structured data. In practice, all three settings arise very often. In [12] we have carried out a detailed study of setting (1). We are currently studying settings (2) and (3).

This paper focuses on setting (3), and examines a specific problem in this setting, namely, *given both text and structured data, can we exploit structured data to achieve better ER over text, compared to ER over text alone?* We first give an informal definition of this problem, then propose several solutions. The key insight behind our solutions is that we can selectively expand the ER query over text with keywords obtained from the given structured data.

The key challenge in realizing the above idea is to discover which parts of structured data should be considered, and which keywords to "pull" from those parts (to be used to augment the query). We note that many variations of query expansion have been considered in the IR community, but few if any works have considered query expansion using structured data.

We then evaluate our solutions on a real-world setting, in which we retrieve information about a researcher from both the World-Wide Web and DBLP, a bibliographic database. Our preliminary experimental results show that selectively using more structured information to expand the text query improves the ER performance on text.

The rest of the paper is organized as follows. We define the ER problem in Section 2. In Section 3 we present several ER methods that can exploit structured information to improve ER over text. We discuss experiment results in Section 4. Finally, we review related work in Section 5 and discuss future research directions in Section 6.

2. THE ENTITY RETRIEVAL PROBLEM

Informally, given a collection C of text documents, a relational database T, and an ER query that specifies the description of a particular real-world entity called *target entity*, the ER problem is to find documents in C and data fragments in T that describes the entity. For example, we may want to retrieve information about a particular researcher from both a bibliographic database (e.g., DBLP) and a text collection (e.g., web pages) given some information about the researcher (e.g., the researcher's name and affiliation as well as his home page).

An ER query consists of two parts: a constraint and a template. The constraint specifies whatever information a user already knows about the target entity. This information typically includes some attribute values in the database and some documents in the text collection. For example, a query may specify the following: (1) the name of the target researcher is "*John Smith*", (2) he has published a paper titled "*A study of entity retrieval*" in the conference *SIGIR 2004*, (3) the text of his home page. The first two specify the values of three attributes in a bibliographic database – author, title, and conference. The last one gives one example of text documents. The target entity is a person.

The template of an ER query specifies what information the user wants returned about the target entity. To continue with the above example, the user may specify that he or she want to retrieve only all papers that the target researcher has written as well as any course that is reading any of these papers. In this case, most of the papers may come from the bibliographic database, while information about courses (that read these papers) may come from the World-Wide Web.

While we could have defined ER on only a relational database or only a text collection, it is interesting to consider ER over both text and relational data. This way it is possible to leverage useful context information from different types of data to improve retrieval accuracy. Indeed, in this paper, we will show that the performance of ER over a pure text collection can be improved by exploiting the structured data related to the target entity in a relational database. As a preliminary study, we will consider a simplified ER problem, though we will also touch on the general ER problem and the major research issues.

2.1 The Simplified ER Problem:

Let $E = \{e_1, ..., e_{|E|}\}$ be a set of real world entities. Let T be a relational table with attributes $A = \{A_1, ..., A_k\}$, such that each tuple in T describes some aspects of entities in E. Let $C = \{d_1, ..., d_n\}$ be a set of documents. A user who is interested in entity e_i would pose a query Q which contains some or all of the following information:

1. A basic description of e_i (e.g., name of a researcher),
2. Structured context information $c_1 = v_1, c_2 = v_2, \ldots, c_k = v_k$, where $c_j \in A$ is an attribute, and v_j is the corresponding value.
3. Unstructured context information in the form of a set of example text documents that are known to be about the target entity.
4. A template that is specified as a set of attributes in A, or in a more expressive language (e.g., SQL) over the schema of table T, to identify the types of information the user would like to obtain about the target entity e_i.

Note that both (2) and (3) can be regarded as providing context information for (1).

We further simplify the above problem by focusing only on ER over text. However, the ideas that we offer here also carry over to the setting of ER over structured data.

3. ER METHODS

Because of the need for semantic-level matching, ER is in general a quite challenging task. A straightforward matching at the syntactic level (e.g., by retrieving researchers with the exact same name) is clearly insufficient to achieve good performance. Intuitively, the ER task involves two subtasks: understanding the target entity referred to by a query and deciding whether a data fragment is about the same entity.

