Published in *Proceedings of SPIRE 2000*,
IEEE Computer Society, Los Alamitos, CA,
USA, 2000. ISBN 0-7695-0746-8

DelfosnetX: A Workbench for XML-based Information Retrieval Systems

M. J. Fernández-Iglesias[*], P. Pavón-Mariño, J. Rodríguez-Estévez, L. Anido Rifón, M. Llamas-Nistal
Grupo de Ingeniería de Sistemas Telemáticos. Depto. de Tecnologías de las Comunicaciones.
Universidade de Vigo – Spain
**Visiting the International Computer Science Institute, Berkeley CA, USA*
manolo@ait.uvigo.es

Abstract

In this paper we present DelfosnetX, an Information Retrieval (IR) system intended to evaluate different relevance analysis and ranking techniques for metadata-enabled IR, and more specifically, XML-based IR. The theoretical background that supports the proposed model is also discussed here.

1. Introduction

In this paper we study the introduction of metadata, and more specifically XML, into the IR domain. We try to evaluate how metadata may improve the features of classical IR systems. For this, we propose DelfosnetX, a workbench intended to analyze the properties of metadata-enabled IR systems.

The introduction of XML provided an open environment to define any metadata scheme. This environment supports the description of already available metadata schemes, like DublinCore[5], and to define new schemes as new tag languages to address new scenarios that may appear.

The rest of this paper is organized as follows. First, we offer further insight on our motivation and objectives. Then, a brief presentation of the state of the art in the field of XML-enabled search engines and IR systems is provided. Section 4 is devoted to briefly discuss some aspects of classical IR theory, which will help to state some notation and concepts. Sections 5 and 6 present the theoretical foundation of DelfosnetX. In section 7 we present DelfosnetX, this section being the core of this paper. At the end of the paper we present some conclusions and discuss briefly our present and future work.

2. Motivation and objectives

Metadata is used to describe the information provided by a document. Therefore, metadata can be used to make explicit higher level information not directly present in the document itself.

IR systems (e.g. search engines) may be provided with this higher level, structured information about document contents. Users may query the system not only about document contents, but also about these higher level descriptions.

Therefore, metadata may improve the features of IR systems. In this paper we propose a theoretical framework to support the design, implementation and evaluation of metadata-enabled IR systems. As stated in the introduction, XML is a suitable model for metadata description, and will be adopted here as the metadata reference scheme. A tool that implements our approach is the second contribution of this paper.

The proposed framework is not domain specific, but valid for different application domains. In fact, we will define a family of models (i.e. a metamodel?) that can be instantiated to be applied to a given, specific application domain. On the other side, we will make use of already available developments from classical IR theory.

3. Current Trends in XML-Enabled Systems

Already available XML-enabled IR systems can be classified into two groups: those that support inter-document searches/queries, and those that support intra-document searches/queries using an XML query language like XQL[7], XML-QL[3] or Lorel[1].

The model presented in this paper can be classified into the first group. Query results will be composed by relevant XML documents, and not by fragments of them. Searches are performed over all the database documents. DelfosnetX will also allow the definition (and therefore testing) of query languages under specific restrictions, as discussed in section 7.4.

The systems presented below are examples of these two approaches:

- XRS[11] is a XML-enabled search engine based on BUS[7] (Bottom Up Scheme). XRS returns elements extracted from XML documents that fulfill a set of requirements in a user query.
- XSet[12] is oriented towards intradocument information retrieval. Queries are defined as XML documents whose tags reflect the corresponding query parameters.

The systems above differ in the way information is internally organized to handle structured XML files and to efficiently support queries. In XRS, information management is based on traditional database technologies, and a set of inverted files to support searches. XSet is based on a set of hash tables that reflect the structured organization of XML documents.

4. Classical information retrieval

In our context, classical IR models are those which do not support metadata information. Classical IR is a well established field and many different approaches have been discussed along the years: vector models, models based on fuzzy sets, models based on neural networks are just some examples[8][4][2]. Indeed, this scenery is further enriched with many different variations and combinations of models.

