IRIT at INEX 2003

Karen Sauvagnat
IRIT/SIG-RFI
118 route de Narbonne
31062 Toulouse cedex 4
+33-5-61-55-68-99
sauvagna@irit.fr

Gilles Hubert
IRIT/SIG-EVI
118 route de Narbonne
31062 Toulouse cedex 4
+33-5-61-55-74-35
hubert@irit.fr

Mohand Boughanem
IRIT/SIG-RFI
118 route de Narbonne
31062 Toulouse cedex 4
+33-5-61-55-74-16
bougha@irit.fr

Josiane Mothe
IRIT/SIG-EVI
118 route de Narbonne
31062 Toulouse cedex 4
+33-5-61-55-63-22
mothe@irit.fr

ABSTRACT

This paper describes the retrieval approaches proposed by IRIT in INEX'2003 evaluation. The primary approach uses Mercure system and different modules to perform content only and content and structure queries. The paper also discusses a second approach based on a voting method previously applied in the context of automatic text categorization.

Keywords

Information Retrieval, XML retrieval, connectionist model, voting method, automatic text categorization

1. INTRODUCTION

XML (eXtensible Markup Language) has recently emerged as a new standard for representation and data exchange on the Internet [29]. If this tendency goes on, XML will certainly become a universal format and HTML (Hypertext Markup Language) will disappear in aid of XML. Consequently, the information retrieval issue in XML collections becomes crucial.

A growing number of approaches are dealing with structured documents like XML. They can be divided into three main groups: database, XML-oriented specific approaches and IR approaches. The database community considers XML collections as databases, and tries to develop models for representing and querying documents, according to the content and the structure of these documents. Many languages have been developed for querying and updating these databases [1][18][24][30][11]. XML specific oriented approaches estimate the relevance of document parts according to the relevance of their structurally related parts. They are also named aggregation-based methods [8][15][7][13][16]. In IR approaches, traditional IR models are adapted to be used on structured collections [17][20][22].

In this paper, we present two IR approaches applied to structured documents retrieval, within the context of INEX'2003: the first approach uses Mercure information retrieval system, while the second one is based on a voting method used initially for automatic text categorization. Section 2 presents the INEX initiative. Section 3 describes the Mercure model, and the INEX search approach with Mercure system is reported in section 4. Section 5 and 6 present first the voting method defined in the context of categorization and then the adaptations we integrated within the INEX'2003 context.

2. THE INEX INITIATIVE

2.1 Collection

INEX collection, 21 IEEE Computer Society journals from 1995-2002, consists of 12 135 (when ignoring the volume.xml files) documents with extensive XML-markup. All documents respect the same DTD.

2.2 Queries

As last year, participants to INEX'2003 have to perform two types of queries. CO (Content Only) queries are requests that ignore the document structure and contain only content related conditions, e.g. only specify what a document/component should be about. CAS (Content and Structure) queries contain explicit references to the XML structure, and restrict the context of interest and/or the context of certain search concepts. Both CO and CAS topics are made up of four parts: topic title, topic description, narrative and keywords.

Within the ad-hoc retrieval task, three sub-tasks are defined: (1) the CO task, using CO queries, (2) the SCAS task, using CAS queries, for which the structural constraints must be strictly matched, (3) the VCAS task, also using CAS queries, but for which the structural constraints can be considered as vague conditions.

3. MERCURE SYSTEM

Mercure is a full-text information retrieval system based on a connectionist approach and modeled by a multi-layer network. The network is composed of a query layer (set of query terms), a term layer (representing the indexing terms) and a document layer [4].

Mercure includes the implementation of retrieval process based on spreading activation forward and backward through the weighted links. Queries and documents can be used either as inputs or outputs. The links between layers are symmetric and their weights are based on the *tf-idf* measure inspired by OKAPI [23] and SMART term weighting.

