Modeling Metadata-Enabled Information Retrieval

Manuel J. Fernández-Iglesias, Judith S. Rodríguez, Luis Anido, Juan Santos, Manuel Caeiro, and Martin Llamas

Grupo de Ingeniería de Sistemas Telemáticos.
Departamento de Ingeniería Telemática
Universidade de Vigo, Spain
`manolo@det.uvigo.es`,
WWW:`http://www-gist.det.uvigo.es/`

Abstract. We introduce a proposal to theoretically characterize Information Retrieval (IR) supporting metadata. The proposed model has its foundation in a classical approach to IR, namely vector models. These models are simple and implementations are fast, their term-weighting approach improve retrieval performance, allow partial matching, and support document ranking. The proposed characterization includes document and query representations, support for typical IR-related activities like stemming, stoplist application or dictionary transformations, and a framework for similarity calculation and document ranking. The classical vector model is integrated as a particular case in the new proposal.

1 Introduction

Metadata are traditionally defined as *data about data*, *information about information.* Metadata facilitate querying over documents by providing semantic tags. Interpreted this way, they could be introduced in the IR domain to improve the relevance of the documents retrieved as a response to a user query. IR systems may be provided with this higher level, structured information about document contents for users to query the system not only about existing document data, but also about these higher level descriptions. This poses the question of how to utilize the provided semantic tags.

Besides, due to the Web's inherent redundancy, lack of structure and the high percentage of unstable data, IR systems for the Web (e.g. search engines) is and will be a hot research topic in the next years. Although some promising results are already available [3] [4], as typically happens in other emerging fields metadata-based IR is far from being stable.

In this line, we feel that a theoretical characterization of metadata-enabled IR will contribute to a solid foundation of the field, which will facilitate the development of tools and methods to analyze existing systems and new proposals, and eventually improve the performance and features of IR systems. This characterization should be abstract enough to be fully independent from metadata models.

In this paper we propose a framework to represent metadata-enabled Information Retrieval that supports relevance calculation. Our proposal is based on the classical vector model for information retrieval [7]. Despite its simplicity, it is generally accepted that this model is either superior or at least as good as known alternatives (c.f., for example, [2], pages 30 & 34).

2 Basic Concepts and Notation

$\mathbb{N}$ and $\mathbb{R}$ represent respectively the set of natural numbers and the set of real numbers. A matrix over $\mathbb{R}$ is a rectangular array of real numbers. These real numbers are named *matrix components*. The symbol $\mathcal{M}_{m\times n}(\mathbb{R})$ denotes the collection of all $m \times n$ matrices over $\mathbb{R}$. The symbol $\mathcal{M}(\mathbb{R})$ denotes the collection of all matrices over $\mathbb{R}$.

Matrices are denoted by capital letters, and $A = [a_{ij}]$ means that the element in the i-th row and j-th column of the matrix A equals a_{ij}. Operations on matrices and particular types of matrices are defined and denoted as usual (see for example [8]). $A^T = [a_{ji}]$ represents the transpose of matrix A. This notation is extended to vectors (i.e. row vector $\boldsymbol{x^T}$ is the transpose of $\boldsymbol{x}$).

We define the *unit vector* $\boldsymbol{u_j^n} = (u_i) \in \mathbb{R}^n$ as an n-dimensional vector such that

$$u_i = \begin{cases} 1 \; j = i \\ 0 \; j \neq i \; 1 \leq i \leq n \end{cases}$$

We define $\boldsymbol{u^n} = \sum_{i=1}^{n} \boldsymbol{u_i^n}$. We omit the superscript n if the dimension of these vectors is clear from the context.

Matrices are associated with particular types of functions called linear transformations. We associate to $M \in \mathcal{M}_{m\times n}(\mathbb{R})$ the function $T_M : \mathbb{R}^n \rightarrow \mathbb{R}^m$ defined by $T_M(\boldsymbol{x}) = M\boldsymbol{x}$ for all $\boldsymbol{x} \in \mathbb{R}^n$.

A subset $V \subset \mathbb{R}^n$ is called a *subspace of* $\mathbb{R}^n$ if $0 \in V$, and V is closed under vector addition and scalar multiplication. Vectors $\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}$ belonging to a subspace $V \subset \mathbb{R}^n$ are said to form a basis of V if every vector in V is a linear combination of the $\boldsymbol{x_i}$ and the $\boldsymbol{x_i}$ are linearly independent. Every vector in V is expressible as a linear combination of the $\boldsymbol{x_i}$. Vectors $\boldsymbol{u_i^n}$, $i = 1, \ldots n$ form a basis for $\mathbb{R}^n$. We name this basis the *canonical basis* of $\mathbb{R}^n$.