The following concepts and notation from classical IR will be used to describe our proposal:

- Queries and documents are represented as *I*-dimensional vectors, where *I* is the size of the dictionary (i.e. the number of distinct terms stored in the IR system).
- A query is then represented as $\vec{q} = (n_{1q}, \ldots, n_{Iq})$, and document *j* is represented as $\vec{d}_j = (n_{1j}, \ldots, n_{Ij})$. Coordinates n_{ij} (n_{iq}) represent the number of times term *i* appears inside document *j* (query *q*). Note that information about the relative position of terms inside a document is *not* considered. The document database in a classical IR system containing *J* documents can be represented by the n_{ij} above, that is $B_{classic} = \{n_{ij}, I=1\ldots I, j=1\ldots J\}$.
- Queries trigger the calculation of a similarity function $sim(\vec{q}, \vec{d}_j)$ that tries to estimate the similarity between the query and the documents in the database. For the calculation of the similarity function, additional information is used besides $\vec{q}$ and $\vec{d}_j$. This information is obtained from a set of statistic functions whose domain is $B_{classic}$ and return a real number. We classify these functions into three groups:
  - *G:* They are global statistics. For example, *N* (size of the database) is a *G* statistic.
  - *T:* They are computed for single terms in the database. For example, $idf_i = \log(N/n_i)$ describes the discrimination power of term *i* (n_i is the number of times term *i* appears in the collection).
  - *D:* They are computed for single documents. For example, u_j is a *D* statistic that represents the number of distinct terms in document *j*.

5. Matrix model

Let B_{ext} be an IR database containing *J* XML documents. Documents in B_{ext} are composed by a hierarchy of nodes, where each node has a tag and a content. This structure results from parsing an XML document (DOM[6] tree).

Database B_{ext} is modeled *as* $B_{ext}=\{n_{ijm}, i=1\ldots I, j=1\ldots J, m=1\ldots M\}$, where *I* is the number of distinct terms in the database and *M* is the number of distinct tags in the XML scheme, and n_{ijm} represents the number of times that term *i* appears in document *j* bound to tag *m*.

From the classical point of view, this model is based on an *IxM*-dimensional vector space, where both queries and documents are *IxM* matrices (i.e. elements in the above vector space). For a given document [d_j], each row is a vector $\vec{d}_j^{\,m}$ that represents the contents of the corresponding document bound to tag *m*. The same applies for a user query:

$$[d_j]^T = (\vec{d}_j^{\,1}, \ldots, \vec{d}_j^{\,M})\,;\ [q]^T = (\vec{q}^{\,1}, \ldots, \vec{q}^{\,M})$$

where $\vec{q}^{\,m} = (q_1^m, \ldots, q_I^m)$, and the q_i^m represent how much the user is interested in documents containing term *i* bound to tag *m*.

We can also define two different dictionaries from B_{ext}: the term dictionary D_t composed by the set of indexed terms in the IR database, and the tag dictionary D_l containing the tags from XML document schemes. The cardinality of D_t (resp. D_l) is *I* (resp. *M*).

Document 1 (d_1)

```
<List>
<Title>
   things to do
</Title>
<Item>read</Item>
<Item>write</Item>
<Item>read</Item>
</List>
```

Document 2 (d_2)

```
<List>
<Item><Abstract>
   do
</Abstract></Item>
<Item>write</Item>
</List>
```

Figure 1. Example documents

In figure 1 we show two XML documents belonging to a simple IR database. Some examples about the information managed for this database are presented below:

- Dictionaries:
  - D_t = {things, to, do, read, write}; I =5
  - D_l = {<List><Title>, <List><Item>, <List><Title><Abstract>}; M=3
- Example queries
  - q_a: Retrieve all documents containing “read” at tag “<List><Item>”.
  - q_b: Retrieve all documents containing “write” and “do” in any tag under tag “<List><Item>.
- Matrices
  - $[d_1]^T$ = [(1 1 1 0 0), (0 0 0 2 1), (0 0 0 0 0)]
  - $[d_2]^T$ = [(0 0 0 0 0), (0 0 0 0 1), (0 0 1 0 0)]
  - $[q_a]^T$ = [(0 0 0 0 0), (0 0 0 1 0), (0 0 0 0 0)]
  - $[q_b]^T$ = [(0 0 0 0 0), (0 0 1 0 1), (0 0 1 0 1)]

There is a class of queries that are common in XML-enabled search engines due to the hierarchy of tag definitions, namely tree queries. These queries ask for terms bound to a given tag and all tags below it. The query $[q_b]$ above is an example of this class of queries, which can be straightforwardly handled by the proposed model.