The query-term links are weighted as follows :

$$q_{ui} = \begin{cases} \dfrac{nq_u * qtf_{ui}}{nq_u - qtf_{ui}} \text{ if } (\mathrm{nq_u} > qtf_{ui}) \\ qtf_{ui} \text{ otherwise} \end{cases} \quad (1)$$

Where:

- q_{ui} : the weight of the term t_i in the query u at the stage s
- qtf_{ui}: the frequency of the query term t_i in the query u
- nq_u: the number of terms in the query u

The term-document link weights are expressed by :

$$d_{ij} = \frac{tf_{ij} * (h_1 + h_2 * \log(\frac{N}{n_i}))}{h_3 + h_4 * \frac{dl_j}{\Delta_l} + h_5 * tf_{ij}} \quad (2)$$

Where:

- d_{ij} : term-document weight of term t_i and document d_j
- tf_{ij}: term frequency of t_i in the document d_j
- N: total number of documents
- n_i: number of documents containing term t_i
- h_1, h_2, h_3, h_4 and h_5: constant parameters
- Δl : average document length
- dlj :number of terms in the document dj

The query evaluation function computes the similarity between queries and documents.

Each term node computes an input value: $In(t_i) = q_{ui}^{(s)}$

and an activation value: $Out(t_i) = g(In(t_i))$, where g is the term layer activation function.

Each term node propagates then this activation value to the document nodes through the term-document links. Each document node computes an input value: $In(d_j) = \sum_i Out(t_i) * d_{ij}$ and an activation value: $Out(d_j) = g(In(d_j))$, where g is the document layer activation function.

Documents are then ranked by ascending order of their activation value.

The ranking function (activation) is modified to take into account term proximity in a document [14]. Thus, documents having close query terms compute a new input value:

$$In(d_j) = \sum_i (Out(t_i) * d_{ij}) * \sum_i \frac{\alpha}{prox_{i,i-1}} \quad (3)$$

Where:

- α is a constant parameter so that $\frac{\alpha}{prox_{i,i-1}} \geq 1$. α is set to 4 for INEX'2003 experiments.
- $prox_{i,i-1}$ is the number of terms separating the query terms t_i and t_{i-1} in the window of α terms in the document. The query terms are ranked according to their position in the query text.

In other words, documents having close query terms (i.e. no more than α words separate two consecutive query terms in the document content) increase their input value.

In addition, we have implemented two modules that are used to process structured documents. The aim of these modules is to filter the most specific[1] and exhaustive[2] elements of the documents returned by Mercure [15].

[1] An element is specific to a query if all its information content concerns the query.

[2] An element is exhaustive to a query if the element contains all the required information.

The first module, which is *content-oriented*, deals with queries composed of simple keyword terms. It glances through documents retrieved by Mercure, and finds elements answering the queries in the most specific and exhaustive way. Element types that can be retrieved are pre-specified manually, according to the DTD of the documents. This module performs as follow: for each document retrieved by Mercure, it searches occurrences of query terms in all pre-specified elements. It returns the elements containing the greatest number of query terms. If more than *k* elements are supposed to be the most specific and exhaustive, the module returns the whole document.

The second module, which is *content-and-structure-oriented*, performs queries containing both explicit references to the XML structure and content constraints. These queries can be divided into two parts : a target element and a content constraint on this target element. As the content-oriented module, the second module browses documents returned by Mercure, and returns specific elements (e.g. target elements) containing the greatest number of query terms specified in the content constraints. If the target elements do not contain any of the terms of the content constraints, the document retrieved by Mercure is removed from the list of results.

Thus, the main difference between the two modules is the way they process the documents structure. In the content-oriented module, elements that can be returned are pre-specified manually without user intervention. The user only gives keywords and cannot express structural conditions in his query. Using the content-and-structure-oriented module, users explicitly give a target element and content constraints on this target element.

As a result for both modules, we obtain a ranked list of elements/documents.

4. THE INEX SEARCH APPROACH WITH THE MERCURE SYSTEM

4.1 Indexing the INEX database and the queries

The INEX collection was indexed in order to take into account term positions in the documents. Terms are stemmed with Porter algorithm and a stop-word list is used in order to remove non-significant terms from the index. No structural information is kept in the index.

Queries are then indexed in two different ways.

