

An extension to the vector model for retrieving XML documents.

Fabien LANIEL, Jean-Jacques GIRARDOT
École Nationale Supérieure des Mines de Saint-Étienne
158 Cours Fauriel
42023 Saint-Étienne CEDEX 2, FRANCE
Email: {laniel,girardot}@emse.fr

Abstract

The information retrieval community has worked a lot on the combination of content and structure for creating preferment information retrieval systems. With the development of new standards like XML or DocBook, researchers got a growing data-base for creating and testing such systems.

Many XML query engines have been proposed, but most of them do not possibly include a ranking system, because, in all the criteria we can extract from a document, it's not easy to know which one cause a document to be more relevant than another.

This paper describes a reverse engineering method to determine which criteria are the best to optimize the system efficiency.

1 Introduction

During the last twenty years, research in Information Retrieval (IR) has concentrated on two domains: flat documents mainly concerned by textual documents, and structured data such as those that are managed by relational data-bases. With the apparition of XML [8], a new standard for semi-structured data, and the very fast development of corpora of XML documents, new challenges are offered to the research community.

As a matter of fact XML, which offers a very versatile format for information and data exchanging and keeping, can handle a large range of usages from little structured textual documents to strongly typed and structured data. There is however a hidden flaw behind this versatility: while most applications know how to read and write XML documents, there exist no tool that can search with efficiency large quantity of XML documents.

Actually, XML documents that are mainly textual with little structured information (like the text of a novel) can

be easily handled as flat documents using the IR approach; similarly, very structured documents (like the output of a program) can be easily mapped to relations, represented in a classical relational data-base, and queried with SQL. Between these extremes, documents that mix textual contents with complex structures are not satisfactorily handled with these approaches. These include most "digital documents", such as literal text transcribed with TEI [7] or Shakespeare's plays [6], scientific documents represented in the *DocBook* [1] format, and most semi-structured information, like those used to constitute catalogs of industrial products, food, furniture, travels *etc.*

In this last case, we expect to use both contents and structures information of the document to reply efficiently to a query. Many models and methods have been proposed [3, 5, 9, 4] with many criteria (often chosen arbitrarily). So, what criteria should we take into account? Number of appearances of terms? Proximity between them? Relative height between elements? *etc.* Furthermore, is there a criterion more important than others, and in which proportion?

In this paper we will present an approach of reverse engineering process to try to answer those questions. We will present, in a first part, the context of this work: INitiative for the Evaluation of XML Retrieval (INEX). Next, we will describe the methodology we used. Finally we will show some results and discuss the approach.

2 Context

The INEX [2] test collection consists of a set of XML documents, topics and relevance assessments. The documents are articles of the IEEE, which are quite structured and the topics relate to the content only or the structure and the content of the documents.

INEX has defined a query language: NEXI, which proposes an important operator for us: *about()*. For example the NEXI query:

```
//article[about(., java)]  
//sec[about(., implementing threads)]
```

represents the sections about *implementing threads* of articles which about *java* in general. The *about()* operator is exactly an IR operator, in other words the query `//article[about(., java)]` can be processed with a classical flat model of IR.

If we return to the first example, we can solve the first part with classical model (`//article[about(., java)]`) giving a global relevance for the document, we can solve the second part too, if we then consider all the sections as separate documents. But it will be a failure to overlook that the sections are descendant of article, since this may have an impact to the ranking; *i.e.*, if two sections have the same score, but the articles which contain them have different global relevancies, it seems logical to rank the section in the most relevant article before the other.

3 Our Approach

It is clear that we need an expression of the relevance of a document that takes into account the contents of individual elements of a document, and the structure of the document itself. If we suppose that (as in flat document retrieval) we can express the relevance R_x of the textual part of any XML element E_x of the document, the relevance of the document is to be expressed as a function of these values R_x that reflects the structure of the document.

The relevance of the document to a specific request is therefore of the form $R = F_{D-R}(R_1, \dots, R_i, \dots, R_n)$, where the F_{D-R} function is specific to the document and the request itself.

In our very simple example, we could think that it is pertinent to select only documents where both conditions on article and section are satisfied. However, relevance is a strange function, and documents that are not detected as talking about *java* or that contain no section with the words "implementing threads" can be judged as relevant by the user.

Computing relevance is therefore not a matter of just "anding" or "oring" results, but rather a problem of finding a convenient equation with appropriate coefficients.