Unfortunately, we generally have no *unique* representation of a real world entity, which makes this task difficult. The

lack of such a unique representation is obvious in text information where the primary representation is words describing an entity. It is also true in relational data, for example, when two people have the same name. It can be argued that entities in relational databases have unique identifiers such as primary key, but again when considering different relations and different mentions of a real-world entity we have no unique representations to identify the entity. Another complication is that there may be variations of spelling or wording for the same attribute value. For example, a name may be abbreviated. The challenge is thus to infer whether an information item is about the same entity referred to by a query based on the generally non-unique descriptions of various kinds of entities.

Since we have two different types of data, a straightforward strategy is to separate ER over text from ER over structured data. However, a complete separation would deprive us of the opportunity to leverage different types of data to enrich our representation of the target entity. Indeed, a main point we make in this paper is to advocate methods that take advantage of *all* information available to us, and in particular, to leverage structured information to improve ER over text.

We now present four different methods for ER over text, corresponding to different types of queries and/or different levels of sophistication in leveraging the related structured data. We assume that we have available a basic text retrieval method that can match any text query with the documents in the text collection and generate a relevance score for each document. The text documents can then be ranked according to their relevance scores. The retrieval results are obtained by applying some pre-determined score threshold cutoff. Currently this threshold is manually set and tuning it is the subject of future work. There are many choices of text retrieval methods (e.g., [35, 9]), but this is not our focus. Instead, we are interested in different ways for constructing a regular text query based on the ER query, especially on how we may exploit structured information to construct a potentially better text query.

The first, also the simplest method, is to use as the text query only the text description of the target entity in the ER query. This method, called Text Only, does not exploit any structured information, and will be used as our baseline method.

The second method uses immediate and highly reliable context information from the database to expand the Text Only query, and is referred to as Add Immediate Structure. The third method (Add All Structures) uses more structured information than the second method, and expands the Text Only query with all structured information that it found to be describing the target entity.

The last method attempts to further improve the quality of the query in Add All Structures, by *selectively* using only the most relevant attribute values to expand the Text Only query, and will be referred to as Add Selective Structures. We now describe these methods in detail.

3.1 Text Only

The Text Only method ignores any structured information in the query, and simply uses the text description of the target entity as the text query for searching the text collection. For example, when the target entity is a person, the person's name can be used directly as the query.

This method only uses the minimum amount of information in the query, and will clearly not perform well when the description is non-discriminative. For example, when the name of a researcher is ambiguous, using only the name may return documents about a different person who happens to have the same name.

When the user can provide some structured information about the target entity, it is intuitively better to enrich our query with such extra structured information. For example, if a user knows one paper published by a researcher, the title and/or the conference information of this paper can then be exploited to expand the query. Note that, when searching information about a researcher, even if the name is not ambiguous, such an expanded query can still be expected to perform better than Text Only, since the additional structured information can be expected to help refine the text query and thus improve the retrieval accuracy. This strategy leads to the method Add Immediate Structure.

3.2 Add Immediate Structure

This method expands the Text Only query with the immediate (one record), but highly reliable (e.g., provided by the user or automatically obtained when the name is not ambiguous) structured information. To obtain the expanded query, we simply treat the database record (e.g., the title, coauthors, and the conference name of the known publication) as additional text and append it to the original text query. More sophisticated expansion methods are possible. For example, we can incorporate the record text with different weights. But we leave this as our future work. This is also true for all other methods that we describe below.

3.3 Add All Structures

Since in general we can expect the retrieval performance to improve as we provide more relevant context to expand the query, we can expand the query with as much structured information as possible. That is, we go further than just adding the most reliable database record(s) and expand the query with any related structured information (about the target entity) that we can obtain from the database. This method is called Add All Structures and would require doing ER over the database.

All the additional structured information is then appended to the original text query. In the case of searching information about a researcher, this corresponds to using the papers published by the researcher that we can obtain from the bibliographic database to expand the query. Once again, this is the simplest expansion method, and more sophisticated methods that involve weighting the structured information differently than the original text may be more appropriate as we may have noise in the obtained structured information.

3.4 Add Selective Structures

While ADD ALL STRUCTURES allows us to use the "maximum" amount of relevant structured information to expand the text query, not the fields in the extracted database records are equally useful. Indeed, some fields (e.g., the publication dates) may be quite distracting, especially if they are not weighted appropriately. Thus one idea to further exploit the database schema is to *selectively* add only those highly relevant attribute values. This is the ADD SELECTIVE STRUCTURES method. To do so, we can expand the query with only the values of the *most relevant attribute*, which is to be selected with an attribute selection method. So devising a good method for selecting more informative, valuable, and disambiguating attributes from the structured data is very important.