CO and CAS queries are first indexed using title and keywords fields, in order to build queries *for Mercure system*. Regarding CO queries, we simply remove terms preceded by minus (which means that the user does not want these terms appear in the results) and keep all the other terms. CAS queries are indexed using terms in the content constraints of the title field and terms of the keyword field. For both types of queries, terms are stemmed with the Porter algorithm and terms appearing in the stop-word list are removed, as it is done for the documents.

Then, CAS queries are re-indexed *for the content-and-structure-oriented module*. Indeed, as explained before, the content-and-structure-oriented module needs the target element of queries in order to process them. Let us take some examples of CAS queries:

Top.	Title field	Description
63	//article[about(.,'"digital library"') AND about (.//p,'+authorization +"access control" + security')]	Relevant documents are about digital libraries and include one or more paragraphs discussing security, authorization or access control in digital libraries.
66	>/article[.//yr <='2000'] //sec[about(.,'"search engines"')]	The user is looking for sections of articles published before 2000, which discuss search engines.
84	//p[about(.'overview "distributed query processing" join')]	The user wants paragraphs that give an overview about distributed query processing techniques with a focus on joins implementations.
90	//article[about(./sec,' +trust authentication "electronic commerce" e-commerce e-business marketplace') //abs[about(.,'trust authentication')]	The user wants to find abstracts or article that discuss automated tools for establishing trust between parties on the internet. The article should discuss applications of trust for authenticating parties in e-commerce.

Table 1: Examples of CAS queries

All the content constraints occurring in the *about* predicates are first indexed for Mercure system, even though they are not on the target element (in topics 63 and 90 for example). Targets elements (*article* for topic 63, *section* for topic 66, *paragraph* for topic 84 and *abstract* for topic 90) are then indexed for the content-and-structure-oriented module.

About 20% of the CAS topics (like topic 66) contain a constraint on the year of publication. This constraint is also stored and will be used to filter results of the content-and-structure-oriented module.

4.2 Retrieval

In both cases (CO queries and CAS queries), a first search is performed with Mercure search engine using the content part of the queries. As a result, a ranked list of 1000 documents is selected for each query. Then, the content-oriented module is used to process the document results of CO queries, and the content-and-structure-oriented module for CAS queries. Both modules return a ranked list of elements/documents, derived from the first ordered list of documents returned by Mercure system.

4.2.1 Retrieval with CO queries

According to the DTD, we have decided to allow the content-oriented module to return only *section* or *abstract* elements. Indeed, section and abstract elements are supposed to be large enough to be exhaustive and small enough to be specific.

If the content-oriented module finds more than two relevant elements (k =2) within a given document, the whole document is returned.

4.2.2 Retrieval with CAS queries

The content-and-structure-oriented module browses documents returned by Mercure, and returns target elements containing the greatest number of query terms specified in *all* the content constraints of CAS queries. If no occurrence of terms contained in the content constraints is found in target elements, the document returned by Mercure is removed from the results list. Indeed, the target element always have a content constraint.

Then, if the query contains a year constraint, elements returned by the content-and-structure-oriented module are filtered, according to the article publication date .

4.3 Submitted runs

The first goal of our experiments in INEX'2003 is to test whether a full-text information retrieval system can be easily adapted to structured retrieval and to evaluate how suitable are the full-text IR based techniques for such kind of retrieval. Our approach can be compared to the fetch and browse method proposed in [5]. No static structure is used a priori and so, all types of XML documents can be processed. The second goal of our experiments is to measure the effect of term positions in INEX query types.

Five runs performed with Mercure have been submitted to INEX'2003. The runs are labeled as follows:

- *pos* indicates that Mercure uses term positions to process queries (*Mercure2.pos_co_ti, Mercure2.pos_cas_ti, Mercure2.pos_vcas_ti*), otherwise runs are based on a Mercure simple search (*Mercure2.co_ti, Mercure2.cas_ti*).
- *Co (Mercure2.co_ti, Mercure2.pos_co_ti), cas (Mercure2.cas_ti, Mercure2.pos_cas_ti)*, and *vcas (Mercure2.pos_vcas_keyti)* indicate the sub-task type, e.g. CO, SCAS or VCAS
- *ti* indicates that only title field of queries was used (*Mercure2.co_ti, Mercure2.pos_co_ti, Mercure2.cas_ti, Mercure2.pos_cas_ti*) , whereas *keyti* indicates that the title and keywords fields were used (*Mercure2.pos_vcas_keyti)*.