3 The Classical Vector Model for Information Retrieval

Basically, an Information Retrieval model (IR model) defines a document and query representation and formalizes users' information needs, that is, it states how documents should be stored and managed, and which information about documents should be kept to efficiently fetch the most relevant documents as a response to a user query. An IR model also defines how and from which information the relevance of a given document should be estimated. In our context, classical IR models are those that do not support metadata information.

Definition 1 (IR model). *Given a collection of documents, an Information Retrieval model is a quadruple* $\mathcal{IR} = (D, Q, \mathcal{F}, R(q_i, d_j))$ *where*

1. *D is the set of* document representations*.*
2. *Q is the set of* queries *(i.e. representations of users' information needs).*
3. *$\mathcal{F}$ is a framework for modeling document representations, queries and their relationships.*
4. *$R(q_i, d_j)$ is a ranking function that returns a real number for every query $q_i \in Q$ and document representation $d_j \in D$. This function defines an ordering among the documents with respect to the query q_i.*

We have chosen the vector model as our foundation because it is simple and implementations are fast, its term-weighting approach improves retrieval performance, allows partial matching, and permits document ranking. The vector model assigns positive real numbers (*weights*) to index terms in queries and documents. Weights will be eventually used to compute the degree of similarity between documents stored in the IR system and user queries. By sorting the retrieved documents according to their similarity, the vector model [2] supports partial (query) matching.

Definition 2 (Vector Model). *Let t be the number of index terms in the system and k_i a generic index term. Let d_j be a generic document, and q a user query. For the vector model, we associate a positive real number $w_{i,j}$ to each pair (k_i, d_j) called* weight of term k_i in document d_j*. We also associate a positive real number $w_{i,q}$ to each pair (k_i, q) called* weight of term k_i in query q*.*

The document vector for d_j is defined as $\boldsymbol{d_j^T} = (w_{1,j}, \ldots w_{t,j})$, and the query vector for q as $\boldsymbol{q^T} = (w_{1,q}, \ldots w_{t,q})$

Queries trigger the calculation of a similarity function $sim(\boldsymbol{q}, \boldsymbol{d_j})$ that tries to estimate the degree of similarity between the query and the documents in the collection. This similarity is estimated from a measure of the distance between the two vectors, and is typically based on the scalar product (e.g. the cosine of the angle between $\boldsymbol{d_j}$ and $\boldsymbol{q}$). The results from the query, a set of relevant documents, is ranked according to their similarity and presented to the user.

The $w_{i,j}$ ($w_{i,q}$) are defined from a set of statistic functions whose domain is $\mathcal{IR}$ and return a real number. We can use statistics computed for single terms in the database (e.g. $idf_i = \log(N/n_i)$, where N is the number of documents in the system, and n_i the number of documents where term i appears, describes the discrimination power of term i), computed por single documents (e.g. the number of distinct terms in document j) or global statistics (e.g. N).

The best known term-weighting schemes for the vector model (*tf-idf*) are given by (variations of):

$$w_{i,j} = \frac{n_{ij}}{max_l\{n_{lj}\}} \times idf_i$$

where n_{ij} is the frequency of index term k_i in document d_j[2].

4 The Metadata Matrix Model for Information Retrieval

We have to define a representation for documents and queries, a framework to define the relationships among documents, queries and representations, and a ranking function. We start from a collection of t metadata documents. In our context, metadata documents are documents composed by a set of nodes, where each node has a tag and a content.

Definition 3 (Metadata document). *Let T and M be two sets known respectively as* term dictionary *and* tag dictionary*. A metadata document is composed by a set of pairs (m, B_m), where m is a tag from M and B_m a list of terms from T.*

In the definition above, B_m is the list of terms bound to tag m. This structure results, for example, from the parsing of an XML document (DOM tree [6]). However, different representations may be produced from a single XML document, depending on the model selected to label nodes or the value equality chosen.

```
Document 1                        Document 2
<List>                            <List>
 <Title>things to do </Title>      <Item><Abstract>do</Abstract></Item>
 <Item>read</Item>                 <Item>write</Item>
 <Item>write</Item>               </List>
 <Item>read</Item>
</List>

(List.Title, things to do)        (List.Item.Abstract, do)
(List.Item, read write read)      (List.Item, write)
```

Fig. 1. Example metadata documents

Example 1. In figure 1 appear two XML documents belonging to a simple collection. In a given application they may be converted into the metadata documents appearing below the corresponding XML documents in the figure. In this example, tags correspond to *Label Path Expressions* [1].