Classical vector-based similarity functions is now calculated from document and query matrices. Now, similarity functions will measure the distance between a document matrix and a query matrix in the corresponding *IxM*-vector space. Retrieved documents are ranked according to the corresponding results.

The statistics available to compute similarities can now be enriched with new ones that take into account metadata information:

- *M*: They are computed for single tags. For example, N_m is an M statistic representing the number of documents having tag m.
- *DM*: They are computed for single tags in a given document. For example, n_{jm} represents the number of terms bound to tag m in document j.
- *TM*: They are computed for single tags for a given term. For example, $idf_{im} = \log(N_m / n_{im})$ represents the discrimination power of term i for contents bound to tag m.

Obviously, other combinations are possible. As far as we know, the evaluation of available metadata-dependent statistics is an open problem.

6. Relevance analysis in a matrix model

The classical similarity function, i.e. the basis for relevance analysis, is a function on vectors. As stated above, the matrix model defines a similarity function on matrices. In this section we will try to link available results for classical models to the new proposed model. First, we will define a tool that will help to establish this link: the projection of a matrix model into a classical one.

6.1. Projections

Let B_{ext} be an extended IR system containing J documents $[d_j]$, $j=1\ldots J$, M tags and I different terms. Let $\vec{P} = (p_1, \ldots, p_M)$ be an M-dimensional projector vector. We define the P-projection of B_{ext} as a classical (i.e. metadata free) database $B_{classic}$ = { $\vec{d}'_j$, j=1…J} composed by J documents $\vec{d}'_j = \vec{P}[d_j] = (d'_1, \ldots d'_M)$, where $d'_i = \sum_{m=1}^{M} p_m n_{mi}$.

Values in $\vec{P}$ weight the relevance that contents bound to tag m will have in the new classical database. The d_i' in the projected document include information about the relative importance of term i depending on the tag it was bound to, as defined by the p_m.

This projection may be also applied to queries. This way, existing classical relevance analysis methods can be straightforwardly applied to a metadata-extended framework.

6.2. Some projection examples

Example 1 A projection vector $\vec{P} = (1\ldots1)$ generates classical databases where all metadata information is lost. Let us apply the projector $\vec{P} = (111)$ to the database in section 5 (see also figure 1). Then, all text bound to any XML tag is equally relevant, and as a consequence XML specific information will not be considered for relevance calculation.

Let $sim(\vec{d}_{j,} \vec{q}) = \cos(\angle \vec{w}_j \vec{w}_q)$, where $\vec{w}_q = \vec{q}$, and $w_{ij} = n_{ij} / n_i$, and n_i is the number of documents where term i appears (a T statistic, cf. Section 4).

For query q_b we have $\vec{d}_{1-proj} = (11121)$; $\vec{d}_{2-proj} = (00101)$; $\vec{q}_{proj} = (00202)$, and N_{things} = N_{to} = N_{read} = 1, N_{do} = N_{write} = 2, and therefore $\vec{n}_i = (11212)$. Then, $\vec{w}_1 = (1\,1\,0.5\,2\,0.5)$, $\vec{w}_2 = (0\,0\,0.5\,0\,0.5)$, and the corresponding values for the similarity function are $sim(\vec{d}_{1,} \vec{q}_b) = 0.277$ and $sim(\vec{d}_{2,} \vec{q}_b) = 1$

We conclude that d_2 is more relevant to the query q_b. Note that all metadata information was lost, and d_2 is composed solely by the terms in the query.

Example 2 A canonical vector $\vec{1}_m$ where $p_i = 0\ \forall i \neq m,\ p_m = 1$ generates classical databases whose documents contain only the information bound to tag *m*.

For this example we generate *M* classical databases from the example database in section 5, one for each XML tag, using the projectors above. Then, *M* similarity results can be obtained for a given query.

We have to select a procedure to combine these *M* values into a single one for ranking purposes. For this example, the procedure selected is based on the similarity estimation for the extended boolean model. We will assume that the query string is composed by a set of subqueries, each bound to an XML tag, combined by boolean operators.