4.4 First results

The following table shows the results of the five runs, in terms of average precision:

Run	Strict quantization		Generalized quantization	
	Average precision	Rank	Average precision	Rank
Mercure2.co_ti	0.0056	50/56	0.0088	48/56
Mercure2.pos_co_ti	0.0344	28/56	0.0172	41/56
Mercure2.cas_ti	0.0719	33/38	0.0612	34/38
Mercure2.pos_cas_ti	0.1641	25/38	0.1499	24/38
Mercure2.pos_vcas_keyti	NC	NC	NC	NC

Table 2: Results of the five runs performed with Mercure system

The first result that can be drawn from Table 2 is that runs using term positions are definitely better that simple search for both query types (CO and CAS). Average precision for runs using term positions (*Mercure2.pos_cas_ti* , *Mercure2.pos_vcas_keyti*, and *Mercure2.pos_co_ti*) is about four times higher than average precision of runs performed with a single Mercure search (*Mercure2.cas_ti* , *Mercure2.co_ti*).

4.5 Discussion and future works

Regarding this year experiments and results, some investigations have to be performed. First of all, for the CO task, elements that can be returned by the content-oriented module are pre-selected manually. These types of elements are not always necessarily the most exhaustive and specific: it depends on the way the DTD was understood by the document creators. Statistics [12] or aggregation methods [7] [13] may be used to find those elements automatically. Then, the content-and-structure-oriented module is not able to perform all the content and structural constraints. Indeed, it processes only content constraint on the target element and year constraints. For example, in topic 90, the first *about* predicate is on sections, whereas the target element is abstract: the module does not insure that the content constraint on sections is respected. However, topics such as topic 84 are fully treated. According to these remarks, the content-and-structure-oriented module seems to be more adapted to the VCAS task. For this purpose, the run *Mercure2.pos_vcas_keyti* was performed and submitted. Finally, query processing is relatively slow, because the modules have to browse all documents returned by Mercure in order to find relevant elements. Regarding these limitations, an indexing model taking into account the structural and content information of documents seems to be necessary.

Moreover, our approach uses the *idf* measure to compute a retrieval status value for documents (and then documents are browsed to return relevant elements). The *idf* measure is also used in [7] and [26], in order to directly return relevant elements. However, term occurrences in elements do not necessary follow a Zipf law [31]. The number of term repetitions can be (very) reduced in XML documents and *idf* is not necessarily appropriate [6][10]. The use *of ief (Inverse Element Frequency)* is proposed in [28] and [9]. An indexing scheme storing different IR statistics might be interesting on the INEX collection: thus, combinations of IR and XML-specific approaches could be tested.

5. A VOTING METHOD FOR INFORMATION RETRIEVAL

The approach proposed is derived from a process for categorisation of textual documents. This categorisation intends to link documents with pre-defined categories. Our approach focuses on categories organised as a taxonomy. The original aspect is that our approach involves a voting principle instead of a classical similarity computing.

Our approach associates each text with different categories as opposed to most of the other categorisation techniques. The association of a text to categories is based on the Vector Voting method [21]. This method relies on the terms describing each category and their automatic extraction from the text to be categorised. The voting process evaluates the importance of the association between a given text and a given category. This method is similar to the HVV method (Hyperlink Vector Voting) used within the Web context to compute the pertinence of Web page regarding the web sites referring to it [19]. In our context, the initial strategy considers that the more the category terms appear in the text, the more the link between the text and this category is strong.

The association principle between a document and categories is composed of different steps:

– Compute the profile of each category. In automatic categorisation, profiles correspond generally to a set of weighted terms [25][27] which can be obtained by training from previous categorised documents.