Starting with a simple topic such as `//article[about(., java)]//sec[about(., implementing threads)]` where Ra_i and $Rs_{i,j}$ are the computed relevancies of article i and each of its sections j , we can say that the relevance of some section is a function of Ra_i and $Rs_{i,j}$. Different models have been proposed in the past, combining Ra_i and $Rs_{i,j}$ with functions such as addition, multiplication, *etc.* We can note that "and" conditions are typically represented by a multiplication or a minimum. If we use these combinations, the generic equation corresponding to our model is:

$$R_i = \alpha * Ra_i + \beta * Rs_{i,j} + \gamma * Ra_i * Rs_{i,j} + \delta * \min(Ra_i, Rs_{i,j})$$

The question is: how should we choose the coefficients?

Fortunately, INEX provides us, not only with queries, but also with assessments to these queries.

The idea presented here is to say that for this specific topic, we can compute Ra and $Rs_{i,j}$, using some well established evaluation method (such as the vector model) for each assessed document, and say that the result R_i is equal to the user estimated relevance of the document¹. In our case, this gives us a set of 2473 equations with 4 unknown quantities: this over-determined system can be solved with mathematical methods (a linear least squares method in our case), giving values for α , β and γ that minimizes the system.

For the chosen model, and the specific query, we discover therefore the most appropriate values to represent the relevance of any unrated new document.

With the query and the assessment table:

article	1	1	1	...	1	2	2	i	...	n
section	1	2	3	...	m_i	1	2	j	...	m_n
user relevance	1/3	0	0	...	1/3	1	0	k	...	2/3

We create the system:

$$\left\{ \begin{array}{l} \alpha * Ra_1 + \beta * Rs_{1,1} + \gamma * Ra_1 * Rs_{1,1} + \delta * \min(Ra_1, Rs_{1,j}) = 1/3 \\ \alpha * Ra_1 + \beta * Rs_{1,2} + \gamma * Ra_1 * Rs_{1,2} + \delta * \min(Ra_1, Rs_{i,j}) = 0 \\ \alpha * Ra_1 + \beta * Rs_{1,3} + \gamma * Ra_1 * Rs_{1,3} + \delta * \min(Ra_1, Rs_{i,j}) = 0 \\ \vdots \\ \alpha * Ra_1 + \beta * Rs_{1,m_1} + \gamma * Ra_1 * Rs_{1,m_1} + \delta * \min(Ra_1, Rs_{1,m_n}) = 1/3 \\ \alpha * Ra_2 + \beta * Rs_{2,1} + \gamma * Ra_2 * Rs_{2,1} + \delta * \min(Ra_2, Rs_{2,1}) = 1 \\ \alpha * Ra_2 + \beta * Rs_{2,2} + \gamma * Ra_2 * Rs_{2,2} + \delta * \min(Ra_2, Rs_{2,2}) = 0 \\ \vdots \\ \alpha * Ra_i + \beta * Rs_{i,j} + \gamma * Ra_i * Rs_{i,j} + \delta * \min(Ra_i, Rs_{i,j}) = k \\ \vdots \\ \alpha * Ra_n + \beta * Rs_{n,m_n} + \gamma * Ra_n * Rs_{n,m_n} + \delta * \min(Ra_n, Rs_{n,m_n}) = 2/3 \end{array} \right.$$

4 Results

We used these three similar topics for testing this method:

- Topic 128: (1623 Equations)

```
//article[about(., intelligent transport systems)]//sec[about(., on-board route planning navigation system for automobiles)]
```

- Topic 141: (2473 Equations)

```
//article[about(., java)]//sec[about(., implementing threads)]
```

- Topic 145: (2687 Equations)

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

¹For most INEX documents, relevancies have not been estimated. We use only documents for which the relevance has been estimated; the corresponding values (0, 1, 2, and 3) are normalized to 0, 1/3, 2/3 and 1.

By solving these three systems we obtain values for the unknown quantities:

Topic	128	141	145
α	0.051	0.014	0.009
β	0.085	0.059	0.123
γ	-0.753	0.769	0.506
δ	0.567	0.265	0.134

Now we can reintroduce these values into each system, compute the score of each answer, order them by growing values and draw the Precision-Recall graphics. The figure 1 shows for each topics the Precision-Recall graphics.

5 Conclusion and Perspective

What conclusions can we draw from these very first experiments? While we have chosen three similar requests, which look like they might be solved by "adding" two conditions, experimental results only partially validate this hypothesis.

However, there are many aspects that impact the results, and which are difficult to take into account.

- Clearly, the relevance assessment made by user is rarely a strict interpretation of the NEXI formulation: a user can also make errors, incorrect judgments², *etc.*
- the function that we use to evaluate the relevance of a passage is quite simple, based on the vector model. It doesn't take into account synonymy or homonymy between words, *etc.*
- the equation system that we obtain is usually ill conditioned, and gives sometimes unstable results.

