Information Retrieval of Text, Structure and Sequential Data in Heterogeneous XML Document Collections

Eugen Popovici

THÈSE

SOUTENUE DEVANT

L'UNIVERSITÉ EUROPÉENNE DE BRETAGNE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ EUROPÉENNE DE BRETAGNE

Mention :

SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

par

EUGEN-COSTIN POPOVICI

Information Retrieval of Text, Structure and Sequential Data in Heterogeneous XML Document Collections

---

Recherche et filtrage d'information multimédia (texte, structure et séquence) dans des collections de documents XML hétérogènes

---

Présentée le 10 janvier 2008 devant la commission d'examen composée de :

M. BOUGHANEM	Professeur, Université Paul Sabatier, Toulouse III	Rapporteur
P. GROS	Directeur de Recherche, INRIA, Rennes	Examinateur
M. LALMAS	Professeur, Queen Mary University of London	Rapporteur
P.-F. MARTEAU	Professeur, Université de Bretagne-Sud	Directeur
G. MÉNIER	Maître de Conférences, Université de Bretagne-Sud	Co-directeur

Abstract

Nowadays digital documents represent a complex and heterogeneous mixture of text, structure, meta-data and multimedia information. The XML description language is now the standard used to represent such documents in digital libraries, product catalogs, scientific data repositories and across the Web. The management of semi structured data requires the development of appropriate indexing, filtering, searching and browsing methods and tools. In particular, the filtering and searching functions of the retrieval systems should be able to answer queries having an incomplete, imprecise or even erroneous knowledge about both the structure and the content of the XML documents. Moreover, these functions should maintain an algorithmic complexity compatible with the complexity of the data while maintaining the scalability of the system.

In this thesis, we explore methods for managing and searching collections of heterogeneous multimedia XML documents. We focus on the flexible searching of structure, text, and sequential data embedded in heterogeneous XML document databases. Based on the structural part of the XML documents, we propose a flexible model for the representation, indexing and retrieval of heterogeneous types of sequential data. The matching mechanism simultaneously exploits the structural organization of the sequential/textual data as well as the relevance and the characteristics of the unstructured content of the indexed documents. We also design and evaluate methods both for the approximate matching of structural constraints in an XML Information Retrieval (IR) framework and for the detection of best entry points to locate given topics in XML Documents. Finally, we explore the use of dedicated hardware architecture to accelerate the most expensive processing steps of our XML IR application.

Résumé

Les documents numériques sont aujourd'hui des données complexes qui intègrent d'une manière hétérogène des informations textuelles, structurelles, multimédia ainsi que des méta-données. Le langage de balisage générique XML s'est progressivement imposé comme support privilégié non seulement pour l'échange des données mais aussi pour leur stockage. La gestion des documents stockés sous les formats XML nécessite le développement de méthodes et d'outils spécifiques pour l'indexation, la recherche, le filtrage et la fouille des données. En particulier, les fonctions de recherche et de filtrage doivent prendre en compte des requêtes disposant de connaissances incomplètes, imprécises, parfois même erronées sur la structure ou le contenu des documents XML. Ces fonctions doivent par ailleurs maintenir une complexité algorithmique compatible avec la complexité des données et surtout avec leur volume toujours en forte croissance, ceci pour assurer le passage à l'échelle des solutions informatiques.

Dans cette thèse, nous étudions des méthodes et développons des outils pour indexer et rechercher des informations multimédia hétérogènes stockées dans des banques de documents XML. Plus précisément, nous abordons la question de la recherche par similarité sur des données composites décrites par des éléments structurels, textuels et séquentiels. En s'appuyant sur la partie structurelle des documents XML, nous avons défini un modèle de représentation, d'indexation et d'interrogation flexible pour des types hétérogènes de données séquentielles. Les principes que nous développons mettent en oeuvre des mécanismes de recherche qui exploitent simultanément les éléments des structures documentaires indexées et les contenus documentaires non structurés. Nous évaluons également l'impact sur la pertinence des résultats retournés par l'introduction de mécanismes d'alignement approximatif des éléments structurels. Nous proposons des algorithmes capables de détecter et de suggérer les « meilleurs points d'entrée » pour accéder directement à l'information recherchée dans un document XML. Finalement, nous étudions l'exploitation d'une architecture matérielle dédiée pour accélérer les traitements les plus coûteux du point de vue de la complexité de notre application de recherche d'information structurée.

Acknowledgments

Many people have accompanied my long journey to this dissertation. To all of them I give my deepest thanks.

I am especially grateful to my thesis committee members: Prof. Mounia Lalmas, Prof. Mohand Boughanem and Senior Researcher Patrick Gros, for accepting to be on my committee and for the feedback they gave on the manuscript.

I owe gratitude to my PhD supervisors, Prof. Pierre-François Marteau and Assist. Prof. Gildas Ménier, for their continuous guidance, support, critical, yet constructive remarks, infinite understanding, and belief in my work. This thesis would not have been written without their constant encouragement.

I would like to thank all the members of the VALORIA laboratory for assuring an enjoyable and fruitful working atmosphere.

I am also grateful to the whole INEX community, for creating the test collection for the evaluation of XML IR which is extensively used in this work.

Last but not the least, I want to thank, Didier Hoareau and Chouki Tibermacine, my work office colleagues for four long years, Alla Silkina, Xiaoqun Zhang and all the other PhD students, for interesting discussions and their friendship.

Finally, I want to thank all my friends and family, for their patience and understanding, and especially my parents and my brother Emil, for his example.

This research was partially supported by the ACIMD – ReMIX (Reconfigurable Memory for Indexing Mass of Data) French grant.

Contents

List of Figures

List of Tables

Part I

Introduction

Chapter 1

Introduction

"It has taken mankind three hundred thousands years to collect twelve exabytes of data. This volume will be doubled within two years."

— *How Much Information* [137]

Contents

1.1 Context and Motivation

Digital data is expanding every day at an increasing rate, both in our business and in our personal activities. This has unprecedented consequences as ”our ability to store and communicate information has far outpaced our ability to search, retrieve and present it” [137].

The time and financial efforts spent to store, manage and organize the ever increasing amount of digital data take a more and more important role in any organization and in our daily activities. The degree to which individuals, rather than organizations, are responsible for generating data has increased. ”We not only have mass production of information, but also the production of information by the masses" [137].

The user have to take control over the deluge of emails, photos, office documents, web and news content. There is a critical need for appropriate tools able to help the users to *look*, *search*, *organize*, *arrange*, *manage*, *control* and *find* relevant information in the growing data chaos.

Our work is placed in the context of *information retrieval* (IR). Information retrieval is the science of searching for information in documents, searching for documents themselves, searching for metadata which describe documents, or searching within databases, whether relational stand-alone databases or hypertextually-networked databases such as the World Wide Web[1].

[1]Wikipedia – The Free Encyclopedia `http://wikipedia.org/`

An information retrieval process begins with a user entering a query into the system. Queries are formal statements of information needs, for example search strings in web search engines. The information retrieval system matches the query with the documents from the collection and returns a ranked list of relevant documents to the query.

Nowadays, the adoption of XML (eXtensible Markup Language) [164] as a standard for the interchange and publication of structured data has created the opportunity to design more focused information retrieval systems. XML documents have a *self-describing* structure. This is a consequence of the W3C specification requirements which states that the structure should be "human-legible and reasonably clear" [164]. Focusing on the text-rich documents published in the XML format, the information retrieval community has enriched query languages for structured documents with text-retrieval concepts like term weighting and relevance ranking. This gave birth to a new research field: *Document Centric XML Retrieval* or *XML Information Retrieval*.

To better use the structural information available in the XML documents, the information retrieval systems must be adapted to take into account both its semantic and logical levels. In long documents such as novels or technical manuals, only a small part of a document may be relevant to the user's query. By making document structure explicit, XML allows information retrieval systems to extract portions of documents. This should "improve the perceived precision" by allowing the user to avoid a complete scan through each result document in order to locate the relevant material [62]. This is the specific context of our work. We address the problem of information retrieval of *document-centric* heterogeneous XML multimedia documents. More precisely we focus on text-rich documents that do not have a fixed structure, but a flexible one, and that may include multimedia data like sequences or time series data.

1.2 Thesis Contribution

In this dissertation we explore methods for managing and searching collections of heterogeneous multimedia XML documents. More precisely, we focus on the flexible searching of structure, text, and sequential data embedded in heterogeneous XML document databases [173]. We also design and evaluate methods for i) approximate matching of structural constraints in an XML IR framework [177, 176, 179, 178], and ii) detecting best entry points for starting to read an XML document [179, 174] on a given topic. Finally, we adapt the proposed approach for approximate search of semi-structured documents on a dedicated hardware memory developed in the context of the ReMIX[2] project [175].

1.3 Thesis Outline

In this section we introduce the structure of the thesis and make a short overview of the content of each chapter.

[2]Reconfigurable Memory for Indexing Mass of Data (ReMIX) `http://www.irisa.fr/remix`

Part One: "Introduction"

- ***Chapter 1: Introduction*** presents the main research subject and shows a brief outline of the structure of the thesis.

Part Two: "XML Information Retrieval"

- ***Chapter 2: XML Information Retrieval*** presents a non-exhaustive review of the state of the art work in XML IR research field. It describes the influence, the main issues and the specificity introduced by the use of the structure of the XML documents all along the indexing and the retrieval processes starting from the query formulation to the results presentation stage.

- ***Chapter 3: SIRIUS XML IR System*** describes our approach to XML information retrieval and its implementation in the SIRIUS[3] prototype developed by the VALORIA laboratory of University of South-Brittany. As its main characteristics SIRIUS merges 1) flexible matching of the XML structure based on an Levenshtein editing distance on the XML tree paths 2) a relevance ranking algorithm for the textual content based on the vector space model and 3) a query enrichment mechanism based on a thesaurus of semantic rules.

- ***Chapter 4: Experimental Evaluation Framework*** is dedicated to the SIRIUS experimental evaluation within the INEX evaluation campaign. We describe in details the INEX evaluation benchmark and show the specific configuration settings used to tune the SIRIUS XML IR system to participate at INEX 2005 and INEX 2006 ad hoc retrieval tasks.

- ***Chapter 5: Approximate Structural Matching for XML IR*** evaluates the influence of the strict and approximate structural matching mechanisms to access relevant information in semi-structured databases.

- ***Chapter 6: Retrieving Best Entry Points in Semi-Structured Documents*** describes a simple, efficient and effective method for finding the *Best Entry Point* (BEP) to start reading an XML document on a given topic

Part Three: "XML Multimedia IR"

XML Multimedia Information Retrieval focuses on methods and techniques for searching multimedia information published as XML documents, and more particularly on sequential data embedded in heterogeneous collections of XML documents.

- ***Chapter 7: IR of Sequential Data in Heterogeneous XML Databases*** introduces our approach for representing, extracting, indexing and retrieving heterogeneous sequential data from collections of heterogeneous XML documents.

Part Four: "XML IR on Specialized Hardware"

To manage, search and analyze important volumes of data within a reasonable response time, scalable/specialized methods and systems are required. The ReMIX

[3]SIRIUS XML IR System `www-valoria.univ-ubs.fr/APRIM/Sirius`

project sustained by the national french initiative ACI *Masse de données*[4] studied, implemented and validated a framework (both hardware and software) dedicated to query large databases.

- ***Chapter 8: ReMIX – Reconfigurable Memory for Indexing Mass of Data*** presents an original hardware memory architecture for both storing very large indexed data structures and allowing fast information retrieval.
- ***Chapter 9: Approximate Search of Semi-Structured Documents Using Dedicated FLASH Memory and FPGA Components*** focuses on the process of designing and implementing an approximate search method for semi-structured documents adapted to the dedicated hardware memory developed in the context of the ReMIX project.

Part Five: "Conclusions"

- ***Chapter 10: Conclusions*** finally closes the thesis by summarizing the main results and contributions and by providing some directions for future work.

[4] ACI Masse de Données `http://acimd.mabri.fr`

Part II

XML Information Retrieval

Chapter 2

XML Information Retrieval

This chapter presents a non-exhaustive review of the state of the art work in XML IR research field. We describe the influence, the main issues and the specificities introduced by the use of the structure of the XML documents all along the indexing and the retrieval processes starting from the query formulation to the results presentation stage.

Contents

2.1 Introduction

Rapid advances in electronics and computer technology have brought in what we call an information age. The exponential growth of the Internet and the Web has flooded us with huge quantities of data in different formats on a wide variety of subjects. To manage this colossal amount of data and extract useful information out of it, we need efficient and effective means of organizing and indexing the data. Though these data are available in different forms and the amount of available multimedia data is rapidly increasing, textual data continues to be a fundamental and very widely prevalent form of storing information.

Not only the volume and the availability of the textual information is increasing, but also the efforts made to organize this information – as denoted by the increasing number of documents published by using the eXtended Markup Language (XML) [164]. One major purpose of the XML markup is the explicit representation of the logical structure of a document.

From an information retrieval point of view, accessing this new type of documents is a challenging task as both the structure and the textual content coexist in the same document. Relying on the structural context should help to better fulfill the users information needs as it allows to retrieve more focused information and, therefore, to increase the precision of the retrieved results. However, along with these uncontested advantages, this also brings in discussion several challenging issues.

This chapter aims to highlight the main challenges related to structured document retrieval and to provide a brief overview of the state of the art solutions used to handle these challenges. In Section 2.1, we introduce preliminary notions like the document, the collection of documents and the characteristics of unstructured, structured and semi-structured data. Next, we present the different document markup types and trace a brief history of the markup languages. An introduction to the XML language follows. We end with the identification of two different views of XML data that lead to two applications fields with distinct requirements: *data-centric* approaches and *document-centric* approaches. In this work, we focus on *document-centric* approaches – which are described further in detail. In Section 2.2, we introduce the conceptual model for XML information retrieval while in Section 2.3 we review previous related works in passage retrieval and Web search. Section 2.4 highlights the main issues and challenges involved in designing XML information retrieval systems and surveys current research approaches to solve them. Section 2.5 focuses on indexing XML documents for XML ranked retrieval while Section 2.6 surveys some of the most effective

models adapted for XML IR. Finally in Section 2.7 we briefly tackle the specific requirements for XML IR evaluation, and conclude the chapter in Section 2.8.

2.1.1 Document and Collection of Documents

At its simplest, a document collection can be any grouping of static flat text-based documents. At its more complex, a document collection becomes a large set of dynamic multimedia hyper-documents, physically distributed on some networking architecture. An illustration of a typical real-world document collection is Wikipedia[1], the biggest multilingual free-content encyclopedia on the Internet.

A document can be defined as a unit of multimedia (generally text) discrete data within a collection. Within the context of a particular document collection a document can be further described based on its representation of a similar class of entities that defines a framework within that collection. Nevertheless, a document can (and generally does) exist in any number or type of collections – from the very formally organized to the very *ad hoc*. Furthermore, a document is potentially an ubiquitous entity as it can be a member of different document collections, or different subsets of the same document collection, and can exist in these different collections at the same time.

2.1.2 Unstructured, Structured and Semi-Structured Documents

Textual information can be broadly classified into three categories based on their structure : (i) unstructured data (ii) structured data and (iii) semi-structured data.

Documents that have relatively little typographical, layout, or markup indicators to denote structure – like most scientific research papers, business reports, legal memorandum, and news stories – are sometimes referred to as free-format, weakly structured or unstructured documents. To the opposite side we have the structured data that refers commonly to data stored in relational databases. On the other hand, documents with extensive and consistent format elements in which field-type metadata can be more easily inferred – such as some e-mail, HTML Web pages, XML documents, PDF files, and word-processing files with heavy document templates or style-sheet constraints – are described as semi-structured documents.

2.1.3 Documents Markup

Markup has its beginnings in the publishing and printing industry where instructions were written on a document to provide the typesetter with instructions as to how things should look. In the electronic world, markup is a code that is included when a document is created, often to provide display instructions, but also to provide meaning or semantics to words or phrases or to provide processing instructions.

The markup code or element is made up of a start and end tag and usually some included text. For instance *and* would make the text between the start and the end markups to appear as italicized[2].

[1] Wikipedia – The Free Encyclopedia `http://wikipedia.org/`

[2] Accordingly to the conventions defined by the HyperText Markup Language (HTML) `http://www.w3.org/MarkUp/MarkUp.html`.

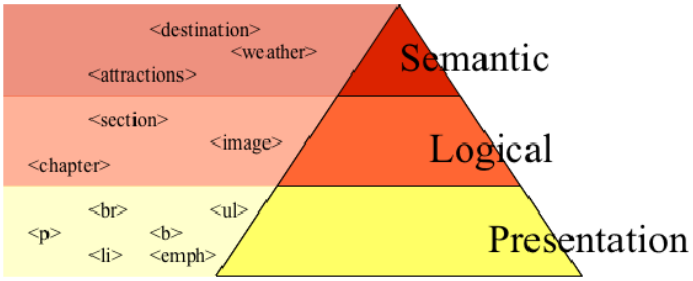


Figure 2.1: Three Layer Structure for XML Document Collections: Semantical, Logical, and Presentation Markup [231].

Presentation, Logical and Semantic Markup. Chiaramella et al. introduced in [47] a three levels taxonomy for the structural markup: *physical*, *logical* and *semantic*.

The Physical Markup of a document defines the way it is presented to the user in terms of page presentation, font sizes and so on. It may be assimilated with an external view of the document.

The Logical Markup express the way a document is logically organized and reflects the discourse structure of the author(s). For instance a document contains a title, then a chapter having a title and several subchapters and so on, each of these parts containing an element of the discourse and having its own internal organization. The logical structure may be seen as an internal view of the document structure.

The Semantic Markup is defined relatively to its meaning for the author of the document or for a specific reader. It corresponds to an organization of the underlying knowledge that constitutes the document content.

More recently, van Zwol et al. [231] used an equivalent three levels classification for the markup of the XML documents: *presentation*, *logical* and *semantic* tags. (see Figure 2.1). The authors used this classification as a basis for a visual query formulation technique for XML retrieval called *Bricks*. Bricks exploits this three layer structure in the retrieval process by adding a priority to each class of structural elements. Semantical elements will receive a high priority, followed by the logical elements, while the presentation elements are given a low priority. Their results showed that this approach may reduce the complexity of the query formulation process and the required knowledge of the underlying document structure for the user.

One difficult problem is to automatically detect the class of each XML tag in order to be used in the information retrieval process. The same tag may have different meanings in different contexts. In an heterogeneous environment, the XML tags semantic/class may be dependent of the specific collection.

Recent approaches [213, 215] use natural language processing techniques and collection statistics to automatically detect between *soft*[3], *hard*[4] and *jump*[5] tags. These approaches are dependent on the collection characteristics and are not yet extended to manage heterogeneous collections of XML documents.

Identifying the tags class may help to determinate if a particular structural element is meaningful for the final user. It may also help in filtering or transforming the structure of the XML documents into a logical and semantically valid structure. Under this hypothesis, using the structural information should bring a positive contribution to the information retrieval process.

In [233] the authors plead for a more meaningful and semantic structure in XML documents. They indicate how this can be achieved by adding additional semantic tags using named entity recognition techniques. They show that XML retrieval systems can effectively exploit the additional structure as it allows for identifying and retrieving more relevant XML elements in the top of the ranking. This is especially true, when the users specify semantically structured clues in the information request on the semantic collection.

2.1.4 A Brief History of Markup Languages

Modern markup languages had their beginnings at a government meeting in Ottawa in 1967 when an American, William Tunnicliffe, made a presentation titled *The Separation of Information Content of Documents from their Format*. A project named *GenCode* stemmed from this work and was expanded upon by a group at IBM that developed *Generalized Markup Language* (GML) in 1969. *Standard Generalized Markup Language* (SGML) was the next iteration on GML and was accepted as an ISO standard in 1986[6].

SGML is a metalanguage for tagging text defining rules for a markup language based on tags. Each instance of SGML includes a description of the document structure called a document type definition. Hence, an SGML document is defined by a description of the structure of the document and the text itself marked with tags which describe the structure. SGML is the precursor of all markup languages, but it is more complicated and more powerful than needed for most applications.

In 1992, the *World Wide Web Consortium* (W3C)[7] propose the *HyperText Markup Language* (HTML). HTML is a simple language based on SGML using predefined tags specified in the HTML Document Type Definition. The HTML language allows to format the way a document is presented into a web browser and facilitates the creation of hyperlinks between documents. Most documents on the web are stored and transmitted in HTML format. One of the main critics brought to the HTML language is that it mixes the documents content with the presentation layer. This makes the updates difficult and reduces its usability as a data exchange format.

[3] *Soft tags* identify significant parts of a text, like quotations, appearance effects, but become "transparent" while reading the text – i.e. presentation markup.

[4] *Hard tags* interrupt the "linearity" of a text and contribute to the structuring of the document. Examples of this type are *'titles*', *'chapters'* and *'paragraphs'* – i.e. logical markup.

[5] *Jump tags* are used to represent particular elements, like margin notes, references to bibliography, or glosses. They are detached from the surrounding text. For example the elements *'comment*' and '*footnote*' may be considered as jump tags.

[6] See A History of Markup Languages `http://careo.prn.bc.ca/losc/mod1t1.html`

[7] The World Wide Web Consortium (W3C) `http://www.w3.org/`

The complexity of the Web has grown significantly since that time and the need for a more powerful markup language to provide structure and meaning within documents, and to facilitate the exchange of data, became apparent. A subset of SGML which should be less complicated than SGML, but more powerful than HTML was seen as a promising candidate. The result was *eXtensible Markup Language* or XML, which became a W3C specification in 1998.

2.1.5 eXtensible Markup Language (XML)

The *Extensible Markup Language* (XML) is a general-purpose markup language. It is classified as an extensible language because it allows its users to define their own tags. Its primary purpose is to facilitate the sharing of structured data across different information systems, particularly via the Internet. It is used both to encode documents and serialize data.

It started as a simplified subset of the *Standard Generalized Markup Language* (SGML), and is designed to be relatively human-legible. By adding semantic constraints, application languages can be implemented in XML. These include XHTML, RSS, MathML, GraphML, Scalable Vector Graphics (SVG) , MusicXML, and many others. Moreover, XML is sometimes used as the specification language for such application languages.

XML does not have many of the restrictions imposed by HTML but on the other hand imposes a more rigid syntax on the markup, which becomes important at processing time. Each document must have a single root element. An element begins with a start-tag and ends with an end-tag enclosed in angle brackets. Both tags must have the same name. In XML, ending tags cannot be omitted. Also, tags for elements that do not have any content, like BR and IMG, are specially marked by a slash before closing angle bracket. All elements and text passages between the start-tag and the end-tag of an element make up its content. An element can have certain attributes that are listed within its start-tag. An attribute is a pair consisting of a name and a value. In Figure 2.2 we show an example of a document marked up in XML . The document starts with a preamble that specifies the version of the XML specification and the name of the character set used within the document. The preamble is followed by the start-tag of the root element *document*. All other elements are nested within that root element. The names for elements and attributes may be freely chosen by the users. Optionally, a Document Type Definition (DTD) or a schema language such as XSD or Schematron can be used to formalize the relationships between elements.

2.1.6 Data-Centric vs. Document-Centric XML Data

Making the distinction between data-centric and document-centric XML data is important in order to settle the context in which the work presented in this thesis is conducted.

Data-centric XML documents are generally for machine-to-machine inter-change of machine-readable data. It is characterized as being highly structured.