Let us assume that XML query substrings for each tag are *or*-ed. Then, $sim_{bool-ext} = \frac{1}{M'}\sqrt{\sum_{m=1}^{M'} sim_m^2}$, where *M'* is the number of non-null similarities. For query q_b we obtain $sim(\vec{d}_{1,}\vec{q}_b) = 0.171$ and $sim(\vec{d}_{2,}\vec{q}_b) = 0.5$. If we compare these results with those from the previous example, we see that the relevance of both documents to the query q_b decreases. This is due to the role played by the metadata. We see that the term “do” is not bound to tag “<List><Item>” in the first document. For the second document we see that, although both query terms are relevant to the query, they are bound to different tags.

7. DelfosnetX: the system

DelfosnetX is an information retrieval system that handles XML documents. Documents managed by DelfosnetX may be defined according to any scheme (DTD file), and different DTDs can be handled simultaneously. The search engine was developed according to the framework described in this paper.

The first aim of DelfosnetX was to provide a workbench to validate our proposal, so it was designed to easily test the performance of different configurations for the matrix model. The system automatically fetches and (re)calculates a comprehensive set of statistics to be used to compute and test different similarity functions and relevance analysis methods. This approach also permits to study the relative performance of classical and metadata-oriented IR systems.

DelfosnetX is a Java-based system that can be accessed through the Web using a standard web browser. An Application Programmer Interface (API) is also provided to easily customize DelfosnetX for particular applications.

At this point we will offer some insight into the architecture selected to support the model discussed in previous sections. First, we will present the components of the system and how they interact. Then, we will justify the relational model selected to support the data structures needed to implement the target IR system. We will also briefly discuss the main features of DelfosnetX, and present the dynamics of a typical user query.

7.1. System Architecture

DelfosnetX is based on a well known paradigm for distributed computing: a three-tier architecture (see figure 2). We divided all the software involved into three separated tiers. At the front-end a thin client was selected to allow efficient access through a networked environment. The middle tier, business logic tier, is responsible for actually implementing the functionality of the whole system by managing the data stored at the back-end tier. This model has been selected because it permits to separate network issues related to remote system access from database access and management, simplifying system maintenance. Whatever change in any of the three tiers can be carried out independently as clear interfaces are provided between the elements involved. The three main components of DelfosnetX are:

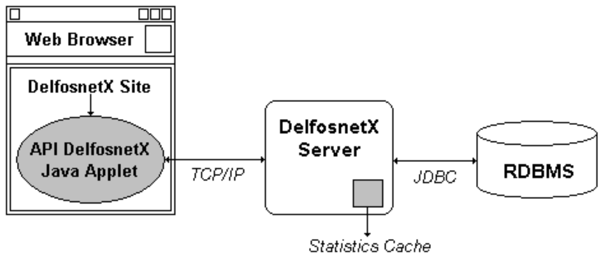

Figure 2. DelfosnetX architecure

- **DelfosnetX Client** Access to the system (user queries, maintenance, user management) is provided by an applet that implements the DelfosnetX Application Programmer Interface (API). Its methods may be invoked by JavaScript or Java code running at the client. A standard WWW browser provides the adequate Graphical User Interface (GUI) for any particular application (see section 8 below).
- **DelfosnetX Server** It implements the DelfosnetX API at the server side and takes care of network connections and user authentication. It also provides basic IR features: relevance analysis, result ranking, etc. It is the only agent that interacts with the database system. This also improves system security and hides implementation details related to low-level data management, preventing direct access from clients to the database. To sum up, it converts a relational database management system (RDBMS) into an IR system. It is Java-based application responsible for implementing the business logic for the whole DelfosnetX system.

- **Relational Data Base Management System** All data needed to implement the functionality of DelfosnetX is managed by the RDBMS. It maintains all the information needed to process user queries, manage documents, users and permits.

7.2. The relational model

Among the available solutions, we have selected the relational model to support the data structures that serve as the foundation of DelfosnetX. In other words, we have constructed an IR system on top of a RDBMS.

As stated before, XML has been selected as the language to define metadata. XML documents follow a simple model: they are composed by a set of tag-term(s) pairs. The system handles XML documents according to this structure, that is, documents are parsed to extract tags and terms bound to them.

In figure 3 we present an outline of the RDB organization. We see that there are seven relations directly related to terms and tags. These relations are summarized in table 1.