Many more experiments (including different evaluations functions for textual elements) clearly need to be conducted to be able to draw firm conclusions. However, we believe that the approach may lead to a progress in many directions, including:

- taking into accounts the profile of the user.
- discovering the best usages of structures for XML information retrieval.
- adapting relevance feedback system

More generally, we can expect such an approach to help designing acceptable models for "and" and "or" operations, used in typical requests on structure and contents of XML documents, therefore allowing us to build better information retrieval systems.

²Actually, when two INEX experts evaluate the same set of documents, they usually totally disagree about which are relevant and which are not.

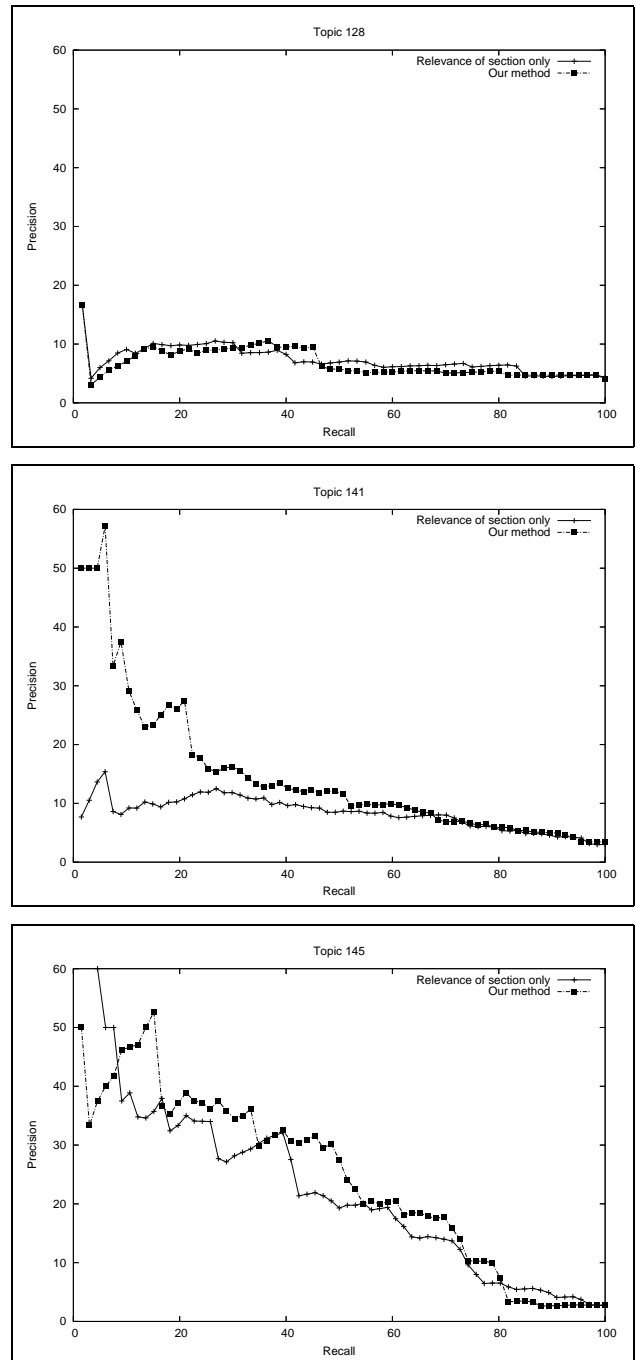


Figure 1. Precision-Recall for each topics

References

- [1] DocBook. <http://www.docbook.org/>.
- [2] Initiative for the evaluation of xml retrieval. <http://inex.is.informatik.uni-duisburg.de/>.
- [3] G. Navarro. *A Language for Queries on Structure and Contents of Textual Databases*. PhD thesis, University of Chile, 1995.
- [4] K. Sauvagnat, M. Boughanem, and C. Chrisment. Searching XML documents using relevance propagation. *SPIRE*, 2004.
- [5] T. Schlieder and H. Meuss. Querying and ranking XML documents. *JASIST*, 53(6):489–503, 2002.
- [6] XML corpus of Shakespeare’s plays. <http://www.ibiblio.org/xml/examples/shakespeare/>.
- [7] TEI Consortium. Text Encoding Initiative, 1987. <http://www.tei-c.org/>.
- [8] World Wide Web Consortium (W3C). Extensible Markup Language (XML), February 1998. <http://http://www.w3.org/XML/>.
- [9] R. Wilkinson. Effective retrieval of structured documents. In *Research and Development in Information Retrieval*, pages 311–317, 1994.