Document-centric XML documents are generally human readable, and may contain some markup to help a person understand the text (e.g., markup indicating italics or footnotes).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<document>
   <title> Slant </title>
   <author> G.Bear </author>
   <text>
      <chapter num="1" >
         Omphalos dominates ...
      </chapter>
      <chapter num="2" >
         Alice Grale believes ...
      </chapter>
   </text>
</document>
```

Figure 2.2: Example of a document marked up in XML.

This distinction is important, as generally, the techniques for processing and querying XML vary depending on its usage; more structured XML documents rely more on its structure to relay or filter information, whereas document-centric XML relies more on content (i.e., the values of data).

In document centric XML retrieval the queries are user information needs that vaguely specify the context of the desired answers. This is different from well-structured XML queries where one tightly specifies what he/she is looking for using a very constraining query language.

2.2 Conceptual Model for XML IR

A standard information retrieval process begins by a user entering a query into the system. Queries are formal statements of information needs, for example search strings in web search engines. In information retrieval a query does not uniquely identify a single object in the collection. Instead, several objects may match the query, perhaps with different degrees of relevancy. To manage this problem, most IR systems compute a numeric score on how well each object in the database match the query, and rank the objects according to this value. The top ranking objects are then shown to the user. The process may then be iterated if the user wishes to refine the query.

Traditional IR uses this paradigm to find relevant documents, e.g. entire books, to a user's information need – in this case the indexed objects are indivisible.

In *structured document retrieval* (SDR) the documents structure is exploited to identify which document components to retrieve. The motivation is that the structure of the documents may improve the search precision by exploiting the visual memory of the users. Therefore *s*tructured document retrieval[8] allows the users to retrieve document components that are more focused to their information need, e.g. a chapter,

[8] *{Structured | Semi-Structured | XML} {Information | Document} Retrieval* are used interchangeably in this thesis to denote the task of retrieving *relevant information* from a collections of semi-structured documents as opposed to *Data Retrieval* which describes the task of accessing the data, usually as database records.

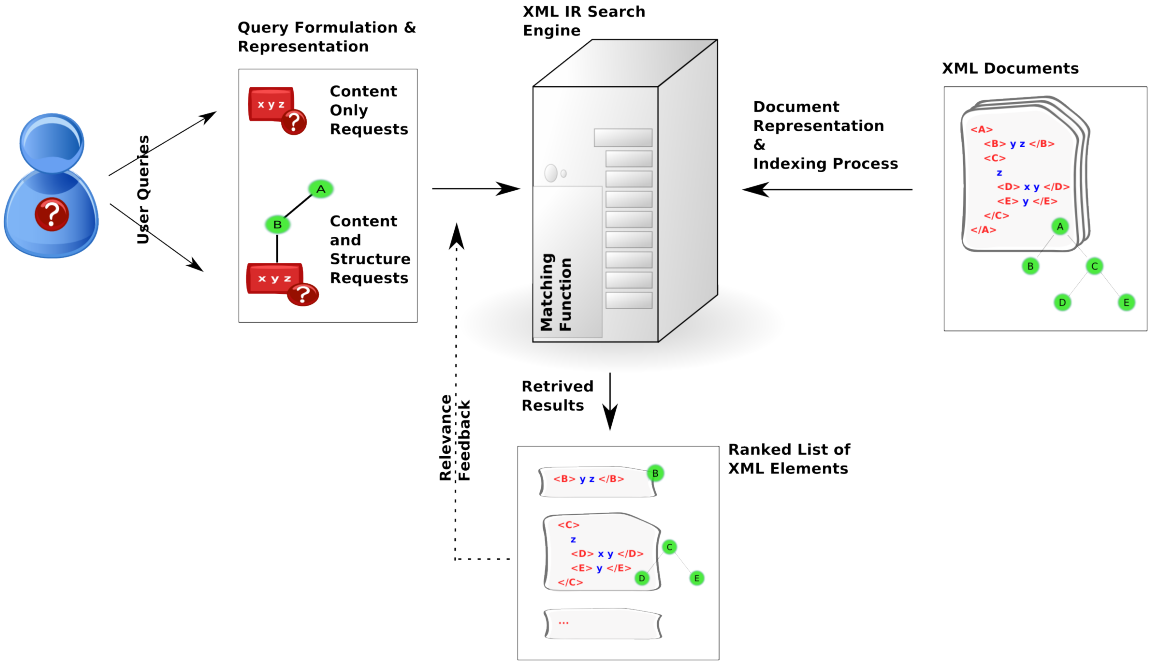


Figure 2.3: Conceptual Model for XML Information Retrieval.

a section, a page, several paragraphs of a book, instead of an entire book – here also fragments of the indexed objects may be retrieved [10, 41].

In particularly, the XML language is becoming an ubiquitous format for information interchange over the Internet due to its powerful structured representation for storing information. In contrast to HTML, which is mainly layout-oriented, XML follows the fundamental concept of separating the logical structure of a document from its layout. XML thus offers the opportunity to exploit the logical structure of documents to allow more precise searching [122].

Indexing, Searching and Ranking XML Documents. Previously, XML documents were *data-centered* and used to store rich-structured data which did not contain much amount of text. But now-a-days, *document-centric* XML documents are being used to store information in the form of structures as well as text. We concentrate on such XML documents from an information retrieval point of view. That is, we study and emphasize problems related to the effective and – in a less measure – efficient access to XML documents repositories.

An *XML information retrieval* system retrieves document components (i.e. XML elements) rather than whole documents in response to a user's query by exploting the logical structure of the documents. This process is illustrated in Figure 2.3.

Extracting information from such documents requires that a query contains both textual keywords as well as structured information. Therefore, the index structures for such XML collections must be adapted to index the structured parts as well as the textual parts of the documents. This may be done at different granularity levels that will directly affect the systems ability to answer focused requests.

Ranking methods must sort the elements so as to return the most relevant results to the given query. They must take into account the keywords relevance, the structural

similarity, the unit of result, the nested information and the global relevance of the documents .

The XML query processing algorithms must be efficient. The most important performance-enhancing factors are the existence of appropriate index structures designed for faster content and structure query processing and top-k query optimization techniques.

2.3 Precursors of XML IR

Long before the emergence of the XML language, studies related to the granularity of the retrieved information for a more focused document access have been conducted. Alternative and earlier technology exist for identifying relevant parts of documents [221] – i.e. *passage retrieval* [38, 99, 105, 194].

Second, works that identify means to exploit the fixed structure of the documents like the weblinks and HTML tags to improve the retrieval effectiveness were proposed in the context of Web information retrieval. Even if the structure of the XML documents is not static and these approaches can not be applied directly to XML information retrieval [33], these may be considered as its precursors [198, 206].

Next, we present early approaches for passage and Web information retrieval as surveyed by Sauvagnat [198] and Sigurbjörnsson [206].

2.3.1 Passage Retrieval

In the area of classical document retrieval, passage retrieval [38, 99, 105, 194] can be seen as a document retrieval extension of a rudimentary data retrieval approach. The aim of passage retrieval is to increase the precision of the retrieved answers and to reduce the cognitive load on the user by filtering relevant from irrelevant content within a document. However, passage retrieval lacks the structural dynamic contexts to which ranking is applied in semi-structured retrieval [33].

Moffat et al. [152] investigate approaches for passage retrieval starting from arbitrary divisions of running text into pieces of similar length, or by taking the document structure into consideration – i.e. the documents are split into sections based on their internal markup. Their whole set of experimental results are difficult to interpret but show that section and document retrieval can improve retrieval performance.

Starting from the observation that most text items are naturally subdividable into recognizable units, such as text sections, paragraphs, and sentences, –Salton et al. [194] evaluate paragraph and section passage retrieval. They use similarity at sentence level for local matching. They provide experiments on an encyclopedia collection and show that section and paragraph retrieval can improve document retrieval performance.

Callan [38] describes experiments with paragraph-based and window-based methods of defining passages. In their experiments, passages based upon paragraph boundaries were less efective than passages based upon overlapping text windows of varying sizes. This result held for both document retrieval based on a single best passage, and document retrieval based upon combining document-level and passage-level evidence. The experimental results also showed that, for document ranking, the combination of passage and document-level evidence was more effective than using passage-level evidence alone.

Kaszkiel and Zobel [105] explored several passage retrieval approaches: fixed-length passages, variable-length passages, discourse structures (e.g., sections and paragraphs), and text tiles[9] [88]. The authors show how passage-level evidence can be used to retrieve documents and compare the effectiveness of the different approaches. In their experiments the fixed-length passages obtained the best results.

Liu and Croft [133] explore the use of generative language models and relevance models to improve document retrieval by using passage retrieval. They show that in the case in which the documents are long and span several topics, the passage retrieval can significantly outperform document retrieval.

More recently, Jiang and Zhai [99] present a method for detecting coherent relevant passages of variable lengths using hidden Markov models (HMMs). The HMM-based method naturally captures the topical boundaries between passages relevant and non relevant to the query. Pseudo-feedback mechanisms can be naturally incorporated into such an HMM-based framework to improve parameter estimation. They show how the HMM method can be applied on top of any basic passage extraction method to improve passage boundaries detection.

As a conclusion, retrieving relevant passages as opposed to whole documents improves retrieval accuracy in general. This technique may be particularly useful when the documents are long and lack the semantic mark-up [221].

Passage vs. Element Retrieval

Although, at first sight, XML retrieval and passage retrieval appear quite different, they share much in common. As with traditional document-centric information retrieval, the user need is loose, linguistic variations are frequent, and answers are a ranked list of relevant results. Furthermore, in focused retrieval, the size of the unit of information retrieved is variable and results within a single document may naturally overlap [222].

In case of marked up documents, the explicit divisions of the document can be used as retrieval units. For semi-structured data such as XML, the retrieval systems have at their disposal an explicit set of logical units, e.g. paragraphs, sections – that can be used as potential answers. In element retrieval the search engine is expected to identify not only which documents are relevant, but also which structures (or elements) within those documents are relevant to an information need. In *XML element retrieval,* the retrieval units are set exclusively to marked-up XML elements. This is the main difference to other focused tasks such as passage retrieval [38, 99, 105, 194] where the systems can freely choose the best retrieval unit [40].

One of the current research trend in XML IR community concerns the relations between passage retrieval and element retrieval in the context of answering an XML retrieval task. Even if at first view XML retrieval is mostly based on element retrieval, experimental studies of the users relevance judgments show that the relation between passage and element retrieval is not a trivial one.

The need for returning ”range results in XML retrieval” rather then single XML elements is first introduced and discussed by Clarke in [48]. The author analyzed the

[9]TextTiling [88] is a technique for subdividing texts into multi-paragraph units that represent passages, or subtopics. To detect a passage, for each phrase, it computes the phrase similarity with its k previous and k next phrases. A rapid change of the similarity between phrases should point a thematic change and delimit a new passage.

INEX 2004 ad hoc task relevance judgments, e.g. based on the IEEE collection[10], and found that out of the 5229 elements judged as highly *exhaustive* and *specific* [11], at least 1700 (32%) are part of a larger range of elements with identical tag names. Since then, more recent studies tackled the relationships between the XML element retrieval and passage retrieval [101, 97, 221, 21, 20].

Itakura and Clarke [97] believe that passages (or elements derived from passages) are better results than XML elements. They propose ranking every possible passage in a document and converting them into elements for comparison to other element retrieval systems. To go from passages to elements either additional content must be added, or some content must be lost. By analyzing the relevance results of 39 INEX 2005 topics on the INEX IEEE collection, they found that element retrieval out performed passage retrieval. Their conclusion was that, the best elements to return are those that best fit the user expected results. Since the results tend to be passages it may be necessary to return multiple consecutive elements to cover a passage.

In [101], Kamps and Koolen analyzed the the INEX 2006 assessments on the Wikipedia collection to find how the relevant text inside a document (i.e. highlighted passages) relates to the document structure (i.e. XML elements). They find that relevant passages may span a range of elements. They specifically look at the relation between passages and container elements – i.e. the shortest XML element containing the whole passage. Container elements are twice as long as the average passage. They show that passages tend to start at the container element's start but passages ended somewhat earlier than the container element. On the other hand, half of the passages have a closely fitting container element (the passage covers 95-100% of the element). Best fitting container elements were paragraphs, sections, list-items, titles and the whole articles. The authors conclude that as half of the passages fit closely with an XML element this seems to support retrieving XML elements, but the fact that the corresponding elements are twice the length of the relevant passage seems to support passages results.

Overall, as the start of a relevant passage coincides with the start of an element, and assuming that the results are displayed in the context of the article (i.e. the users presumably start reading at the beginning of the element, and returning trailing non-relevant text is not penalized) retrieval of XML elements seems a good approach [101].

2.3.2 Web Information Retrieval

Using the Web Link Structure

The web links carry implicit useful information about the relevance of the web pages they point to. This can provide a mean to rank and filter the web pages. For instance a link from page A to page B may be considered in most cases as a recommendation of the page B by the author of the page A. In addition, even if the primary aim of the links is to provide a mean to navigate through the structure of a website, they can also be seen as indications of semantic proximity between web pages [44].

A number of works exists that propose ways to use the hyperlink structure between documents to improve retrieval effectiveness. We briefly survey here some of the most

[10] See a description of the IEEE collection in section 4.2.1 on page 63.

[11] See a description of the two different dimensions: *exhaustivity* and *specificity* used to judge the pertinence of the result elements in section 4.2.3 on page 69.

common Web IR methods for analyzing the hyperlink structure: *Page Rank* [34, 162] and HITS *(Hypertext Induced Topic Search)* [112].

Brin and Page [34] use the notion of popularity propagation as the base of their *PageRank* algorithm successfully implemented into the Google search engine. PageRank uses the hyperlink structure of the Web to view inlinks into a page as a recommendation of that page from the author of the inlinking page. However, inlinks from good pages (highly revered authors) should carry much more weight than inlinks from marginal pages. Each web page can be assigned an appropriate PageRank score, which measures the importance of that page.

Another popular approach is the Kleinberg's HITS [112] algorithm. The HITS algorithm improves the popularity propagation by introducing the notion of page relevance within the method used to compute the web search results relevance. The HITS IR method defines authorities and hubs. Authority value estimates the value of the content of the page (inlinks); hub value estimates the value of its links to other pages (outlinks). The HITS thesis is that good hubs point to good authorities and good authorities are pointed to by good hubs. HITS assigns both a hub score and authority score to each web page. To compute the relevance of pages to the users query, HITS is applied at querying and not at indexing time as opposed to the *PageRank* algorithm.

In Web information retrieval [35], link analysis has been successfully applied to different tasks such as pages ranking, pages crawling, pages classification/categorization and as source of evidence for detecting related or duplicated pages. A more detailed description of current link analysis algorithms can be found in [123].

Using the Structural Markup

In [244], Wilkinson uses a structured collection with assessments at element level to compare document retrieval and element retrieval. It shows that element retrieval can be used to improve the overall performances of document retrieval. He also shows that ranking documents first, and then selecting the elements to retrieve is not a good strategy. Finally, the mixture of local and global evidence is proposed as an effective strategy to obtain better results for both document and element retrieval.

In [153], Myaeng et al. propose a retrieval model based on inference networks that allows to retrieve parts of SGML documents. The appropriate unit of retrieval is defined by the user as a structural constraint. Their experiments show that the system can handle a variety of structural queries and that both element-based passage retrieval and specific weights associated to elements-type can ameliorate the retrieval effectiveness at document level.

In [31], Bordogna and Pasi propose a flexible query language for Web pages that allows to express both conditions on the documents structure and conditions on the topics of interest. The first condition acts as a soft filter so as to reduce the set of documents on which to evaluate the second condition. The main idea is to use the logical structure of the documents when computing the weights of the indexed terms. First, the terms are indexed according to the documents sections. Each section is associated with a fuzzy membership function, and the terms weights in the main sections are computed by using aggregate functions. Flexibility is achieved by allowing users to specify both a linguistic quantifier such as 'most' for qualifying the global composition of the documents into a number of sections and preferences on the desired documents sections.

2.4 XML IR Challenges

Although the structure of the XML documents can be used as a backbone to answer more specific and focused results – i.e. that contain the information sought by the users and the least amount of irrelevant information – deciding which level of the tree is *the right unit of retrieval* (for instance the whole book, a chapter, a section, a paragraph, a table, a figure, etc.) for answering a query with regard to both content and structure represents a challenging problem [16].

In this section, we discuss the main challenges associated with scoring and returning XML documents elements as answers for user's focused information needs. We also briefly illustrate how these challenges have been or are currently being addressed by the XML IR research community as surveyed by Amer-Yahia and Lalmas [16], Baeza-Yates and Lalmas [24] and Amer-Yahia et al. [10].

2.4.1 Term and element statistics

Most retrieval approaches rely on term distribution statistics to determine the relevance of a document. The most common measures being term frequency, *tf*, and document frequency *df*. *tf* is the number of occurrences of a term in a document and *df* is the number of documents in which the term appears. In XML retrieval the indexed statistics must be adapted to the granularity level of the indexing and retrieval units. The problem arise when one needs to compute the importance of a term within an element, since elements in XML documents are nested. The importance of a term within an element is dependent on the importance of the term within the elements composing that element or even of the term importance within the neighboring elements [110]. For instance, suppose that a section element is composed of two paragraph elements. How should we compute the element term frequency, *etf* and element frequency *ef* values of a term in the paragraphs and the section?

One approach in [142] is to compute statistics for predefined element types specified at indexing time independently. This avoids problems arising from nested elements as elements of different types are treated separately. – see Section 2.5.5 for a more comprehensive description of this approach.

Another approaches [77, 201, 234] gather term statistics only for the leaf-elements. The scores are then propagated in the XML tree to compute the relevance scores of the inner-nodes. To this end various propagation weights dependent of the tree distance between the current node and its leaf-elements descendants may be used. The aim is to gradually attenuate the contribution of the leaf-elements to the score of their ancestors.

Other approaches consider all the elements of the collection [207], or classes of elements of the same type [218] to compute the terms statistics. When computing the element term frequency, the textual content of the current node and of all its descendants are usually merged [157, 218]. Finally document frequency is used in [50].

2.4.2 Structure Statistics

In a collection of XML (heterogeneous) documents we have generally a large number of XML tags. Not all elements are equally relevant to an information need. How can we find which type of element is a satisfactory answer? The structure in XML documents

should be used to decide which elements are likely to produce higher user satisfaction. This is usually done by computing some statistics on the structure itself [16].

One approach [142, 50] is to index and/or retrieve only those element types with the highest distribution of relevant elements in (available) relevance assessment sets. For instance, section, abstract, sub-section, and paragraph element types are assessed as relevant elements for a test collection composed of a subset of IEEE Computer-Society scientific articles.

Another approach is to take the element length into account. This can be done either by using a fixed threshold (usually expressed in terms of number of words) either by using the average length of the elements [207].

Both strategies discard from the result list elements which are often considered to be "too small" to act as a meaningful retrieval unit. For instance presentation tags such as italics that delimit only a few words. This has to be done carefully, as [201] showed, that even if the small elements should not be returned to the users, they might still influence the scoring of their enclosing elements.

2.4.3 Relationships Statistics

The logical structure of the XML markup describes explicit relationships between the XML documents components. In the XML tree, a node may have a parent, children, ancestors, descendants, preceding and following nodes. This rich structural relationships information should be exploited when computing the XML elements relevance scores. The challenge is than to determine how to estimate (or learn) relevant relationships statistics and what parameters to use (e.g. size, number of children, depth, distance, the relevance of the root element, the relevance of the previous/following or surrounding nodes).

A simple but effective method is to integrate the global relevance of the article (i.e. the relevance of the root element) in the XML elements score computation. Several approaches (e.g. pivot [143], root-based contextualization [19] or article level language model [207]) obtained significant improvements when using this particular relationship.

The structural organization of the collection – i.e. the structural relationships between the documents, like their distributions in different folders or sub-collections – can also be used to increase the retrieval performances [183].

In [184] the structural relationships between the elements returned by the retrieval system and highly relevant elements from the assessment set are analyzed. The discovered links are introduced in the retrieval model as virtual propagation/reinforcement paths between specific elements types and used to improve the ranking.

Some approaches, like those based on Bayesian networks [170, 235], or hierarchical language modeling [157] can capture the strength of the structural relationships through learning.

These models need extensive training, which is computationally expensive. A simplified hierarchical language model that reduces the parameter estimation complexity was proposed in [158].

Other approaches propagate terms statistics in the XML tree by using propagation weights that depend on the elements relationships [77, 78, 201, 75]. Several propagation methods are described more in details in the next paragraphs but all share the characteristic that the parameters used to weight the different elements relationships

are usually empirically determined, which consist by itself a rather difficult task.

2.4.4 Relevance Propagation

In the case where the XML nodes are considered as *disjoint-nodes* – see Section 2.5.5 – techniques able to aggregate term statistics and to propagate the relevance of the leaf-nodes to the upper-levels in the XML tree are required. This may raise several questions: How to aggregate term and/or relationship statistics? Which sub-element(s) contribute best to the content of its parent element and vice versa? How to propagate scores between relevant elements? Is there any influence between sibling or neighboring elements?

In the HyRex system [78], Govert and al. introduce an *augmentation method* for dealing with XML documents. In their approach, standard term weighting formulas are used to index the so called *index nodes* of the document. Index nodes are not necessarily leaf nodes, because this structure is considered to be too fine-grained. However, index nodes are disjoint. In order to allow the nesting of nodes, in case of high-level index nodes comprising other index nodes, only the text that is not contained within the other index nodes is indexed. For computing the indexing weights of inner nodes, the weights from the most specific index-nodes are propagated towards the inner nodes. During propagation, however, the weights are down-weighted by multiplying them with a so-called *augmentation factor*. In case where a term at an inner node receives propagated weights from several leaves, the overall term weight is calculated by assuming a probabilistic disjunction of the leaf term weights. This way, most specific elements are preferred during retrieval.

The approach applied in the XFIRM system [199, 198] is also based on an augmentation method. However, in theirs approach, all leaf nodes are indexed, because the authors start from the assumption that even the smallest leaf node may contain relevant information. The propagation of relevance values in the document tree is function of the distance that separates the nodes in the tree. Furthermore, each node in the document tree is assigned a relevance value which is function of the relevance values of the leaf nodes it contains. Nodes containing a larger number of relevant leaf-nodes are favored in the final ranking. Terms that occur close to the root of a given sub-tree are considered to be more significant for the root element then terms occurring at deeper levels of the sub-trees. Also the nodes containing a larger number of relevant leaf-nodes are favored in the final ranking.

The *GPX* [76, 77], the $B^3 - SDR$ [229] and the *XSee* [234] systems use an approach based on the construction of a collection sub-tree that consists of all elements (nodes) containing one or more of the query terms. Leaf nodes are assigned a score using a simple $tf \cdot idf$ variant, and scores are propagated upwards in the document XML tree, so that all ancestor XML elements are ranked. The approach implemented in the GPX system, even if "heuristic-based" [77], proved to be versatile and to consistently produce good performance results for a variety of retrieval tasks. For this reason, [229, 234] extended the GPX system approach to reward elements that partly fulfill the structural constraints of the information needs and to penalize those elements that contain excessive elements in their path.

Yet, another approach exist. The so-called *reading paths* [75] are used to propagate information – i.e. relevance – from each section to the following section, where the importance of a section decrease with the distance between sections. This conforms

with the human reading memory hypothesis. That is, generally a reader is assumed to read the sections of a document in their appearing order. Therefore, the information that is read at the beginning of the reading path has more importance than the others, considering that it is used afterward as reading memory. Thus far, the evaluation results for this approach were not conclusive.

2.4.5 Overlapping Elements – Removing Nested Redundant Information

In an XML retrieval setting, to identify the most appropriate elements to return to the user is not an easy task. The system has to find out which are the most *exhaustive* – i.e. highly relevant – and *specific* – i.e. not discussing unrelated topics – elements in the tree. Given an XML document collection, a structured document retrieval system could present to the user any of the marked up elements. The possible retrieval units vary widely in size and type; from small elements like italics and section titles to large elements like document bodies or whole articles.

Due to the inherent nested structure of the XML documents – i.e. logical markup (paragraph, section, article) overlaps with presentation markup (italics, emphasize) – the result lists may contain a large number of overlapping elements. This is because when a specific element contain the query terms, all of its ancestors will contain them too. Thus all elements in the same path that contain that element may be relevant (with different degrees) to the query. This is known as the *overlap problem* [109].

Presenting a list of overlapping XML element retrieval results to a user is not a reasonable thing to do. Empirical evaluation has shown that users are irritated by receiving a ranked list of overlapping elements [220]. It is the task of the system to decide which units are the most suitable in order to produce results lists without overlapping elements.

Several types of approaches have been applied to remove the redundant nested information from the results list. Common approaches return recursively the highest scored element from each path [77, 201] as long as it does not overlap with an already selected result, or discard elements of a given structural type [235]. In these approaches, the information retrieval systems rely completely on the underlying retrieval models to produce the best rankings. Thus, the assumption is that the most appropriate element in a path has been assigned a higher score than the rest. The structure is not considered when selecting the nonoverlapping elements.

In [50, 143], the structural relationships between elements are taken into account when removing the overlapping information. [50] iteratively adjust the score of those elements contained in or containing higher ranked elements in the result list while in [143] the elements are selected by using heuristics based on the distribution in the tree structure of the relevant elements.

Another approach [151] merges the relevance score of the elements, their size and the amount of irrelevant information contained by their children elements into an *utility function*. An element whose utility value is higher to that of the sum of the utility value of its children is selected as best element, otherwise, the children elements whose utility values are above some threshold are selected.

2.4.6 Structure Constraints

Specifying an information need including both content and structural constraints is not an easy task for an end user, in particular for semi-structured data with a wide variety of tag names. This is a valid assumption in the context of heterogeneous XML collections as the ones currently available on the Web.

Even if both strict and vague interpretations of the structural constraints were proposed [223], the vague interpretation is prefered as considered most suited to the user's both, information needs, and its own knowledge of the collection structural organization. The motivation for this states in the fact that "although the users may think they have a clear idea of the structural properties of the collection, there are likely to be aspects to which they are unaware" [16]. In this case, even if the result elements do not exactly meet the structural conditions expressed in the query, they must be returned if they can satisfy the user information need .

To answer structural constraints in a vague manner, a common and effective approach is to to manually build a dictionary of tag synonyms [143, 201, 200].

A more extreme case is to simply ignore the structural constraints in either the support elements, the target elements, or both [19]. More details about the current approximate structure matching schemes for vague interpretation of structural requests can be found in Section 3.1.3 while a survey of automatic schema matching approaches is available in [181].

2.4.7 XML IR Interfaces

Results Presentation

The presentation of results may also pose a number of problems. For multiple relevant fragments, possibly originating from different documents, an unified visualization method must be found. Since several elements may belong to the same document some results may contain others and the presentation as a simple ranked list may not be appropriate. For a single result element one has to decide whether the element should be shown out of context, or within the context of the document it belongs to. In the latter case, one has to decide how this context should be displayed [66].

Up to now, few approaches tackled this issues. Among them we may cite the HyGATE interface of the HyREX IR system [81] that uses a treemap-based visualization method to present the document structure and *TextBars* – an extension of Hearts's *TileBars* [87] – to present the relative relevance of elements within in a document. HyGATE use the so-called *Partial Treemaps* which display results by omitting nodes that are not a retrieved node or an ancestor of a retrieved node. Similar with the treemap concept, a *DocBall* is used in [53] to represent the structure of a single structured document.

Daffodil [67] and the XMLFind systems [104, 209, 206] display lists of ranked XML fragments. The XML fragments are clustered per article and presented in their original document order. The relevance in the XML tree is shown by using a heat-map [209] while a table of contents provides easy access to different document granularity levels.

We cited here only interfaces that were specifically designed for XML retrieval. For a brief presentation of information retrieval interfaces providing access to sub-document-level results we refer to [206, page 143]. Additionally, INEX Interactive track investigates users behaviors when interacting with elements of XML documents.

An overview can be found in [220].

Query Formulation

The presence of the structure in XML documents allowed more complex and precise searching paradigms. Nonetheless, this enhancement brought a new burden to the end-user as the structured query languages such as XQuery and XPath became more and more complex. These query languages are rather intended for experts and not for the end-users.

To help the user formulate their requests, a query language usually based on a subset of XPath navigational axes and standard IR keywords operators may be used – e.g. NEXI [226]. Also, some approaches use form-based interfaces that provide hints about the presentation, logical or semantic relevance classes of the different structural tags from the collection [231]. A rather different approach is taken by the systems that allow the users to specify their content and structure information needs using the natural language [214, 247].

In this section we have presented some of the main issues and challenges arising when designing XML information retrieval systems. We have also presented some approaches to manage these challenges, mainly the ones introduced within the INEX initiative for the evaluation of XML retrieval [64, 68, 70, 69, 71].

2.5 Indexing XML Documents

The indexing process allows to extract the search keys or terms out of the document collection. For flat, unstructured documents, the textual content is processed in order to find and weight the most significant terms in the collection. For XML documents, the structure has to be taken into account. This process raises several research questions: i) What is the influence of the structure on the terms statistics and weights used in the retrieval models? ii) What is the appropriate granularity for an indexing unit? iii) How do we index the structural features of the XML documents? iv) How do we represent and preserve the nested hierarchical relations between the indexed elements? v) What kind of indexing structures are the most appropriate to index both content and structure for relevance ranking?

2.5.1 Indexing Process

The straight sequential search strategy may not be efficient enough to access vast amounts of information within acceptable response times. To solve this access problem an *index* may be created to improve the access time and/or to ensure a different or more complex access patterns to the sought piece of information. The goal of storing an index is to optimize the speed and performance of finding relevant documents to a search query. Without an index the search engine would scan every document in the corpus, which would take a considerable amount of time and computing power. The trade offs for the time saved during information retrieval are the additional computer storage required to store the index and a considerable increase in the time required for an update of the index to take place.

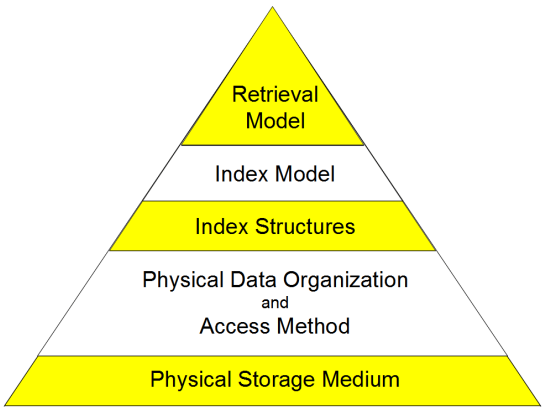


Figure 2.4: Index Layers.

2.5.2 Index Layers

An *index* is intimately related with the *retrieval model* used to access the data since the later is based on information produced by the former. This relation concerns both the index capability to answer correctly to the search operators defined by the retrieval model and space-time optimizations for frequent access/update patterns.

A retrieval system efficiency is determined by several different layers: the retrieval model, the index model, the index structures, the physical data organization and access method, and the physical storage medium (i.e. hard-disk, RAM memory, FLASH disk, etc) – see Figure 2.4. None of these layers are independent since each model or layer is constrained and/or optimized by/for the specificity and characteristics of the underneath layers.

Conversely, a designer aiming to optimize a specific retrieval model had to optimize the layers stack from top to bottom, down to the physical storage medium. For instance, an attempt to use specialized hardware for XML IR is presented in Chapter 9. This case is not common as the design and implementation costs are high.

2.5.3 Indexing Term Weights

In classical document retrieval, *indexing* is the process in which a document collection is parsed, assigning an identifier to each document and keeping track of the number of times each term is observed within a document (*tf* or term frequency). Other parameters of statistical value like the document length or number of unique words per document are extracted in this step as well.

Most retrieval approaches rely on term distribution statistics to determine the relevance of a document. A term is a word or number found within the text of a document, or just the *stem* of a word. Stemming words and thus, mapping multiple different words into a single stem has proven useful to compensate for semantically equivalent, but syntactically different word forms. One common practice in information retrieval is to extract the stems from the indexed terms by applying the Porter[12] stemming algorithm [180].

Term distributions and in particular values such as the inverse document frequency *idf* may vary widely throughout a document collection. Hence, the significance

[12]The Porter Stemming Algorithm `http://tartarus.org/~martin/PorterStemmer/`

of a term within a document and thus, its relevance, highly depends on the context. In most existing retrieval approaches, the context is always the complete document collection. For example, the *idf* is a static value determined when the document collection is indexed.

In XML retrieval the indexed statistics must be adapted to the granularity level of the indexing and retrieval units. Among the different approaches, computing and indexing term statistics per element, selected type/category of elements, document and collection levels is a common design decision. We have presented several approaches for computing term statistics adapted to XML IR in Section 2.4.1.

2.5.4 Index Structures

The standard index format for storing term frequencies is the *inverted file* format [245]. An inverted file is an indexed sequential file that maps term identifiers to lists of pairs of document identifiers and counters. These lists are also called postings lists.

In document retrieval, the relatively simple inverted file access structure has proven superior [26, Chapter 8] to more complex structures like B-trees, hashtables, signature files, suffix trees or suffix arrays for indexing, index storage, and query processing, in particular for large amounts of data [245, page 109] or even data at Web scale [34].

A plethora [136] of alternative indexing models and structures were designed for indexing semi-structured data, both within the DB and IR communities. The main concern in databases approach being the optimization of complex structural requests, efficient storage and updates and so on.

In IR community and among the XML information retrieval systems providing content-oriented relevance ranking, the popularity of the inverted lists remained uncontested. A straightforward approach is to expand the postings list with information about the context of occurrence of the indexed terms in the XML tree [76]. Other popular approach consist in combining the inverted lists with different structural indexes – see Section 2.5.6. Efficient implementations of XML retrieval ranking based on inverted lists and top-k query processing algorithms can be found in [106, 219].

2.5.5 Indexing Unit

The document retrieval unit assumption does not hold anymore in the case of XML documents. The XML documents are trees that have a hierarchical structure of nested elements (sub-trees). To what extent the information contained in an XML element will influence its ancestors, its descendants or its neighbors? This problem echos on the choice of the indexing unit granularity.

We note here that the granularity of the retrieval unit may be different of the granularity of the indexing unit. A retrievable unit may be an indexing unit or be composed of several indexing units. The reverse is not true. The granularity of the indexing unit determines the smallest granularity level for the retrieved results. For instance, if we decide to index an XML document at section level, we will not be able to retrieve results at paragraph level.

There is no established technique of choosing what a good indexing or retrieval unit is. The indexing process may be additive – the terms of the leafs nodes are considered as valid index terms for all their ancestors – or disjunctive – the index terms are local to their nearest XML element in the tree hierarchy.

Next we present some of the the current indexing approaches by following the classification introduced in [24, 10].

XML Sub-Trees

A first approach is to index and retrieve separately XML sub-trees [8, 103, 208]. This is close to traditional IR – each XML element is a *bag of words* consisting of itself and its descendants and can be scored as an ordinary plain text document [10]. This approach has the advantage that it is a well-understood problem and applying the ranking function is straightforward. On the negative side, indexing XML sub-trees leads to a large amount of redundant information in the index. Another problem is the fact that the terms statistics computed at sub-tree level may not reflect the real retrieval significance of search terms. This is due to the fact that in the case of structured documents, the terms occurrences within the XML sub-trees may not conform to the Zipfs [253] and Luhn [135] distribution laws [214].

Disjoint Elements

Another approach consist in considering the XML elements as disjoint indexing units [65, 78, 157]. The index terms are local to their nearest XML element in the tree hierarchy. Therefore, the textual content of the elements can be retrieved as the union of one or more disjoint indexing units. The major advantage of this approach is the fact that it avoids redundancy in the index. One of its main difficulties is the computation of a score at a higher – i.e. non-leaf – level in the XML tree. For this purpose relevance propagation[13] [76, 77, 229, 234] or augmentation [78, 168] methods and mixture of element specific language models have been proposed [157].

Distributed Units

A third type of approaches indexes separately particular types of elements [142, 143, 141] – e.g. create separate indexes for articles, abstracts, sections, subsections, subsubsections, paragraphs, etc. This allows to compute statistics tailored to particular types of structural elements. The queries are submitted to all the indexes and the retrieved results are merged and combined after score normalization. The main advantages of this approach compared with the disjoint element strategy is that it avoids score computation and propagation which can be expensive at retrieval time. From this point of view the index redundancy may be seen as a way of pre-computing propagation at indexing time. Also, the XML specific relevance propagation strategies [77, 234] requires nontrivial parameters settings that may need training data. In contrast, the indexing and retrieval models used in this framework can use standard and well-known IR techniques. As a negative side, the retrieval model is completely dependent of the particular types of elements retained beforehand at indexing time. This decision is manually made per application and collection basis and it can not evolve with the collection structural characteristics. To cope with this problem, an algorithm to automatically detect the component types for any given collection with respect to the minimum size of the components to be indexed and to the number of indices to create was introduced in [141].

[13] An early suggestion for a multimedia retrieval model allowing upward and downward attributes values propagation in structured documents was made by Chiaramella et al. in [47].

2.5.6 Indexing the Structural Information

Indexing the structural information of documents is of major importance for the XML IR process. This is due to the fact that the structural expressivity power of the query languages and the whole retrieval process depends – or at least should depend – on the structure.

We present and discuss in this section a non exhaustive XML document indexes taxonomy. It is useful to classify the indexing methods based on their structure, query expressivity and processing strategy. A straight relation exists between the strategy used to index the structural components and the power and expressivity of the structured query language. We present the indexing paradigms in the increasing order of the amount of structural information that they take into account by following the classifications introduced by Luk et al. [136] and more recently by Catania et. al. [43].

Flat-File Indexing

XML documents can be treated as simple (flat) text files and the indexing process is similar to indexing in conventional IR systems. In this view the XML tags are discarded and only the content of the XML documents is indexed [136]. The obvious advantage is the fact that any standard IR system can be employed for this task. Nevertheless, the structural part of the document can not be used either in the request – i.e. *keyword-only queries* – or in an implicit manner in the retrieval process if the user does not want to add structural features to his queries.

An alternative approach is to use the XML tags as indexed terms – i.e. <*author*> – or to completely ignore the surrounding angle brackets – for instance <*author*> will be indexed as *author*. Anyway, the additional index terms of the XML tags may influence the document terms statistics used in the retrieval model which can be either a desirable or undesirable effect [136].

Tag-Based Indexing

This is perhaps the simplest semi-structured indexing method for XML documents [136]. Tag-based indexing is an approach that links each indexed term with the structural field in which it occurs. This allows to maintain a structural trace at a single depth-level of the context from which the structural term was extracted. In this case the overall tree structural organization of the document is lost. The XML document is represented as a bag of words in context.

To allow searching restricted to certain tags, index terms are constructed by combining the tag name with the terms from the content – i.e. *tag and keyword queries*. For instance XSEarch engine [52] use the following syntax *"author: Bayeza-Yates"* to search for *Bayeza-Yates* in the <*author*> context. XSEarch may also consider semantically related tags as query results.

Path-Based Indexing

The path-based indexing techniques have as objective to efficiently retrieve documents or documents fragments containing known sequences of values for elements/attributes tags or certain attributes values. Each index term is associated with its XML context or path starting from the root of the tree to the deepest node containing the term. For

instance, *Bayeza-Yates* which is the name of the author of an article will be indexed as *"/article/author/name/Bayeza-Yates"*. For efficiency purpose, the tag names or the whole path can be replaced by OIDS (Object Identifiers).

To answer *path and keywords queries,* a common approach is to mix structural path-based indexing with text indexing techniques like the inverted lists [245, 26]. This allows to associate each word entry with its path position in the host document. This approach may sustain sophisticated conditions on structure "ala" XPath [45] or NEXI Content-And-Structure (CAS) queries [226]. Nevertheless, finding ancestor-descendant relationships among the nodes of the indexed terms is less efficient than in the tree-based indexing approaches.

Tree-Based Indexing

In tree-based indexing an unique object id is associated with each node or element of the XML tree. The indexed terms point to the object id of the element that contains them. This allows to precisely locate where the indexed terms occurred at indexing time and to recreate the hierarchical relations of the elements [205].

For instance, the GPX system [77] uses the complete XPath specification of the nodes locations within the XML tree – including their order position – as OID for the indexed elements. In this case, the B*ayeza-Yates* term points to the unique XPath context – *"/article[1]/author[2]/name[1]"* within the given document. Furthermore, this is associated with a reference to the document id and to the position of the indexed term within the current XML element in order to allow the processing of content and structure requests.

The tree-based approach supports twig queries by performing structural joins between the indexed nodes. Most of approaches process each of the root-to-leaf paths in the query twig separately and then merge the results. One of the most efficient method for processing twig patterns is the holistic twig join algorithm [36]. The efficiency of the joins are determined by dedicated node encoding schemes. These schemes can determine the position of the indexed nodes in the XML documents tree structure without accessing the original XML documents.

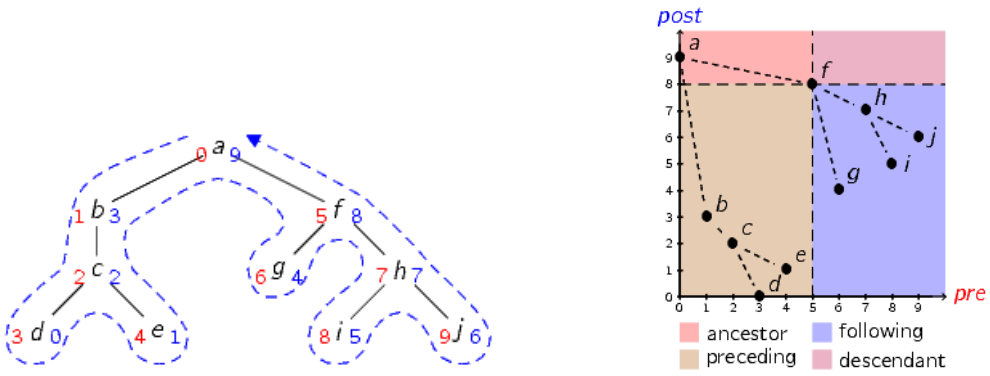
Labeling Schemes. Many labeling and encoding schemes for XML nodes were proposed to define a trade off between space occupancy, information content and suitability to updates [51]. Among them we can cite ORDPATHs [160] implemented in the MS SQL Server[14] or Dewey IDs [85].

One common labeling node scheme is the region-based approach that encodes each element by its position based on the tree preorder and postorder traversal algorithm – see Figure 2.5 a) . The graphic on the left sketches the derivation of postorder ranks (red numbers) for an example tree.

This encoding allows to efficiently characterize XML axis regions: – i.e. ancestor, descendant, following and preceding node relationships – by simple algebraic operations. For instance, *v'* is an ancestor of *v* if $pre(v') < pre(v) \wedge post(v') < post(v)$.

This characterization is illustrated in Figure 2.5 b) , where the nodes of the XML tree from the previous figure have been mapped into the two-dimensional pre/post plane, using their pre and post values as coordinates. Each node in this plane induces a partitioning into four disjoint regions (illustrated for node *f*): 1) the ancestors of *f,*

[14]Microsoft SQL Server `http://www.microsoft.com/sql/`



a) b)

Figure 2.5: Encoding XML Document Regions using Pre/Post Order Traversal.Encoding XML Document Regions using Pre/Post Order Traversal a) Derivation of Post Order Traversal Ranks b) The Pre/Post Plane [82].

2) the descendants of f, 3) the following nodes of f and 4) the nodes preceding f. This characterization of document regions applies to all nodes in the plane. This means that we may pick any node v and use its location in the plane to efficiently solve structural relationships [82].

Grust et al. equally use references to the label and to the parent of the current node to efficiently compute XPath location steps in their XPath Accelerator approach [82]. XPath Accelerator can also answer partial structural queries – i.e. queries that do not start with the root node of the XML document. An example of a system that uses region-based labeling scheme in an XML IR framework with relevance ranking is the XFIRM system [201].

Sequence-Based Indexing

Recently, a new indexing paradigm was proposed in order to increase the efficiency of processing branching queries on XML documents – the sequence-based indexing [241, 185, 240]. In sequence-based indexes, XML documents and branching queries are represented as sequences, and subsequence matching provides the query answers. Thus, unlike other approaches, sequence based XML indexing uses the tree structure as the basic query unit, thus avoiding the need to disassemble a structured query into multiple subqueries [43].

Two approaches were used to transform twig queries and structured XML data into sequences, namely the tree depth-first traversal implemented in the Virtual Suffix Tree approach [241] and the Prüfer codes [185]. A virtual suffix tree [241] codifies XML documents and queries as sequences of pairs, each representing a node and the path (including node content) to reach it, according to a tree preorder visit. In [185], the XML documents and queries are coded as sequences of labels, corresponding to Prüfer sequences.

False hits can be generated when querying XML tree structures using sequence-based representations. That is, a subsequence match will not always correspond to a subtree match between the query and the documents. Therefore, a suplimentary refinement step is needed to eliminate the false hits. To avoid this problem [240] pro-

posed classes of sequencing methods to preserve query equivalence between a structural match and a subsequence match.

However, sequence-based indexes do not directly support queries with selection conditions over internal nodes and they are not yet extended to include content relevance ranking.

For more information on indexing, searching and querying XML documents, we suggest the comprehensive survey of Luk et al. [136]. A more recent XML document indexes classification is introduced by Catania et al. in [43], while Amer-Yahia and Lalmas [16] provide a classification and brief description of the expressivity of XML search languages and current challenges in scoring XML elements.

2.6 XML Retrieval Models

One central problem for information retrieval systems is to predict which documents are relevant and which are not. A ranking algorithm that establishes an order based on the relevance of the retrieved documents operates according to distinct premises about the notion of document relevance. The way these different assumptions are modeled within the IR system, the way the relevance scores are derived and interpreted constitutes a *retrieval model* [26].

In classical information retrieval, to determine relevance, a numerical weight is assigned to each term-document pair in a document collection. The weight represents the term's significance within the document content. The relevance-based ordering of documents with respect to a query, also called ranking, is obtained by aggregating the query term weights for every document and determining the highest value. Most document retrieval approaches use the *bag-of-words* paradigm as their underlying model. The main assumption of the bag-of-words model is that the documents are represented by independent terms and their multiple occurrences within a single document.

Standard document retrieval and its underlying ranking models are discussed in detail in the classical books by van Rijsbergen [228], Salton [196], and Baeza-Yates and Ribeiro-Netto [26]. In this section we survey some of the existing extensions of classical document retrieval approaches to support semi-structured documents retrieval. These extensions try to take into account the structure or, more often the impact of the structure on the information unit characteristics [214]. We focus mainly on the (most successfully) retrieval approaches introduced within the INEX evaluation campaigns.

2.6.1 Vector Space Model

The *vector space model* [197, 196] (VSM) is a well-known IR model for representing text documents as vectors of identifiers, such as, for example, index terms. The similarity in vector space models is determined by using associative coefficients based on the inner product of the document vector and query vector, where word overlap indicates similarity. The inner product is usually normalized. The most popular similarity measure is the cosine coefficient, which measures the angle between the document vector and the query vector.

To apply the vector space model to XML documents, a similarity score must be computed between the request and each XML element from the document. Also, the

similarity measure must be extended in order to evaluate relations between structure and content. In this case, each index term should be encapsulated by one or more elements. An usual approach is to index XML *sub-trees* – see Section 2.5.5 – by propagating the terms of the leaf nodes in the document tree.

An attempt made as early as 1993 by Fuller et al. in [72] generalized the vector space model with the aggregation of relevance scores in the documents hierarchy. The basic idea is that the extended vector space model can be applied recursively to every sub-tree in a hierarchy to aggregate scores. The authors also suggested that the contribution of the content of the child and current node could be weighted differently.

Schlieder and Meuss introduce in [203], a query model based on VSM and tree matching techniques: this allows to express queries without perfectly knowing the document structure. The document collection is seen as a single tree where documents are sub-trees. Query is also a tree and every logical document rooted at a node with the same label as the query root is a potential candidate to be returned as result. But instead of returning a ranked list of document elements, a ranked list of whole documents was returned.

In [79], Grabs et Scheck propose a modified VSM that evaluate the importance – i.e. ief_{cat} inverse element frequency – of an indexed term from a given element as a function of its importance within the elements of the same type or category.

One of the successful aplication of VSM to XML retrieval is the JuruXML search engine [142] developed by the IBM research lab at Haifa. Instead of keeping term statistics at the document level they stored them at XML component level – see Section 2.5.5. Each component was thus separately ranked by the similarity measure with its own kind of *tf-idf*. When using fine-grained indexes, data that is outside their scope, but which is not indexed, may be required to obtain a better ranking. As a solution to this problem, [143] applied a document pivot scaling [208] for which the scores of the elements from each index is scaled by the score of their parent article. One major drawback of this approach is the fact that the above algorithm requires some prior knowledge on the collection such as the element types to index in each of the indexes. Therefore, the authors extended their work by providing in [141] a "component indexing algorithm" for creating a small number of indexes for any given collection. The system can also handle vague structural constraints by integrating a context resemblance function between the users requests represented as XML fragments [41] and the XML documents.

The XXL system [216] uses a ranking function based on the *tf* and *idf* term statistics. It allows vague path conditions on the XML structure integrated in a SQL syntax flavor (select-from-where). More flavors and extensions of the VSM model applied to XML IR can be found in [86, 93, 75, 76, 54, 130, 179, 229, 234].

2.6.2 Language Model

Statistical language models were first introduced for information retrieval by Ponte and Croft [172]. A language model estimate the relevance of a document by calculating the probability that the document generates the query terms. This is done by using two different probability distributions: the foreground and background models. In XML element retrieval, the *documents* in the retrieval model correspond to elements. The foreground model $P(t_i|e_j)$ estimates the probability of a term t_i given a particular element e_j, and $P(t_i)$ estimates the probability of the term t_i in natural

language. These two distributions are linearly combined to give a final score. The likelihood for a query $q = (t_1, t_2, ..., t_n)$ to be generated from an element e_j it is given by:

$$\begin{aligned} P(q|e_j) &= P(t_1, t_2, ..., t_n|e_j) \\ &= \prod_{i=1}^{n}(\lambda \cdot P(t_i|e_j) + (1-\lambda) \cdot P(t_i)) \end{aligned}$$

where n is the query length.

In order to avoid assigning a value of 0 to the entire product of probabilities when a single query term t_i does not occur in the document (i.e. the sparse data problem [91]), a linear combination of the probabilities, also known as linear interpolation smoothing, can be applied. The Jelinek-Mercer smoothing [91] parameter in the above formula controls the emphasis given to the evidence collected by the model (foreground) or by the collection model (background). Background probabilities assure that common terms contribute less to the final ranking. In XML retrieval the smoothing parameter may also introduce a length bias. For instance, larger elements are returned when using higher lambda values [100].

The foreground and background probabilities, $P(t_i|e_j)$ and $P(t_i)$ are defined as maximum likelihood estimators and computed using specific collection statistics. For the foreground probability, usually a maximum-likelihood estimator based on term frequency (the number of times a term t_i occurs in an element e_j) is used. For estimating the background probability, common estimators are collection or document frequencies (the number of times a term t_i occurs in the collection/unique documents).

The language modeling approach allows to combine "non-content" features of elements (or documents) with the scoring mechanism by using element prior probabilities $P(e_j)$. The ranking is produced by computing the relevance of an element e_j to a given query q as follows:

$$P(e_j|q) \propto P(e_j) \cdot P(q|e_j)$$

where $P(e_j)$ is the prior probability of relevance for element e_j and $P(q|e_j)$ as defined above.

A very simple, yet effective approach is to use the element size as a length prior ($P(e_j) = size(e_j)$). Different lengths priors and their effects for XML retrieval are analyzed in [103]. Structural features, like the overall organization of the documents within the collection are introduced in [183].

Recent approaches [20, 21] use discourse features, like semantic passages, as source of evidence for enhancing language models for XML retrieval. The authors use a topic segmentation algorithm based on lexical cohesion and develop the notion of the so-called *topic shifts*. A topic-shift is detected as changes in the vocabulary of adjacent text segments occurs. These are used as priors to tune their language model for XML retrieval.

Another popular technique in language modeling approaches is the so called *hierarchical language models* [157] or *mixture models* [206]. In these models relevancy is estimated as a linear interpolation of different language models.

Sigurbjörnsson et al. [207] studied the effect of selective indexing strategies by using a multinomial language model as a linear interpolation of three language models for *element*, *document* and *collection*.

Ogilvie and Callan [156, 157, 158] applied a hierarchical language model to XML retrieval. For each node in the XML document tree (e.g. title, abstract, body, section,

sub-section etc.) a language model is considered. For leaf nodes, the language model is based only on the text contained by the nodes itselfs. For nodes situated at upper levels in the XML tree, the language model is estimated by the linear interpolation of the language model formed from the text in the node itself and that formed from its children [156]. In [157] they extend their approach by adding a language model for the element's parent into the linear interpolation. Interpolation parameters are estimated using a generalized expectation maximization algorithm in [158].

The language model is used in the TIJAH system [150, 151] on a three-level database architecture composed of conceptual, logical and physical levels. The language model is implemented at the conceptual level, then at the logical level it is translated into expressions constructed on a probabilistic region algebra and executed at the physical level using the MonetDB database kernel.

2.6.3 Probabilistic Model

In the traditional probabilistic model [190] retrieval is considered as a two class Bayesian decision problem, and ranking is based on the binary independence model [136]. The probabilistic model is applied to XML documents in [117, 66].

Lalmas [117] extends the FERMI structured multimedia information retrieval model introduced in [47] by applying the Dempster-Shafer's theory [204] to combine different evidence sources. In this paradigm, we search to find evidence that the document is relevant to a query. Each occurrence of a query term in the document is partial evidence that the document is relevant. It is shown that the theory provides a rule, the Dempster's combination rule, that allows the expression of the uncertainty with respect to parts of a document. Each piece of evidence is weighted by a belief of relevance, which can be specified in the query. The Dempster-Shafer combination of those belief weights sets the total relevance of the document [136].

Another example is the HyREX system [78] that uses probabilistic inference and path algebra to transform queries into facts and rules in *pDatalog* – i.e. probabilistic Datalog. The XIRQL query language [66] implemented in HyREX extends the XPath operators with operators for relevance-oriented search and vague searches on non-textual content. Documents are then sorted by decreasing probability that their content is relevant to the one specified by the user.

A fusion approach for multiple probabilistic searches against different XML components combined and weighted using a logistic regression-based algorithm is presented in [124, 125]. Finally, OKAPI-based models adapted for XML IR are presented in BM25E [134], TopX [217, 219] and TReX [9].

2.6.4 Machine Learning

A retrieval model for XML documents based on *Bayesian Networks* is proposed by Piwowarski et al. in [169, 170]. The structure of the Bayesian networks reflects directly the documents hierarchy. Each document node X is associated with a random variable that can be in any of the three states: *Irrelevant*, *Big* or *Exact*. The retrieval process starts from the root node of the document and propagates downwards. The probability that a node X has a particular state v given a query q, $P(X = v_x|q)$ is calculated as a linear combination of conditional probabilities of its parent nodes and various local baseline models. Conditional probabilities are learned from a labeled collection of structured documents – which is composed of documents, queries and their associated

assessments – using a cross-entropy training criterion and by minimizing the entropy function via gradient descent method. An approach to structured document retrieval based on a generalization of Bayesian Networks – *influence diagrams* – is presented in [56].

Vittaut and Gallinari propose in [235, 236] a *machine learning ranking model* for XML element retrieval. The model is used to automatically learn a set of weighted parameters. These weights are used to combines features characterizing the elements to be ranked which depend on the element itself, its parent element and the document containing that element. The model learns how to combine these features in an optimal way according to a loss function from a set of examples composed of query and assessed elements pairs. The proposed approach shown to effectively improve the results of a baseline OKAPI model adapted to XML retrieval.

In this section we have introduced some of the existent retrieval models for structured IR. More information about the past and recent retrieval approaches for XML retrieval can be found in [119, 136, 163, 198] and in the annual INEX workshop proceedings [64, 68, 70, 69, 71].

We note here that there is not a clear winner for any of these retrieval approaches. As stated in [16], an uniform and controllable platform on which all the proposed retrieval approaches and their various parameters could be studied and compared is required in order to obtain fundamental results regarding the best practices in XML retrieval. Unfortunately, this still represents a major challenge for XML IR research.

2.7 XML IR Evaluation

The predominant approach to evaluate a system retrieval effectiveness is to make use of test collections and appropriate effectiveness scoring methods. A test collection consist of a set of documents, a set of user requests (the so-called topics, or queries) and relevance assessments of the documents with respect to the queries.

The implicit assumption in traditional test collections is that the atomic retrieval unit is at document level. This is not a valid assumption in the case of XML retrieval systems that may return document components instead of full documents as response to a user's information need.

In order to evaluate how effective an XML IR system is, we need to dispose of a testbed that integrates the new granularity of the retrieval unit defined by the structural aspects in the evaluation paradigm [16].

Since 2002, the *INitiative for the Evaluation of XML Retrieval*[15] (INEX) started to address this issue. The aim of the INEX initiative is "to establish an infrastructure and provide means, in the form of large test collections and appropriate scoring methods, for evaluating content-oriented XML retrieval systems" [122]. We provide on overview of the evaluation methodology developed in INEX and a detailed description of the test collection, the topics, the retrieval tasks, the relevance judgments and the evaluation measures in Chapter 4.

[15]The INitiative for the Evaluation of XML Retrieval (INEX) `http://inex.is.informatik.uni-duisburg.de/`

2.8 Conclusion

We have presented past and current work in XML IR by highlighting the main challenges of the field. We have also taken a close look at the indexing process and retrieval models and at the way they were adapted and enhanced to handle semi-structured information retrieval.

We present our proposals for managing and searching semi-structured documents and their implementation within the SIRIUS XML IR system [177, 179] in the next chapter, while the evaluation settings and the experimental investigations that we conducted in order to validate our approach are described in Chapters 4, 5 and 6.

Chapter 3

SIRIUS XML IR System

"This much is already known: for every sensible line of straightforward statement, there are leagues of senseless cacophonies, verbal jumbles and incoherences. [...] The certitude that some shelf in some hexagon held precious books and that these precious books were inaccessible, seemed almost intolerable. [...] There are official searchers, inquisitors."

— Jorge Luis Borges, The Library of Babel

In this chapter we present our approach to XML IR and its implementation within SIRIUS – a lightweight indexing and search engine for XML documents developed at the VALORIA laboratory of the University of South-Brittany. Preliminary versions of the approach presented in the current chapter were published in [177, 176, 179, 178].

Contents

3.1 Introduction

The widespread use of XML in digital libraries, product catalogs, scientific data repositories and across the Web prompted the development of appropriate searching and browsing methods for XML documents. XML IR should provide the possibility of querying the information acquired by a system having an incomplete or imprecise knowledge about both the structure and the content of the XML documents [66]. The purpose of an XML IR system is to retrieve a ranked list of relevant - i.e. the most *specific* and the most *exhaustive* – XML elements.

In this chapter we describe our approach to XML IR and its implementation in the SIRIUS prototype developed by the VALORIA laboratory of the University of South-Brittany. As its main characteristics SIRIUS merges i) flexible matching of the XML structure based on an Levenshtein editing distance on the XML tree paths ii) a relevance ranking algorithm for the textual content based on the vector space model and iii) a query enrichment mechanism based on a thesaurus of semantic rules. Finally we present the evaluation results for the proposed approach obtained within the INEX 2005[1] and INEX 2006[2] evaluation campaigns.

3.1.1 Document Structure and IR

XML documents are semi structured documents that contain structural annotations in contrast with plain - i.e. unstructured - documents used in classical IR. This structure reflects (at least partially[3]) the logical structure of the documents and therefore should be taken into account in the retrieval process. This is expected to improve the quality of the retrieved results [26]. More, mixing content and structure in the IR process will allow the users to express more powerful queries and to search not only whole documents, but also relevant document components [198].

3.1.2 Strict and Vague Interpretation of XML-Retrieval Queries

When looking for information including structural constraints, we perform a content and structure (CAS) query [226]. When specifying these structural constraints users may have a clear picture of what they mean. For example when searching for *Baeza-Yate*s as the *author* of an article they do not consider as relevant answers articles *citing Baeza-Yate*s. However, when searching for *sections* of articles about *string processing* they are likely to be satisfied with a *subsection* of an article on the topic. The interpretation of the structural constraints may be *strict,* or *vague* - i.e. they are considered as *structural hints* [226, 223].

XML *data-centric* approaches like XPath [60] and XQuery [63] uses a strict match of the structure and of the content of XML documents.

Currently, the World Wide Web Consortium (W3C)[4] propose XQuery 1.0 and XPath 2.0 Full-Text [42, 193, 11] that extends data-centric query languages with full-text search

[1] INitiative for the Evaluation of XML Retrieval (INEX), April 2005 - December 2005 `http://inex.is.informatik.uni-duisburg.de/2005/`

[2] INitiative for the Evaluation of XML Retrieval (INEX), March 2006 - December 2006 `http://inex.is.informatik.uni-duisburg.de/2006/`

[3] If this is not the case, we should preprocess and classify the tags according to their real semantic for the end user (see Section 2.1.3 on page 12).

[4] The World Wide Web Consortium (W3C) `http://www.w3.org`

capabilities like relevance weighting and ranking. Nevertheless, the matching of the structural constraints remains strict.

In the context of a heterogeneous environment, like the web, XML documents may have multiple sources and serve different purposes. In this view, it is unlikely that a user has a perfect knowledge of the structure of the documents to be queried. Even in cases when the structure of the documents from the target collection is completely known - i.e. available DTD or XML Schema [28, 138] - the common users have difficulties to formulate queries about the structure of the documents [223, 159, 226]. As a consequence, the structural constraints expressed by the users may be seen as structural hints. These are hardly intended to dictate the exact structure of the query result; rather they provide a loose example of the information the user is interested in [32]. Their strict interpretation may ignore potentially useful information and negatively influence the relevance of the retrieved results. At the extreme, no result will be returned at all - resulting in *silence*. Therefore, in an XML information retrieval process the structural constraints should be in general interpreted vaguely.

3.1.3 Approximate Structure Matching

"A fool sees not the same tree that a wise man sees."

— William Blake

Considerable effort was spent in the IR community, starting with the SIGIR workshops on XML IR [23, 22, 192, 6] and with the INEX evaluation campaigns [64, 68, 70, 69, 225] for designing new approaches for XML IR. Among them, several approaches include specific algorithms for integrating the structure and the structural organization of the XML tags in an approximate way within the relevance ranking process.

A first approach is to use the structural constraints as simple filtering conditions. [216, 208].

Most of the systems implementing a vague approach for the structural constraints use classes of equivalent tags. These classes are manually generated starting from the DTD(s) of the collection [201]. Additionally, specific heuristics (i.e. statistics of the relevant results [151], structural enforcement [143]) are added to weight the relative importance of the tags in the collection. In [86] the XML tags are transformed using manually built mappings to meta data with relevant semantic for the user, while [19] replace them with $*$ - i.e. any element.

To cope with the heterogeneity of the tags in the collections, some approaches [216, 80] consider the use of dictionaries, thesaurus or ontologies to extract semantic related terms or concepts for building the XML tags equivalence classes.

[230] introduces *path factor extension* to account for the number of structural tags matched and *request penalty factor* to penalize excessive elements in the retrieved answer path not specified in the user query. Their approach does not handle the hierarchical organization of the XML tags.

In [249] the structural heterogeneity is handled by using distances reflecting the notion of structural proximities (i.e. horizontal distance - number of intermediary sibling nodes, and vertical distance - the number of nested levels).

Another level of vagueness allowed on the structural constraints concerns the switch or equivalence between the role of XML elements names and the XML attributes or between the XML element content and attributes values [25, 66]. This

approximation may be regarded as a design decision which is application dependent. One reason is the fact that the XML standard [164] defines no specific order for the pairs of attributes and attributes values associated to the same XML element. Conversely, the XML elements and their content must respect the *document order*. This difference is important in a *document-centric* context as the *document order* is highly relevant for the information retrieval process.

Anyhow, in all the previous approaches, the organization of the structural tags is not regarded as flexible.

More elaborate methods explore the structural organization of the XML tags by using *graph* [55, 139], *tree* [32, 203, 15] or *path* [41] based approximate matching.

We cite here only methods integrating hierarchical structure approximation and textual content ranking. Early seminal work for approximate match of semi-structured data modeled as graphs is the OEM model implemented in the Lore system [145].

The approaches trying to approximately match complete trees are based on classical tree edit distances [212] and have a high complexity - i.e. $O(\mid N_1 \mid \cdot \mid N_2 \mid \cdot depth(T_1) \cdot depth(T_2))$, where $\mid N_i \mid$ is the number of nodes of the tree T_i - for *ordered trees*[5] [202]. In [251], Zhang et al. gave a proof that unordered tree embedding problem is $NP - Hard$. This complexity may be reduced depending on the assumptions about the XML trees and of the operations allowed [13]. The high computational complexity of this kind of tree matching is not suited for the data sets sizes we intend to process.

Amer-Yahia et al. explored in [12] the complexity and the expressivity of path scoring and tree based queries for XML IR. They experimentally showed that the path based scoring provides very high precision while improving score computation time.

In our approach we introduce algorithms designed to manage approximate search in heterogeneous XML Document databases. We propose specific data structures designed to the indexing and retrieval of information elements in heterogeneous XML data bases (originated from a set of WEB pages for instance). The indexing scheme is well suited to the management of various contextual searches, expressed either at a structural level or at an information content level. We design an approximate query language that merges a matching process based on a modified editing distance with a text ranking model at a low-level, and provides boolean and fuzzy merging operators for answering complex requests at a higher level. The complexity of main algorithms is studied and the performance is analyzed. The implementation described highlights the mixing of structured information presented as field/value instances and free text elements. Further, we evaluate its effectiveness and efficiency for a variety of XML IR tasks within the framework provided by the INEX 2005 [69] and INEX 2006 [71] evaluation campaigns.

The work presented in this chapter has similarities with the XML fragment approach [41]. In addition we introduce extensions for managing attributes and attributes values as well as semantic similarity between the structural tags. Also, the prototype implementation including the index structures, the definition of the query language and the processing scheme for complex requests is quite different.

[5] A tree is ordered if the left-to-right order of sibling nodes is fixed.

3.2 Document Model

XML documents are generally represented as rooted, ordered, and labeled trees in which each node corresponds to an element and each edge represents a parent-child relationship.

Each XML element in an XML document may be composed of a set of possible nested XML elements, textual pieces of information (TEXT or CDATA), unordered $< attribute, value >$ pairs, or a mixture of such items.

An example of an XML document extracted from the Reuters Corpus[6] together with its ordered tree representation are given in Figure 3.1. and Figure 3.2.

3.2.1 XML Context

According to the tree structure, every node n inherits a path $p(n)$ composed with the nodes that link the root n_0 to node n. This path is an ordered sequence of XML elements potentially associated to unordered $< attribute, value >$ pairs $A(n)$, that determines the XML context in which the node is occurring. This path can be represented as follows:

$$p(n) =< n_0, A(n_0) >< n_1, A(n_1) > ... < n, A(n) >$$

Let n be named the *terminal node* of $p(n)$. A tree node n containing textual/mixed information can be decomposed into textual sub-elements or strings. Each textual terminal t (or token, word, lemma, string...) is also linked to $p(n)$ and may be seen as a terminal node without any attribute. In this case the contextual occurrence of t will be the sequence:

$$p(t|t \in n) =< n_0, A(n_0) >< n_1, A(n_1) > ... < n, A(n) >< t, \oslash >$$

3.3 The Index Model

The indexing process is based on the creation of an enriched inverted list designed for the management of the XML contexts. For this model, the entries of the inverted lists are all the valid textual terminals t of tree nodes. A textual terminal t of a node n is associated with a list of reference locators $\{rl_i\}$. Each reference locator rl has attached four pieces of information:

$$rl \rightarrow < docId >, < startNodeId, endNodeId >, < wordOffset >, < ctxtId >$$

- a link to the URI of the document , – this is used in order to retrieve the original XML document;
- a range-based labeling scheme that assigns to each node n two numbers that denote the start and end points of an interval $< startNodeId, endNodeId >$. The two values form a range (order, order+size). The labels are assigned such that a child's interval is contained in all it's parents' intervals. An example is given

[6] Reuters Corpus, Volume 1, English language, 1996-08-20 to 1997-08-19 (Release date 2000-11-03, Format version 1, correction level 0) `http://trec.nist.gov/data/reuters/reuters.html`

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<newsitem itemid="15569" id="root" date="1996-08-27" xml:lang="en">
    <copyright> (c) Reuters Limited 1996
    </copyright>
    <title> UK: UK shares start weaker as Wall Street weighs.
    </title>
    <headline> UK shares start weaker as Wall Street weighs.
    </headline>
    <dateline> LONDON 1996-08-27
    </dateline>
    <text>
        <p> London shares started sharply lower on Tuesday after the long holiday
            weekend as a disappointing performance on Wall Street and technical
            positions in the market sparked an early markdown, traders said.
        </p>
        <p> The FTSE 100 opened some 18.0 points lower at 3,889.5 as shares re-
            trenched after the string of all-time highs last week.
        </p>
    </text>
    <metadata>
        <codes class="bip:countries:1.0">
            <code code="UK">
                <editdetail attribution="Reuters BIP Coding Group" ac-
                  tion="confirmed" date="1996-08-27" />
            </code>
        </codes>
        <dc element="dc.date.created" value="1996-08-27" />
        <dc element="dc.publisher" value="Reuters Plc" />
        <dc element="dc.date.published" value="1996-08-27" />
        <dc element="dc.source" value="Reuters" />
        <dc element="dc.creator.location" value="LONDON" />
        <dc element="dc.creator.location.country.name" value="UK" />
    </metadata>
</newsitem>
```