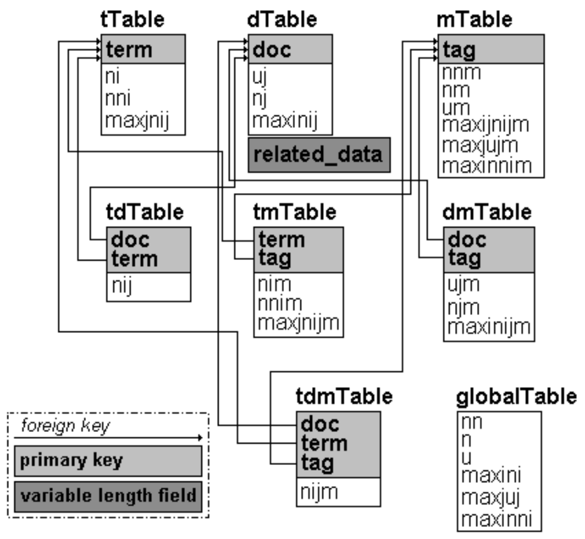

Figure 3. Database organization

Despite of terms being bound to a tag, DelfosnetX has been designed to support queries both for terms interpreted as *free* (i.e. not bound to any tag) and as terms bound to a tag. Information stored in tables T, D and TD allow relevance calculation without any tag influence (as if plain documents were indexed). In this way, classical and matrix similarities can be tested in DelfosnetX.

Table 1. The relational model

Rel.	Record info	Record field example
T	Terms	*nni*: # of docs. containing term i
M	Tags	*nm*: # of terms bound to tag m
D	Documents	*uj*: # of unique terms in doc. j
TM	Tag-term pairs	*nim*: # of times i is bound to m
TD	Term-doc pairs	*nij*: # of apparitions of i in j
TDM	Term-doc-tag triplets	*nijm*: # of times I appears binded to m in j.

In figure 3 also appears a relation not described in table 1. This relation stores global statistics. For example, entry *nn* stores the size (number of documents) of the database.

All the relations described above are indexed to minimize response time. A thorough analysis, supported by well established database theory, has been performed to select and adequate indexing scheme. The most convenient index set is described in table 2.

Table 2. Index set for the relational model

Relation	Indexing field(s)
T	term
M	label
D	document
TM	term and tag
DM	tag and document
TD	term and (tag,document)
TDM	(term,tag) and (term,tag,document)

Note that RDBMS indexes are, from an implementation point of view, inverted files. Inverted files are a classical approach in the IR world to support document retrieval. In other words, we rely on the indexing scheme provided by the RDBMS, which in turn is implemented as an inverted file set, to support queries. This approach speeded up system development and permitted us to devote most of our efforts to issues related to retrieval and relevance analysis.

Furthermore, the relational approach offers a great flexibility to define and maintain statistics to support relevance analysis. Note that one of our main objectives was to develop a platform to study several approaches to relevance calculation/analysis for multimedia documents. The availability of a set of structured statistics will permit us to easily define customary relevance approaches maintaining fairly reasonable response times.

7.3. Data and metadata

Metadata enabled systems can follow two parallel approaches.

- Traditional approaches for IR systems rely on document contents for retrieval and ranking. In this case the XML document (used for indexing and

relevance calculation), is the only item the user wants to retrieve.
- On the other side, XML can be used to store additional information available about document contents, which does not need to be made explicit in the documents themselves. That is, documents are retrieved according to data about the data, i.e. according to available metadata.

For example, an image database will have, for each stored image, an attached XML metadocument reflecting all the information relevant to the final user (e.g. creator, content description, format, image digest, watermark info, location, ...). Image retrieval is not based directly on document content, but on available information about document contents.

DelfosnetX allows both kinds of approaches. Every indexed XML can have associated information (i.e. image), that is stored in *related_data* variable length field. Each system built on top of DelfosnetX can define the contents it associates to its documents (i.e. a serialized Java object with the image plus any particular information). We feel that this design concept gives flexibility and strength to the system.

7.4. DelfosnetX functionality

The present version of DelfosnetX addresses definition and testing on the following key aspects of IR systems:

- Similarity functions to estimate the relevance of a given document with respect to a user query.
- Stoplists for document and/or query filtering.
- Stemming functions for document and/or query filtering.