One such method we devise and use is to score each attribute in the database schema based on its frequency in the top a few text documents retrieved by previous methods. A method for calculating the effectiveness of each attribute and selecting the best to use in ADD SELECTIVE STRUCTURES is as follows:

- W = *set of documents returned by* TEXT ONLY *method.*
- D = *The first* $\frac{size(W)}{4}$ *documents in* W
- *For each attribute* i *and all its values in the database schema;* $a_{i,j}$, *let* $s_{i,j}$ = *the number of occurrence of* $a_{i,j}$ *in* D
- *let the effectiveness score of attribute* i *be:* $score_i = \sum_j s_{i,j}$
- *normalize* $score_i$ *according to the size of* D *and number of values of* a_i
- BEST ATTRIBUTE = $argmax_i\{score_i\}$

The assumption is that an attribute can be expected to be more useful if it occurs more frequently in the top text documents. Once again, we concatenate the selected attribute values with the original text query to generate an expanded query.

4. EMPIRICAL EVALUATION

We evaluate our solutions using a bibliographic database (DBLP) and a collection of Web pages (constructed using Google). The ER task is to retrieve information about a researcher from both the database and the Web-page collection, given the researcher's name and some citation information from the bibliographic database. Our goal is to study whether the performance of ER over Web pages can be improved as we exploit more structured information.

4.1 Data Set and Evaluation Measures

Data Set: Our structured data source is the DBLP bibliography database at $www.informatik.uni-trier.de/\ ley/db$. DBLP lists more than 460,000 articles and has become a very popular bibliography search engine for the computer science research community. Our test bed consists of 11 researchers from DBLP whose names are ambiguous. Out of these researchers, 36% of the researchers have totally ambiguous names, meaning that when searching the Web with their names, we come across different real world people (entities) with completely identical names. For example "Amit Singhal" (one of our queries) appears to match two people – one is a researcher in IR, while the other is a researcher in system architecture. 45% of the names are partially ambiguous, meaning that there are entities with similar names differing only in the middle name, or initials. For example "Richard Cox" and "Richard V. Cox" are two researchers in this group.

Our text collection is constructed as follows. We use the name of each of the 11 researchers as a simple keyword query to search with Google, and take the top 100 Web pages as a "working set". We combine these working sets for all the 11 researchers to form our text collection. Since we have deliberately chosen ambiguous names, a working set generally has documents about distinct entities that share the same name, thus our text collection is adequate for discriminating different entity retrieval methods.

In order to measure the ER accuracy, we create a gold standard by manually reviewing these results and judging whether they are relevant to the corresponding entity query (i.e., whether they are truly about the target researcher). In this way, we can create a judgment file similar to those used in standard IR evaluation [37]. Note that although our judgments are not on the entire Web, they are still very useful for comparing different methods.

Evaluation Measures: We use precision, recall, and the $F1$ measure to evaluate the ER results. Given a set of result pages for a researcher query, precision is the percentage of pages retrieved that are relevant (i.e., truly about the target researcher), and the recall is the fraction of all relevant pages in our working set that are returned as results. $F1$ is a combination of precision and recall, and will be our primary measure when comparing different methods. It is given by $F1 = \frac{2PR}{P+R}$, where P and R are precision and recall respectively.

4.2 Implementation of ER Methods

Our general experiment procedure is to use one of our methods to construct a text query, which is then used to search the text collection with a standard IR method. We use the Lemur toolkit [26] to index our web text collection and perform text retrieval using the popular vector-space retrieval model [32] with BM25 TF weighting [30]. We used the default parameter seating provided in Lemur. Since we use a constant score threshold for cutoff, it is important to make the scores of all documents for all queries comparable. We thus normalize all the similarity scores by dividing the scores by the maximum score given by the top-ranked document. The cutoff value here is set manually and we will further look into it in future.

The implementation of the TEXT ONLY method is trivial, as it simply involves taking the name of the researcher as the text query. The ADD IMMEDIATE STRUCTURE method generates a text query by manually picking one citation that belongs to the target researcher from DBLP and concatenating the text in all fields with the researcher's name. This simulates a query posed by a user who has some limited additional information about the target researcher.