– Extract automatically the concepts describing a document and their importance for the document. The extraction is based on a set of rules to treat, for example, document tags, and processes to treat synonymy and to remove stop words.

– For each category of the hierarchy, compute a score with a voting function which measures the representativity of the category according to the text. Different functions can be used as voting function based on measures such as term importance in text and in hierarchy, text size, hierarchy size, number of terms describing a category that appear in the text.

– Sort the winning categories according to their score, and eventually select the best categories (for example, scores greater than a fixed threshold, or n greatest scores).

We have studied different voting functions whose results are presented in [2][3]. The voting function must take into account the importance in the document of each term describing the category, the discriminant power of each term describing the category, the category representativity within the document. The function providing the best results is described as follows :

$$Vote(E_H, D) = \sum_{\forall t \in E} \frac{F(t,D)}{S(D)} \cdot \frac{S(H)}{F(t,H)} \cdot e^{\frac{NT(E,D)}{NT(E)}} \qquad (1)$$

where

E_H corresponds the category E in the hierarchy H

D is a document

$\frac{F(t,D)}{S(D)}$	This factor measures the importance of the term t in the document D. F(t,D) corresponds the number of occurrences of the term t in the document D and S(D) corresponds to the size (number of terms) of D.
$\frac{S(H)}{F(t,H)}$	This factor measures the discriminant power the term t in the hierarchy H. F(T,H) corresponds to the number of occurrences of the term t in the hierarchy H and S(H) corresponds to size of H.
$\frac{NT(E,D)}{NT(E)}$	This factor measures the presence rate of terms representing the category in the text (importance of the category). NT(E) corresponds to the number of terms in the category E and NT(E,D) corresponds to the number of terms of the category E that appear in the document D

The above function (1) considers the two factors as equivalent: the importance of a term in the document and the discriminant power of this term in the hierarchy. Applying the exponential function to the third factor (i.e. the presence rate of terms representing the category in the text) aims at accentuate its importance.

The function is completed with the notion of *coverage*. The aim of the coverage is to ensure that only categories enough represented in a document will be selected for this document. The coverage is a threshold corresponding to the percentage of terms from a category that appear in a text . For example, a coverage of 50% implies that at least half of terms describing a category have to appear in the text of a document to be selected.

6. THE INEX SEARCH APPROACH WITH A VOTING METHOD

6.1 Evolution if the categorisation process

From the topic point of view, CO and CAS topics are constituted of different informative parts (title, keywords, description) that can be exploited to construct their profile. Although our method can use all the possible parts we first focused on to the title and keyword parts for the INEX'2003 experiments. For both topic types, stop words are removed and optionally terms can be stemmed using Porter algorithm.

For CAS topics, an additional step identify the structural constraints indicated in a topic. All the structural constraints defined on target elements of topics are taken into account and stored to be processed in a post categorisation step to filter the results issued from the categorisation step. Only, the results having expected xpaths are kept. About content structural constraints (e.g. about(.//p,'+authorization +"access control" +security') or .//yr <='2000') only constraints on the year of the article are taken into account and stored to filter the results. More complex content constraints have not been treated for INEX'2003. Next experiments are planned about the extension of the voting method to take into account such constraints.

From the INEX collection point of view, the documents are considered as sets of text chunks identified by xpaths. For each document, concepts are extracted automatically with the different xpaths identifying the chunks where they appear and their importance in the chunk is calculated. For INEX'2003 experiments all XML tags have been taken into account.

The voting method is applied without any modification. Topics are considered as categories to which document elements have to be assigned. The result is constituted of a list of topics associated to each chunk of text (identified by its xpath) for each document.

6.2 Experiments

Our experiments aim at evaluating the efficiency of the voting function and estimating the adaptations needed for the categorisation process in a context such as INEX'2003.

Four runs based on the voting method were submitted to INEX'2003. The main parameter that distinguishes the runs is to apply or not a coverage (C50 corresponds to apply a coverage of 50% i.e. half of the terms describing the topic must appear in the text to keep the topic, C0 corresponds to no coverage). No stemming process has been applied for the submitted although it can be added. The tcXX% parameter specifies that only the elements having a score over a given percentage of the best score will be kept (e.g. tc50% indicates that only the elements having a score over the half of the best score are kept in the result)..