The DelfosnetX API enables the user to perform the actions summarized below:
- Register/unregister a custom-defined matrix similarity function. A Java `.class` file implementing a concrete interface should be provided (see section 5).
- Register/unregister a stemming function. A Java `.class` file implementing a concrete interface should be provided.
- Register/unregister stoplists. A text file with the list's stopwords should be provided.
- Upload an XML file into the database. A registered stoplist and a registered stemmer may be selected to be applied to the document. The system will parse the document, apply the stoplist, apply the stemmer and update the index set. Actually, this action does not immediately update the database, but schedules this action to be performed during the next offline actualization.
- Delete a document from the databbase. As in the previous case document deletion is not performed online.
- Perform a query according to the matrix model. Documents are retrieved and classified according to their relevance. The selected stoplist and stemmer will be applied to the query, and relevance analysis will be based on the selected similarity function. The system filters the query using the selected stoplist, and generates a set of potentially relevant documents ranked according the number of times query terms appear in each document. This set may be optionally truncated to the *n* potentially more relevant documents if desired, *n* being specified by the user. This preliminary selection may improve response time for big collections. The target similarity function will be applied only to this truncated set of documents.
- Calculate a precision-recall set of points according to the standard process presented in [2]. A query and *a priory* relevant document set are passed as parameters.
- Offer direct (read) access to the statistics stored in the database.
- Many available test collections offer several *a priori* relevant document sets for a given query[9]. Each set generates distinct precision-recall points. The system permits to specify several relevant document sets to calculate a set of precision-recall sets. This feature speeds up this kind of tests.
- All actions related to system maintenance. A matrix of permits can be defined, to grant or deny access for every user and API function.

As previously stated, DelfosnetX is based on the matrix model presented in section 5. This model is a generalization of classical vector models. As a consequence, DelfosnetX may also be used as a benchmark for this kind of systems.

Next, we will present two examples to illustrate basic DelfosnetX operation.

Adding documents To add an XML document to the database, the user invoke the following method from the API:

```
public FloatMatrix addDocument(URL url,
     Byte[] relatedData, Locale language,
     String idStoplist, String idStemmer)
```

where `url` identifies a link to the XML document to be indexed; `relatedData` points to actual data attached to the XML document, if any; `language` identifies the unicode set for the document; and `idStoplist` and `idStemmer` are identifiers for a registered stoplist and stemming function.

Similarity calculation Some commands (e.g. queries, precision-recall matrix calculation) require the calculation of similarity values. Registered similarity functions are implemented as Java classes according to a predefined interface. Then, DelfosnetX invokes the *calculate* method from the appropriate class to get the needed similarity values:

```
float calculate (FloatMatrix queryMatrix,
     String doc, Object params, DBMatrix db,
     Vector rankingDocs)
```

`queryMatrix` stores the user's query; `doc` points to the target document; and `params` is a generic object defined by the user that the system passes to the similarity function.

`db` is an object whose methods provide statistics for similarity calculation. For example, `db.idfi("calcium")` returns IDF_i for the term "calcium". The method `generateProjection(Vector tags)` returns a `DBMatrix` object where only the selected tags are taken into account for statistic generation, as discussed in section 6.1.

`rankingDocs` stores other documents that will be ranked in this query. Probabilistic-like similarity functions may user this information.

8. Some results and ongoing work

Preliminary tests have been performed based on the cystic fibrosis reference collection provided by TREC[9]. This database includes 1239 XML documents related to cystic fibrosis. It also provides 99 natural language queries together with *a priori* results for four different user groups (see table 3).

Table 3. Cystic fibrosis refcol. User groups

Group	Description
Score 1	Relevant docs. for authors
Score 2	Relevant docs. for other physicists in the field
Score 3	Relevant docs. for post-doc researchers
Score 4	Relevant docs. for other medical bibliographist

DelfosnetX is queried for all the 99 queries to obtain the precision-recall matrix in every case. Each query is performed three times using three different similarity functions:

- Classical Salton and Buckley, discarding all metadata information.
- Matrix similarity based on the extended boolean method applied to Salton and Buckley, where the corresponding projections are *and*-ed (see section 5).
- Matrix similarity based on the extended boolean method applied to Salton and Buckley, where the corresponding projections are *or*-ed.

For the matrix similarity case, tags where weighted as reflected in table 4. Note that these values affect the final performance. DelfosnetX may be used to tune up these weights to get better results.

Table 4. Tag weights for cystic fibrosis refcol.