4.1 Metadata Documents in a Matrix Model

In our proposal, documents and queries will be represented as matrices:

Definition 4 (Metadata Matrix Model). *Let t be the size of the term dictionary and k_i a generic index term. Let m be the size of the tag dictionary and l_j a generic tag. Let d_k be a generic document, and q a user query. For the matrix model, we associate a positive real number $w_{i,j,k}$ to each triplet (k_i, l_j, d_k)*

called weight of term k_i in document d_k when bound to tag l_j. *We also associate a positive real number* $w_{i,j,q}$ *to each triplet* (k_i, l_j, q) *called* weight of term k_i in query q when bound to tag l_j.

The document matrix for d_k *is defined as* $D_k = [w_{i,j,k}]$*, and the query matrix for* q *as* $Q = [w_{i,j,q}]$.

From the classical point of view, this model is based on a $t \times m$-dimensional real subspace, where both queries and documents are elements from $\mathcal{M}_{t\times m}(\mathbb{R})$. For a given document matrix $D_k = (\boldsymbol{d_{k,1}}, \ldots, \boldsymbol{d_{k,m}})$, each column is a vector $\boldsymbol{d_{k,j}}$ containing the weights of the terms bound to tag j in the corresponding document . The same applies to user queries: a query matrix $Q = (\boldsymbol{q_1}, \ldots, \boldsymbol{q_m})$ is composed by m vectors $\boldsymbol{q_j^T} = (w_{1,j,q}, \ldots w_{t,j,q})$, and the $w_{i,j,q}$ represent how much the user is interested in documents containing term i bound to tag j.

We can see that the proposed matrix model extends the classical vector model: the vector model is a matrix model where $m = 1$.

Example 2. At the bottom of figure 1 appear two metadata documents belonging to a simple collection (c.f. example 1). Let us assume that $w_{i,j,k} = n_{ijk}$, where n_{ijk} is the raw frequency of term i in document k for tag j. Some examples about the information managed for this IR system are presented below:

- Dictionaries: $T = \{things,\ to,\ do,\ read,\ write\}$; $t = 5$;
  $M = \{List.Title,\ List.Item,\ List.Item.Abstract\}$; $m = 3$
- Example queries: q_a = {Retrieve docs. containing *read* at tag *List.Item* }; q_b = {Retrieve docs. containing *write* or *do* in any tag at or below *List.Item* }
- Matrices: $D_1 = (\ (1, 1, 1, 0, 0)^T,\ (0, 0, 0, 2, 1)^T,\ (0, 0, 0, 0, 0)^T\ )$
  $D_2 = (\ (0, 0, 0, 0, 0)^T,\ (0, 0, 0, 0, 1)^T,\ (0, 0, 1, 0, 0)^T\ )$
  $Q_a = (\ (0, 0, 0, 0, 0)^T,\ (0, 0, 0, 1, 0)^T,\ (0, 0, 0, 0, 0)^T\ )$
  $Q_b = (\ (0, 0, 0, 0, 0)^T,\ (0, 0, 1, 0, 1)^T,\ (0, 0, 1, 0, 1)^T\ )$

4.2 Documents as Linear Transformations

Document representations are $t \times m$ matrices. Matrix algebra states that these matrices can be seen as the representation of a linear transformation between two subspaces. These subspaces will be defined as the *tag subspace* and *term subspace.*

Definition 5 (Tag subspace, content profile). *Let* m *be the number of distinct metadata tags in the collection. We define* $V_l \subset \mathbb{R}^m$ *as the subspace whose vectors define* content profiles. *The* i*-th coordinate of a vector* $\boldsymbol{v} \in V_l$ *defines the weight of tag* l_i *in the corresponding content profile. Null coordinates in* $\boldsymbol{v}$ *represent tags not participating in the profile.*

A canonical base for V_l *is composed by the set of content profiles* $\boldsymbol{u_j^m}$, $1 \leq j \leq m$. *Each* $\boldsymbol{u_j^m}$ *represents a profile with a single tag. This canonical base represents the tag dictionary.*

Content profiles represent the relative weights of metadata tags in a given context. For a given collection, they define how relevant is a given tag or set of tags with respect to the others, and characterize the tag dictionary.