Figure 3.1: An excerpt of an XML document extracted from the Reuters Corpus.

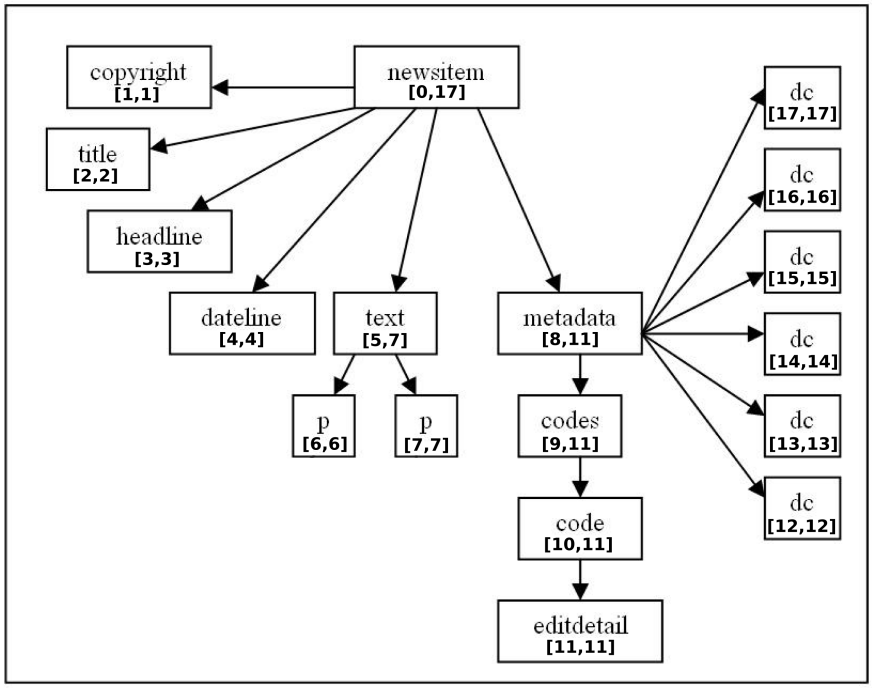


Figure 3.2: The ordered tree representation of the XML document from Figure 3.1.

in Figure 3.2. This labeling scheme was introduced in the XISS system [132] and used to accelerate the computation steps for the path constraints involving ancestor-descendant relationships between elements.

- an index specifying the location of the token t within the document $< wordOffset >$, – this index is used to perform various operations involving the words positions like phrase search or proximity words search within a given word distance interval;

- a link $< ctxtId >$ toward its XML context $p(n)$, – this context is used to apply and compute the structural constraints.

To each $< ctxtId >$ we associate an XML context $p(n)$ defined by:

- the ordered sequence of XML tags id $< tagId >$,

$$ctxId \rightarrow < tagId >_0 < tagId >_1 \ldots < tagId >_m$$

- the ordered sequence of unordered sets of $< attributeId, valueId >$ pairs associated with each XML element of the first sequence. If no attributes are associated to an XML element, the null value $\oslash$ is used to maintain the correspondence between the indexes of the two sequences.

$$ctxId \rightarrow < \{(attributeId, valueId)_k\} >_0 < \oslash >_1 \ldots\ldots < \oslash >_m$$

Example For the XML document from Figure 3.1 the inverted list associated with the *london* and *points* entries are:

$london \rightarrow [< 0 >, < 4,\ 4 >, < 21 >, < 4 >],\ [< 0 >, < 6,\ 6 >, < 23 >, < 6 >]$

$points \rightarrow [< 0 >, < 7,\ 7 >, < 48 >, < 6 >]$

where we consider that the $< fileId >$ associated with the XML document is 0.

The *london* entry appears in two different locations in the document content at word offsets 21 and 23 counting from the beginning of the text, in the nodes labeled *dateline [4, 4]* and *p [6, 6]* respectively (see Figure 3.1 and Figure 3.2). It also occurs as an attribute value in the node labeled *dc [16, 16]*, but this occurrence is not indexed by default as the XML node contains no valid textual data.

The XML contexts <*ctxtId*> starts to be numbered at the root of the XML tree. This assigns the 0 value to the */newsitem* context. The pairs of attributes and attributes values are indexed relatively to the position of the tag element in the XML context as shown below. The XML contexts referred by the *london* entry in the inverted list are */newsitem/dateline* (<*ctxtId*>=4) and */newsitem/text/p* (<*ctxtId*>=*6*).

For space efficiency purposes, an XML context is indexed by using a dictionary mechanism and it is related to a unique *ctxtId* key value. For instance, the XML contexts associated with the nodes *p [6, 6] and p [7, 7]* are identical (i.e. - */newsitem/-text/p*) and will refer to the same key <*ctxtId*>=*6*. Therefore, the inverted list entry located in the node labeled *p [7, 7]* (i.e. *points*) refers to <*ctxtId*>=*6*. There is no direct link between the preorder position of the node in the XML tree and its *ctxtId* key value.

$0 \rightarrow < newsitem_{Id} >_0$

$$0 \rightarrow \left\langle \begin{array}{ll} (itemid_{Id}, 15569_{Id}), & (id_{Id},\ root_{Id}), \\ (date_{Id},\ 1996-08-27_{Id}), & (xml : lang_{Id},\ en_{Id}) \end{array} \right\rangle_0$$

$1 \rightarrow < newsitem_{Id} >_0 < copyright_{Id} >_1$

$$1 \rightarrow \left\langle \begin{array}{ll} (itemid_{Id}, 15569_{Id}), & (id_{Id},\ root_{Id}), \\ (date_{Id},\ 1996-08-27_{Id}), & (xml : lang_{Id},\ en_{Id}) \end{array} \right\rangle_0 < \oslash >_1$$

...

$4 \rightarrow < newsitem_{Id} >_0 < dateline_{Id} >_1$

$$4 \rightarrow \left\langle \begin{array}{ll} (itemid_{Id}, 15569_{Id}), & (id_{Id},\ root_{Id}), \\ (date_{Id},\ 1996-08-27_{Id}), & (xml : lang_{Id},\ en_{Id}) \end{array} \right\rangle_0 < \oslash >_1$$

...

$6 \rightarrow < newsitem_{Id} >_0 < text_{Id} >_1 < p_{Id} >_2$

$$6 \rightarrow \left\langle \begin{array}{ll} (itemid_{Id}, 15569_{Id}), & (id_{Id},\ root_{Id}), \\ (date_{Id},\ 1996-08-27_{Id}), & (xml : lang_{Id},\ en_{Id}) \end{array} \right\rangle_0 < \oslash >_1 < \oslash >_2$$

...

3.4 The Retrieval Scheme

Most of the time, for large heterogeneous databases, one cannot assume that the user knows all of the structures - even in the very optimistic case, when all the structural properties are known. Some straightforward approaches (such as the XPath [60] or XQuery [63] search scheme) may not be efficient in these cases. As the user cannot be aware of the complete XML structure of the data base due to its heterogeneity, efficient searching should involved exact and approximate search mechanisms.

The main structure used in XML is a tree: It seems acceptable to express a search in term of tree-like requests and approximate matching. The matching tree process involves mainly elastic matching or editing distance [212]. As described in Section 3.1.3 the matching complexity is too high to let these approaches perform well for large heterogeneous databases with documents with a high number of elements (as nodes). Ménier and Marteau proposed in [148] to focus on path matching rather than on tree matching - in a similar way with the XML fragment approach [41]. The request should be expressed as a set of path $p(r)$ that is matched with the set of sub-path $p(n)$ in the document tree. This breaks the algorithmic complexity of the tree matching techniques while still providing high precision results [12]. This *low-level* matching only manage sub path similarity search with conditions on the elements and attributes matching. This process is used to design a more *higher-level* request language: a full request is a tree of low-level matching goals (as leafs) with set operators as nodes. These operators are used to merge leaf results. The whole tree is evaluated to provide a set of ranked answers. The operators are classical set operators (intersection, union, difference) or dedicated fuzzy merging processors.

3.4.1 Approximate Path Search

Let R be a low-level request, expressed as a path goal p^R ended with a textual terminal t, in which the tuples $< attribute, value >$ are replaced by conditions or constraints. These constraints Ct set conditions to be fulfilled on the attributes and attributes values (for instance, Ct may indicate that the attribute date should have a date value > 2005).

$$p^R =< n_0, Ct(n_0) >< n_1, Ct(n_1) > ... < n_i, Ct(n_i) >< t, \oslash >$$

To allow a flexible interpretation of the structural constraints, we propose to evaluate the similarity σ between p^R (coding a path with constraints) and p_i^D (a path of the tree T^D associated to an index document D) as follows:

$$\sigma(p^R, p_i^D) = \frac{1}{1 + \delta_L(p_i^D, p^R)}$$

where δ_L is a dedicated editing distance (see [239]).