Tag	Weight
Majorsubj Topic	7
Minorsubj Topic	6
Title	8
Abstract	3
Other tags	1

We obtained 3x4x99 precision-recall point sets. To analyze these results, we calculated for each set the corresponding average precision as the arithmetic mean of the precision values. This statistic favors those systems that rank higher the most *a priori* relevant documents.

Results are summarized in figures 4 to 7. Each graphic corresponds to one of the scores in table 3. They display query number vs. average precision for the three similarity functions defined, queries being sorted in ascending order by average precision for the classical case (no metadata information).

We can extract the following conclusions:

- For each query, the performance for both extended boolean approaches (*and*, *or*) are similar, although results are slightly better for the *or* case.
- There are remarkable differences between metadata and metadata-free queries. Depending on the query, we can get clearly better or worse results introducing metadata.
- On average, global results are comparable for the metadata and metadata-free cases.

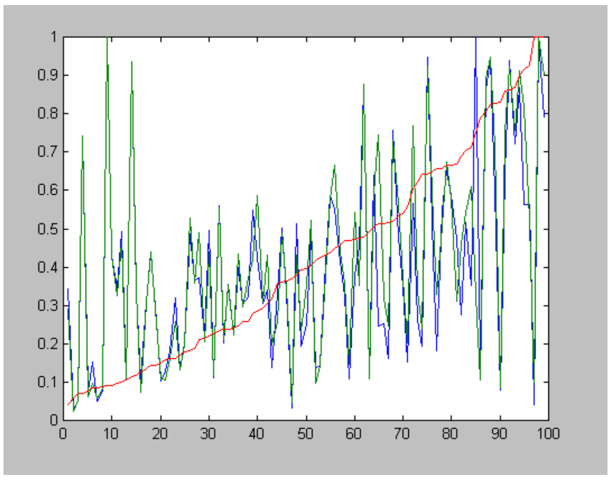

Figure 4. Results for score 1

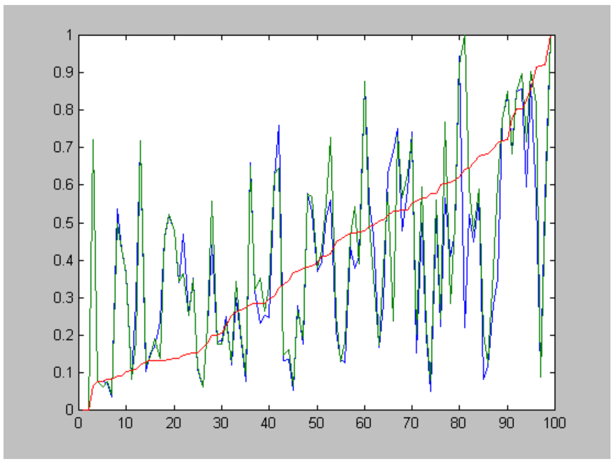

Figure 5. Results for score 2

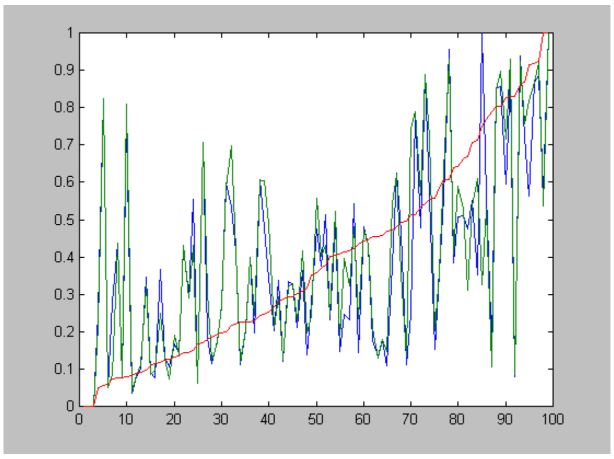

Figure 6. Results for score 3

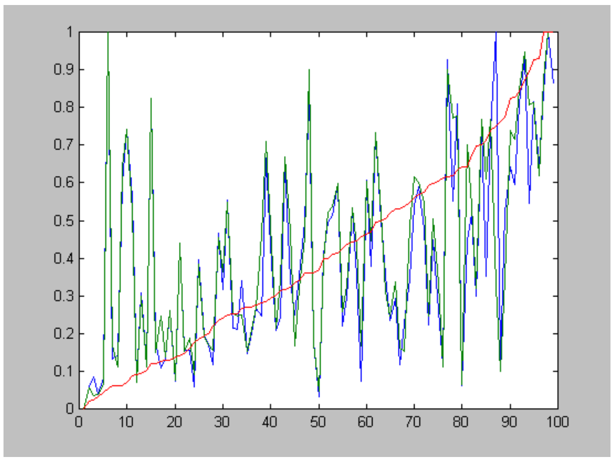

Figure 7. Results for score 4

Note that this test is only an example of the kind of analysis that can be performed using DelfosnetX. This system can be easily tuned up to study a broad spectrum of relevance analysis methods based on the matrix model presented above. Indeed, DelfosnetX can be easily tailored to particular applications where there exists enough understanding of the underlying domain as to define a well-suited relevance/ranking strategy.

Currently, we are tuning up the following DelfosnetX based systems:

- A search engine for non-text documents (audio, video, images, etc.) An XML file is bound to each piece of multimedia information. This file identifies the location of the multimedia file an provides textual (structured) information about it. We are testing several configurations of the matrix model (e.g. projections, query method, etc.) to find the most suitable ones for this application.
- An online Internet quality-of-service analysis tool. Information about performance and quality of service for monitored Internet sites or documents is kept as structured XML data (roundtrip times, hop counts, packet sizes, delays, connection establishment parameters, etc.). Our aim is to tune this tool to help final users to select the best location (i.e. the one that likely will guarantee the best quality of service) to download a given document or to analyze different routes to a given service. Users would query for a document, and the system will respond with a ranked list of locations based on the computed estimation of the quality of service.