Definition 6 (Term subspace, content suit). *Let t be the number of distinct index terms in the collection. We define $V_k \subset \mathbb{R}^t$ as the subspace whose vectors define* content suits*. The i-th coordinate of a vector $\boldsymbol{v} \in V_k$ defines the weight of term k_i in the corresponding content suit. Null coordinates in $\boldsymbol{v}$ represent terms not participating in the suit.*

A canonical base for V_k is composed by the set of content suits $\boldsymbol{u_j^t}$, $1 \leq j \leq t$. Each $\boldsymbol{u_j^t}$ represents a content suit with a single term. This canonical base represents the term dictionary.

Content suits represent sets of index terms and their relative weight for a given context. They characterize the term dictionary for a collection. Then, we can associate a linear transformation $T_{D_k} : V_l \to V_k$ to a document representation $D_k \in \mathcal{M}_{t \times m}(\mathbb{R})$. This transformation assigns content suits to content profiles, that is, given a content profile, the linear transformation associated to a document representation D_k returns the content suit corresponding to that content profile for that document.

As stated in definition 4, a document matrix defines the weights of terms when bound to a given tag. Given a content profile defining the relative relevance of tags, a document matrix defines the corresponding content according to that content profile.

Additionally, for a given collection content profiles can be used to obtain new representations of documents where metadata information is *filtered out* according to the profiles. As a consequence, this interpretation of document matrices as linear transformations formalizes the generation of classical vector models from matrix ones.

Example 3. Let us consider matrix D_1 in example 2 (c.f. also figure 1). This matrix corresponds to the linear transformation $D_1\boldsymbol{x} = \boldsymbol{y}$ where $\boldsymbol{x} \in V_l$ and $\boldsymbol{y} \in V_k$. Content profile $\boldsymbol{x^T} = (1\ 0\ 0)$ defines a context where we are only interested in terms bound to tag *List.Title*. The corresponding content suit is $\boldsymbol{y^T} = (1\ 1\ 1\ 0\ 0)$, which represents the raw frequencies of the terms bound to that tag in the original document.

For content profile $\boldsymbol{x^T} = (1\ 1\ 1)$ (i. e. a profile where all tags are equally relevant), we get $\boldsymbol{y^T} = (1\ 1\ 1\ 2\ 1)$, that is, a content suit having all terms in the original document. Term weights in this suit correspond to the raw frequencies of index terms in the original document.

Note that $\boldsymbol{x^T} = (1\ 0\ 0)$, when applied to all documents in the collection, will generate a classical vector model where documents will only retain information bound to tag *List.Title*, whereas $\boldsymbol{x^T} = (1\ 1\ 1)$ will generate a classical vector model where documents have the same index terms and all structure provided by metadata is lost.

4.3 Queries and Ranking as Linear Transformations

Content profiles and content suits also apply to queries. Query matrices can also be considered linear transformations from the profile subspace to the suit subspace. In this case, a content profile represents at what extent users are interested in terms bound to a given set of tags, whereas content suits represent the relative relevance of a set of index terms for a given query.

On the other side, in section 3 we defined a ranking function that returns a real number for each query $q_i \in Q$ and document representation $d_j \in D$. This function defines an ordering among the documents w.r.t. the query q_i. Besides, queries can be interpreted as functions that take as a parameter a query matrix and return a ranking vector. This idea can also be formalized using subspaces and linear transformations:

Definition 7 (Ranking subspace, ranking). *Let N be the size of the collection. We define $V_r \subset \mathbb{R}^N$ as the subspace where vectors in V_r define* rankings*. The i-th coordinate of a vector $\boldsymbol{v} \in V_r$ defines the relative position of document d_i in a ranking of documents. Null coordinates in $\boldsymbol{v}$ represent documents not participating in the ranking.*

A canonical base for V_r is composed by the set of rankings $\boldsymbol{u_j^N}$, $1 \leq j \leq N$. Each $\boldsymbol{u_j^N}$ represents a ranking with a single document.

Example 4. For the collection in example 2, ranking $\boldsymbol{v^T} = (1, 0)$ represents a ranking for query q_a whereas $\boldsymbol{v^T} = (0.1714, 0.5)$ is a ranking for query q_b (c. f. section 5)

5 Relevance analysis in a matrix model

Relevance analysis will based on the same principles as the classical vector model. In our case, queries trigger the calculation of a similarity function $sim(Q, D_j)$ to estimate the degree of similarity between a query and the documents in the collection.