The aim of our approach is to provide a vague interpretation of the structural constraints by combining tags name matching techniques, nodes attributes conditions evaluation, and path edit distances. For this purpose, we designed an editing pseudo-distance using a customized cost matrix to compute the similarity $\sigma(p^R, p_i^D)$ between the request path p^R and an indexed path p_i^D. This scheme, also known as modified Levenshtein distance [131], computes a minimal sequence of elementary transformations to transform p_i^D into p^R. The elementary transformations are described below:

Substitution: a node n in the path p_i^D is replaced by a node n^R from p^R for a cost $C_{subst}(n^R, n)$. Since a node n not only stands for an XML element, but also for attributes or attribute relations Ct , we compute $C_{subst}(n^R, n)$ as follows:

$$C_{subst}(n^R, n) = h(\mu(n^R, n), \alpha(n^R, n))$$

where:

- $\mu(n^R, n)$ is the substitution cost for the nodes n^R and n (i.e. strict match or by using the semantic similarity between the nodes names – see Section 3.4.4) – without taking into account the attributes conditions, $\mu(n^R, n) \in [0, 1]$,
- $\alpha(n^R, n)$ stands for the degree of satisfaction of the conditions stated in the node n^R related to the attributes values in n, $\alpha(n^R, n) \in [0, 1]$,
- h is the merging function for the element substitution and the satisfied conditions on attributes. By design, $h(\mu(n^R, n), \alpha(n^R, n)) \in [0, 1]$, so, $C_{subst}(n^R, n)$ varies between $C_{subst}^{min} = 0$ (perfect matching) and $C_{subst}^{max} = 1$ (no matching at all).

We use a linear merging function to mix $\mu(n^R, n)$ and $\alpha(n^R, n)$:

$$h(\mu(n^R, n), \alpha(n^R, n)) = (1 - \lambda_1) \cdot (1 - \mu(n^R, n)) + \lambda_1 \cdot (1 - \alpha(n^R, n)))$$

with $\lambda_1 \in [0, 1]$. The value of λ_1 can be tweaked to emphasize the importance of the constraints on the attributes and attributes values.

Deletion: a node n in p_i^D is deleted for a cost $C_{del}(n)$ between $C_{del}^{min} = 0$ and $C_{del}^{max} = 1$.

Insertion: a node n is inserted in p_i^D for a cost $C_{ins}(n)$ between $C_{ins}^{min} = 0$ and $C_{ins}^{max} = 1$.

Let $T_{p_i^D \to p^R}$ the set of all the possible transformations (seen as sequences of elementary transformations: substitution, deletion, insertion) to transform p_i^D into p^R. Each transformation τ from $T_{p_i^D \to p^R}$ has a global cost $C(\tau)$ computed as the sum of the costs of each elementary operation.

For example, when matching the path request */doc/article(>= id 1)/bib/* with the indexed path */ieee/article(id=1)/ref/bib/* the minimum global cost is given by the substitution between the *ieee* and *doc* tag elements and the deletion of the *ref* tag (see Figure 3.3). We note here that a deletion operation in one of the contexts can be viewed as an insertion operation in the other context and vice versa. In this example, the *article* tag in the request has associated an attribute constraint *article(>= id 1)* that is satisfied by the attribute value of the *article* element in the index path *article(id=1)*. Therefore, no supplementary cost is added to the global cost of matching the two contexts.

The Wagner and Fisher algorithm [239] computes the minimum cost C_k^* among the costs for the transformations belonging to $T_{p_i^D \to p^R}$:

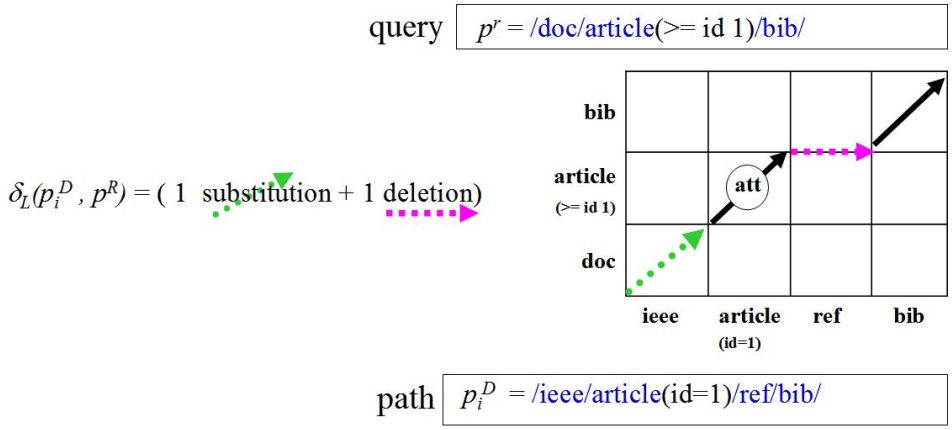


Figure 3.3: Approximate path search with conditions on attributes and attributes values.

$$\delta_L(p_i^D, p^R) = C_k^* = Min_{\tau \in T_{p_i^D \to p^R}} C(\tau) = C(\tau^*)$$

Let τ^*be the optimal transformation (with a minimal cost). Finally the similarity σ between a p^R (coding a path with constraints) and p_i^D is given by:

$$\sigma(p^R, p_i^D) = \frac{1}{1 + C(\tau^*)}$$

This approximate path search mechanism has a complexity of $O(length(p^R) \cdot length(p_i^D))$ [239] where:

- $length(p^R)$ stands for the length of the request path p^R
- $length(p_i^D)$ stands for the length of the indexed path p_i associated with the document D.

This complexity remains acceptable for this application since 99% of the XML documents available on the web have fewer than 8 levels and their average depth $depth(T^D)$ is 4 [149].

Given p^R and p_i^D, the similarity between the two contexts $\sigma(p^R, p_i^D) \to 0$ when the number of mismatching nodes and attribute conditions between p^R and p_i^D increases. More precisely, in the worst case, when the two contexts have no single node in common, the similarity is given by:

$$\begin{aligned} \sigma(p^R, p_i^D) \quad = \quad & 1 \;/\; (\; 1 + Min \left\{ length(p^R), length(p_i^D) \right\} \cdot C_{subst}(n^R, n) + \\ & \left| length(p^R) - length(p_i^D) \right| \cdot Min \left\{ C_{del}(n), C_{ins}(n) \right\} \;) \; . \end{aligned}$$

This corresponds with the operation of matching by substitution all the nodes from the shorter context and either deleting or inserting (whichever operation is less expensive) the remaining/required nodes. For a perfect match $\sigma(p^R, p_i^D) = 1$, i.e. all the elements and the conditions on attributes from the request p^R match XML elements and attributes in p_i^D.

3.4.2 Textual Content Ranking Scheme

Extensions of the main classical retrieval models were brought by the IR community in order to index and search semi-structured documents. Of the main retrieval models applied to XML IR we may cite: probabilistic models [117, 66], language models [157, 151] or the vector space models [41, 216, 203]. A Bayesian framework for XML document retrieval is presented in [171, 237].

Our purpose here is not to design a new relevance ranking formula, but to integrate an approximate structure matching scheme in an XML IR framework. Secondly, we want to evaluate its benefits and drawbacks in both effectiveness and efficiency.

With this in mind, we choose to use a simplification of the TFIDF ranking formula [195] to compute the relevance (i.e. rsv stands for *relevance status value*) of a textual XML node n to a content oriented (or content only - CO) request co^R:

$$rsv(co^R|n) = \nu \cdot \sum_{i=0}^{k} \underset{t \in n}{Max}\{\mu(t_i^R, t)\} \cdot idf(t_i^R)$$

where:

- k is the number of terms t_i^R in the content only request co^R,
- $\mu(t_i^R, t)$ is the substitution cost (i.e. strict match or by using the semantic similarity – see Section 3.4.4) between the request token t_i^R and the index term t,
- ν is a normalization constant:

$$\nu = \frac{1}{\sum_{i=0}^{k} idf(t_i^R)}$$

- $idf(t)$ is a weighting factor specifying the discriminant power of the term t in the collection of XML documents:

$$idf(t) = 1 - ln\frac{1 + |D(t)|}{1 + \mid D \mid}$$

- $\mid D \mid$ is the number of documents in the collection,
- $|D(t)|$ is the number of documents in the collection containing the term t.

3.4.3 Computing Element RSV

The first task in XML ranked retrieval is to produce the relevance scores values for all the nodes in the XML collection starting with the lowest-level in the tree hierarchy (i.e. the leaf nodes) and up to the document level (i.e. the root node).

As a request R may relate to both the structure p^R and the content co^R of an XML element it is quite natural that both axes be involved when computing the relevance score $rsv(R|n)$ of an XML node n. Therefore, we need an aggregation method to merge the relevance scores obtained on the structure $\sigma(p^R, p_i^D)$ and the textual content $rsv(co^R|n)$. We use a weighted linear combination as for the merging of XML elements and attributes conditions:

$$rsv(R|n) = h(rsv(co^R|n), \sigma(p^R, p_i^D))$$

where

$$h(rsv(co^R|n), \sigma(p^R, p_i^D)) = (1 - \lambda_2) \cdot rsv(co^R|n) + \lambda_2 \cdot \sigma(p^R, p_i^D)$$

with $\lambda_2 \in [0, 1]$. The value of λ_2 can be tuned to emphasize the importance of the structural versus textual content matching.

3.4.4 Lexical Semantic Enrichment

The μ heuristic is designed to take into account some semantic variations (for instance, *'paper'* is semantically near to *'document'* or *'article'*) for the XML elements names $\mu(n^R, n)$ or between the query terms and the content of the textual nodes $\mu(t_i^R, t)$. It involves the following semantic/terminology relations to set up a request enrichment process or to define custom costs for matching semantic related XML tags:

equivalence : $eq\, arg_1, arg_2, ..., arg_n$
states that $arg_1, arg_2, ...$ and arg_n are considered as identical or equivalent terms in the context of a specific application.

synonymy : $sy\, arg_1, arg_2, ..., arg_n$
where $arg_1, arg_2, ..., arg_n$ belong to the same synonymy class.

hyponymy : $ho\, arg_1(arg_2, ..., arg_n)$
where *ho* specifies that arg_1 has $arg_2, ...$ and arg_n as hyponyms (terms more specialized and more precise than arg_1). For instance

$$ho\, EuropeanCountry\, (UK,\, Germany,\, France,\, Italy...)$$

states that the concept *'EuropeanCountry'* has *'UK'*, *'Germany'*, *'France'*, *'Italy'*..., as specializations.

hyperonymy : $hr\, arg_1(arg_2, ..., arg_n)$
where *hr* states that arg_1 has hyperonyms $arg_2, ...$ and arg_n (i.e. more general terms than arg_1). For instance

$$hr\, Brittany\, (France,\, Celtic\, Area, ...)$$

states that the concept '*Brittany*' has '*France*' and '*Celtic Area*' as generalizations.

Using these relations, it is possible to tune the normalized substitution costs $w_i \in [0..1]$ as follow:

$\mu(n^R, n) = w_{eq}$ if n^R and n are equivalent,

$\mu(n^R, n) = w_{sy}$ if n^R and n belong to the same synonymy class,

$\mu(n^R, n) = w_{hr}$ if n^R is hyperonym of n,

$\mu(n^R, n) = w_{ho}$ if n^R is hyponym of n.

```
(OR
  (SEQ European countries)/1.0
  (SEQ European states)/0.8
  (SEQ United kindom)/0.7
   UK/0.7 France/0.7 Germany/0.7 Italy/0.7 ...
)
```

Figure 3.4: The enriched version of the '*European countries*' request expressed in the SIRIUS query language.

In the current implementation, the weights can be defined by the user. The default values are issued from previous experimentations aiming to integrate natural language techniques into the information retrieval process [57]. We use the following costs to model the XML tags substitution or to account for the conceptual relations between the textual terms of the XML nodes:

$\mu(n^R, n) = w_{eq} = 1.0$ if n^R and n are equivalent,

$\mu(n^R, n) = w_{sy} = 0.8$ if n^R and n are synonyms,

$\mu(n^R, n) = w_{hr} = 0.7$ if n^R is hyperonym of n,

$\mu(n^R, n) = w_{ho} = 0.1$ if n^R is hyponym of n.

More complex heuristics can be used to assign the semantic weights by taking into account:

- an attenuated distance between the concepts associated to n^R and n [188],
- or the global context of occurrence - i.e. the XML path or the semantic context itself.

For instance <*it*> or <*italic*> may be considered as equivalent elements if they have an <*html*> element as ancestor.

For text or linguistic values, specific heuristics $\mu(t_i^R, t)$ (similarly to $\mu(n^R, n)$) can be used. In addition, some rewriting rules defining simple automata are proposed to encode more complex synonymy relations. For instance, '*state*' and '*country*' can be considered as synonyms in the context of adjectives like '*united*' or '*European*'. This allows to extend the synonymy notion to expressions or association of lexical concepts.

The expanded version of the content only request '*European countries*' expressed in the SIRIUS query language (see Section 3.5) is given in Figure 3.4. A file containing '*European states*' can thus be retrieved, with a similarity score depending on the weight given to synonymy, hyponymy and hyperonymy relations.

3.5 The SIRIUS Query Language

3.5.1 Path Constraints

The path constraints can be expressed by using an ordered sequence of elements names separated by slashes "/". They are delimited by square brackets and begin

```
[/newsitem/metadata/dc/]
```

Figure 3.5: Simple path constraint.

and end with a "/". The use of wild-card '*' operator for matching exactly one tag name is allowed. An example of a simple path constraint is shown in Figure 3.5.

Each tag name may be followed by complex constraints on attributes and attributes values – see the next section.

3.5.2 Attributes Constraints

The $\mu(n^R, n)$ heuristic weights the element names semantic similarity, whereas $\alpha(n^R, n)$ manages the fulfillment evaluation of the constraints on attributes.

For numerical values, the use of fuzzy operators inspired by the fuzzy logic [250] may be adequate. For attributes with linguistic values, models based on heuristics like $\mu(n^R, n)$ could be implemented. We note here that $\alpha(n^R, n)$ is a composite function that merges the degrees of satisfaction of the attributes conditions specified on the n^R node.

Some XML elements may have associated unordered pairs of $< attribute, values >$ as shown in the excerpt from Figure 3.6 extracted from the Reuters corpus.

```
<dc element="dc.date.created" value="1996−08−27" />

<dc element="dc.date.published" value="1996−08−27" />
```

Figure 3.6: XML elements with attributes and attributes values.

When searching for information in a specific XML context, it may be quite useful to be able to filter the results using the attribute and/or the attribute values. This may be used in order to rank the retrieved results or to eliminate those that are not corresponding to a strict selection criterion. For instance, the '*date*' field may be used to extract only articles written after '*1996-08-01*'.

The BNF grammar of the attributes and attributes values constraints language is given in Figure 3.7.

This is not an exhaustive list, but gives a basic set of operators that can be easily extended. The degree of fulfillment of the attributes constraints $Ct_i(n)$ for a node n is recursively evaluated using the α heuristic as follows:

$\alpha\left(and\ Ct_1\left(n\right), Ct_2\left(n\right), ..., Ct_k\left(n\right)\right) = Min_i\left\{\alpha\left(Ct_i\left(n\right)\right)\right\}$

$\alpha\left(or\ Ct_1\left(n\right), Ct_2\left(n\right), ..., Ct_k\left(n\right)\right) = Max_i\left\{\alpha\left(Ct_i\left(n\right)\right)\right\}$

$\alpha\left(not\ Ct_i\left(n\right)\right) = 1 - \alpha\left(Ct_i\left(n\right)\right)$

$\alpha\left(exist\ \_attribute\right) = 1$ if the '*_attribute*' exists for the node n, else 0.

$\alpha\left(==\ \_attribute\ \_value\right) = 1$ if '*_attribute*' exists and has the '*_value*', else 0. The same principle applies for $>$, $<$, $\geq$ and $\leq$.

Constraint ::= Op_{n-ary} *Sequence_Of_Constraints* | Op_{unary} *Constraint* | Op_1 *Attribute Value* | Op_2 *Attribute* | Op_3 *Sequence_Of_Values*

Sequence_Of_Constraints ::= *Constraint* | *Constraint Sequence_Of_Constraints*

Sequence_Of_Values ::= *Value* | *Value Sequence_Of_Values*

Op_{n-ary} ::= *and* | *or*

Op_{unary} ::= *not*

Op_1 ::= > | < | >= | <= | == | *near*

Op_2 ::= *exist*

Op_3 ::= *in*

Attribute ::= *Attribute_Name*

Value ::= *integer* | *real* | *char* | *string*

Figure 3.7: BNF grammar for attribute constraints.

```
1 [/newsitem
2  /metadata
3  /dc ( AND ( OR ( (== element dc.date.published)
4                   (== element dc.date.created) )
5            (>= value 1996-08-27) )
6  /]
```

Figure 3.8: Example of complex constraints on attributes and attributes values.

$\alpha\,(in\ \_attribute\ \_value_1\ \ \_value_2 ...\ \_value_j) = 1$ if '*_attribute*' exists for the node n and if its value belongs to the set $\{\ \_value_1\ \ \_value_2 ...\ \_value_j\}$, else 0.

$\alpha\,(near\ \_attribute\ \_value) =\triangle\ (value\,(attribute)\,,\ \_value)$ if '*_attribute*' exists for the node n, else 0.

$\triangle\ (value\,(attribute)\,,\ \_value)$ is a similarity function with values in $[0..1]$ expressing the *proximity* between the value '*_value*' and the value of the attribute '*_attribute*'.

For $\triangle\ (value\,(attribute)\,,\ \_value)$, we use a simple editing distance normalized by the value $Max\,(length\,(\ \_value)\,, length\,(value\,(attribute)))$.

Using these operators, we may express more elaborate constraints on attributes and attributes values associated to an XML element. For instance, the constraint from Figure 3.8 applied to the XML document from Figure 3.1 select documents that have been either published or created after '*1996-08-27*'.

3.5.3 Complex Requests

Complex requests $R_{complex}$ are built using the low-level request R described in Section 3.4.1 and merging operators (boolean or specialized operators). Namely a complex request is a tree of low-level requests R as leafs. Each node supports an operator performing the merging of the descendant results. Currently, the following merging operators are implemented in the system for the low-levels management:

or, and : n-booleans or n-set. $(or\, R\, R')$ merges the set of solutions for R and R'. $(and\, R\, R')$ selects only the answers belonging to both answer sets.

without : this operator can be used to remove solutions from a set. For instance, $(without\, R\, R')$ delivers the set of solutions for R minus the solutions for R'.

seq : merges some of the inverted list to provides a simple sequence management. For instance, $(seq\, warning * error)$ express the search of a sequence of texts items.

same+ : should be related to the or operator. The or operator is a simple set merging operator, whereas $same+$ is a dedicated operator that takes into account the number and the discriminating power of the retrieved terms/elements in the collection. We used a dedicated (see Section 3.4.2.) TFIDF-like function for this purpose (TFIDF stands for Term Frequency / Inverse Document Frequency, see [195]).

in : express boolean contextual relations (in = inside elements with the specified path p^R),

in+ : add structural matching information to the set of solutions. The structural conditions are interpreted vaguely as described in Section 3.4.1. It performs a weighted linear aggregation between the conditions on structure and the set of solutions (see Section 3.4.3).

filter : takes into account the relevance of the global context of occurrence of an element answer. The current implementation works at a document level, but arbitrary levels in the tree hierarchy may be defined. Filter is a binary set operator. It selects from the second set received as argument all the elements coming from document trees containing at least one relevant answer in the first set. The relevance of a returned element is computed as the arithmetic average between its relevance in the second set and the weight associated with the most relevant answer of the same document in the first set.

The system analyzes a complex request $R_{complex}$ and produce a set of weighted results. Let $r(R_{complex}) = \{(n_i, w_i)\}$ the set of weighted results produced by the system, where n_i is a an XML element node and $w_i \in [0..1]$ a weight showing its relevance to the request. Let R_i be a complex request $R_{complex}$, or a simple (low level) R request. The similarity computation for a complex request involves modifications of the relevance associated with a result element (i.e. $w_i \in [0..1]$) and is performed recursively starting at the leafs of the request tree:

$$r(or\ (R_0, ..., R_n)) = \{(n_i, w_i)\} \text{ with } w_i = Max_k\,(w_k) \text{ where } (n_i, w_k) \in \bigcup_j^n r\,(R_j);$$

$r(and\,(R_0,...,R_n)) = \{(n_i,w_i)\}$ with $w_i = Min_k\,(w_k)$ where $(n_i,w_k) \in \bigcup_j^n r\,(R_j)$;

$r(without\,(R_0,R_1)) = \{(n_i,w_i)\}$ where $w_i \in r\,(R_0)$ and $w_i \notin r\,(R_1)$;

$r(seq\,(t_0,t_1,...,t_n)) = \{(n_i,w_i)\}$ where $w_i = 1$ if the request terms $t_0,t_1,...,t_n$ occurs in sequence and belong to the same context/leaf – i.e. $\{t_0,t_1,...,t_n\} \in n_i$ – else 0.

$r(in\,(p^R,R_0,...,R_n)) = \{(n_i,w_i)\}$ with $w_i = Min\left\{Min_k\,(w_k)\,,\Delta\left(p^R,p\,(n_i)\right)\right\}$ where:

- $p(n_i)$ the XML context of the XML element n_i ; and
- $\Delta\left(p^R,p\,(n_i)\right) = 1$ if $p^R \equiv p(n_i)$, 0 if not; and
- $(n_i,w_k) \in \bigcup_j^n r\,(R_j)$;

$r(in+\ (p^R,R_0,...,R_n)) = \{(n_i,w_i)\}$ with

$w_i = (1-\lambda_2)\cdot Min_k\,(w_k) + \lambda_2 \cdot \Delta\left(p^R,p\,(n_i)\right)$ where:

- $p(n_i)$ the XML context of the element n_i ; and
- $\Delta\left(p^R,p\,(n_i)\right) = \sigma(p^R,p\,(n_i))$ representing the structural similarity between the two XML contexts (see Section 3.4.1); and
- $\lambda_2 \in [0..1]$ a parameter used to emphasize the importance of the structural versus textual content matching; and
- $(n_i,w_k) \in \bigcup_j^n r\,(R_j)$;

$r(same+\ (R_0,...,R_n)) = \{(n_i,w_i)\}$ where $w_i = rsv(co^R|n_i)$ (see Section 3.4.2) and $(n_i,w_i) \in \bigcup_j^n r\,(R_j)$;

Let n^D be a result element descendant of document D.

$r(filter\,(R_0,R_1)) = \{(n_i,w_i)\}$ with $w_i = \frac{w_j + Max_k(w_k)}{2}$ where

$\forall_D\left(\forall_i\left(n_i^D,w_j\right) \in r\,(R_1)\ \textbf{and}\ \forall_k\left(n_k^D,w_k\right) \in r\,(R_0)\right)$.

3.6 Prototype Implementation

This section reports on SIRIUS, a lightweight indexing and search engine for XML documents. The retrieval approach implemented is document oriented. It involves an approximate matching scheme of the structure and textual content. Instead of managing the matching of whole XML trees, SIRIUS splits the documents structure in a set of paths. In this view, the request is a path-like expression with conditions on the attribute values. We present and discuss the system architecture together with its main functionalities and characteristics.

3.6.1 System General Architecture

The general organization of the system is shown in Figure 3.9. During the indexing process, the structure of the XML documents is extracted by using the Ælfred XML parser[7]. The extracted paths are used in order to construct the inverted lists and stored within repositories based on the QDBM[8] and Berkeley[9] database libraries. The XML contexts are coded to serve in the analysis phase of the retrieval process. The requests are expressed in a specialized language and their content enriched based on a thesaurus of semantic rules. The structural part of the requests is compared with the indexed structures using a mechanism inspired by the techniques for computing editing distances. The weights of the cost matrix are adapted to the search process and allow matching the elements and attributes specified in the request with the structure of the indexed XML documents. The matching process may eventually include information fetched in the enrichment process. In perspective, a relevance feedback algorithm could be used to tune the XML contexts cost matrix.

3.6.2 GUI

In this section we present the main characteristics of a graphical user interface (GUI) aiming at providing end users with focused access to relevant information. We show an example of a straightforward web implementation of the SIRIUS *system oriented* graphical interface in Figure 3.10.

Similarly to popular search engines like Google or Yahoo! the interface displays a ranked lists of elements sorted accordingly to their relevance to the user request. Each result refers to the XML document from which it was extracted and its position within the document as an XML fragment. The relevance score is indicated both as a color code and a numerical value. To provide the users with an indication about the reasons for which the elements were selected, query dependent text snippets are associated with each retrieved element and the searched terms are highlighted.

Within the above example, the output of the XML retrieval system was assumed to be a ranked list of XML elements, ordered by their presumed relevance to the query, whether overlapping elements were allowed or not. However, user studies [220] suggested that users were expecting to be returned elements grouped per document, and to have access to the overall context of an element. For this purpose we rank and display the relevant documents (the fetching phase) and then the elements within the fetched documents (the browsing phase). In the fetching phase, documents had to be

[7]The Ælfred XML Parser `http://saxon.sourceforge.net/aelfred.html`
[8]QDBM: Quick Database Manager `http://qdbm.sourceforge.net/`
[9]Oracle Berkeley DB `http://www.oracle.com/database/berkeley-db/db/`

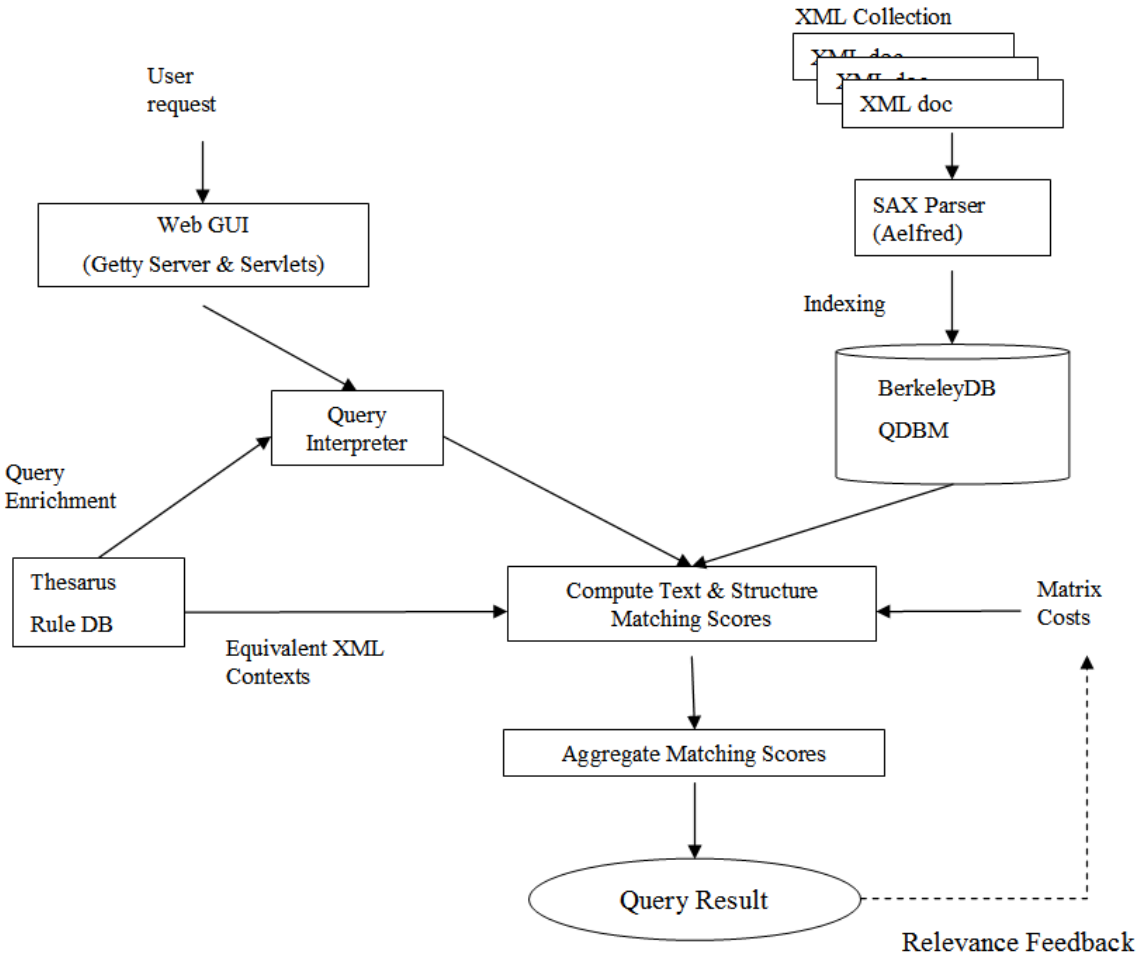


Figure 3.9: SIRIUS General Architecture

ranked according to their global relevance to the user request. In the browsing phase, ranking had to be done according to the relevance of the retrieved elements in the document compared to other elements in the same document. However, for presentation purposes, the selected relevant elements are returned in their original document order and they are not allowed to contain overlapping information. The selection process may imply the use of a minimum threshold value for the elements relevance score, a maximum number of retrieved elements per document; or both. In Figure 3.11 we show a snapshot of the SIRIUS *user oriented* graphical interface. In the example we use a limit of five relevant elements per document and a *Max* function to compute the documents global relevance score from the scores of its components.

3.7 Conclusions

Our main contributions are:

- We have proposed specific data structures dedicated to the indexing and retrieval of information elements embedded within heterogeneous XML databases. The indexing scheme is well suited to the characterization of various contextual searches, expressed either at a structural level or at an information content level.

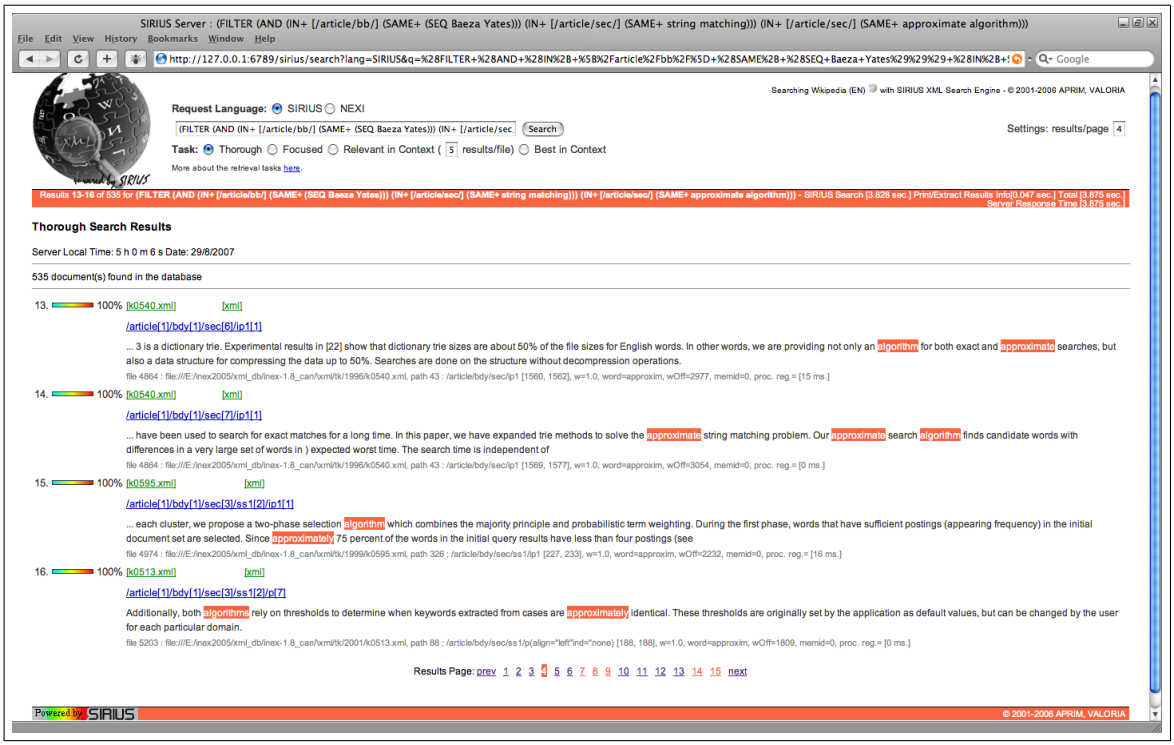


Figure 3.10: SIRIUS system oriented graphical interface.

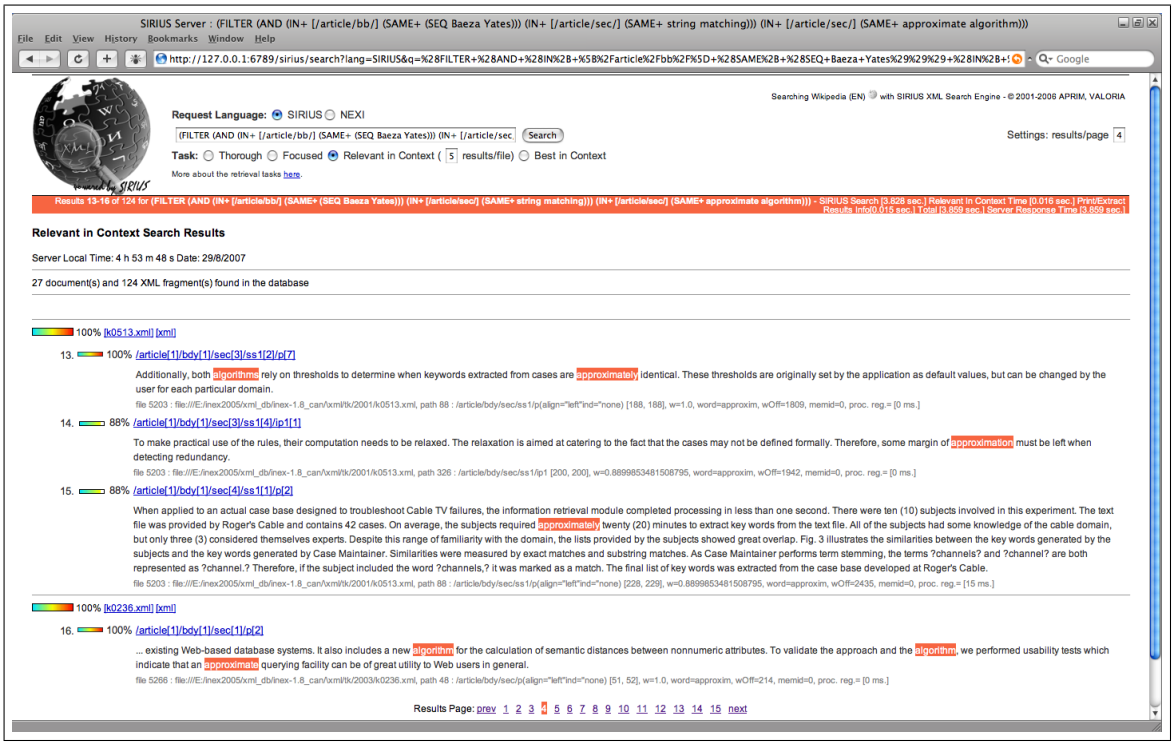


Figure 3.11: SIRIUS user oriented graphical interface.