9. Concluding remarks

The need for efficient information retrieval and management tools for the Web and the apparition of advanced markup and metadata methodologies determined the evolution of IR techniques to take into account metadata information. As a consequence, research is necessary to study the real contribution of metadata to the performance of IR systems. A suitable theoretical framework to formally characterize the different aspects of this may be helpful.

In this paper we have presented a matrix-based characterization for metadata-based IR where both documents and user queries are modeled as matrices. These proposals adapts well to XML metadata and easily integrates previous results from classical IR.

DelfosnetX was conceived as an Internet-oriented workbench to study and test the properties of our proposal. This system is in an advanced implementation phase, and some promising results have already been obtained. Nevertheless, more results are needed, to be tested against other results in the Academia, to be able to answer the question: how should metadata be handled to take all relevant information from available data?

10. Acknowledgments

This work was funded by the European Community and the Spanish Ministry of Education under grant 1FD97 0282.

References

[1] S. Abiteboul, D. Quass, J. McHugh, J. Windom, and J. Wiener, “The Lorel query language for semistructured data”, *International Journal on Digital Libraries*, April, 1997.

[2] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*, Addison-Wesley, 1999.

[3] A. Deutsch, M. Fernández, D. Florescu, A. Levy, and D. Suciu, *XML-QL: A query language for XML*, Technical Report, W3C, August, 1998, http://www.w3.org/TR/1998/NOTE-xml-ql-19980819.

[4] W.F. Frakes and R. Baeza-Yates, *Information Retrieval. Data Structures and Algorithms*, Prentice-Hall, 1992.

[5] DublinCore Working Group, *DublinCore Metadata for Resource Discovery*, Technical Report, DubliCore, 1998, ftp://ftp.isi.edu/in-notes/rfc2413.txt

[6] W3C DOM Working Group, *Document Object Model*, Technical Report, W3C, December, 1998, http://www.w3.org/DOM

[7] J. Lapp, J. Robie and D. Schach, “XML Query Language (XQL)”, *QL-98-The Query Languages Workshop*, W3C, December 1998, http://www.w3.org/TandS/QL/QL98/pp/xql.html

[8] D. Shin, “BUS: An effective indexing and retrieval scheme in structured documents”, *Procs. of Digital Libraries 98*, 1998

[9] A. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*, McGraw-Hill, 1983

[10] W. M. Shaw, J. B. Wood, R. E. Wood, and H. R. Tibbo, “The cystic fibrosis database: content and research opportunities”, *Library and Information Science Research*, 13, 1991, pp. 247-366.

[11] XRS: http://dlb2.nml.nih.gov/~dwshin/xrs.html.

[12] XSet: http://www.cs.berkeley.edu/~ravenben/xset.