Classical vector-based similarity functions are now calculated from document and query matrices. Similarity is based on a measure of the distance between a document matrix and a query matrix in the corresponding $t \times m$ subspace. 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, like statistics computed for single tags (e.g. the number of documents containing tag j), computed for single tags in a given document (e.g. the number of terms bound to tag j in document k), or computed for single tags for a given term (e.g. $idf_{ij} = \log(N_j/n_{ij})$ represents the discrimination power of term i for contents bound to tag j). Obviously, other combinations are possible. As far as we know, the evaluation of useful metadata-dependent statistics is an open problem.

Existing results for the vector model can be translated to the matrix model if we note that queries and documents are elements from a $t \times m$ subspace.

Similarity functions based on the scalar product of vectors are typically used in the classical vector model. For example

$$sim(q,d) = \frac{< \boldsymbol{q},\ \boldsymbol{d} >}{|\boldsymbol{q}||\boldsymbol{d}|} = \frac{q_1 d_1 + \ldots + q_n d_n}{|\boldsymbol{q}||\boldsymbol{d}|}$$

measures the similarity of document d with respect to query q as the cosine of the angle between $\boldsymbol{d}$ and $\boldsymbol{q}$.

For the matrix model, we can also define a similarity function having the properties of a scalar product:

$$sim(Q,D) = \frac{< Q,\ D >}{|Q||D|} = \frac{tr(Q^T D)}{|Q||D|} = \frac{tr(Q^D T)}{|Q||D|} = \frac{\sum_i \sum_j q_{ij} d_{ij}}{|Q||D|}$$

where tr represents the trace of a matrix. In the new scenery, this approach has further advantages due to the properties of the scalar product combined with those of the product of matrices.

Proposition 1. *Let D and Q be respectivelty a document matrix and a query matrix. Let T_T be a transformation in the term subspace and T_M a transformation in the tag subspace. Then*

$$< Q,\ DT_T > = < QT_T^T,\ D > \ ;\ < Q,\ T_M D > = < T_M^T Q,\ D >$$
$$< Q,\ T_{M1} T_{M2} D T_{T1} T_{T2} > \ = \ < T_{M2}^T T_{M1}^T Q T_{T2}^T T_{T1}^T,\ D >$$

The proof is based on the algebraic manipulation of the expressions above.

Property 1 establishes a relation between transformations on queries and documents insofar similarity calculation is concerned, which opens the door to the methodical characterization of transformations and their influence on the computed similarity in a metadata-enabled world. Besides, as the vector model is a particular case of the matrix model, available results for the former can be systematically adapted to the latter.

Example 5. Let us take the collection in figure 1. Transforming documents using $\boldsymbol{u^T} = (1\ 1\ 1)$ generates classical vector models where all metadata information is lost. For each index term k_i, the resulting vector $\boldsymbol{d_k} = D_k \boldsymbol{u}$ has as weights $w_{i,k} = \sum_j w_{i,j,k}$.

Assuming in this example that weights correspond to raw frequencies, all text bound to any tag will be equally relevant. After this transformation being applied to all documents, we get a classical IR vector model where metadata-specific information will not be considered for relevance calculation. Vectors for the transformed model are

$$\boldsymbol{d_1^T} = (1,1,1,2,1),\ \boldsymbol{d_2^T} = (0,0,1,0,1),\ \boldsymbol{q_a^T} = (0,0,0,1,0),\ \boldsymbol{q_b^T} = (0,0,2,0,2)$$

To calculate the similarity between documents and queries we can now apply known results for this classical model. Let $sim(d_j, q) = \cos(\angle \boldsymbol{w_j w_q})$, where

$\boldsymbol{w_q} = \boldsymbol{q}$, and $w_{ij} = n_{ij}/n_i$, n_i being the number of documents where term i appears.

For query $q = q_b$ we have $n_{things} = n_{to} = n_{read} = 1$, $n_{do} = n_{write} = 2$, and therefore $\boldsymbol{n_i^T} = (1\ 1\ 2\ 1\ 2)$. Then, $\boldsymbol{w_1^T} = (1\ 1\ 0.5\ 2\ 0.5)$, $\boldsymbol{w_2^T} = (0\ 0\ 0.5\ 0\ 0.5)$, and the corresponding values for the similarity function are $sim(d_1, q_b) = 0.277$, $sim(d_2, q_b) = 1$.