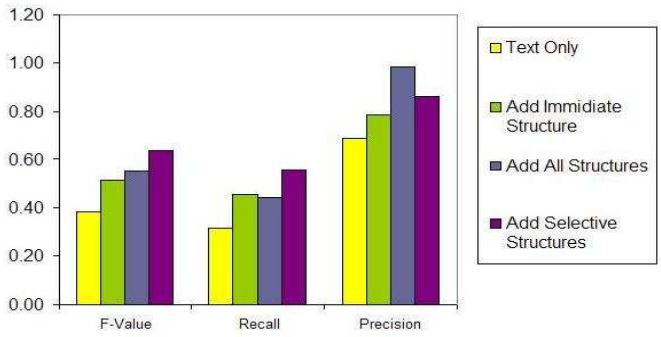

Figure 1: Effect of adding structured information to the ER query.

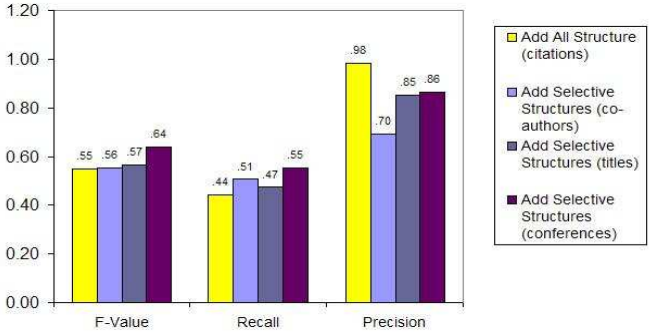

Figure 2: Comparing the effectiveness of using different attributes to expand the ER query.

The ADD ALL STRUCTURES method assumes that we can perform ER over the database with reasonable accuracy, a task that we have studied in [12]. The method takes all papers that are published by the target researcher and concatenate all of them as text with the researcher's name to generate an expanded text query. We implement this method by exploiting the DBLP web server to find relevant citations to the target researcher. Specifically, we search DBLP with the researcher's name and manually choose all the papers that belong to the target researcher. In this way, all the chosen papers would be relevant, and thus the ADD ALL STRUCTURES method is quite similar to relevance feedback in pure text retrieval.

The ADD SELECTIVE STRUCTURES method is implemented in almost exactly the same way as ADD ALL STRUCTURES with only one difference: while the ADD ALL STRUCTURES method adds all the text information from a relevant paper, which includes coauthors, titles, dates, and conferences, the ADD SELECTIVE STRUCTURES method only adds all the information in the values of the most highly scored attribute, which is the conference name. We score each attribute of the database schema by a formula based on normalized frequency of the values of the corresponding attribute in the top few web pages from previous text retrieval results, and only add the text values of the attribute with the highest score.

4.3 Result Analysis

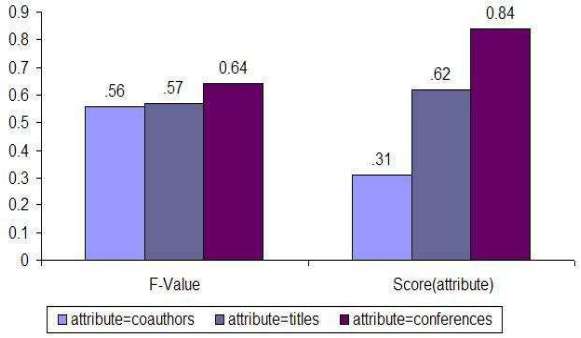

Figure 3: Correlation between the attribute scores and the F-Values of using them.

4.3.1 *Comparison of the four proposed methods*

In Figure 1, we compare the four proposed methods in terms of all three measures. The baseline is the TEXT ONLY method which uses only the researcher's name as the query. As expected, ADD IMMEDIATE STRUCTURE performs much better than the baseline by all three measures, indicating that the added paper provides very useful context to focus the results on the target person entity.

As we use more structured information in the ADD ALL STRUCTURES method, we see that we can further improve the precision and F1 value, though the recall decreases slightly. The increase of precision and decrease in recall suggest that the number of pages returned is reduced. This may be caused by a dramatic increase in the similarity score of a few web pages (e.g., the publication page of the researcher); with the same relative score threshold, the effect is a decrease in the number of pages that can pass the threshold.