- We have implemented and evaluated a new search mechanism based on a set of tree paths matching that involves a modified Levenshtein editing distance and information fusion heuristics.
- We have developed a fully functional XML IR system. The implementation that is described highlights the mixing of structured information presented as field/-value instances and free text elements.
- We have experimentally evaluated the proposed approach and the system implementation within the INEX 2005 and INEX 2006 evaluation campaigns with encouraging results. Details about the experimental setup, the retrieval tasks and the evaluation results can be found in the next three chapters.

Chapter 4

Experimental Evaluation Framework

This chapter is dedicated to the SIRIUS experimental evaluation within the INEX evaluation campaign. We present the INEX evaluation benchmark and how we tuned the SIRIUS XML IR system to participate to INEX 2005 and INEX 2006 ad hoc retrieval tasks.

Contents

4.1 Introduction

Much of the research and development in information retrieval is aimed at improving the effectiveness and efficiency of retrieval.

In a system designed for data retrieval, the *efficiency* aspects are fundamantal and are usually measured in terms of the computer resources used, such as the storage space and the C.P.U. time.

In a system designed for providing information retrieval, the user query request is inherently vague, the retrieved documents are not exact answers and have to be

ranked according to their relevance to the query. Such relevance ranking introduces a component which is not present in data retrieval systems and which requires to evaluate the *effectiveness* of the answer set [26].

Such an evaluation is usually based on a test reference collection and on an evaluation measure. The test reference collection consists of a collection of documents, a set of example information requests, and a set of relevant documents (provided by experts) for each example of information request. Given a retrieval strategy, the evaluation measure quantifies the *similarity* between the set of documents retrieved and the set of relevant documents provided by the experts.One of the most consistent and popular benchmark for evaluating information retrieval systems is the TREC collection for *Text REtrieval Conference*[1]. The goal of the TREC conference series is to encourage research in information retrieval from large text applications by providing a large test collection, uniform scoring procedures, and a forum for organizations interested in comparing their results.

Typically, the retrieval systems compare their approaches based on their appropriate retrieval of objects (documents, concepts, etc.) that should be retrieved and the non-retrieval of those that should not. The effectiveness of the information retrieval systems is commonly measured in terms of precision and recall. The *precision* is the ratio of the number of relevant documents retrieved to the total number of documents retrieved, and *recall* is the ratio of the number of relevant documents retrieved to the total number of relevant documents (both retrieved and not retrieved).

For evaluating the effectiveness of content-oriented semi-structured documents retrieval, specialized test collections, tasks and metrics were introduced by the INEX – for *INitiative for the Evaluation of XML Retrieval*[2]) – evaluation campaigns started in 2002.

This chapter is dedicated to the SIRIUS experimental evaluation within the INEX 2005 and INEX 2006 evaluation campaigns. We present the INEX evaluation benchmark and relate on our experience on adapting and using the SIRIUS XML IR system for participating at the INEX ad hoc retrieval tracks.

4.2 INEX Evaluation Campaigns

The *INitiative for the Evaluation of XML Retrieval* (INEX) is an international campaign that provides a benchmark in the form of large test collections and appropriate scoring methods for evaluating the effectiveness of content-oriented XML retrieval systems.

Evaluation is carried out using test collections assembled specifically for evaluating particular retrieval tasks. A test collection consists of a document collection, a set of user requests (i.e. topics) and relevance assessments. The characteristics of traditional test collections have been adjusted to appropriately evaluate XML retrieval effectiveness: the document collection comprises documents marked up in XML, the topics specify requests relating to both content and structure, and the relevance assessments are made at element level. In addition, relevance is measured such that it appropriately quantifies the system's ability to return the correct granularity of XML elements.

[1]Text REtrieval Conference (TREC) `http://trec.nist.gov/`

[2]INitiative for the Evaluation of XML Retrieval (INEX) `http://inex.is.informatik.uni-duisburg.de/`

Since its start in 2002, INEX has provided a forum for the discussion of XML retrieval related issues. In the past five years INEX has constantly grown and evolved to incorporate new retrieval tasks, scenarios, and collections.We make use of the INEX 2005 and INEX 2006 data sets to evaluate our retrieval approach. This section briefly describes the settings used in INEX 2005 and INEX 2006 regarding collection, tasks, topics, assessments, and evaluation metrics. We focus our explanation on the retrieval tasks used to evaluate the approach presented in this thesis. Detailed information on these and other tasks can be found in the workshop proceedings [69, 71].

4.2.1 Document Collections

The IEEE Collection. The *inex-1.8* document collection contains 16819 articles taken from 24 IEEE Computer Society journals, covering the period of 1995-2004. The total size of the source files in their canonical form is about 750 MB. The collection contains 141 different tag-names composing 7948 unique XML contexts by ignoring the attributes and the attributes values. The maximum length of an index path is 20, while the average length is 8. These statistics are computed from the viewpoint of the retrieval system. That is, we use the XML tag equivalence classes[3] in concordance with [210]. Also, the XML contexts associated to empty elements or containing only stop words do not count in our statistics.

The structure of an XML document extracted from the INEX IEEE document collection is given as example in Figure 4.1.

The Wikipedia Collection. The INEX 2006 document collection for the ad hoc track is based on the main English collection of the Wikipedia XML Corpus[4] [59]. The INEX 2006 collection consists of 4.6 GB of text marked-up in XML. The collection is made of 659,388 English articles extracted from the Wikipedia[5] project. The structural part of the collection corresponds to the Wikipedia templates (about 5000 different tags). On average an article contains 161.35 XML nodes, where the average depth of an element is 6.72.

An excerpt of an XML document extracted from the Wikipedia XML Corpus is given as example in Figure 4.2 while its XML tree representation is shown in Figure 4.3.

4.2.2 Topics

The INEX campaigns distinguished two types of topics: Content-Only (CO) topics and Content-And-Structure (CAS) topics. These topic types reflect two types of users with

[3]We create structural equivalence classes for the tags defined as interchangeable in a request. In the INEX IEEE collection there are several tags used interchangeably (for historical paper-publishing reasons). Tags belonging to the following groups are considered to be equivalent and can be used interchangeable in a query.

Paragraphs: ilrj, ip1, ip2, ip3, ip4, ip5, item-none, p, p1, p2, p3

Sections: sec, ss1, ss2, ss3

Lists: dl, l1, l2, l3, l4, l5, l6, l7, l8, l9, la, lb, lc, ld, le, list, numeric-list, numeric-rbrace, bullet-list

Headings: h, h1, h1a, h2, h2a, h3, h4

[4]Wikipedia XML Corpus `http://www-connex.lip6.fr/~denoyer/wikipediaXML/`
[5]Wikipedia - The Free Encyclopedia `http://en.wikipedia.org/`

```
<?xml version="1.0" encoding="ISO
    −8859−1"?>
<article>
  <fm>     ...
    <ti>IEEE Transactions on</ti>
    <atl>Tries for  ... </atl>
    <au>
      <fnm>Ricardo</fnm>
      <snm>Baeza−Yates</snm>
      <aff>University of ...</aff>
    </au>
    <au>...</au>
     ...
  </fm>
  <bdy>
   <sec>
     <st>String Matching</st>
     <p>...approximate algorithm...</p>
       ...
   </sec>
   <sec>
     <st>...</st>
      ...
     <ss1>...</ss1>
     <ss1>...</ss1>
      ...
   </sec>
    ...
  </bdy>
  <bm>
    <bib>
      <bb>
        <au>...</au>
        <ti> ... </ti>
         ...
      </bb>
       ...
    </bib>
  </bm>
</article>
```

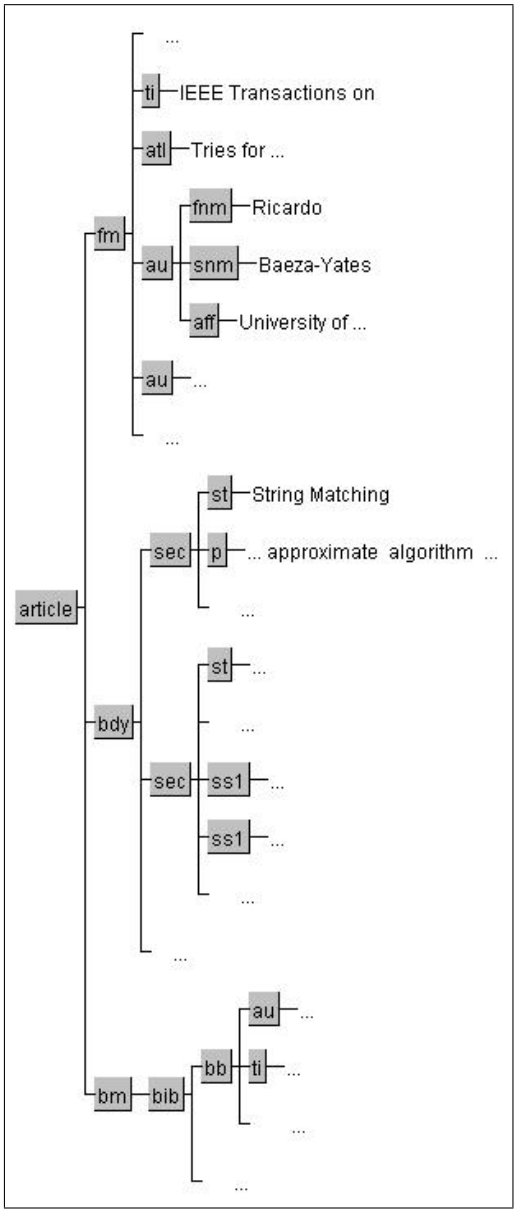


Figure 4.1: An excerpt of an XML document extracted from the INEX IEEE document collection associated with its XML tree representation.

```
<?xml version="1.0" encoding="ISO−8859−1"?>
<article>
  <name id="13772">History of Poland</name>
  <body>
    ...
    <section>
      <title>Early history of Poland...</title>
    </section>
    <section>
      <title>The Jagiellon Era...</title>
    </section>
    <section>
      <title>The Polish−Lithuanian Commonwealth...</title>
    </section>
    <section>
      <title>Partitioned Poland (
        <collectionlink xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple"
            xlink:href="35799.xml"> 1795 </collectionlink>
        –
        <collectionlink xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple"
            xlink:href="34594.xml"> 1918 </collectionlink>
        )
      </title>
      ...
      <p>Polish independence ...</p>
      <p>Following the
        <collectionlink xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple"
            xlink:href="10581.xml">French</collectionlink>
        emperor
        <collectionlink xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple"
            xlink:href="69880.xml">Napoleon I</collectionlink>
        ‘s defeat of Prussia, a Polish state was again set up in
        <collectionlink xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple"
            xlink:href="34745.xml">1807</collectionlink>
        under French tutelage as the
        <collectionlink xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple"
            xlink:href="366190.xml">Duchy of Warsaw</collectionlink>
        ...
      </p>
      <p>With Napoleon‘s defeat...</p>
    </section>
    ...
  </body>
</article>
```

Figure 4.2: An excerpt of an XML document extracted from the Wikipedia XML Corpus.

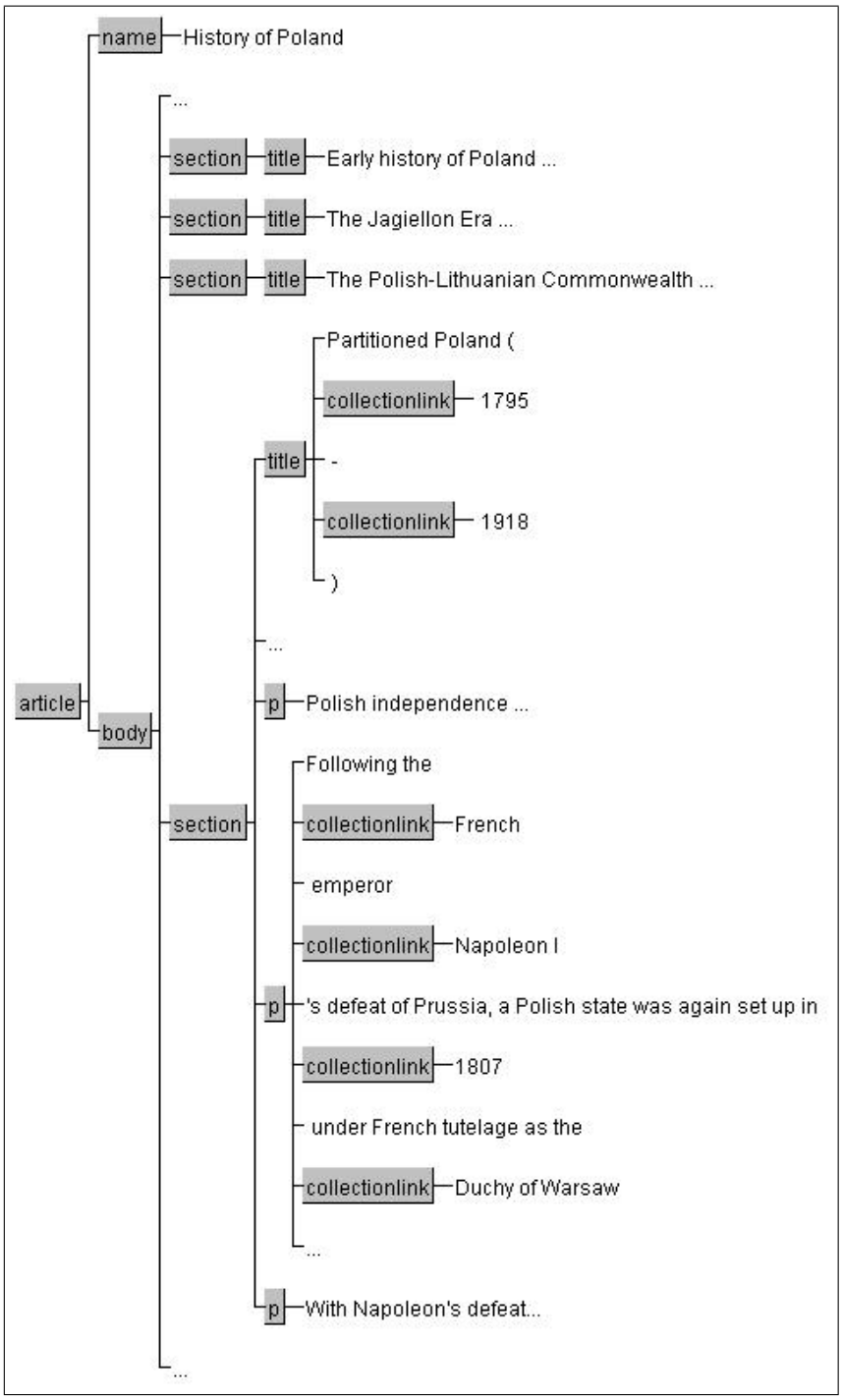


Figure 4.3: The XML tree of the document from Figure 4.2.

varying levels of knowledge about the structure of the searched collection. The first type relates to ignorant users who do not have any knowledge of the document structure or who choose not to use such knowledge. The latter type relates to users that are likely to use any knowledge about the document structure that they may possess.

Content-Only (CO) topics contain search terms as in the traditional requests used in information retrieval test collections. These topics do not include any specific reference to the document structure.

Content-And-Structure (CAS) topics are requests that contain conditions referring both to content and structure of a document. CAS topics are topic statements that contain explicit references to the XML structure, and explicitly specify the contexts of the user's interest (e.g. target elements) and/or the contexts of certain search concepts (e.g. containment conditions). More precisely, a CAS query contains two kinds of structural constraints: where to look (i.e. the support elements), and what to return (i.e. the target elements).

An INEX topic consists of *title*, *description* and *narrative* fields. For CO topics, the *title* is a sequence of terms. For CAS topics, the *castitle* is expressed using the NEXI query language [226], which is a variant of XPath defined for content-oriented XML retrieval evaluation.

Both CO and CAS titles are made of terms, i.e. words or phrases, where the latter are encapsulated in double quotes. Furthermore the terms can have either the prefix '+' or '-', where '+' is used to emphasize an important concept, and '-' is used to denote an unwanted concept. An example of a CAS topic is given in Figure 4.4. This is a stand alone CAS topic that has no CO equivalence as the *title* field is empty.

INEX 2005. At INEX 2005 the CO topics were extended to *Content-Only+Structure (CO+S)* topics. The aim was to enable the comparison of system performance across two retrieval scenarios (on the same topic): when structural hints are taken into account and when these hints are ignored [118].

Content-Only+Structure (CO+S) topics are CO topics that include an additional field called *castitle*, which is a representation of the same information need contained in the *title* field of the CO topic but including additional knowledge in the form of structural constraints.

For the ad hoc track of the INEX 2005 campaign a total of 40 CO and 47 CAS topics were selected by the organizers.

None of the 47 official CAS topics contained references to attributes or attributes values. This may be explained by the fact that the INEX IEEE document collection is considered to have no attribute or attribute value with practical interest for a real end user – see "A Note on Attributes" in [226].

INEX 2006. In the INEX 2006 campaign, the topics have a uniform format (see Figure 4.5). An official INEX 2006 topic must contain a CO formulation (i.e. the *title* element must be completed) and may have a CAS formulation as required for the INEX 2005 CO+S topics. We note here that a CO formulation can be translated with no loss of semantics into a CAS formulation [226]. The reverse is not true. The

```
<?xml version="1.0" encoding="ISO−8859−1" ?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd" >
<inex_topic topic_id="280" query_type="CAS" ct_no="137" >
  <InitialTopicStatement>find sections describing ways to use approximate string
      matching.</InitialTopicStatement>
  <title></title>
  <castitle>
    // article[about(.//bb, Baeza−Yates) and about(.//sec, string matching)]//sec[about(.,
        approximate algorithm)]
  </castitle>
  <description>find sections about approximate algorithms in works about string matching
      citing Baeza−Yates.</description>
  <narrative>I am interested in fast ways to use approximate string matching in the
      context of text searching. I hope to increase the flexibility of a search engine I am
      implementing − and am looking for novel ways to solve the string edit problem. I
      noticed that Baeza−Yates has published extensively on text searching and expect
      works citing Baeza−Yates to be more relevant than works not citing him. Sections
      focusing on approximate string matching algorithms will be considered relevant.
  </narrative>
</inex_topic>
```

Figure 4.4: INEX 2005 CAS topic 280.

```
<?xml version="1.0" encoding="ISO−8859−1"?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd">
<inex_topic topic_id="289" ct_no="2">
  <title>
    emperor “Napoleon I“ Polish
  </title>
  <castitle>
    //∗[about(., emperor “Napoleon I“ Polish)]
  </castitle>
  <description>I want to know everything about the emperor Napoleon I and Polish people.
      </description>
  <narrative>Polish history is closely related to Napoléon I of France. But also, Napoléon I
      knew very well some Polish people (among which Marie Laczynska and the Prince
      Poniatowski). I want to know about the big History (how Napoléon had influence on
      the history of Poland) and the “small“ history (Napoléon mistress, marshals, etc.). My
      aim is simply to know better the ins and outs of the question, and to understand how
      much personal relationships of Napoleon influenced his behaviour as a head of state.
      Relevant elements should make me able to give a summary of this subject.</narrative
      >
  <ontopic_keywords>“duchy of Warsaw“, “Marie Laczynska“, “countess Malewski“, Eylau,
      “Prince Poniatowski“, Russia</ontopic_keywords>
</inex_topic>
```

Figure 4.5: INEX 2006 topic 289.

```
<?xml version="1.0" encoding="ISO−8859−1"?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd">
<inex_topic topic_id="406" ct_no="198">
  <title>
    book architecture
  </title>
  <castitle>
    //template[about(.//@name,book reference)]//∗[about(.,architecture)]
  </castitle>
  <description>Show me books about architecture</description>
  <narrative>After coming home from a trip to venice, the user is intrigued to read more
      about architecture and city planning. He wants to find books about architecture and is
      therefor looking for references including the author and title. The user is not interested
       in landscaping and even less in computing.
  </narrative>
  <ontopic_keywords>+house</ontopic_keywords>
</inex_topic>
```

Figure 4.6: INEX 2006 topic 406.

INEX 2006 format also provides the possibility to add relevant *ontopic_keywords* for a request.

From the set of 125 official INEX 2006 topics on the Wikipedia collection, only topic 406 includes a constraint on attribute and attribute values – see Figure 4.6.

4.2.3 Pertinence Judgments

At INEX 2005 the relevance judgments are given in two different dimensions: *exhaustivity* (E) and *specificity* (S). The exhaustivity dimension reflects the degree to which an element covers a topic and the specificity dimension reflects how focused the element is on that topic. Thus, to assess an XML element, participants are asked to highlight the relevant parts of each element (specificity) and to use a three-level scale [0, 1, 2] to define how much of the topic that element covers (exhaustivity). For later usage in the evaluation metrics, the specificity dimension is automatically translated to a value in a continuous scale [0...1], by calculating the fraction of highlighted (relevant) information contained by that element. The combination of the two dimensions is used to quantify the relevancy of the XML elements. Thus, a highly relevant element is one that is both, highly exhaustive and highly specific to the topic of request. The following year, INEX 2006 dropped the exhaustivity dimension, and relevance was defined only along the specificity dimension.

4.2.4 Retrieval Tasks

The main retrieval task to be performed in INEX is *ad hoc retrieval*. This can be described as a simulation of how a library might be used, and involves the searching of a static set of documents using a set of topics. In INEX, the library consists of XML documents, the queries may contain both content and structural conditions, and in response to a query, arbitrary XML elements may be retrieved.

Two ad hoc retrieval sub-tasks that depend on how the structural constraints are expressed were identified. In the content-only (CO) sub-task, it is left to the retrieval

system to identify the most appropriate XML elements to return to the user. For the CAS sub-task,

the most specific document components, which are relevant to the topic of request and match, either strictly or vaguely, the structural constraints specified in the CAS topic, had to be retrieved. Three different strategies have been defined, depending on the preferred output format of an XML retrieval system.

The Focused Task. In the focused task, the goal is to find the most exhaustive and specific elements on a path. Once the element is identified and returned, none of the remaining elements in the path should be returned. In other words, the result list should not contain overlapping elements. This is a user-oriented task since the underlying assumption is that users do not want to see the same information twice.

The Thorough Task. In the thorough task, the aim is to retrieve all highly exhaustive and specific elements in the collection, regardless whether they overlap or not. Hence, retrieval systems are simply asked to return elements ranked by their relevancy to the topic of request. This is a system-oriented task and its goal is to evaluate whether retrieval systems are capable of locating all the relevant elements in the collection.

The Fetch & Browse Task. In a *fetch & browse* strategy, we assume that a user is interested in highly relevant elements contained within highly relevant documents. The fetch & browse task aim is to first identify relevant documents (the fetching phase), and then to identify the most exhaustive and specific elements within the fetched documents (the browsing phase). In the fetching phase, documents had to be ranked according to how exhaustive and specific they were. In the browsing phase, ranking had to be done according to how exhaustive and specific the relevant elements in the document were, compared to other elements in the same document [122].

In 2005, no explicit constraints were given regarding whether returning overlapping elements within a document was allowed. The rationale was that there should be a combination of how many documents to return, and within each document, how many relevant elements to return.

In 2006, the same task, renamed the *relevant in context sub-task*, required systems to return for each article an unranked set of non-overlapping elements, covering the relevant material in the document. In addition, a new task was introduced in 2006, the *best in context sub-task*, where the aim was to find the *best-entry-point*, here a single element, from where a user should start reading articles with relevant information. This sub-task can be viewed as the extreme case of the fetch & browse approach, where only one element is returned per article [122].

4.2.5 Evaluation Measures

Since its launch in 2002, INEX has been challenged by the issue of how to measure an XML information access system's effectiveness. Due to the lack of an atomic predefined unit of retrieval as well as the increased richness of the user's interaction with the system (i.e., browsing), users have access to other, structurally related components from a returned result element. *Near-misses* are elements, which may be themselves not exactly relevant to the user's query, but from where users can access relevant

content. Starting from this observation, XML retrieval approaches should be partially rewarded by the evaluation measure for finding such elements, as it is still better to return near-misses than irrelevant elements [16].

Overlapping is another specific issue to be considered with care when evaluating XML retrieval approaches. In INEX, the recall-base (the set of relevant elements for each given query) consists of a large proportion of overlapping elements (if an element is relevant, so is its parent element). This so-called overpopulated recall-base can lead to misleading effectiveness results because the recall-base contains more relevant elements than an ideal system should in fact retrieve. In fact, perfect recall can only be reached by systems that return all the relevant elements of the recall-base, including all the overlapping elements [16].

To manage these specific challenges, the INEX 2005 and INEX 2006 evaluation campaigns used the eXtended Cumulated Gain (XCG) metrics [108]. These measures are specially designed for evaluating XML element retrieval and therefore, unlike traditional IR evaluation measures, they address XML element retrieval issues such as overlap and near-misses. In this section, we briefly outline their main characteristics, and refer to [69, 71] and [122] for a more detailed description.

The XCG metrics are an extension of the Cumulated Gain (CG) metrics [98] that consider dependency between XML elements (e.g., overlap and near-misses). The XCG metrics include a *user-oriented* measure called normalized extended cumulated gain (nxCG) and a *system-oriented* measure called effort-precision/gain-recall (ep/gr).

User-oriented measures allow to reason about a system's ability to satisfy users. These metrics typically focus on the early ranks of a system's output as users are more likely to limit their search to these results.

System-oriented measures allow system developers to obtain an overall picture of the system retrieval performance.

In comparison to the common IR evaluation measures, nxCG corresponds to a precision measure at a fixed cut-off, and ep/gr provides a summary measure related to mean average precision (MAP) .

To apply the metrics, the two relevance dimensions, exhaustivity and specificity, are mapped to a single relevance scale by using two different quantization functions. These functions model different user preferences. The *strict* one models a user who only wants to see highly relevant elements (E = 2 , S = 1) and the *generalized* one allows different degrees of relevance. More formally:

$$quant_{strict}(e,s) = \begin{cases} 1 & if\, e = 2\, and\, s = 1, \\ 0 & otherwise. \end{cases}$$

$$quant_{gen}(e,s) = e \cdot s$$

For example, a strict quantisation function is used to evaluate retrieval methods with respect to their capability of retrieving highly relevant elements. A generalised function is used to credit retrieved elements according to their degree of relevance, thus also allowing to reward less relevant elements. The latter is important as it allows considering near-misses when calculating effectiveness performance.

In INEX 2006, as the exhaustivity dimension was dropped, the quantization function simply maps an element to its specificity value s. Only the results using the generalized function were reported.

Reporting Results. For the experiments presented in this thesis, we report the following numbers:

nxCG[i]: For a given rank i, the value of nxCG[i] reflects the relative gain the user accumulated up to that rank, compared to the gain he/she could have obtained if the system would have produced the optimum best ranking.

MAep: is the uninterpolated mean average effort-precision and is calculated by averaging the effort-precision values obtained for each rank where a relevant document component is returned.

4.3 SIRIUS @ INEX

In this section we relate on our experience on adapting and using the SIRIUS XML IR system in the INEX 2005 and INEX 2006 ad hoc retrieval tracks.

4.3.1 Indexing the INEX 2005 and INEX 2006 Collections

Indexing the IEEE Collection. The collection is pre-processed by removing the *volume.xml* files and transforming all the XML documents in their canonical form[6] .

At indexing time, the least significant words are eliminated using a stop list. The terms are stemmed using the Porter algorithm [180]. We index only the ALPHANUMERIC words as defined in [226] (like *iso- 8601*). We did not index the attributes, the attributes values, and empty XML elements. This allowed important performance gains both in indexing and querying time as well as disk space savings. The index size is about 1.28 times the size of the initial collection.

The index model – see Section 3.3 – is implemented on top of the Berkeley DB library[7] using a combination of BTrees and Hashtables structures.

Figure 4.7 shows the evolution of the indexing time using a computer with a P IV at 2.4 GH processor and 1.5 GB of RAM in function of the volume of the indexed data set. The time required to create the inverted lists for about 750 MB of data (the inex-1.8 IEEE collection in its canonical form) is about 38 Mn. The index size is about 1.28 times the size of the initial collection. This quasi-linear evolution shows that the indexing of large document XML databases is a conceivable objective in the given hypothesis. Nevertheless, we must underline the fact that taking into account the attributes and their values multiplies the number of structural contexts to be indexed (7948 in the current configuration). This may have as consequence a significant increase in the indexing time and space.

Indexing the Wikipedia Collection. SIRIUS has the capability of using *indexing profiles* for a specific collection. The indexing profiles are composed of rules defining how the structure and the content of each specified XML tag should be indexed. By default, all the non empty XML tags are fully indexed. Using these profiles we may decide or not to index the attributes associated to a given tag, to index only the content of the *presentation tags* or *jump tags* – see Section 2.1.3 on page 12, or to completely ignore some *logical tags* for a specific collection. The use of indexing profiles may

[6]Canonical XML Processor `http://www.elcel.com/products/xmlcanon.html`

[7]Berkeley DB Library `http://www.oracle.com/database/berkeley-db/index.html`

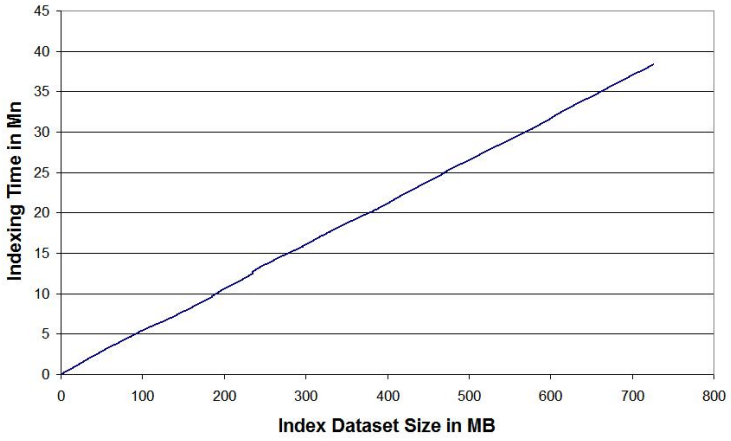


Figure 4.7: Indexing Time for the inex-1.8 IEEE Collection.

	Ignore tags	**Ignore tag attributes**
Presentation tags	emph2, emph3, emph4, sup	table, tr, td, font
Jump tags	collectionlink, unknownlink, outsidelink, languagelink	
Logical tags	title, name, image, caption	

Table 4.1: Indexing rules for the Wikipedia collection

reduce significantly the volume of the requested disk space for the index and improves the system performances both in indexing and retrieval time.

We use the rules shown in Table 4.1 to index the Wikipedia collection. This indexing profile was manually defined as we assumed that the jump and presentation tags contained information that should not be retrieved out of their context. The logical tags <*name*>, <*title*> and <*caption*> are of a particular importance for the Wikipedia collection, as this will ensure that the <*title*> of a <*section*> will always be retrieved with the <*section*> itself, that the <*name*> of an <*article*> will be retrieved with the whole <*article*>, and that the <*caption*> of a <*figure*> or <*table*> will be retrieved only associated to the element to which they are referring to.

The Wikipedia collection is processed using an XML SAX parser and standard methods for stop words removal and stemming. At indexing time, the most frequent words are eliminated using a stop list. The XML elements containing no valid textual content after stop words removal are not indexed. The index terms are stemmed using the Porter algorithm [180].

The inverted file index was constructed in parallel on a 4 computer cluster by using a *Physical Document Partitioning* approach [26]. The index time has a linear dependence relative to the number of indexed documents and unique attribute values. The total size of the index is situated between 4 GB and - 6.6 GB (i.e between 86% and 122% of the initial database size). The difference shows the storage space savings that can be attained when the above indexing rules are used.

$$\begin{array}{lll} \delta_L(//article//bb, & /article/bm/bib/bibl/bb/au/snm &) = 0 \\ \delta_L(//article//bb, & /article/bm/app/bib/bibl/bb/au/snm &) = 0 \\ \delta_L(//article//bb, & /article/fm/au/snm &) = 1 \end{array}$$

Figure 4.8: Example of distances between structural contexts.

4.3.2 Structural Weighting Scheme for INEX

The NEXI language [226] allows only the descendant relationship between the nodes in a path. Therefore the XML path expressed in the request is interpreted as a *subsequence* of an indexed path, where a subsequence need not necessary consist of contiguous nodes.

To model this, we relaxed in [177] the weights of the path editing distance $\delta_L(p_i^D, p^R)$ – see Section 3.4.1 in order to allow node deletions in the indexed paths p_i^D without any penalty:

Deletion: a node n in p_i^D is deleted for a cost $C_{del}(n) = C_{del}^{min} = 0$,

Insertion: a node n is inserted in p_i^D for a cost $C_{ins}(n) = C_{ins}^{max} = 1$, and

Substitution: a node n^R in the path p^R is replaced by a node n from p_i^D for a cost $C_{subst}(n^R, n)$. Since a node n not only stands for an XML element, but also for attributes or attribute constraints Ct, we compute $C_{subst}(n^R, n)$ as follows:

$$C_{subst}(n^R, n) = \begin{cases} C_{subst}^{max} = 1 & if \ \left(n^R \neq n\right) \\ C_{subst}^{Ct} = \frac{1}{2} & if \ \left(n^R = n\right) \wedge \left(\exists\, Ct^i \in \left\{Ct\left(n^R\right)\right\},\ Ct^i\left(n\right) = false\right) \\ C_{subst}^{min} = 0 & if \ \left(n^R = n\right) \wedge \left(\forall\, Ct^i \in \left\{Ct\left(n^R\right)\right\},\ Ct^i\left(n\right) = true\right) \end{cases}$$

where:

- C_{subst}^{max} is the substitution cost for the nodes n^R and n – without taking into account the attributes conditions and without allowing any semantically relaxation for the tag names (i.e. full substitution),
- C_{subst}^{Ct} represents the cost associated with a valid match at element level but unsatisfied attribute constraints Ct stated in the request for n^R that should apply to the attributes and attributes values in n (i.e. attribute substitution),
- C_{subst}^{min} stands for a perfect match, both at element and attributes levels. The XML tags are strictly equivalent and all the attribute constraints Ct stated in the request for the node $n^R$are satisfied by the attributes and attributes values in n (i.e. perfect matching, no substitution).