6.3 Results

The following table shows the preliminary results of the four runs based on the voting method :

Run	**Strict quantization**		**Generalized quantization**	
	Average precision	**Rank**	**Average precision**	**Rank**
VotingNoStemTKCO tc75%C0nonorm	0.0012	54/56	0.0041	56/56
VotingNoStemTKVCAS C50nonorm	NC	NC	NC	NC
VotingNoStemTKSCAS tc50%C0nonorm	0.0626	34/38	0.0746	31/38
VotingNoStemTKVCAS tc50%C0nonorm	NC	NC	NC	NC

Table 3: Results of the 4 runs performed with the voting method

Results for VCAS topics are not yet known.

6.4 Discussion and future works

Regarding the experiment that were performed and the obtained results we can notice that:

- the voting method applied without coverage tends to promote short chunks of text that have only one term in common with the topic. Introducing coverage intends to correct this since short chunks of text that have several terms in common with the topic are less frequent than longer ones. We plan to study changes made to the voting function to evaluate their impact on results notably with regard to the size of text chunks.
- The elementary level has been considered to identify the different chunks of text. This choice leads to miss complex chunks of text constituted of different elementary chunks with high voting scores. A rebuilding of complex chunk should be integrated in the process.
- Structural constraints defined on the content of topics have not been taken into account. This aspect constitutes the main axis of study to extend the voting method. The main idea is to integrate the constraint when computing the voting score in order to promote relevant text chunks regarding content which respect the structural constraints without eliminating relevant chunks (regarding content) but that do not satisfy the constraints.

7. ACKNOWLEDGMENTS

We thank Cécile Laffaire for her participation in this work.

8. REFERENCES

[1] Abiteboul, S. , Quass, D., Mc Hugh, J., Widom, J., Wiener, J-L.. The Lorel query language for semi-structured data. International Journaf on Digital Libraries, 1(1), 68-88, 1997.

[2] Augé, J., Englmeier, K., Hubert, G., Mothe, J. Catégorisation automatique de textes basée sur des hiérarchies de concepts , 19ième Journées de Bases de Données Avancées, Lyon, p. 69-87, 2003.

[3] Augé, J., Englmeier, K., Hubert, G., Mothe, J. Classification automatique de textes basée sur des hiérarchies de concepts , Veille stratégique, scientifique et technologique, Barcelone, p. 291-300, 2001.

[4] Boughanem, M., Chrisment, C., Soule-Dupuy , C. Query modification based on relevance back-propagation in ad-hoc environment. Information Processing and Management, 35 (1999), 121-139, 1999.

[5] Chiaramella, Y. , Mulhem, P. , Fourel, F. A model for multimedia search information retrieval. Technical report, Basic Research Action FERMI 8134, 1996.

[6] Fuhr, N., Gövert, N., Röelleke, T. Dolores: a system for logic based retrieval of multimedia objects. In Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1998, Melbourne, Australia, pages 257-265. ACM Press, 1998.

[7] Fuhr, N., Grossjohann, K. XIRQL: A query Language for Information Retrieval in XML Documents. In Proceedings of the 24th annual ACM SIGIR conference on research and development in Information Retrieval, New Orleans, USA, pages 172-180. ACM Press, 2001.

[8] Fuller, M., Mackie, E., Sacks-Davis, R., Wilkinson, R. Structural answers for a large structured document collection. In Proc. ACM SIGIR, pp. 204-213. Pittsburgh, 1993.

[9] Grabs, T., Sheck, H.J. ETH Zürich at INEX : Flexible Information Retrieval from XML with PowerDB-XML. In INEX 2002 Workshop Proceedings, p. 35-40, Germany, 2002.

[10] Grabs, T. Storage and retrieval of XML documents with a cluster of databases systems. PhD thesis, Swiss Federal Institute of Technology Zurich, 2003.