Finally, by comparing ADD ALL STRUCTURES with ADD SELECTIVE STRUCTURES, we see that selectively using the values from the most effective attribute to expand the query helps improve F1 and recall, though the precision is decreased. This means that the number of pages returned is likely increased when using only the selected attribute values, and the reason may be the top score is relatively low compared with using all the attribute values.

Overall, the F1 value consistently increases as we selectively use more structured information. It is interesting to note that ADD SELECTIVE STRUCTURES is better than ADD IMMEDIATE STRUCTURE by all three measures, whereas ADD ALL STRUCTURES has a lower recall than ADD IMMEDIATE STRUCTURE.

This suggests that selectively adding the values from the most effective attributes is a more robust way of expanding the query than using all the attribute values. Intuitively, this also makes sense, as the selected attribute, which is the conference name, is indeed a good discriminator for identifying pages about the target researcher; other fields, such as co-authors and titles may introduce noise.

4.3.2 *Effectiveness of Attribute Selection*

ADD SELECTIVE STRUCTURES uses a frequency-based scoring formula to select the best attribute for query expan-

sion, which is shown to perform better than using all the attributes. In Figure 2, we look into the possibilities of using different attributes, and compare the results from using the co-authors, paper titles, and conference names, respectively. We see that the conference name attribute is indeed the best for query expansion.

To further analyze the effectiveness of our attribute scoring method, we plot the scores of the three attributes together with their corresponding F1 values in Figure 3. We see that there is a positive correlation between the score of an attribute and the F1 value of using the attribute for query expansion, indicating that our attribute scoring method is effective.

5. RELATED WORK

The topic of integrating structured data and text has received much attention recently, in both the database and IR communities. Works have focused on keyword search over structured data [21, 19, 1], adding text search extensions to database query languages (e.g., XQuery) [3, 14], integrating querying database, web, and XML [17], translating relational querying into fuzzy-search IR style querying and ranking relational queries [2, 7], and also combing structure indices and inverted ones [28]. Several works use structured queries to query meta-data, and use meta-data to improve local search [22, 25].

A key commonality underlying the above works is that they integrate text and structured data at the *syntactic* level. In contrast, our work is at a more *semantic* level, focusing on gluing together disparate data fragments that belong to the same real-world entity.

Entity retrieval is related to the entity matching, *a.k.a.* tuple matching, problem, which has been studied extensively in the database and AI communities (e.g. [36, 8, 24, 39, 5, 23, 4, 33, 18, 20, 16, 29]). But the two problems differ in important aspects. Entity matching decides if two given tuples refer to the same real-world entity, whereas entity retrieval retrieves all tuples (or data fragments in general) that belong to a single entity. As such, ER is a more general problem, and can leverage existing entity matching techniques.

There is also a huge amount of work on XML retrieval where semi-structured queries are supported (e.g., the recent ACM SIGIR workshops on XML retrieval [10, 27] and the INEX workshop on evaluating XML retrieval [15]). While XML document collections also involve both text and structures, they are already "integrated" in some sense. Our problem deals explicitly with integrating a *separate* text collection with a relational database. Moreover, while most of the work on XML retrieval focuses on designing a query language that would allow a user to constrain keyword matching to certain XML tag path as well as the selection of the right information segment to retrieve, we emphasize leveraging structured information to refine a text query and improve entity retrieval performance on text.

The work [34] performs IR over documents that contain both free text and semantically enriched markup. However, it does not address the specific issue of entity retrieval. Our idea of exploiting structured data to refine a text query is related to query expansion and relevance feedback commonly used in information retrieval [31, 13, 35]. It differs from relevance feedback in that we use relevant *structured* information and exploit structures to selectively choose text fragments for query expansion, while the standard relevance feedback uses text documents to expand the query.

Our work is also related to the pseudo feedback technique in IR [11, 6, 38], which exploits likely relevant information in an unsupervised way, but our methods are applied to structured data, rather than the unstructured text documents, and we make a distinction between different attributes and further select most useful attributes. There are also some works on TREC web track task of topic distillation which involves finding relevant home pages, given a broad query (topic). This task could be similar to our work in case we would have some ambiguous topics, which is not the case in topic distillation task.