To illustrate the alignment mechanism we show in Figure 4.8 the distances between the path requested in the INEX 2005 CAS topic 277 *//article[about(.//bb, Baeza-Yates)]* searching works citing Baeza-Yates and three paths extracted from the IEEE collection (an excerpt showing the structure of the documents in the collection is given in Figure 4.1).

In the first two cases the requested path *//article//bb* is a subsequence of the indexed paths and therefore the editing distance is 0 independently of the fact that

the paths have a different number of nodes. In the last case, where *Baeza-Yates* is the author of the article, the editing distance is 1 highlighting the mismatch of the requested *bb* node from the indexed path.

The weights used to compute the structural similarity relate to an end user having *precise* but *incomplete* information about the XML tags of the indexed collection and about their ancestor-descendant relationships.

The structural similarity takes into account the order of occurrence of the matched nodes and the number of nodes with no matching in the request. It heavily penalizes any mismatch relatively to the information provided by the user but it is independent to mismatches/extra information extracted from the indexed paths.

4.3.3 Translating NEXI to SIRIUS Query Language

We use automatic transformation of the INEX topics expressed in NEXI [226] to SIRIUS recursive query language.

To translate CO topics we use the *same*+ operator for weighting the textual content, the *seq* operator for strict phrase match and the *without* operator for the '-' sign in a straight forward way – see Section 3.5.3 for definitions of the operators. The '+', sign and the numerical expressions are ignored.

For CAS topics, we have two cases:

1. simple queries of the form //*A[B]* and
2. complex queries of the form //*A[B]*//*C[D]*.

For the simple CAS type queries, the translation process involves the *in*+ operator to approximate match the structural constraints and the *same*+ and *seq*[8] operators for processing the textual content. A translation example for INEX 2005 topic 277 is shown in Figure 4.9.

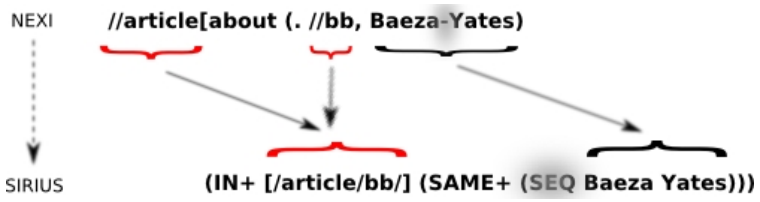


Figure 4.9: Translating a simple CAS topic to SIRIUS query language.

For translating complex queries of the form //*A[B]*//*C[D]* we introduce the *filter* operator aiming to solve element containment relationships (see Subsection 3.5.3 and Subsection 4.3.4). An exemple is shown is Figure 4.10

4.3.4 Processing NEXI Requests

Processing CO requests. CO queries are INEX topics containing only textual search terms (i.e. see the *title* part in Figure 4.4). We compute the relevance score for all the leaves elements of the XML tree containing at least one of the searched terms using a variant of the TF-IDF ranking scheme (see Section 3.4.2). In our approach we consider

[8]The *seq* operator is used to translate both the quotes " " and the dashes.

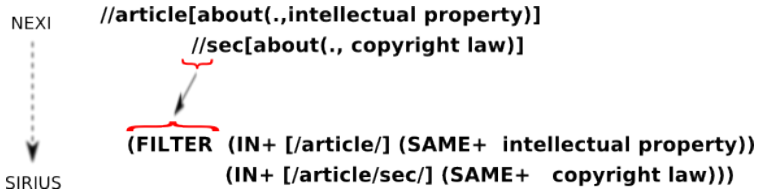


Figure 4.10: Translating a complex CAS topic to SIRIUS query language.

the XML element containing a searched term as the basic and implicitly valid unit of retrieval regardless of its size.

Processing CAS requests. For CAS topics, we have two cases: (1) simple queries of the form *//A[B]* – i.e. the request specifies only the target elements; and (2) complex queries of the form *//A[B]//C[D]* – i.e. the request specifies both target (i.e. *//C[D]*) and support (i.e. *//A[B]*) elements.

Processing the Support and Target Elements. For simple type queries of the form *//A[B]* like *//template//*[about(.,architecture)]* (see topic in Figure 4.4), we rank the textual content of the nodes using the same ranking scheme as for the CO requests. The structural constraints from the requests are interpreted as structural hints [226].

We compute the similarity between the structural constraints expressed in the request – i.e. *//template//** – and the XML paths of the candidate fragments using a modified editing distance (see Section 3.4.1) involving specific heuristics for attributes and attributes values [179]. Finally we merge the content and structural match scores using a weighted linear aggregation method (see Section 3.4.3).

Processing the Containment Conditions. To process complex queries of the form *//A[B]//C[D]* (see the castitle part in Figure 4.4) we compute the relevance for both the support elements *//A[B]* and target elements *//C[D]*.

We note here, that the structural position of the target elements relative to the support elements – *//A//C,* is explicitly taken into account when evaluating the relevance of the target elements *//A//C[D]*.

If the structural constraints for the support elements – *//A* are evaluated vaguely, there are cases in which the support elements are not necessarily ancestors of the target elements. Therefore, we select for the final set of answers, only the target elements that have at least a relevant support element occurring in the same document. This is a relaxation from the strict ancestor-descendant relation between the support and target elements. In this case, a support element is considered to be be any relevant element, where the relevance is evaluated on both content and structure in either a strict or vague manner, to the query *//A[B]*. The logic behind this is that if a relevant support element exists in a document, its weight should be propagated using a *Max* function to the root node of the XML tree that is an ancestor – i.e. support element – for all the elements of the tree. This rule applies inclusively to all the target elements.

The similarity computation for a complex request involves modifications of the relevance associated with a result element. The relevance of a result element is computed

as the arithmetic average between the relevance of the target element and the maximum relevance of its support elements.

Formally, let $\{(e_i, w_i)\}$ the set of target results, $\{(e_j, w_j)\}$ the set of support elements, where e_i is a an element of the result and $w_i \in [0..1]$ its relevance weight. Let e^D a descendant of document D. The set of weighted results produced by the system is $\left\{\left(e_i^D, w_i^{'}\right)\right\}$ with $w_i^{'} = (w_i + Max_j(w_j))/2$ where $\exists e_j^D \in \{(e_j, w_j)\}$.

Using this approach, the target elements without support elements are discarded from the final answers, while the ones supported by highly relevant elements are boosted in the final ranking. The final results are sorted by relevance values and the top N results returned.

4.4 Conclusion

In this chapter we have focused on the evaluation of XML information retrieval systems. We have introduced the INEX evaluation benchmark and detailed the SIRIUS XML IR system experimental settings used for the INEX ad hoc retrieval tracks.

In the next two chapters we evaluate the relevance gain to information access brought by the use of structural approximate matching mechanisms. We also show that, despite the lightweight characteristics of SIRIUS, we are able to retrieve highly relevant non overlapping XML elements and to obtain quite good results for low values of recall. Finally, we present and evaluate a simple, efficient and effective approach for retrieving best entry points in semi structured documents.

Chapter 5

Approximate Structural Matching for XML IR

In this chapter we evaluate the influence of the strict and approximate structural matching mechanisms to access relevant information in semi-structured databases. Preliminary versions of the results presented in this chapter were published in [176, 178].

Contents

5.1 Introduction

Our aim is to determine to what extent the structural constraints should be taken into account in an XML IR process. We present and evaluate both the response time efficiency and the effectiveness of different retrieval approaches with emphasis on the structural approximate matching strategies. We report the retrieval performances of a lightweight XML indexing and approximate search scheme currently implemented in the SIRIUS XML IR system for different degrees of structural constraints relaxations. SIRIUS retrieves relevant XML elements by approximate matching both the content and the structure of the XML documents. Finally, we compare the evaluation results of the SIRIUS vague structural approach against the official results of the INEX 2005 and INEX 2006 campaigns.

5.2 Retrieval Strategies

We evaluate the following structural retrieval scenarios on the same topic set: the first one takes into account explicit structural constraints using different degrees of structural constraints relaxations; the second one ignores the explicit structural constraints while the information about the hierarchical organization of the searched information is maintained.

For all the evaluated runs we use the same basic retrieval approach, namely:

- Automatic transformation of the *title* and *castitle* part of the INEX topics expressed in NEXI [226] to SIRIUS recursive query language (see Figure 5.1).
- The XML elements directly containing the search terms are considered as independent and the only valid units of retrieval;
- IDF-like weighting for the leaf nodes containing the researched terms (see Section 3.4.2 on page 50);
- Strict and approximate structural match using a modified editing distance on the XML paths with conditions on attributes and attributes values (see Section 3.4.1 on page 47 and Section 4.3.2 on page 74);
- Weighted linear aggregation for content and structure matching scores[1].

Text Matching Strategies

We evaluate two retrieval strategies for matching the textual content:

SAMEPLUS: – flexible sequence matching – we implemented a relaxed sequence search based only on the *same+* operator. These runs rank as best results the XML elements that contain all the researched terms without taking into account their order of occurrence. XML elements that contain only a part of the research terms are also retrieved and ranked based on the number and the discriminating power of the enclosed terms.

SEQ: – strict sequence matching – we used a strict *seq* operator for phrase matching inside the *same+* operator – where strict stands for all the words appearing in sequence in the textual content (ignoring the stop list words) of the same XML node.

Structural Matching Strategies

In the CAS requests the structural constraints are explicitly stated in the topics and can refer to where to look for the relevant elements (i.e. support elements) and what types of elements to retrieve (i.e. target elements). Structural constraints can be interpreted as either strict – S, or vague – V, and these interpretations can be applied to both support and target elements, giving a total of four strategies:

VVCAS: the structural constraints in both the target elements and the support elements are interpreted as vague.

[1] We use equal weights for all the reported runs.

SVCAS: the structural constraints in the target elements are interpreted as strict and the structural constraints in the support elements are interpreted as vague.

VSCAS: the structural constraints in the target elements are interpreted as vague and the structural constraints in the support elements are interpreted as strict.

SSCAS: the structural constraints in both the target elements and the support elements are interpreted as strict.

The strategies employing different degrees of structural relaxations are compared with a strategy that uses only content only (CO) information extracted from the CAS topics.

COCAS: the explicit structural constraints in both the target elements and the support elements are ignored. The hierarchical organization of the searched information is preserved. We note here that the containment conditions of the support and target elements are processed in the same manner for all the strategies (see Section 4.3.4 on page 76).

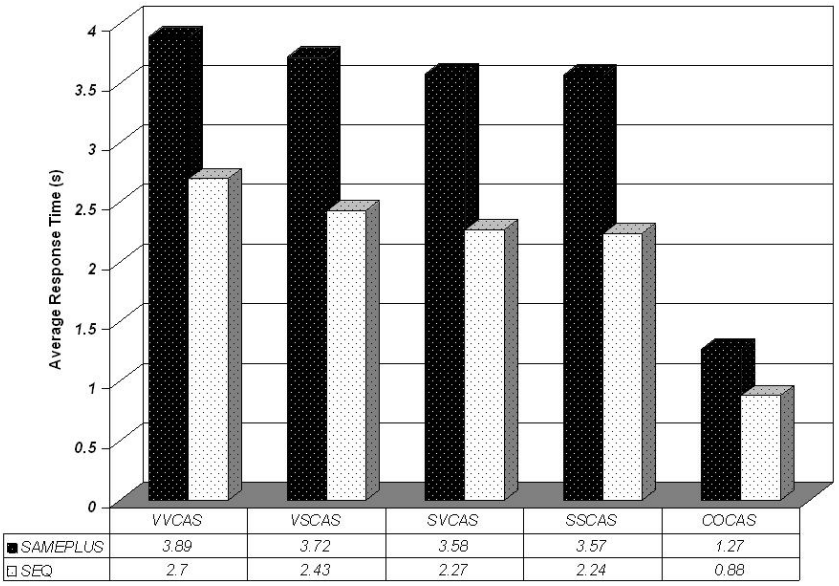
According to the different interpretations of the structural constraints for each search strategy, we automatically translate the CAS topics expressed in NEXI [226] to SIRIUS query language. An example of translation for the INEX 2005 CAS topic 280 (given in Figure 4.4 on page 68) is shown in Figure 5.1. The vague/strict interpretation of the structural constraints are translated using the *in+*, respectively *in* operators, while the containment conditions are translated using the *filter* operator.

```
COCAS_SEQ          ( FILTER ( AND ( SAME+ ( SEQ  Baeza Yates ) ) )
                                  ( SAME+   string matching ) ) )
                            ( SAME+  approximate algorithm )
COCAS_SAMEPLUS     ( FILTER ( AND ( SAME+ Baeza Yates)
                                  ( SAME+ string matching ) )
                            ( SAME+ approximate algorithm ) )
VVCAS_SAMEPLUS     ( FILTER ( AND ( IN+ [/article/bb/]  ( SAME+ Baeza Yates ) )
                                  ( IN+ [/article/sec/] ( SAME+ string matching ) ) )
                            ( IN+ [/article/sec/] ( SAME+  approximate algorithm ) ) )
VSCAS_SAMEPLUS     ( FILTER ( AND ( IN  [/article/bb/]  ( SAME+ Baeza Yates ) )
                                  ( IN  [/article/sec/] ( SAME+ string matching ) ) )
                            ( IN+ [/article/sec/] ( SAME+  approximate algorithm ) ) )
SVCAS_SAMEPLUS     ( FILTER ( AND ( IN+ [/article/bb/]  ( SAME+ Baeza Yates ) )
                                  ( IN+ [/article/sec/] ( SAME+ string matching ) ) )
                            ( IN [/article/sec/] ( SAME+  approximate algorithm ) ) )
SSCAS_SAMEPLUS     ( FILTER ( AND ( IN [/article/bb/]  ( SAME+ Baeza Yates ) )
                                  ( IN [/article/sec/] ( SAME+ string matching ) ) )
                            ( IN [/article/sec/] ( SAME+  approximate algorithm ) ) )
```

Figure 5.1: Translating INEX 2005 CAS topic 280 to SIRIUS query language for different interpretations of structural constraints.

5.3 Evaluating the Efficiency of Different Retrieval Strategies

The graphic from Figure 5.2 shows the average response time for different degrees of structural constraints approximation and text matching strategies. The evaluation results are obtained on a computer equipped with a Pentium IV processor at 2.4 GHz and with 1.5 GB of RAM memory. The set of requests is composed of 17 complex CAS topics[2]. The requests are evaluated on the inex-1.8 IEEE collection (i.e. 750 MB).



	VVCAS	VSCAS	SVCAS	SSCAS	COCAS
SAMEPLUS	3.89	3.72	3.58	3.57	1.27
SEQ	2.7	2.43	2.27	2.24	0.88

Figure 5.2: Average response time for different degrees of structural constraints approximation.

The average response time takes values between 0.88 s for a strict sequential search ignoring the structural constraints and 3.89 s for a flexible sequential search requiring to approximate match the structural constraints expressed in the request with the structure of the indexed XML documents. The computing time associated with the complexity of the approximate search algorithms on the documents structure can not be neglected, but is not prohibitive – particularly in the perspective of the parallelization of the algorithms in a context aiming to manage large databases.

5.4 Evaluating the Effectiveness of Text Matching Strategies

We report results for the system-oriented and user-oriented INEX 2005 official evaluation measures: the effort-precision/gain-recall (ep/gr) metric using the MAep value and the normalized extended cumulated gain (nxCG) metric at ranks 10, 25 and 50. All the results are calculated using a strict quantization function and by allowing the occurrence of overlapping elements.

[2]These are the INEX 2005 official topics used for evaluating the CAS runs.

By analyzing the SIRIUS evaluation results from Table 5.1 and by comparing them with the INEX 2005 official results (see Figure 5.3) we observe that the system has a good precision for low values of recall. This behavior is due to the fact that the runs used strict constraints for phrase searching and the topic set was rich (7 among the 10 assessed topics used sequence search) in this kind of hints. The restrictive interpretation of the *seq* operator improved the system precision for the first ranked results. In the same time, this behavior has penalized the system overall quality performance. One explanation is that the system stops to return elements when running out of good answers. This has important implications, as the system is not increasing its information gain until reaching the limit of 1500 allowed answers by returning imprecise/less perfect results.

The flexible phrase search strategy based only on the *same+* operator looses between 13.38% of the system precision for nxCG@10 and 41.08% for nxCG@50 versus the strict sequence search strategy, but we obtain an obviously improved overall effort-precision/gain-recall curve (gain of 14.57% on the MAep metric (overlap=off, quant=strict)). We could further improve these results by defining a new operator combining *same+* ranking with *seq* ranking strategies.

When compared with the INEX 2005 official results (see Figure 5.3), the SIRIUS evaluation results are rather encouraging. In particular, the best values reported for nxCG@{10,25,50} (overlap=off, quant.=strict) could be ranked unofficially on the first three positions (from 28 submissions) for the INEX 2005 VVCAS task. The best overall performance is obtained by the flexible matching sequence strategy with MAep=0.0558 (overlap=off, quant= strict) that is equivalent with a non official 5^{th} place.

	nxCG@10	**nxCG@25**	**nxCG@50**	**MAep**
VVCAS_SAMEPLUS	0.1444	***0.2022***	***0.1889***	***0.0558***
VVCAS_SEQ	***0.1667***	0.1689	0.1339	0.0487
Gain in %	-13.38	19.72	41.08	14.58

Table 5.1: Gain in % for the text matching strategies evaluated by using a vague interpretation of the structural constraints. (Task: VVCAS, Quantization: strict, Overlap: off).

5.5 Evaluating the Effectiveness of Approximate Structural Matching Strategies

The objective of our study is to evaluate the influence of the strict and approximate structural matching mechanisms to access relevant information in semi-structured databases. To evaluate the different retrieval strategies against the same users' expectations, we use the non filtered pertinence judgments of the INEX 2005 VVCAS task.

We present in Figure 5.4, Figure 5.5 and Table 5.3 the evaluation results for different degrees of structural constraint relaxations for the target and support elements (VV, VS, SV, SS). The effectiveness of the results are compared with a retrieval strategy searching for nested CO relevant information (see Table 5.5).

By analyzing the data from Table 5.3 we observe that the strategies implementing a strict interpretation of the target elements – i.e. SV and SS – returned identical

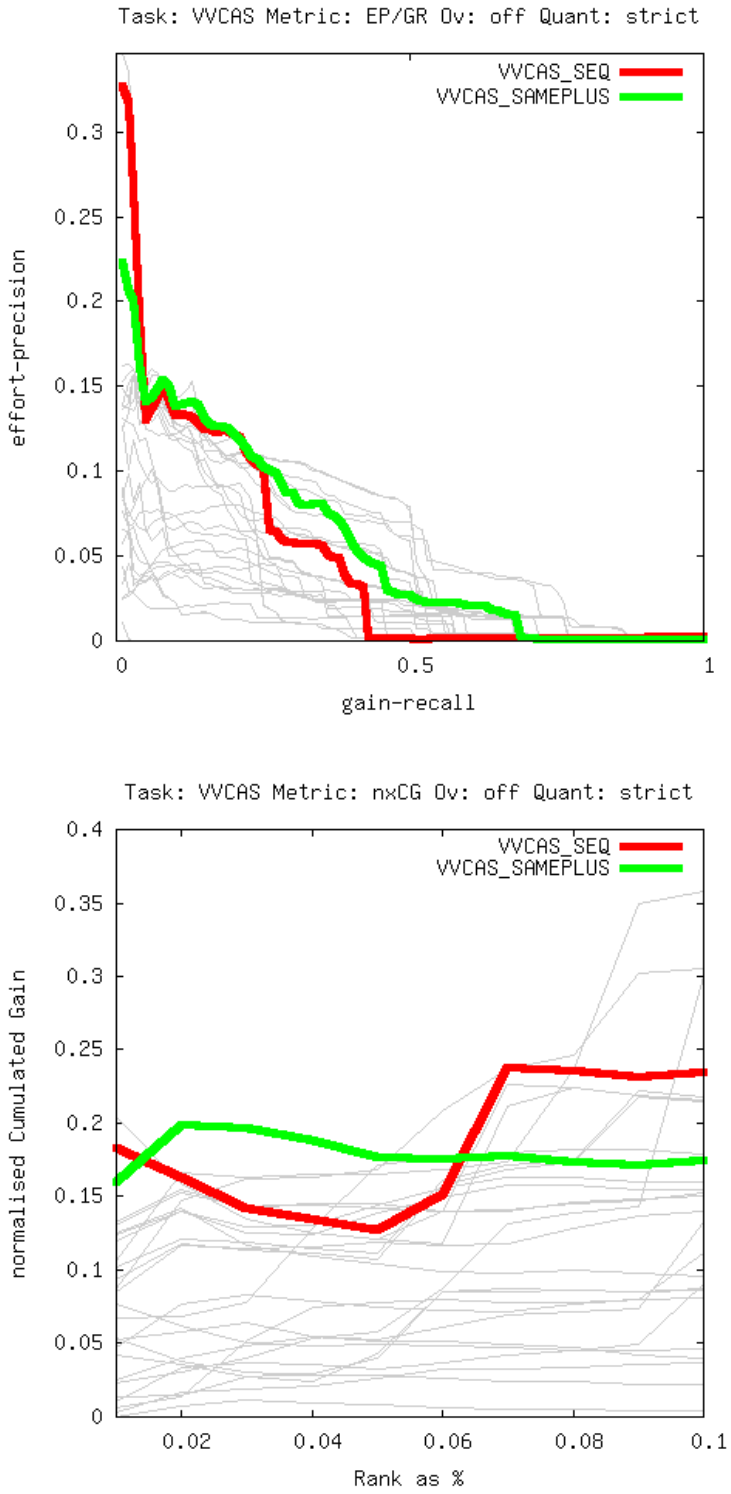


Figure 5.3: SIRIUS evaluation results compared with the INEX 2005 VVCAS task official results. Metric: ep/gr – top & nxCG – bottom, Quantization: strict, Overlap: off.

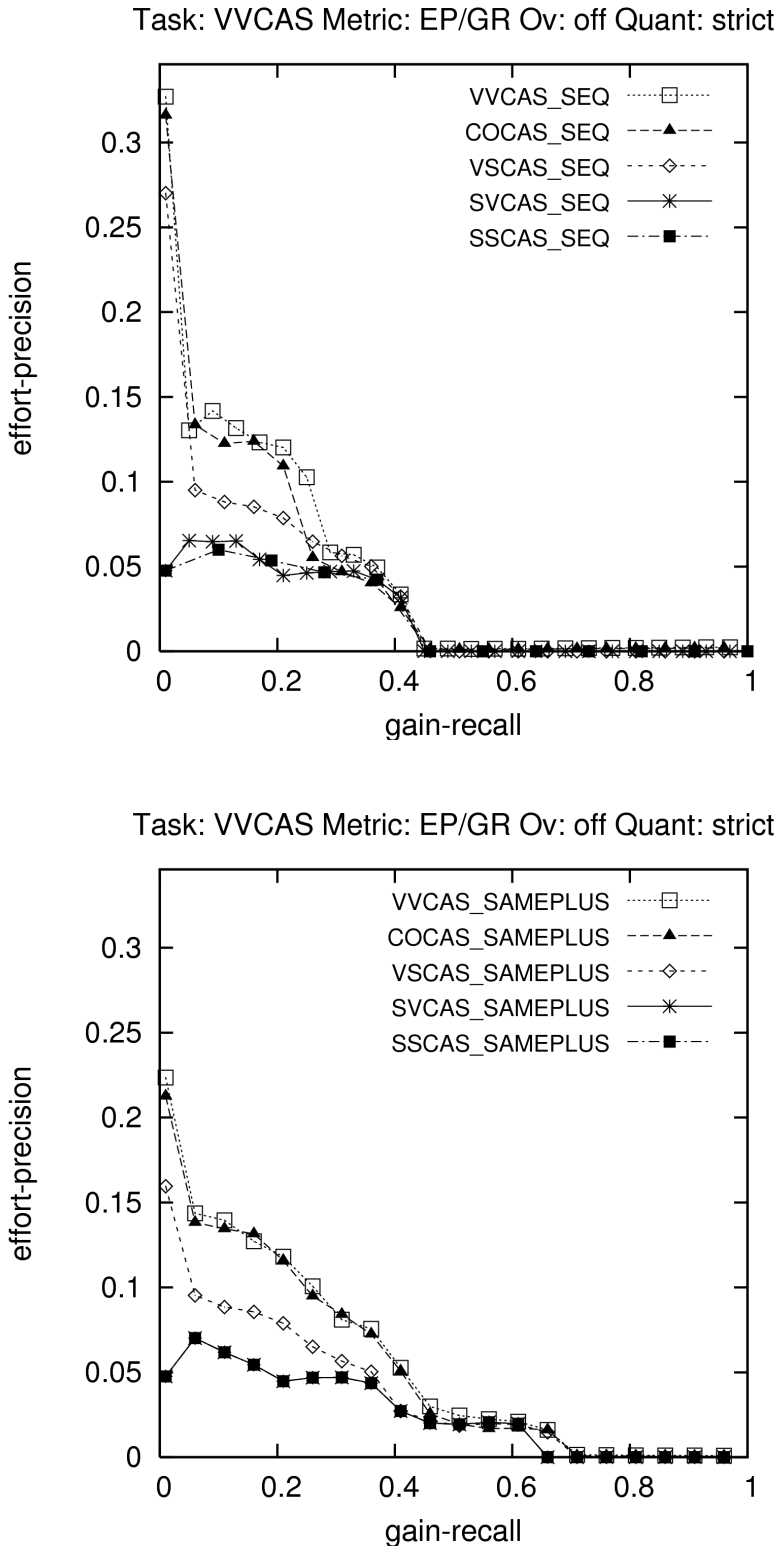


Figure 5.4: Evaluating different degrees of structural constraints approximation (Task: VVCAS, Metric: ep/gr, Quantization: strict, Overlap: off).

values. This observation is supported by the experiments conducted in [223] on the totality of the INEX 2005 submissions for the CAS sub-task. As a consequence, and to simplify the further analysis of the results we will refer from now on only to the SV strategy.

The strict interpretation of the structural constraints seems to have a negative effect on the quality of the retrieved results in our approach – globally relevant elements (as considered by the users) may be rejected from the list of the retrieved answers because they partially match or do not match at all the structural constraints. The recorded global performance degradation evaluated by the MAep measure is about -51% for the SV strategy and stands between -20.09% to -29.14% for the VS strategy using the *seq* and respectively the *same*+ operators (see Table 5.1). These results can be explained by the fact that the users, even experts, are particularly bad at formulating topics involving the structure of the XML documents [224]. Another possible explanation is the fact that the users may have a poor knowledge of the exact structural location of the information they are interested in.

Using vague structural constraints for both the target and the support elements – i.e. the VV strategy – augments the quality of the retrieved results compared with the CO strategy. The most significant improvement is obtained for the first retrieved answers evaluated with the nxCG@10 measure – i.e. a gain of 18.7% for the flexible sequence search strategy – VVCAS_SAMEPLUS, and of 15.44% for the strict sequence search strategy – VVCAS_SEQ. The overall gain recorded by the MAep measure and calculated with a strict quantization function was of 6.28% for the VVCAS_SAMEPLUS strategy and of 8,7% for the VVCAS_SEQ strategy.

As an overall conclusion, the approximate structural matching mechanisms seem more appropriate then the strict matching techniques to answer to the user structured information needs.

		nxCG@10	**nxCG@25**	**nxCG@50**	**MAep**
SEQ	SSCAS	0.0556	0.0711	0.0644	0.0217
	SVCAS	0.0556	0.0711	0.0644	0.0217
	VSCAS	0.1111	0.1022	0.0939	0.0358
	COCAS	0.1444	0.16	0.1273	0.0448
	VVCAS	***0.1667***	***0.1689***	***0.1339***	***0.0487***
SAMEPLUS	SSCAS	0.0556	0.0711	0.0644	0.0259
	SVCAS	0.0556	0.0711	0.0644	0.0259
	VSCAS	0.0889	0.0933	0.0889	0.0372
	COCAS	0.1222	0.1933	0.1822	0.0525
	VVCAS	***0.1444***	***0.2022***	***0.1889***	***0.0558***

Table 5.3: nxCG[i] and MAep evaluation results for different degrees of structural constraints approximation (Task: VVCAS, Quantization: strict, Overlap: off).

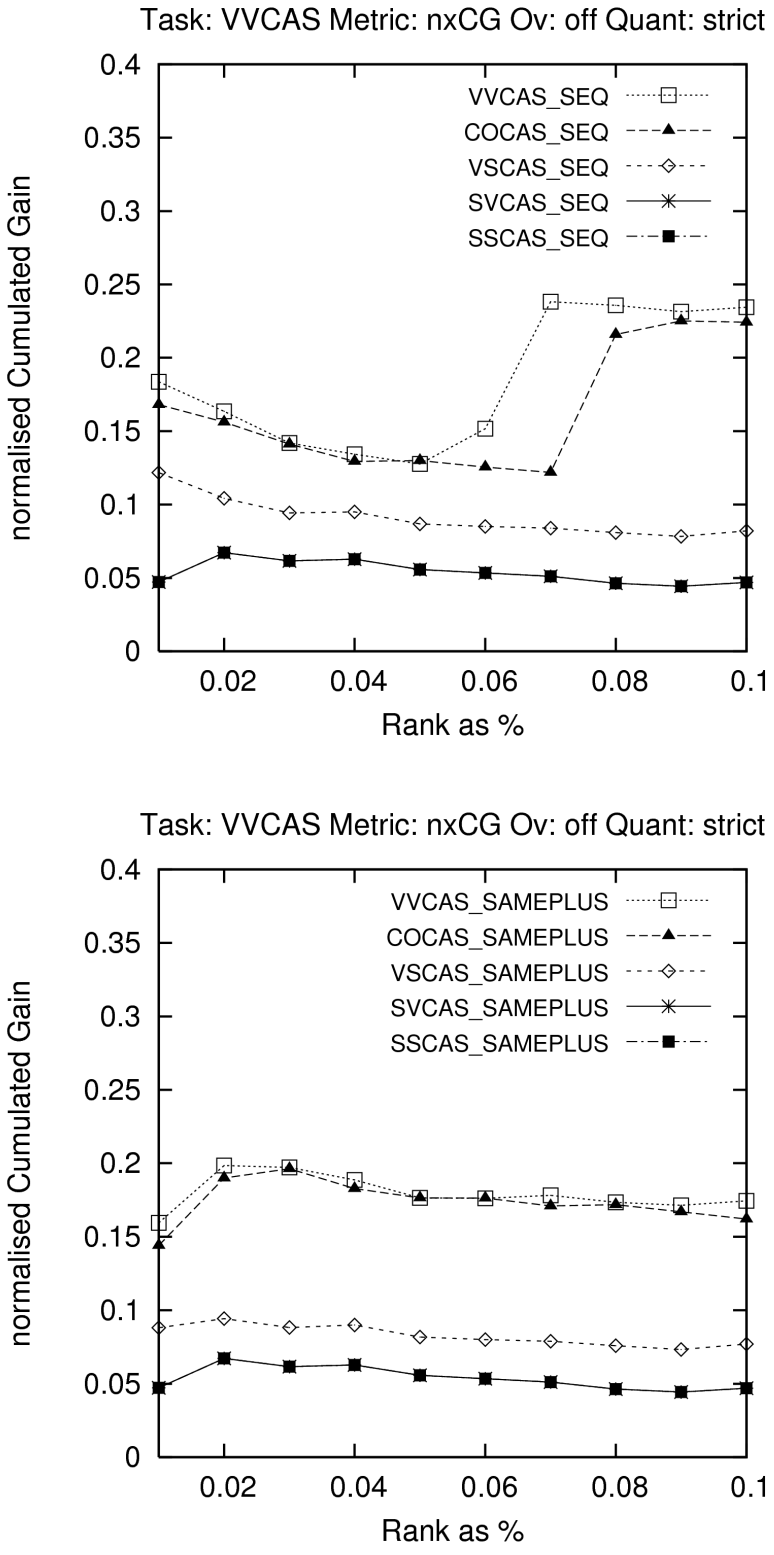


Figure 5.5: Evaluating different degrees of structural constraints approximation (Task: VVCAS, Metric: nxCG, Quantization: strict, Overlap: off).

		nxCG@10	nxCG@25	nxCG@50	MAep
SV, SS vs. CO	SAMEPLUS	-54.5	-63.22	-64.65	-50.67
	SEQ	-61.5	-55.56	-49.41	-51.56
VS vs. CO	SAMEPLUS	-27.25	-51.73	-51.21	-29.14
	SEQ	-23.06	-36.13	-26.24	-20.09
VV vs. CO	SAMEPLUS	18.17	4.6	3.68	6.29
	SEQ	15.44	5.56	5.18	8.71

Table 5.5: Gain in % introduced by the different degrees of structural constraints approximation (VV, VS, SV, SS) compared with the plain CO retrieval strategy evaluated on the INEX 2005 VVCAS task.

5.6 Evaluating the Effectiveness of Approximate Structural Matching for Focused XML IR

In this section we evaluate the influence of the strict and approximate structural matching mechanisms to access relevant information in semi-structured document collections using a focused retrieval strategy. In focused XML retrieval, information retrieval systems have to find out which are the most appropriate retrieval units and return only these to the user, avoiding overlapping elements in the result lists. For a brief description of the focused retrieval strategies implemented in the SIRIUS XML IR system, see Section 6.2 on page 97.

We evaluate two approaches: one when structural constraints are taken into account using different degrees of structural constraint relaxations and the other when the explicit structural constraints are ignored. The COS requests are equivalent with the CAS ones. Therefore we can use the same structural interpretations of the structural constraints for the target and support elements as for the evaluation of the VVCAS task (VV, VS, SV and SS). The difference appears at the COCOS strategy that uses genuine CO topics containing only textual keywords as in the standard search engine queries. The COCOS topics make absolutely no reference to the structure of the XML documents.