[11] Grosjohann, K. Query Formulation and Result Visualization for XML Retrieval . In Proc of the SIGIR 2000 Workshop on XML and Information Retrieval. Athens, Greece, 2000.

[12] Hatano, K., Kinutani, H., Watanabe, M. An Appropriate Unit of Retrieval Results for XML Document Retrieval. In INEX 2002 Workshop Proceedings, p. 66-71, Germany, 2002.

[13] Kazai, G., Lalmas, M., Roelleke, T. Focused document retrieval, 9th International Symposium on string processing and information retrieval, Lisbon, Portugal, September 2002.

[14] Kean, E.M. The use of term position devices in ranked output experiments, Journal of Documentation, v.47 n.1, p.1-22, March 1991 .

[15] Lalmas, M. Dempster-Shafer's theory of evidence applied to structured documents : Modeling uncertainty. In Proc. ACM-SIGIR, pp. 110-118. Philadelphia, 1997.

[16] Lalmas, M., Roelleke, T. Four-valued knowledge augmentation for structured document retrieval. International Journal of Uncertainty, Fuziness and Knowledge-based systems (IJUFKS), Special issue on Management of uncertainty and imprecision in multimedia information systems, 11(1), 67-86, February 2003.

[17] Larson, R. R. . Cheshire II at INEX : Using a hybrid logistic regression and Boolean model for XML retrieval. In INEX 2002 Workshop Proceedings, p. 2-7, Germany, 2002

[18] Levy, A., Fernandez, M., Suciu, D., Florescu, D., Deutsch A. . XML-QL : A query language for XML. World Wide Web Consortium technical report, Number NOTE- xml-ql-19980819, 1998.

[19] Li, Y. Toward a qualitative search engine , IEEE Internet Computing, vol. 2, n° 4, p. 24-29, 1998.

[20] Ogilvie, P., Callan, J. : Languages models and structured documents retrieval. In INEX 2002 Proceedings, p. 18-23, Germany, 2002.

[21] Pauer, B., Holger, P. Statfinder . Document Package Statfinder, Vers. 1.8, may 2000.

[22] Piwowarski, B., Faure, G.E. , Gallinari, P. . Bayesian Networks and INEX. In INEX 2002 Workshop Proceedings, p. 7-12, Germany, 2002.

[23] Robertson, SE , Walker, S. Okapi/Keenbow at TREC-8. In Proceedings of the TREC-8 Conference, National Institute of Standards and Technology, pages 151-161, 2000.

[24] Robie , J., Lapp, J., Schach, D. XML Query Language (XQL). Proceedings of W3C QL'98 (Query Languages 98). Massachussets, 1998.

[25] Salton, G., « The SMART Retrieval System », Experiments in automatic document processing, Prentice Hall Inc., Englewood Cliffs, NL, 1971.

[26] Theobald, A., Weikum, G. The Index-Based XXL Search Engine for Querying XML Data with Relevance Ranking. In C. S. Jensen, K. G. Jeffery, J. Pokorny, S. Saltenis, E. Bertino, K. Bäohm, and M. Jarke, editors, Advances in Database Technology - EDBT 2002, 8th International Conference on Extending Database Technology, Prague, Czech Republic, volume 2287 of Lecture Notes in Computer Science, pages 477-495. Springer, 2002.

[27] Van Rijsbergen, K. Information Retrieval . Butterworths, London, Second Edition, 1979. http://www.dcs.gla.ac.uk/Keith/Preface.html

[28] Wolff, J.E., Flörke, H., Cremers, A.B. Searching and browsing collections of structural information. In Proc of IEEE advances in digital libraries, pp. 141-150. Washington, 2000.

[29] World Wide Web Consortium. Extensible Markup Language (XML) 1.0. http://www.w3.org/TR/REC-xml , Oct. 2000.

[30] World Wide Web Consortium. Xquery 1.0: an XML query language. http://www.w3.org/TR/xquery/ , Aug. 2003.

[31] Zipf, G. Human Behaviour and the Principle of Least Effort. Addison-Wesley, 1949.