6. CONCLUSIONS AND FUTURE WORK

Many real-world applications increasingly involve both structured data and text. Hence, managing both in an efficient and integrated manner has become a critical topic that has received much attention. To date, however, little attention has been paid to semantic issues in the integration of text and data.

In this paper we introduced an issue that belongs to the above realm: ER. Given data fragments that describe various aspects of a real-world entity, find all other data fragments as well as text documents that describe that same entity. As such, ER arises in numerous information management applications. It addresses a semantic aspect of integration of text and data.

We then considered how ER over text documents can be significantly aided by the availability of structured data. We developed several solutions and empirically evaluate them on the problem of retrieving information related to researchers from Web documents, by leveraging the structured data of DBLP. The experimental results show the promise of our approach.

As a new retrieval problem, ER poses many interesting and challenging research questions: (1) What is an appropriate query language for ER? A simple combination of a SQL-like query language with an IR-style keyword query is inadequate for a user to express all known information about a target entity. (2) What is an appropriate formal retrieval framework for entity retrieval? Presumably, such a retrieval framework should include, at the minimum, a model of entities, which is missing in all current retrieval frameworks. (3) What are the best strategies and methods for ER? We have shown that leveraging context clues from different data types is clearly beneficial. But the methods we have experimented with are heuristic. It is necessary to develop a more principled ER method that can exploit useful information from *all* types of data, including both text and structured data. One attractive strategy is to perform "mutual feedback", i.e., using the most reliable relevant information from the structured data to improving ranking of text documents, and at the same time, using reliable text documents to help select the most relevant structured data fragments.

Currently we are extending the methods studied in this paper to incorporate ER techniques we have recently developed over relational data. Our long-term goal is to develop a formal model and a general solution for ER. We are also developing novel techniques (e.g., those that exploit the link structure of the Web) to improve retrieval accuracy and exploring applications of ER in information integration.

Acknowledgments: We thank the anonymous reviewers for invaluable feedback on the paper.

7. REFERENCES

[1] S. Agrawal, S. Chaudhuri, and G. Das. DBXplorer: A system for keyword-based search over relational databases. In *Proc. of International Conf. on Data Engineering (ICDE)*, 2002.

[2] S. Agrawal, S. Chaudhuri, G. Das, and A. Gionis. Automated ranking of database query results. In *Proc. of Conf. on Innovative Data Systems Research (CIDR)*, 2003.

[3] S. AmerYahia, C. Botev, and J. Shanmugasundaram. A full-text search extension to XQuery. In *Proc. of WWW Conf.*, 2004.

[4] R. Ananthakrishna, S. Chaudhuri, and V. Ganti. Eliminating fuzzy duplicates in data warehouses. In *Proc. of Int. Conf. on Very Large Databases (VLDB)*, 2002.

[5] M. Bilenko and R. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *Proc. of the ACM SIGKDD Conf. on Knowledge Discovery and Data Mining (KDD)*, 2003.

[6] C. Buckley. Automatic query expansion using SMART: Trec-3. In D. Harman, editor, *Overview of the Third Text Retrieval Conference (TREC-3)*, pages 69–80, 1995. NIST Special Publication 500-225.

[7] S. Chaudhuri, G. Das, V. Hristidis, and G. Weikum. Probabilistic ranking of database query results. In *Proc. of Conf. on Very Large Databases (VLDB)*, 2004.

[8] W. Cohen. Integration of heterogeneous databases without common domains using queries based on textual similarity. In *Proc. of then ACM Conf. of Special Interest Group on Management of Data (SIGMOD)*, 1998.

[9] W. B. Croft and J. Lafferty, editors. *Language Modeling and Information Retrieval*. Kluwer Academic Publishers, 2003.

[10] A. S. D. Carmel, Y. Maarek. XML and information retrieval: a SIGIR 2000 workshop. *ACM SIGIR Forum*, 34(1):31–36, 2000.

[11] D. A. Evans and R. G. Lefferts. Design and evaluation of the CLARIT TREC-2 system. In *Proc. of the Second Text REtrieval Conference (TREC-2)*, 1994.

[12] H. Fang, R. Sinha, W. Wu, A. Doan, and C. Zhai. Entity retrieval over structured data. Technical report, University of Illinois at Urbana-Champaign, Department of Computer Science, Jan. 2004.