INEX 2005 COS Focused Task Results

We evaluate the approximate structural match for the the *maxHA*[3] focused retrieval strategy. This strategy performed among the best focused strategies implemented in the SIRIUS XML IR system during the INEX 2005 evaluation campaign (see [177] for more detailed evaluation results). We present in Figure 5.6 and Table 5.6 the evaluation results for different degrees of structural constraint relaxations for the target and support elements (VV, VS, SV, SS). The effectiveness of the results are compared with a content only (CO) retrieval strategy – see Table 5.7.

All the strategies are evaluated against an ideal recall database that includes no overlapping information (see [122] for details on the methodology used to build the ideal recall database).

[3]See Section 6.2 on page 97 for a description of the focused retrieval strategies implemented in SIRIUS.

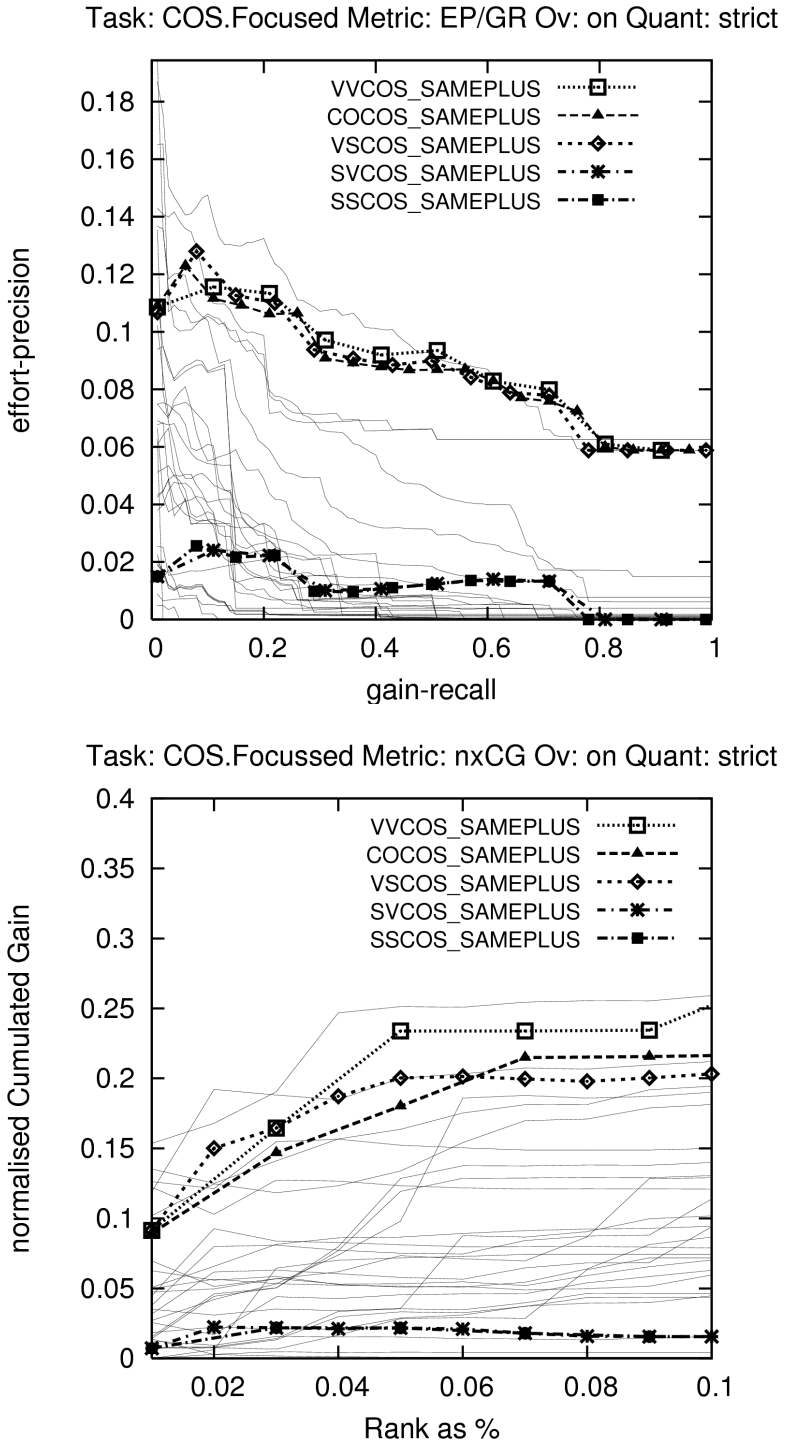


Figure 5.6: Evaluation results for the SIRIUS focused retrieval approach compared with the official results of the INEX 2005 COS Focused task. Metric: ep/gr – top & nxCG – bottom, Quantization: strict, Overlap: on.

By analyzing the data from Table 5.6 we observe that the strategies implementing a strict interpretation of the target elements – i.e. SV and SS – returned identical values.

As for the VVCAS task, the strict interpretation of the structural constraints seems to have a negative effect on the quality of the retrieved results in our approach. The recorded global performance degradation evaluated by the MAep measure is about -86.5% for the SV/SS strategies. A small increase on the MAep values may be observed for the VS and VV strategies versus the CO strategy (1.53% for the VS strategy and 4.58% for the VV strategy when using the *same*+ operator – see Table 5.7).

Using vague structural constraints for both the target and the support elements – i.e. the VV strategy – augments the quality of the first retrieved results compared with the plain CO strategy. The most significant improvement is obtained on the nxCG@25 and nxCG@50 measure – i.e. a gain of 22.63% respectively of 21.16% for the flexible sequence search strategy.

The negative evaluation results recorded for the strict interpretation can be explained by the fact that the structural constraints used for the INEX 2005 COS topics did not seem to correspond to actual hints; instead they appear to be a function of the document collection rather than the query [224].

This behavior was confirmed by the experimental results obtained within the INEX 2005 VVCAS task [177]. This indicates (usual disclaimers apply) that the structural hints, and jointly, the modified editing distance on the XML paths improves the system retrieval performances. This is according to other results [200] from the literature that also reported improvements for the use of vague structural constraints on the INEX 2005 test collection.

		nxCG@10	**nxCG@25**	**nxCG@50**	**MAep**
SAMEPLUS	SSCOS	0.0118	0.0235	0.0224	0.0115
	SVCOS	0.0118	0.0235	0.0224	0.0115
	VSCOS	0.1059	0.162	0.1853	0.0865
	COCOS	0.1	0.1321	0.1668	0.0852
	VVCOS	***0.1059 (4)***	***0.162 (3)***	***0.2021 (2)***	***0.0891 (2)***

Table 5.6: nxCG[i] and MAep evaluation results for different degrees of structural constraints approximation for the SAMEPLUS strategy (Task: COS Focused, Quantization: strict, Overlap: on). The best results are emphasized and compared with the INEX 2005 official results. The ranks/27 submissions are given in parentheses.

		nxCG@10	**nxCG@25**	**nxCG@50**	**MAep**
SAMEPLUS	***SV, SS vs. CO***	-88.2	-82.21	-86.57	-86.5
	VS vs. CO	***5.9***	***22.63***	11.09	1.53
	VV vs. CO	***5.9***	***22.63***	***21.16***	***4.58***

Table 5.7: Gain in % introduced by the different degrees of structural constraints approximation (VV, VS, SV, SS) compared with the plain CO retrieval strategy evaluated on the INEX 2005 COS Focused task.

When comparing the proposed strategy with the INEX 2005 COS focused task official evaluation results (see Figure 5.6), the results are rather encouraging. In particu-

lar, the best values reported for nxCG@{10, 25, 50} (overlap=on, quant.=strict) could be ranked unofficially on the 4^{th}, 3^{rd} and 2^{nd} positions (from 27 submissions). Looking at the system global retrieval performance measured with the MAep metric (overlap=on, quant= strict), we could obtain a 2^{nd} place for the flexible matching sequence strategy (see Table 5.6 – the ranks/27 submissions are given in parantheses).

INEX 2006 Focused Task Results

In INEX 2006 evaluation campaign the CO, CAS and COS topics received an uniform format (see Figure 4.5 on page 68). A single generalized quantization measure is used as only the specificity dimension was maintained to evaluate the relevance of the retrieved elements. The quantization function simply maps an element to its specificity value s. Official evaluation results for the INEX 2006 Focused task were reported only for the nxCG metric using the generalized quantization function.

The results reported in Figure 5.7 and Table 5.8 are obtained by using the SIRIUS *maxMRD*[4] focused retrieval approach and evaluated using the Wikipedia XML collection. The SIRIUS focused evaluation results are compared against the official results of the INEX 2006 Focused task (see [179] for more detailed evaluation results).

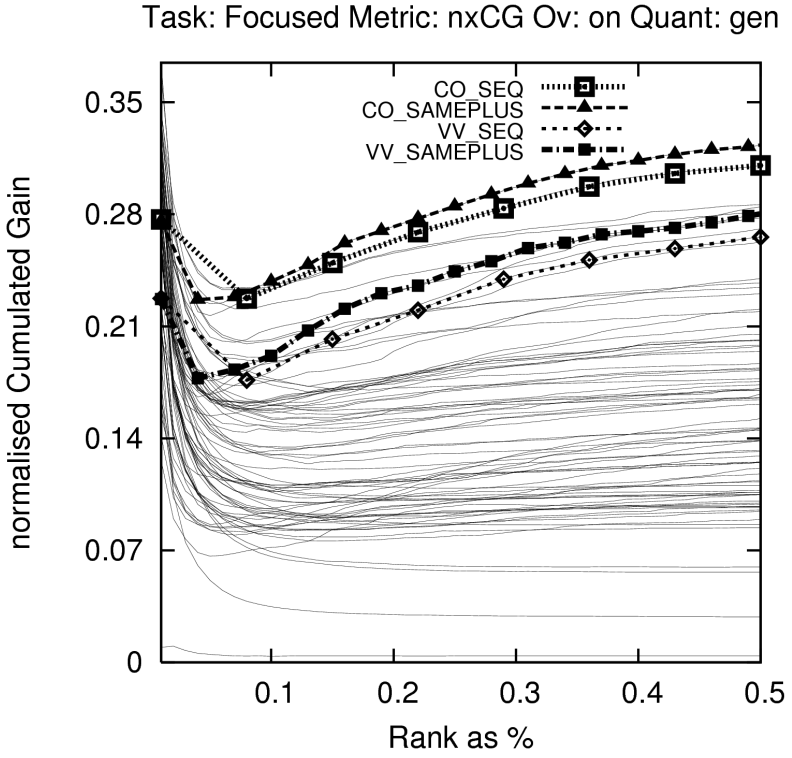


Figure 5.7: Evaluation results for the SIRIUS focused retrieval approach compared with the official results of the INEX 2006 Focused task. Metric: nxCG , Quantization: generalized, Overlap: on.

By analyzing the overall comportment of nxCG curves from Figure 5.7 we observe that SIRIUS runs have a good recall. The results show the system ability to retrieve

[4]See Section 6.2 on page 97 for a description of the focused retrieval strategies implemented in SIRIUS.

		nxCG@5	nxCG@10	nxCG@25	nxCG@50
SEQ	CO	***0.2840 (46)***	0.2679 (28)	0.2469 (10)	0.2211 (7)
	VV	0.2301 (64)	0.2196 (61)	0.2004 (33)	0.1709 (32)
SAMEPLUS	CO	0.2832 (48)	***0.2752 (23)***	***0.2475 (9)***	***0.2289 (4)***
	VV	0.2297 (66)	0.2218 (59)	0.2039 (31)	0.1782 (23)

Table 5.8: nxCG[i] evaluation results for the SIRIUS focused retrieval strategy using a content only – CO, and a vague interpretation of structural constraints – VV (Quantization: generalized, Overlap: on). The ranks compared with 85 official submissions for the INEX 2006 Focused task are given in parantheses. Best results are emphasized.

		nxCG@5	nxCG@10	nxCG@25	nxCG@50
VV vs. CO	SAMEPLUS	**-18.89**	-19.4	**-17.62**	**-22.15**
	SEQ	-18.98	**-18.03**	-18.83	-22.7
SAMEPLUS vs. SEQ	VV	**-0.17**	1	**1.75**	**4.27**
	CO	-0.28	**2.72**	0.24	3.53

Table 5.9: Gain in % introduced by a vague interpretation of structural constraints (VV) compared with a plain content only (CO) approach. Gain in % for a flexible (SAMEPLUS) versus a strict (SEQ) sequence matching strategies. Results are evaluated within the INEX 2006 Focused task using a *maxMRD* focused retrieval strategy.

highly relevant non overlapping XML elements. SIRIUS obtained several best top ten rankings out of 85 official submissions for the 25-50 first retrieved answers using the nxCG@25 and nxCG@50 metrics (see Table 5.8).

The results from Table 5.9 show that the runs using structural constraints were consequently outperformed by the runs using content only conditions, while the runs using strict constraints for phrase searching were outperformed by their relaxed variants.

The improvement brought by the use of the structural hints observed on the INEX 2005 test collection and topics [177] was not confirmed by the evaluation results of the INEX 2006 Focused task [179]. We note here that, even if improvements could not be recorded on the average retrieval performances of the system when using all the topics, specific topics get their effectiveness boosted by the use of the structural constraints. For instance, topic 406 – the only INEX 2006 topic that has specified structural constraints on attributes and attributes values (see Figure 4.6 on page 69 – shows more effective retrieval curves for the runs using a vague approximation of the structural constraints (i.e. VV) versus their CO competitors – see Figure 5.8.

In this example we used different weightings – i.e. 0.1 and 0.5 – to emphasize the importance/weight of the structural hints in the score of the retrieved results. The curves shows that the approach using equal weights to fusion the content and structural matching scores (i.e. *w0.5*) get better results than the approach that emphasize the textual matching (i.e. *w0.1*) – which outperforms the approach that completely ignores the structural constraints (i.e. CO). This behavior shows that, at least for some CAS topics, the structural hints may increase significantly the effectiveness of the content only approach.

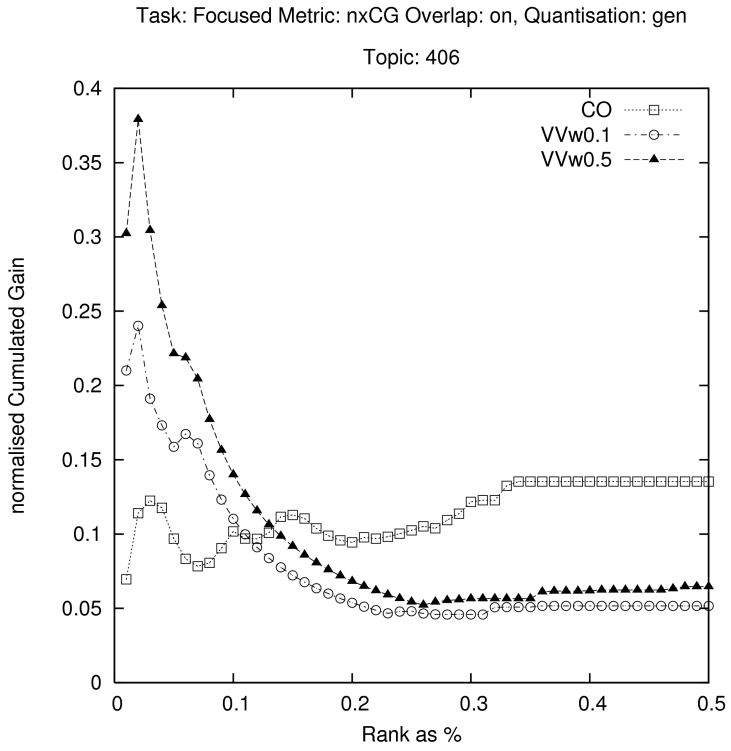


Figure 5.8: Evaluatin results for topic 406 of the INEX 2006 campaign. Task: Focused, Metric: nxCG, Quantization: generalized, Overlap: on, Focused retrieval approach: *avgMRD*, Text matching operator: *SAMEPLUS*.

5.7 Conclusion

In this chapter we have evaluated the relevance gain to information access brought by the use of structural approximate matching mechanisms. Our experiments at INEX 2005 have showed that taking into account the structural constraints improved the retrieval performances of the system and jointly showed the effectiveness of the proposed weighted editing distance on the XML paths for the VVCAS and COS Focused tasks. We have also showed that the approximate structural matching mechanisms are more effective than the strict matching techniques for answering to the user structured information needs.

Further, we have evaluated our approach for the structural constraints approximate matching using the INEX 2006 test collection and topics. With the new experimental settings for the Focused task, we observed a decrease in the average retrieval effectiveness of the approach using structural hints versus the plain CO approach [179]. We note here that, even if improvements could not be recorded for all the topics on average, specific topics got their effectiveness boosted by the use of the structural constraints. This behavior shows that, at least for some topics, the structural hints may significantly increase the effectiveness of the content only approach.

It is not yet clear if these results are due to changes brought to the evaluation method between the INEX 2005 and INEX 2006 campaigns – i.e. different quantization functions, different relevance scales, different strategies to compute the ideal recall database – either due to the users ”clumsiness” in adding valid and useful struc-

tural constraints to the CO topics.

More detailed experimental studies analyzing and comparing the use of the structural hints within the XML IR requests of the INEX 2005 and INEX 2006 campaigns are necessary to better understand the reasons for this behavior.

The experiments have also showed that the approximate search inside XML elements implemented using our *same+* operator improves the overall performance of the ranking, compared to a strict sequence search (*seq* operator), except for low recall values. The complementarity of the two operators call for the design of a new matching operator based on their combination to further improve the retrieval performance.

Compared with the official results of the INEX evaluation campaigns, the lightweight indexing and approximate search scheme implemented in the SIRIUS XML IR system, obtained good quality results in the range of the 10-25-50 first ranked answers. This is encouraging since the first ranked elements are the ones end users will most probably browse.

Chapter 6

Retrieving Best Entry Points in Semi-Structured Documents

Focused structured document retrieval tries to make use of the concept of best entry point (BEP) which is intended to define from a user's perspective the starting-point from which browsing relevant document components should be optimally initiated [121]. In this chapter we describe a simple, efficient and effective method for providing BEPs candidates in XML documents. Experiments conducted within the framework of INEX 2006 evaluation campaign ranked the proposed approach on the 1^{st} place out of 77 official submissions for the Best In Context Task. Secondly we compare the effectiveness of the approach with a standard 'flat' document retrieval system that returns document snippets as BEPs based on the Google search engine. Preliminary versions of the approach and results presented in this chapter were published in [179, 174].

Contents

6.1 Introduction

In traditional (flat) information retrieval, query results that are presented to the user have typically the form of a list of matching documents. For structured (from weakly to semi-structured) or focused information retrieval in large hyper document collections this alone is not a satisfactory option: the user also needs to know where into the documents he can find relevant text fragments.

Structured document retrieval makes use of document components as the basis of the retrieval process, rather than complete documents. Within this context the concept of Best Entry Points (BEPs) has been defined as document components from which the user can initiate its browsing to get optimal access to relevant document components [186].

The basic characteristics of BEPs are presented in [186] while [187] investigates different types of best entry points in structured document retrieval and their usage and effectiveness in real information search tasks. Algorithms for automatic identification of BEPs were implemented and empirically evaluated in [110, 121] on experimental data from the Shakespeare dataset[1]. The overall effectiveness of the algorithms was found to yield poor results [186]. Also the size of the test collection used for evaluation was rather modest for a real application-case (i.e. less than 10 MB). This pioneer experimental dataset may not provide enough information about the scalability of the proposed algorithms.

In this chapter we focus on automatic detection of the one ”true” BEP – i.e. a unique BEP per semi-structured document given a query as required in the Best In Context task of INEX 2006. The aim of the task is to first identify relevant articles (the fetching phase), and then to identify the element corresponding to the best entry points for the fetched articles (the browsing phase). In the fetching phase, articles should be ranked according to their topical relevance. In the browsing phase, we have a single element whose opening tag corresponds to the best entry point for starting to read the relevant information in the article [49].

Recent BEPs studies [107] suggested that the users have a strong preference for the most specific, most focused components that contain the most amount of relevant information and the least amount of irrelevant content. They also identified several types of BEPs of which ”Start reading here” BEPs – i.e. a leaf level entry point, representing the point where the users would prefer to be directed to and where they would likely appreciate to start reading the text. This is the BEP type the most related to the INEX 2006 Best In Context task requirements. This was the most popular BEP type occurring in the Shakespeare collection. 62% of the ”Start reading here” BEPs were the first leaf nodes occurring in a sequence of relevant leaf nodes. This confirms the findings from [209] in which a focused access experiment on the Wikipedia collection[2] was carried out. The experiment showed that 55% of the users' accesses went to the first section of the articles. This emphasizes the importance of the document order for the automatic detection of BEPs. Starting from these experimental observations we propose an algorithm for detecting the BEPs that returns *the first most relevant non-overlapping focused elements* to the requested topic.

[1] Shakespeare Plays Dataset `http://metalab.unc.edu/bosak/xml/eg/shaks200.zip`

[2] Wikipedia, the free encyclopedia `http://wikipedia.org/`

This event marked the beginning of the period known as "Nobles' Commonwealth" when the state was ruled by the "free and equal" *Polish* nobility (szlachta). The Lublin Union of 1569 constituted the

<collectionlink xmlns:xlink=http://www.w3.org/1999/xlink
xlink:type="simple" xlink:href="343234.xml">
Polish-Lithuanian Commonwealth
</collectionlink>

as an influential player in Europe ?

Figure 6.1: Example of XML element with mixed content extracted from the Wikipedia XML corpus [59].

6.2 Focused Retrieval Strategy

The aim of the *Focused* retrieval strategy is to find the most exhaustive and specific element in a path. In other words, the result list should not contain any overlapping elements. In our approach we consider the XML element directly containing a searched term as the basic and implicitly valid unit of retrieval regardless of its size. This approach implements ”naturally” a focused strategy as it returns the most focused elements containing the search terms. However, for XML nodes with mixed content – like *paragraphs* or *sections* including presentation tags (i.e. *italics*, *emphasized*) or jump tags (i.e. *collectionlink*, *outsidelink*) – overlapping elements may occur in the result list.

For instance, both the *p* (paragraph) and the *collectionlink* elements of the excerpt from Figure 6.1 will be retrieved by a request aiming to extract relevant XML elements for the term '*Polish*'.

In order to remove the overlapping elements from the results list we have implemented a two phases post filtering process [177].

6.2.1 Elements Scores Aggregation

First, we aggregate the relevance of the elements in the answer list in order to reflect the relevance of their descendant elements (if any). The weights are calculated in a bottom-up manner starting from leaves to the highest non overlapping nodes composing the answer by using two strategies:

max – the max relevance value is propagated recursively to the highest non overlapping elements; and

avg – the relevance of a node is computed as the arithmetic average of all its descendant relevant nodes including its own relevance.

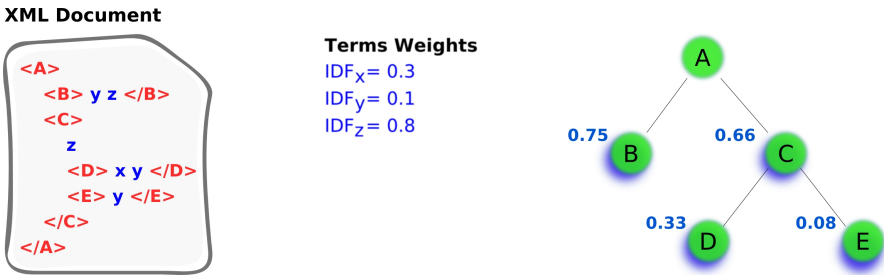


Figure 6.2: XML document with mixed content and term weights.

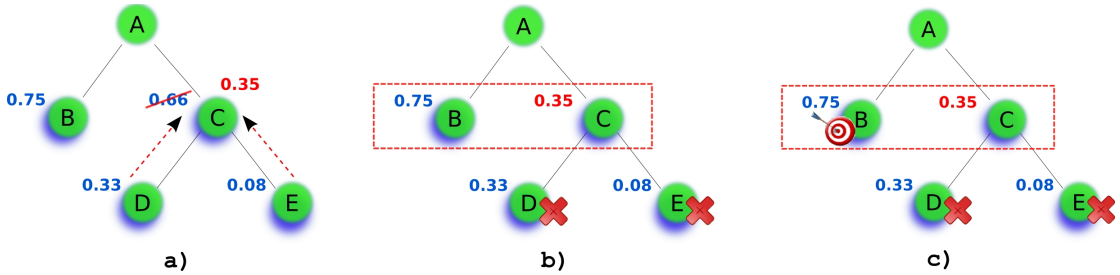


Figure 6.3: Example of a BEPs retrieval strategy. a) Elements score aggregation using the *avg* strategy b) Removing overlapping elements using the *MRD* strategy c) BEPs selection heuristic.

6.2.2 Removing Overlapping Elements

Second, to select the non-overlapping elements we experiment with the following strategies:

HA – the highest ancestor in the answer list is selected among the relevant overlapping elements;

MR – the most relevant answer is selected recursively from the answer list as long as it does not overlap with an already selected element – i.e. for equally relevant overlapping elements we choose either the descendant (MRD) or the ancestor (MRA).

6.3 BEPs Selection Heuristic

After the overlap removing process, the first most relevant element in the document order is selected as the BEP for the document. Its weight is propagated at the document level and used to rank the files by relevance.

We illustrate the process of detecting a BEP using the *avgMRD* focused retrieval strategy in Figure 6.3 for the '*x y z*' request on the XML document given as example in Figure 6.2.

For instance, by applying the proposed method on the XML document presented in Figure 4.2 on page 65, the path */article[1]/body[1]/section[4]/p[2]* was selected as BEP for the INEX 2006 topic 289 presented in Figure 4.5 on page 68.

```
emperor "Napoleon I" Polish site:http://en.wikipedia.org/wiki
```

Figure 6.4: Using the Google search engine to answer INEX 2006 CO topics.

6.4 Google @ INEX 2006 Best In Context Task

To show the benefits brought by our BEPs detection method we define as our baseline a standard 'flat' document retrieval system based on the Google[3] search engine that retrieves documents and document snippets as BEPs. The 'flat' runs are composed only of results at document level while the 'bep' runs try to match the snippets associated with the retrieved documents to BEPs. We then compare the effectiveness of the proposed approach against the baseline runs. The evaluation is performed on a snapshot of the Wikipedia collection.

6.4.1 Retrieval Settings

We retrieve results for the INEX 2006 CO topics using the Google search engine restricted to the English part of the Wikipedia collection. For instance, the INEX 2006 CO topic 289 from Figure 4.5 on page 68 is transformed and submitted to Google search engine as shown in Figure 6.4.

The implementation is based on the Google SOAP Search API[4] . This service allows querying a maximum of 1000 requests per day, where a request may retrieve 10 results at most. The maximum depth that can be reached for a given topic is limited to 1000 results. The 'Automatic Filtering'[5] of the results was set to 'false' in our experiment which allowed us to obtain all the relevant results for the requested topic. The retrieved experimental datasets are available online[6].

6.4.2 Flat Runs

The 'flat' runs are composed only with results having the BEPs set to document or */article[1]* level (see an exemple of the structure of the XML document in Figure 4.2 and Figure 4.3 on page 66). From the set of results returned by Google we remove all the answers that do not refer to a valid document id in the INEX 2006 ad hoc collection [59]. The valid answers are selected using the title of the articles (i.e. *'/article[1]/name[1]')* and ranked starting from 1 to MAX by preserving their original retrieval order.

We identified a number of 14832 valid documents out of 31517 retrieved by Google (i.e. 47%) for all the 111 assessed CO topics of INEX 2006 (i.e. this represents an average of 133.6 valid results at document level per topic).

[3]Google `www.google.com`

[4]Google SOAP Search API `http://code.google.com/apis/soapsearch/`

[5]When enabled, filtering takes the following actions: i) Near-Duplicate Content Filter = If multiple search results contain identical titles and snippets, then only one of the documents is returned; and ii) Host Crowding = If multiple results come from the same Web host, then only the first two are returned (see `http://code.google.com/apis/soapsearch/reference.html`).

[6]Google @ INEX 2006 Datasets `http://www-valoria.univ-ubs.fr/APRIM/Sirius/Resources.php`

```
{
TM = 0.172802
Q = "emperor Napoleon I Polish site:http://en.wikipedia.org/wiki"
CT = ""
TT = ""
CATs =
{
<EMPTY>
}
Start Index = 1
End Index = 10
Estimated Total Results Number = 379
Document Filtering = false
Estimate Correct = false
Rs =
{
...
[
URL = "http://en.wikipedia.org/wiki/History_of_Poland"
Title = "History of Poland - Wikipedia, the free encyclopedia"
Snippet = "Following the French <b>emperor Napoleon</b> I&#39;s defeat
of Prussia, a <b>Polish</b> state was <br> again set up in 1807 under
French tutelage as the Duchy of Warsaw.  <b>...</b>"
Directory Category = {SE="", FVN=""}
Directory Title = ""
Summary = ""
Cached Size = "75k"
Related information present = true
Host Name = ""
],
...
}
}
```

Figure 6.5: An excerpt of a retrieved result using the Google SOAP Search API.

6.4.3 Approximate Matching of Snippets to BEPs

The ranked list of documents returned as results by standard web search engines such as Google provides the users with a 'snippet' summarizing the relevant text found in the document. We provide in Figure 6.5, an example including a snippet obtained by using the Google SOAP Search API. The retrieved result, answers the INEX topic 289, which searches information about the emperor Napoleon I and the Polish people (see Figure 4.5 on page 68 and Figure 6.4).

We use the content of the snippet to approximately match a text region in the retrieved document. A snippet is mainly a query-biased summary formed by several semantically contiguous text parts surrounding the search terms – see Figure 6.6. To match a snippet to a BEP we first parse and extract the original contiguous text segments from the snippet[7]. The content of both the extracted text segment, and the XML

[7] In the example given in Figure 6.6 the text segments are delimited by ....

```
"<b>Napoleon</b> I, <b>Emperor</b> of the French, King of
Italy, Mediator of the Swiss Confederation <br> and <b>...</b>
<b>Napoleon</b>, in an attempt to gain increased support from
<b>Polish</b> <b>...</b>"
```

Figure 6.6: An example of snippet composed of several semantically contiguous text parts .

document content are normalized – i.e. the punctuation marks, the presentation and the XML markup, as well as the stop words are ignored. Next, we perform a search of the text extracted from the snippet within the content of the XML document and select the index position of the first matching character. We use this index position to search within the stacks constructed while parsing and normalizing the XML document content. The stacks keep information about the index position of the normalized textual content starting character for each XML path. Finally, we set the BEP path as the most focused XML element containing the index position of the first matching character. If no match is found for the current text segment, we advance to the following segment until either a match occurs or all the content of the snippet is discarded.

This is very similar with selecting the *first relevant character* [102] from the document as a BEP. As Kamps et al. show in [102] by analyzing the relevance assessments of the INEX 2006 BEPs task, many of the best entry points coincide with the first relevant character. This shows a strong relation between the BEPs and relevant text, where the text relevance is decided by the topic assessors that highlight relevant text passages.

The approach described above, naturally reproduces the behavior of an user that submit a keyword query to a standard search engine, that decides of the relevance of the fetched documents by reading the document snippets, and that uses the local search functionality of its browser to find short phrases extracted from the snippet in order to be pointed directly to the relevant information. Furthermore, this approach may take advantage of the eventually more evolved natural language techniques used by the current search engines to extract document snippets. The capacity of the search engines to process and extract the relevant text surrounding the search terms may also simulate the role of the real topic assessors that highlight relevant text passages.

For instance, using this method, we were able to match the snippet from Figure 6.5 to the '/*article[1]*/*body[1]*/*section[4]*/*p[2]*' path in the file id 13772 of the Wikipedia XML corpus [59] given as example in Figure 4.2 on page 65.

Only 5241 BEPs were identified for 14832 valid results at document level using the described method (i.e. 35%). This may be due to the fact that the content of the Wikipedia articles are changing at a higher rate than their main subject – i.e. the article titles. If no valid match could be detected, the BEP path was set by default to '/*article[1]*'. Another problem may appear if a match is found at a different structural position than its original location in the XML document. This may occur due to a significant change between the two snapshots of the collection used for evaluation, and may have a negative effect on the 'document snippet → BEP' strategy evaluation results.

6.5 Evaluation Framework

A preliminary evaluation [179] of the proposed approach was conducted within the framework of the Best In Context task [49] of the INEX 2006 evaluation campaign. INEX 2006 provides an experimental evaluation framework constituted of a test collection, tasks / requests, relevance judgments and evaluation measures.

The test collection [59] is composed of 659,388 English articles in XML format extracted from Wikipedia and totaling 4.6 GB. A set of 111 content only (CO) and content and structure (CAS) topics with their associated BEPs assessments were used for evaluation. In INEX 2006, the assessors were requested to indicate one and only one BEP for every document that has relevant content. For example, the BEP assessment for the topic 289 (see Figure 4.5 on page 68) set the BEP path for the XML document shown in Figure 4.2 on page 65 to '/*article[1]*/*body[1]*/*section[4]*/*p[2]*'.

6.5.1 Best In Context Task Evaluation Metrics

Runs for the BIC task are evaluated with two metrics [120]:

- a set based measure, BEPD (For BEP-Distance); and
- an extension of precision recall (EPRUM).

Both metrics use a base score for an element x, which is defined as 0 if x does not appear in a relevant document, i.e. a document containing a Best Entry Point (BEP). Otherwise, there exists a BEP b in the x's document and the measure is:

$$s(x, b) = \frac{A \cdot L}{A \cdot L + d(x,b)}$$

which is between 0 and 1 where:

- $d(x, b)$ is the distance (in number of characters) between the beginning of element x and the beginning of element b,
- L is the average document length (in characters), and
- $A > 0$ is a parameter.

Note that high values of A (e.g. 10) tend to give a score of 1 to any answer in a relevant document, hence the score does not discriminate whether x is near to or far from the BEP (b). Whereas, low values of A (e.g. 0.1) favor runs that return elements (b) very close to a BEP [120]. Official evaluation results at INEX 2006 were evaluated with A equal to 0.01, 0.1, 1, 10, 100.