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 only by the terms in the query.

Example 6. A unit vector $\boldsymbol{u_j}$ generates classical IR systems whose documents contain only the information bound to tag j. For a given tag dictionary T of size m, we can generate m classical IR systems projecting the original IR system using $\boldsymbol{u_j}$, $j = 1 \ldots m$. Then, m similarity results can be calculated for a given query. We have to select a procedure to combine these 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[2]. We will assume that the query string is composed by a set of *o*r-ed subqueries, each bound to a tag.

For the extended boolean model $sim_{bool-ext} = \frac{1}{m'}\sqrt{\sum_{m=1}^{m'} sim_m^2}$, where m' is the number of non-null similarities. For the query $q = q_b$ we have:

Projection	$\boldsymbol{d_1^T}$	$\boldsymbol{q^T}$	$sim_i(d_1, q)$
$\boldsymbol{u_1}$ *List.Title*	(1 1 1 0 0)	(0 0 0 0 0)	0
$\boldsymbol{u_2}$ *List.Item*	(0 0 0 2 1)	(0 0 1 0 1)	0.1714
$\boldsymbol{u_3}$ *List.Item.Abstract*	(0 0 0 0 0)	(0 0 1 0 1)	0
Document 1: $sim_t(d_1, q) = 0.1714$			
Projection	$\boldsymbol{d_2^T}$	$\boldsymbol{q^T}$	$sim_i(d_2, q)$
$\boldsymbol{u_1}$ *List.Title*	(0 0 0 0 0)	(0 0 0 0 0)	0
$\boldsymbol{u_2}$ *List.Item*	(0 0 0 0 1)	(0 0 1 0 1)	0.707
$\boldsymbol{u_3}$ *List.Item.Abstract*	(0 0 1 0 0)	(0 0 1 0 1)	0.707
Document 2: $sim_t(d_2, q) = 0.5$			

If we compare the results above with those from example 5, we see that the relevance of both documents to the query q_b decreases. This is due to the role played by 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.

6 Concluding remarks

The need for efficient information retrieval an management tools for the Web, and the introduction of advanced markup, determined the evolution of Information Retrieval techniques to take into account metadata. 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 problem may be helpful.

In this paper we have introduced a matrix-based characterization for metadata-based IR where documents, user queries, and document and term transformations are modeled as matrices. This proposal is independent of the metadata model as long as metadata documents can be represented as a collection of tag-value pairs. Therefore, it can be easily applied to several metadata and tag (meta)language proposals, like XML or XML-derived languages. Furthermore, it seamlessly integrates previous results from classical IR.

The proposal can also be seen as an extension of the vector model for IR. It is complete in the sense that it provides a document and query representation framework and a ranking function. We hope that this characterization will contribute to construct a solid foundation for metadata-enabled IR.

Presently, we are using the proposal discussed in this paper to evaluate different relevance calculation methods for metadata IR using DelfosnetX[5]. Indeed, 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 allows to study the relative performance of classical and metadata-oriented IR systems. An Application Programmer Interface (API) is also provided to easily customize DelfosnetX for particular applications.

References

1. S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. Wiener. The lorel query langauge for semistructured data. *International Journal of Digital Libraries*, 1(1):68 – 88, 1997.
2. R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval.* Addison-Wesley, 1999.
3. F. J. Burkowski. An algebra for hierarchically organized text-dominated databases. *Information Processing & Management*, 28(3):333–348, 1992.
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. M. Fernández, P. Pavón, J. Rodríguez, L. Anido, and M. Llamas. Delfosnetx: A workbench for XML-based information retrieval systems. In *Procs. of the 7th International Symposium of String Processing and Information Retrieval*, pages 87–95. IEEE Comp. Soc. Press, 2000.
6. A. L. Hors, P. L. Hégaret, L. Wood, G. Nicol, J. Robie, M. Champion, and S. Byrne, editors. *Document Object Model Level 2 Core Specification.* W3 Consortium, 1998. W3C Recommendation.
7. G. Salton and M. E. Lesk. Computer evaluation of indexing and text processing. *Journal of the ACM*, 15(1):8–36, 1968.
8. H. Schneider and G. P. Barker. *Matrices and Linear Algebra.* Dover Books on Advanced Mathematics. Dover Pubns, 2nd edition, 1989.