[13] N. Fuhr and C. Buckley. A probabilistic learning approach for document indexing. *ACM Transactions on Information Systems*, 9(3):223–248, 1991.

[14] N. Fuhr and K. Grossjohann. XIRQL: A query language for information retrieval in XML documents. In *Proc. SIGIR-2001.*

[15] N. Fuhr and M. Lalmas. Report on the INEX 2003 workshop. *ACM SIGIR Forum*, 38(1):46–51, 2004.

[16] H. Galhardas, D. Florescu, D. Shasha, and E. Simon. An extensible framework for data cleaning. In *Proc. of Int. Conf. on Data Engineering (ICDE)*, 2000.

[17] R. Goldman and J. Widom. WSQ/DSQ: A practical approach for combined querying of databases and the web. In *Proc. of the ACM Conf. of Special Interest Group on Management of Data (SIGMOD)*, 2000.

[18] L. Gravano, P. Ipeirotis, N. Koudas, and D. Srivastava. Text join for data cleansing and integration in an RDBMS. In *Proc. of Int. Conf. on Data Engineering (ICDE)*, 2003.

[19] L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram. XRANK: Ranked keyword search over XML documents. In *Proc. of the ACM Conf. of Special Interest Group on Management of Data (SIGMOD), year=2003.*

[20] M. Hernandez and S. Stolfo. The merge/purge problem for large databases. In *Proc. of then ACM Conf. of Special Interest Group on Management of Data (SIGMOD)*, 1995.

[21] V. Hristidis, L. Gravano, and Y. Papakonstantinou. Efficient IR-style keyword search over relational databases. In *Proc. of the 29th Conf. on Very Large Data Bases (VLDB)*, 2003.

[22] B. Kelly. Using metadata to improve local searching. *Exploit Interactive*, 2000. http://www.exploit-lib.org/issue5/metadata/.

[23] S. Lawrence, K. Bollacker, and C. L. Giles. Autonomous citation matching. In *Proc. of the 3rd Int. Conf. on Autonomous Agents*, 1999.

[24] A. McCallum, K. Nigam, and L. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proc. 6th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, 2000.

[25] J. Milstead and S. Feldman. Metadata: Cataloging by any other name. *Online Magazine*, pages 24–31, Jan/Feb 1999.

[26] P. Ogilvie and J. Callan. Experiments using the lemur toolkit. In *Proceedings of the 2001 TREC conference*, 2002.

[27] N. F. R. Baeza-Yates and Y. Maarek. Second edition of the xml and information retrieval workshop. *ACM SIGIR Forum*, 36(2), 2002.

[28] J. F. N. Raghav Kaushik, Rajasekar Krishnamurthy and R. Ramakrishnan. On the integration of structure indexes and inverted lists. In *Proceedings of ACM SIGMOD 2004*.

[29] V. Raman and J. Hellerstein. Potter's wheel: An interactive data cleaning system. In *The VLDB Journal*, pages 381–390, 2001.

[30] S. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of SIGIR'94*, pages 232–241, 1994.

[31] G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 44(4):288–297, 1990.

[32] G. Salton, C. S. Yang, and C. T. Yu. A theory of term importance in automatic text analysis. *Journal of the American Society for Information Science*, 26(1):33–44, Jan-Feb 1975.

[33] S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In *Proc. of 8th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, 2002.

[34] U. Shah, T. Finin, and A. Joshi. Information retrieval on the semantic web. In *Proc. of The Conf. on Information and Knowledge Management (CIKM)*, 2002.

[35] K. Sparck Jones and P. Willett, editors. *Readings in Information Retrieval*. Morgan Kaufmann Publishers, 1997.

[36] S. Tejada, C. Knoblock, and S. Minton. Learning domain-independent string transformation weights for high accuracy object identification. In *Proc. of the 8th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining*, 2002.

[37] E. Voorhees and D. Harman, editors. *Proceedings of Text REtrieval Conference (TREC1-9)*. NIST Special Publications, 2001. http://trec.nist.gov/pubs.html.

[38] J. Xu and W. Croft. Query expansion using local and global document analysis. In *Proceedings of the SIGIR'96*, pages 4–11, 1996.

[39] W. Yih and D. Roth. Probabilistic reasoning for entity and relation recognition. In *Proc. of International Conf. on Computational Linguistics (COLING)*, 2002.