BEP-D

The BEP-D metric is the sum of all the single scores $s(x, b)$ over elements x of the run divided by the total number of best entry points. The measure is then averaged over runs (queries).

Metric	A=0.01	A=0.1	A=1	A=10	A=100
BEPD	0.1959 (1)	0.2568 (2)	0.3642 (6)	0.5596 (6)	0.7556 (7)
EPRUM	0.0407 (1)	0.0579 (8)	0.0873 (13)	0.1489 (16)	0.2193 (35)

Table 6.1: INEX 2006 Best In Context Task official evaluation results for the *avgMRD* strategy. The ranks / 77 submissions are given in parentheses.

EPRUM-BEP-Exh-BEPDistance

The EPRUM metric is an extension of precision recall that can capture the scenario of a user consulting the context of the retrieved elements. This is modeled with a parameter, which is the probability that a user goes from a returned element x to a BEP b.

In the context of the BIC task, this probability is defined as $s(x, b)$ for any BEP b. This behavior is defined stochastically, that is we only know that the user has seen the BEP with probability $s(x, b)$.

Precision at recall r is defined as the ratio, for the user to achieve a recall r, of the minimum expected search length for the ideal run to the run's minimum expected search length.

Precision at rank k is defined as the expected search length (for the ideal run) for a user to achieve the same recall as the one achieved by the evaluated run divided by k.

In both cases the ideal run is the list of BEP. Both definitions reduce to the classical precision and recall when the standard user model is assumed, where the parameters (i.e. the probabilities) are either 0 or 1.

6.6 Experimental Results

We first present the INEX 2006 Best In Context task official evaluation results. Then we evaluate the efficiency of different focused retrieval strategies for the automatic detection of BEPs. Next, we have two objectives: first to evaluate the gain brought by the use of BEPs versus a standard 'flat' document retrieval approach; and second to compare the performance of our system with a well-known commercial search engine like Google on a realistic application-case.

The runs use no constraints for phrase matching and the documents were preprocessed for stop words removal and stemming. At indexing time the most frequent presentation tags and jump tags (i.e. *emph*, *collectionlink*, *languagelink*, etc.) were ignored (see Table 4.1 on page 73).

6.6.1 INEX 2006 Best In Context Task Official Results

We present in Figure 6.7 and Table 6.1 the INEX 2006 Best In Context task official results compared with the *avgMRD* focused retrieval strategy (see Section 6.2) for detecting the BEPs. The submitted run obtained several top ten results and was ranked on the 1^{st} place out of 77 official submissions by both the BEPD and EPRUM metrics when using A=0.01 – which rewards methods retrieving elements very close to a BEP.

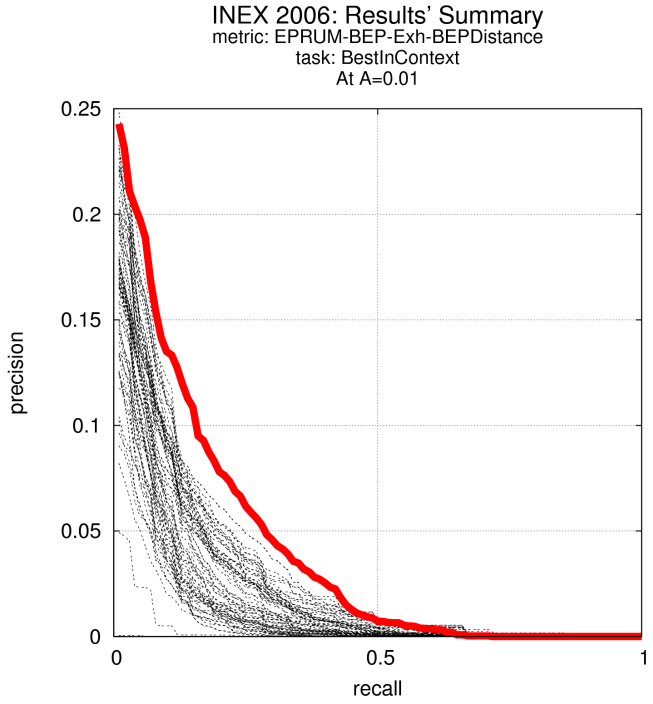


Figure 6.7: INEX 2006 Best In Context Task Official Results, Metric:EPRUM-BEP-Exh-BEPDistance, A=0.01.

6.6.2 Evaluating Different Focused Retrieval Strategies for the Automatic Detection of BEPs

We experimented with different settings for aggregating the elements relevance score while selecting the non overlapping answers (see Tables 6.2, 6.3 and Figure 6.8). With the exception of the *avgMRA* focused retrieval strategy that recorded a maximum gain of 1.4% on BEPD and 2.25% on the EPRUM measure for A=0.1, the other settings did not perform better than the run officially submitted to the INEX 2006 Best In Context task. The best performing strategies, *avgMRD* and *avgMRA* seem to have a similar behavior as shown in Figure 6.8. Therefore, in the following experiments we choose to use the run officially submitted to INEX 2006 for the comparison against the baseline. As a general observation, it seems that the *avg* aggregation strategy produces better results than the *max* aggregation independently of the strategy used to select the non-overlapping elements.

6.6.3 BEPs versus Document Retrieval

In Table 6.4, Table 6.5 and Figure 6.9 we present experimental results that show the superiority of the 'bep' approach versus its 'flat' competitor. The gain obtained by the *avgMRD* focused retrieval strategy is situated between 193.26% for A=0.01 and 20.36% for A=100 on the BEPD metric (see Table 6.4), – and 347.25% for A=0.01 and 41.21% for A=100 on the EPRUM metric (see Table 6.5). These experimental results seem to validate the utility of the focused approach from the users' point of view.

Surprisingly, slightly negative gains (with a maximum of -5.03% for the BEPD metric and -15.14% for the EPRUM metric) are recorded by the *google-bep* run versus the

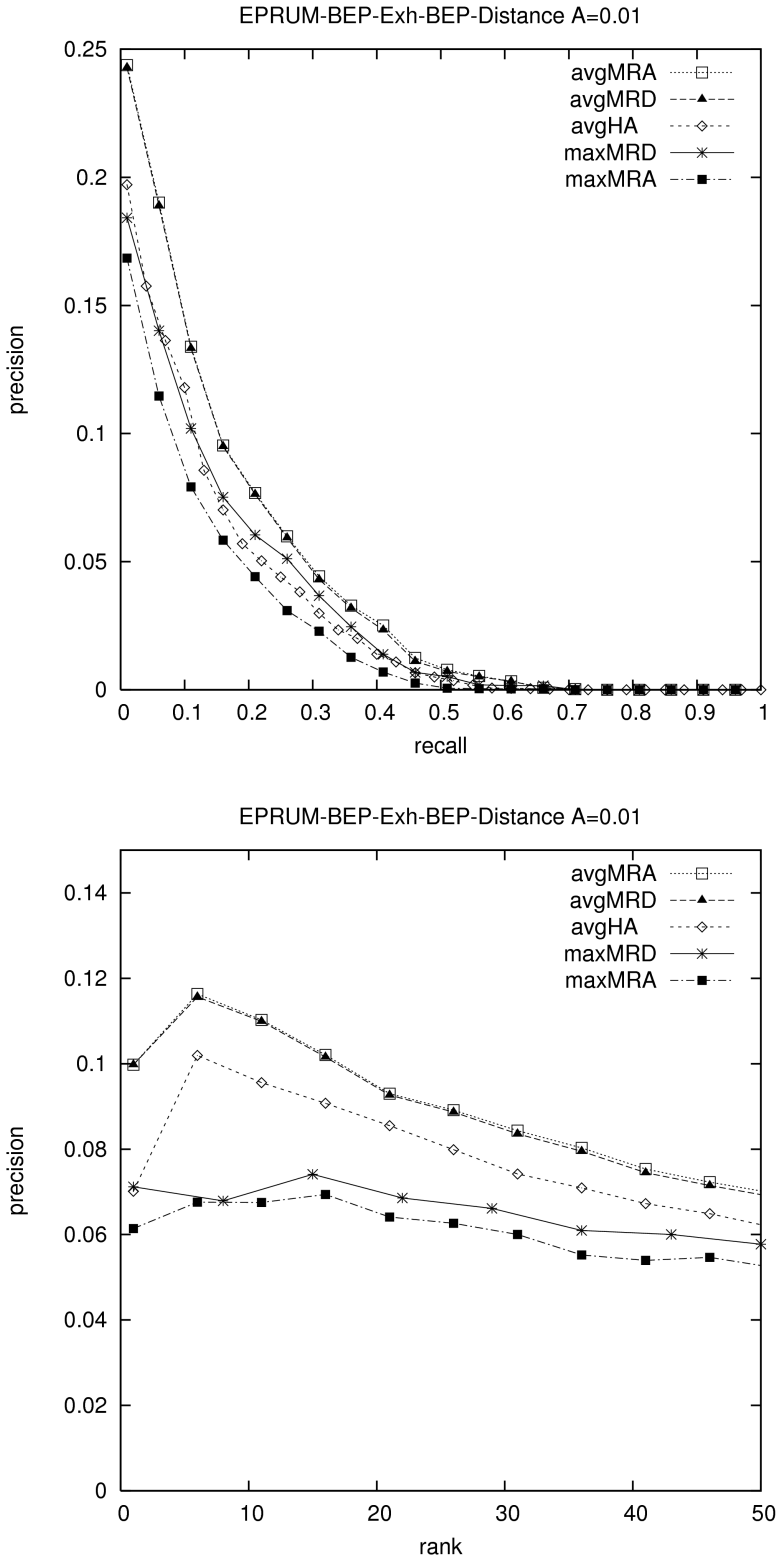


Figure 6.8: Precision at recall r values – top; and Precision at rank k values – bottom; for different methods of relevance node aggregation and overlap removing. Metric: EPRUM-BEP-Exh-BEPDistance, A=0.01.

	A=0.01	A=0.1	A=1	A=10	A=100
avgMRA	0.197	0.2604	0.3689	0.562	0.7558
avgMRD	0.1959	0.2568	0.3642	0.5596	0.7556
maxMRD	0.1642	0.2272	0.3315	0.4993	0.6453
maxMRA	0.1448	0.2377	0.3571	0.507	0.6427
avgHA	0.1518	0.2373	0.3473	0.4882	0.6161

Table 6.2: BEP-Distance results for different methods of relevance node aggregation and overlap removing.

	A=0.01	A=0.1	A=1	A=10	A=100
avgMRA	0.0412	0.0591	0.0873	0.1498	0.2195
avgMRD	0.0407	0.0578	0.0784	0.1489	0.2193
maxMRD	0.0313	0.0475	0.0743	0.136	0.1941
avgHA	0.0301	0.0486	0.0863	0.1132	0.1471
maxMRA	0.0234	0.0476	0.3473	0.1391	0.1919

Table 6.3: EPRUM-BEP-Exh-BEPDistance results for different methods of relevance node aggregation and overlap removing.

	A=0.01	A=0.1	A=1	A=10	A=100
avgMRD	0.1959	0.2568	0.3642	0.5596	0.7556
avgMRDflat	0.0668	0.1579	0.2749	0.4395	0.6278
Gain in %	*193.26*	*62.63*	*32.48*	*27.33*	*20.36*
google-bep	0.0522	0.0974	0.1415	0.2105	0.2914
google-flat	0.0446	0.0992	0.149	0.2142	0.2949
Gain in %	*17.04*	*-1.81*	*-5.03*	*-1.73*	*-1.19*

Table 6.4: BEP-Distance results and gains in % between the 'bep' and the 'flat' runs for the *avgMRD* focused retrieval strategy and for the baseline runs.

	A=0.01	A=0.1	A=1	A=10	A=100
avgMRD	0.0407	0.0578	0.0873	0.1489	0.2193
avgMRD-flat	0.0091	0.0259	0.0488	0.0888	0.1553
Gain in %	*347.25*	*123.17*	*78.89*	*67.68*	*41.21*
google-bep	0.0201	0.0471	0.0732	0.1156	0.1757
google-flat	0.0212	0.0555	0.0844	0.1234	0.1787
Gain in %	*-5.19*	*-15.14*	*-13.27*	*-6.32*	*-1.68*

Table 6.5: EPRUM-BEP-Exh-BEPDistance results and gains in % between the 'bep' and the 'flat' runs for the *avgMRD* focused retrieval strategy and for the baseline runs.

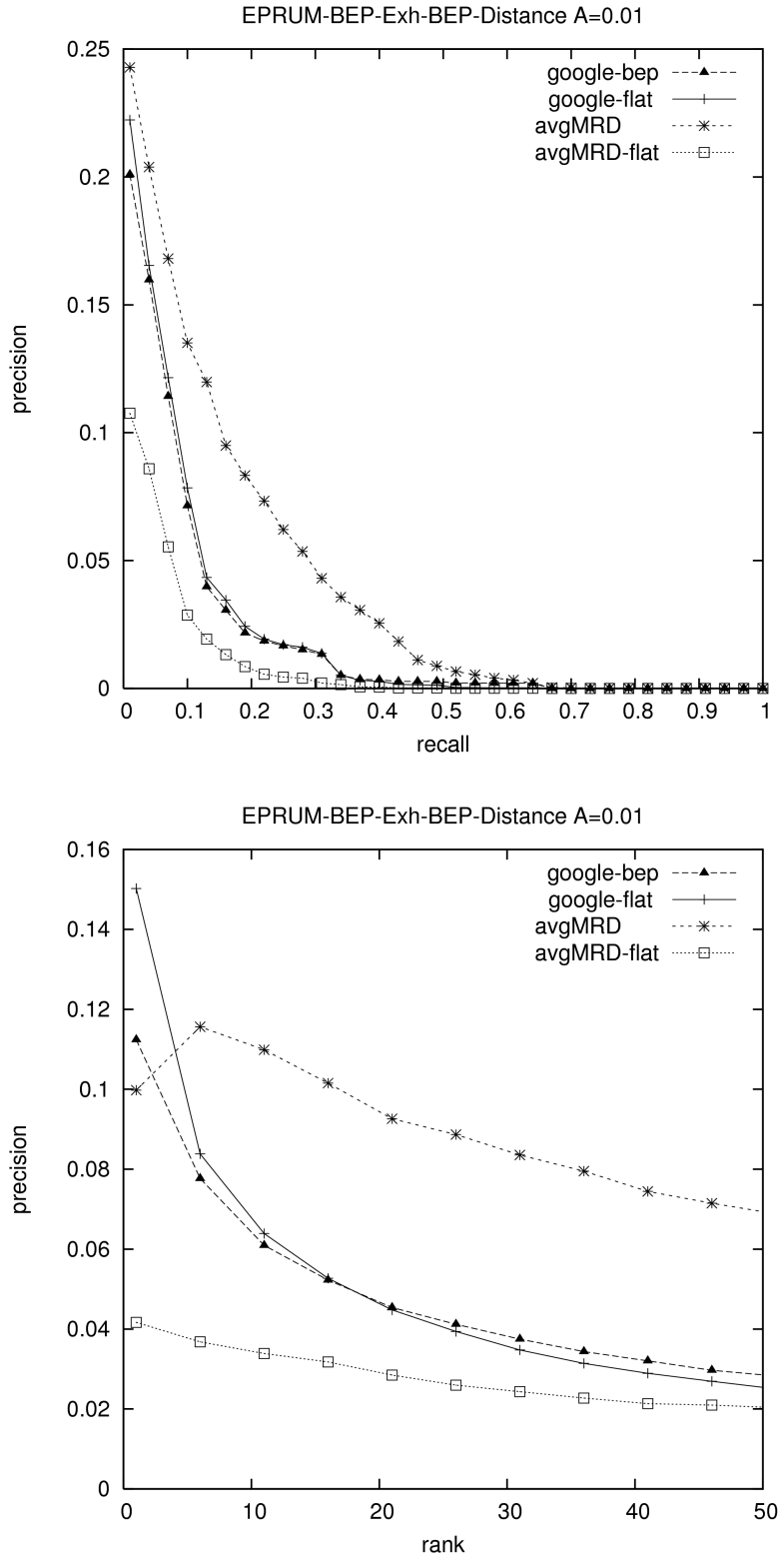


Figure 6.9: Precision at recall r values – top; and Precision at rank k values – bottom; for the *avgMRD* focused strategy and the baseline runs based on the Google search engine. Metric:EPRUM-BEP-Exh-BEPDistance, A=0.01.

		A=0.01	A=0.1	A=1	A=10	A=100
BEPD	**bep**	275.29	163.66	157.39	165.84	159.3
	flat	49.78	59.17	84.5	105.18	112.89
EPRUM	**bep**	102.49	22.72	19.26	28.81	24.82
	flat	-57.08	-53.33	-42.18	-28.04	-13.09

Table 6.7: Gains in % between the *avgMRD* focused retrieval strategy and the baseline runs using the Google search engine.

google-flat run. A single but significant positive exception of 17.04% is recorded for the BEPD metric when using A=0.01 – which favors runs that return elements very close to a BEP. This seems to show that the 'document snippet → BEP' matching strategy was indeed effective. The gain obtained by the BEPs strategies decreases while the value assigned to the A parameter increases, as more importance is given to the document ranking strategy. When evaluated with the EPRUM metric, the *google-flat* run is very competitive at document ranking, and therefore, is quite difficult to improve its results. This is not the case of the *avgMRD-flat* strategy that has rather poor results for document ranking and can fully take advantage of the contribution of the BEPs retrieval strategy. Nonetheless, the *google-bep* negative gains may indicate the fact that our 'document snippet → BEP' matching strategy is not excelling at detecting focused relevant information inside the retrieved documents. We remind here that both the positive and the negative results are due to only 35% of the valid 'document snippet → BEP' matches – the rest of the 65% of the results being strictly equivalent.

6.6.4 Real Application-Case

In Table 6.7 and Figure 6.9 we compare the retrieval performances of our XML IR system against a baseline that uses the Google search engine within the framework of the Best In Context Task of the INEX 2006 evaluation campaign. We use the English part of the Wikipedia collection and the Wikipedia XML Corpus [59] (i.e. 4.6 GB) to retrieve BEPs for 111 CO requests.

The results show a significant overall gain of the proposed focused strategy for the detection of BEPs against the baseline 'document snippet → BEP' run. The gain stands between 275% for the BEPD metric at A=0.01 and 159% at A=100, – and between 102% for the EPRUM metric at A=0.01 and 24.82% at A=100.

When comparing 'flat' runs – i.e. at document level – the BEPD measure reports gains between 49.78% at A=0.01 and 112.89% at A=100, – while EPRUM metric yields very poor results – i.e. 57.08% at A=0.01 and 13.9% at A=100 (see Table 6.7).

These results emphasize the fact that the proposed focused approach is competitive at detecting the relevant information within a retrieved document but yields poor performances at ranking the relevant documents. In order to improve the retrieval performances of the method, the ranking of documents based only on their BEP relevance (see Section 6.3) could be enhanced with a global relevance value computed at document level – i.e. TFIDF [195] for example.

By analyzing the first 150 results returned (see Figure 6.9) we observe that the Google runs are rather precision than recall oriented and that they outperform the focused BEP retrieval for the first 10 results retrieved (see Figure 6.9 – bottom).

As far as system efficiency is the concern, the average response time for 125 CO topics was 2.7 sec. on a computer with a 2.4 Ghz Pentium IV processor and 1.5 GB of RAM.

An extension of the approach using a vague interpretation of the structural constraints specified in the CAS topics is reported in [179]. The evaluation results did not show an improvement of the retrieval performances of the base method.

6.7 Conclusion

We have described an efficient and effective method to automatically identify BEPs in semi-structured documents. The method was evaluated on a realistic experimental dataset based on the Wikipedia collection within the context of the INEX 2006 evaluation campaign. The evaluation results were quite encouraging as it was ranked on the 1^{st} place out of 77 official submissions for the Best In Context task by both the BEPD and EPRUM metrics for A=0.01 – which rewards methods retrieving elements very close to a BEP. A comparison with a standard 'flat' information retrieval system returning documents snippets as BEPs was performed based on the Google search engine and showed the pertinence of the proposed algorithms for the focused access to the documents components. This may help users to obtain better access to relevant information inside XML documents. As a perspective to this work, we will explore the impact of aggregating BEPs heuristics and document ranking heuristics in order to better support the users' information-seeking behavior and to improve the overall performance of the search strategy.

Part III

XML Multimedia IR

Chapter 7

IR of Sequential Data in Heterogeneous XML Databases

The XML language is a W3C standard sustained by both the industry and the scientific community. Therefore, the available information annotated in XML keeps and will keep increasing in size. Furthermore, not only the volume of the XML information is increasing but also the various structures and media described are becaming more complex. The nowadays documents have evolved from simple plain texts to documents containing a mixture of textual, multimedia, and meta data information. These documents may have complex and heterogeneous structures and contents like sequential or time series data. In this chapter we introduce a retrieval scheme designed to manage sequential data in an XML context. The sequence extraction and matching scheme relies on two levels of approximation: on the structural localization/organization of the sequential data and on its content. To this end, we merge methods developed in two different research areas: XML information retrieval and sequence similarity search. A preliminary version of the approach and results presented in this chapter was published in [173].

Contents

7.1 Introduction

As mentioned above, the XML language is a W3C standard that has rapidly been adopted and sustained by both the industry and the research community. In the recent years, we are witnessing an increasing volume of XML digital information produced through day-to-day or by specialized scientific activities. The XML documents evolved from plain structured text representations to documents having complex and heterogeneous structures and contents: multimedia description (MPEG7-DDL [166]) and synchronization (SMIL [2]), mathematical formulas (MathML [3]), time series or sequences. We focus on the last two mentioned categories that are a ubiquitous form of data in financial, medical, scientific, musical or biological applications.

Flexible querying of scientific experimental results, patients medical records, financial summaries, musical pieces or biological sequences published as XML documents are only a few examples of applications that involve managing sequential data in an XML context. We can thus state that there is a real need for high-performance systems and methods able to extract, index, and query heterogeneous types of sequential information from heterogeneous collections of XML documents.

In musical and biological fields, special DTDs have been designed for MIDI files – MidiXML [4], musical scores – MusicXML [5] and biological sequential data [191] representation. In these cases the applications handle normalized sequential data and *data-centric* oriented XML documents. Data-centric documents have usually fairly regular structure, fine-grained data and little or no mixed content.

A heterogeneous collection of XML documents contains many un-normalized or various kinds of sequential data. A *document-centric* view of the collection is well suited to this kind of data. Document-centric documents have less regular or irregular structure, larger grained data and lots of mixed content. Furthermore, the order in which sibling elements and PCDATA occurs is almost always significant.

XML information retrieval is closely related to the document-centric view and provides the possibility of querying the information acquired by a system having an incomplete or imprecise knowledge about both the structure and the content of the XML documents [14, 15, 41, 66].

One basic requirement of an XML query engine based on information retrieval concepts is to dispose of vague predicates/specialized similarity operators to adequately manage different data types [66] and to improve the precision of the IR system [61].

The approaches proposed in [15, 41, 66] study the flexible querying on both XML structure and content (usually text), but do not specifically take into consideration

sequential/time series data, nor its organization within the XML documents, which is the focus of our approach.

7.2 Background and Related Works

Multimedia information systems are widely recognized to be one of the most promising fields in the area of information management [26, ch. 11]. As the name literally implies, designing a multimedia (i.e. more than a single media) information retrieval system requires to mix knowledge, techniques and tools from several domains. For this reason, the development of a multimedia system is substantially more complex than a traditional information system. Multimedia systems must have the capabilities to store, retrieve, transport and present data with very heterogeneous characteristics such as text, structures, images, sound, video or sequential/time series data [26, ch. 11]. Important research efforts and well established research communities sustained the development of strong and independent research fields (in text, images, audio, video, sequences, or time series retrieval). Compared to similarity on individual object domains, complex multimedia documents that feature many different media objects have received little attention, so far [211]. Previous research works in multimedia information retrieval commonly focused on combining textual and image retrieval techniques. A recent survey on indexing and retrieval of multimodal documents for text and images can be found in [46].

The work most related to ours is mainly in the area of XML retrieval of atomic multimedia objects, and in particular images. We will focus here on approaches that are able to integrate the structure of the documents in the retrieval process of multimedia documents/components.

One of the incipient papers that tackled the information retrieval of multimedia objects in an explicit hierarchical structured context is [47]. The authors proposed a model for structured multimedia documents and studied the notion and the impact of the structure on the information retrieval process. They proposed to merge IR and hypermedia for querying and browsing multimedia documents. Their model allows to manipulate the logical hierarchical structure of the documents, the semantic content, the attributes of the document components and the navigation links. They also introduced the notion of *index objects* defined as classes of structural objects which, depending on application requirements, correspond to retrievable objects. In our proposal, the extracted sequences can be seen as an instantiation of the index objects concept.

$POQL^{MM}$ [89] is a general purpose query language for structured multimedia documents. It employs regular path expressions to compute sets of objects or sets of attribute values which can be reached via links in a query and allows the combination of various feature extraction operators for different media types. The structure of the documents that should be retrieved can be specified in much detail but a flexible interpretation is not allowed.

In [113] the authors introduce a method based on XLink and XPath integration to exploit the intersection of hierarchical and linking information within XML-based multimedia documents. They define the *regional knowledge* of a multimedia object as the combination of the textual content of its surrounding XML elements. They identify three kinds of such elements: caption or description, sibling text information and hierarchical text information. The contribution of the different regional knowledge levels

to the multimedia object description may be weighted by using a *relevance strength* function. The function is dependent on the tree level distance between the specific region and the element containing the multimedia object. They also propose to integrate two kind of *linking knowledge:* a *textual linking* which can be used to link the multimedia object with different textual descriptions or metadata, and a *multimedia linking* between low-level feature of similar multimedia objects. The last proposal is similar to our sequence extraction process that implicitly create virtual links between the sequence symbols. The difference consists in the fact that we use type compatibilities and similarities between the structural locations of the symbols to decide if a "virtual link" can be created – i.e. the extracted symbols occur in the same sequence. In their further works [114, 116], the authors investigate the impact of the XML logical structure on the retrieval of XML multimedia objects. More precisely, they evaluate the impact of the combination of different disjoint text level regions surrounding a multimedia element on the retrieval effectiveness of the multimedia objects . The evaluation was carried out on a test collection that was adapted for XML multimedia retrieval starting from a subset of the INEX IEEE collection.

Recent XML-related work was carried out as part of the INEX multimedia tracks [232, 243]. The INEX multimedia track[1] provides an evaluation platform for the retrieval of multimedia document fragments – i.e text, structure and images. The objective of the INEX multimedia track is to exploit the XML structure that provides a logical level at which multimedia objects are connected, to improve the retrieval performance of an XML-driven multimedia information retrieval system [243]. The task set for the multimedia track is to retrieve relevant document fragments based on an information need with a structured multimedia character. A structured multimedia document retrieval approach should be able to combine the relevance of the different media types into a single (meaningful) ranking that is presented to the user [243].

Simple fusion techniques are used to merge similarity scores for text/structured and multimedia data. For instance [126] fusion multiple search algorithms on image low-features with an XML retrieval approach based on a TF-IDF variant. In [96, 95] a simple linear interpolation is used to merge relevance score from a full-text IR system (Zettair), a native XML database system (eXist) and the GNU Image Finding Tool (GIFT).

[92] uses the text surrounding the image and the structure of the documents (i.e. the descendants nodes, the brothers nodes, the ancestors nodes and the image name) to judge the relevance of an image element. Finally the relevance values are propagated in the document tree and combined with textual and structural constraints to obtain a final score for the XML multimedia fragment.

In [230] a principal component analysis (PCA) is used to derive a composite ranking for a set of XML elements that have a multimedia character. Three strategies are defined: annotation-based (which uses the caption of an image to find related images using a keyword-based search), content-based multimedia retrieval (which uses PCA to derive a composite ranking for text and images present within an XML element) and a combination of the first two strategies.

In [115] a Bayesian network incorporating element language models for the retrieval of a mixture of text and image is proposed. An element language model is applied upon each XML element. The framework combines the language models asso-

[1]INEX Multimedia Track `http://inex.is.informatik.uni-duisburg.de/2006/mmtrack.html`

ciated with the elements used to perform the retrieval of the multimedia content using the inference network model. An element-based collection language model is used in the element language model smoothing. Both the semantic level (i.e. the names of the elements) and the logical level (i.e. the hierarchical organization of the elements) of the structure are considered. This approach showed promising results when evaluated on the INEX 2005 multimedia collection.

All the above XML multimedia approaches concerns the retrieval of multimedia fragments with focus on images and are not appropriate for the retrieval of heterogeneous sequential data.

An interesting related approach is [246] that propose an XML repository for molecular sequence data annotated in XML. The model proposed allows for flexible sequence search but only strict structural matching for the annotation tags.

Another related works concerns the similarity search of sequential/time series data and structure approximate search methods [37]. The similarity search of sequences in biological applications are widely discussed in [83] while [90] survey recent methods for efficient retrieval of similar time sequences. Some other related works can be found in [161] which surveys recent research topics in the musical information retrieval field.

7.3 Challenges in XML Sequential Data IR

In this section we try to emphasize the differences between the retrieval of atomic multimedia objects, e.g. a single image, and sequences of multimedia objects (e.g. sequential data, musical scores, a sequence of temporal ordered photos). What are the main differences?

Sequential data is not well defined as an atomic object since:

- it may cross structural boundaries of the XML elements (i.e. Midi XML format),
- it may have a variable number of symbols or atomic units,
- it may be (re-)ordered, or constrained to preserve their original document/collection order.

Therefore, the sequential data must first be detected and extracted (mined) from the XML documents. For image retrieval for instance, we have the objects in hand and we try to extract useful features to obtain a good description of the object. When considering mining XML documents for sequential data, we had to ”reverse engineering” this process: since we have to detect and extract the sequential/time series data from their XML description. Then we can apply specific transformations to extract features suitable for the indexing and search processes.

After the sequence extraction, we should take into account the structural organization of the sequences into the retrieval process. For this purpose we have to preserve the structural information/organization of the indexed sequences, where the sequences may have an homogeneous or an heterogeneous structure. Finally, we have to weight and merge the sequential content and textual or structural information to compute a final ranking score.

To provide adequate approximate operators for managing sequential/time series data in a heterogeneous XML environment, in our work, we merge methods developed

in two different research areas: XML information retrieval and sequence similarity search.

In Section 7.4 we introduce the underlying data model of our application and we identify relations between the structure of the XML documents and several common types of sequential data. In Section 7.6 we present a hybrid indexing scheme allowing the implementation of semi-structured and sequential searching operators. In Section 7.7 we devise an approximate searching scheme for ranking the results by taking into account similarities between both the structural location and the content of the sequential data with the user requests. In Section 7.9 we carry out preliminary experiments dedicated to MIDI files retrieval embedded in heterogeneous XML databases. Finally, in Section 7.10 we summarize our conclusions and present some perspectives for our work.

7.4 Data Model

An XML document can be represented by an ordered tree whose nodes contain heterogeneous pieces of information (TEXT or PCDATA such as (parts of) sequences or time series). Each XML element may be related to attributes "name–value" fields, and each attribute value may contain (a part of) a sequential data.

For instance a phone number (a whole sequence) or a musical note (a sequence symbol) can be seen either as the content of an XML element node or as an XML attribute value.

In order to describe a sequence embedded in a heterogeneous XML environment we concurrently use its structural location in the collection – i.e. the set of XML contexts associated with the sequence symbols, and its content – i.e. the sequence symbols values.

7.4.1 XML Context

XML documents are generally represented as rooted, ordered, and labeled trees in which each node corresponds to an element and each edge represents a parent-child relationship – see Figure 7.1 and Figure 7.2 for an example.

According to the tree structure, every node n of the XML tree inherits a path $p(n)$ composed with the nodes that link the root to the node n. More precisely, $p(n)$ is an ordered sequence of nodes $p(n) = n_0 n_1 ... n_i ... n_d n$, where n_0 is the root node and $d + 2$ the length of the sequence. An unordered set of $< attribute, value >$ pairs $A(n_i) = \{(a_j, v_j)\}$ may be attached to each node n_i of the ordered sequence, so that $p(n)$ can be represented as follows:

$$p(n) =< n_0,\ A(n_0) >< n_1,\ A(n_1) > ... < n_n,\ A(n_n) >$$

A node n in the document tree can be decomposed into structured and unstructured sub-elements. Moreover, an Unstructured Sub-Element (USE) or an attribute value v_j may be decomposed into tokens t_i (or words, symbols). Each token t_i is related to an XML context $p(n)$ that characterizes its occurrence within the document. For instance, in the example from Figure 7.1 the $TextEvent$ node value "$date : 8 - 19 - 91$" is associated with the following XML context

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<MIDIFile>
  ...
  <Track Number="0">
    ...
    <Event>
        <Absolute>0</Absolute>
        <TextEvent>date:8-19-91</TextEvent>
    </Event>
    ...
  </Track>
  <Track Number="1">
    ...
    <Event>
        <Absolute>576</Absolute>
        <NoteOn Channel="1" Note="71" Velocity="127" />
    </Event>
    ...
    <Event>
        <Absolute>768</Absolute>
        <NoteOn Channel="1" Note="76" Velocity="127" />
     </Event>
     ...
  </Track>
  ...
</MIDIFile>
```

Figure 7.1: An excerpt of an MidiXML [4] file.

$$
\begin{aligned}
p('date:8-19-91') \quad \rightarrow \quad & < MIDIFile,\ \{\emptyset\} > \\
& < Track,\ \{(Number,\ 0)\} > \\
& < Event,\ \{\emptyset\} > \\
& < TextEvent,\ \{\emptyset\} >
\end{aligned}
$$

7.4.2 Sequential Data

The XPath 1.0 Recommendation [60] defines the concept of *document order* as the order in which the first character of the XML representation of each node occurs in the XML representation of the document after the expansion of general entities, except for namespaces and attribute nodes whose document order is application-dependent. This represent the linear or sequential reading order of the document from left to right by ignoring the order of the attributes and attributes values which is not specified.

The XML document structure and the document order encode useful and potentially (semantically) rich information about the sequential organization of the data. We describe and formalize hereinafter our approach in exploiting this kind of information for XML sequence extraction and representation.

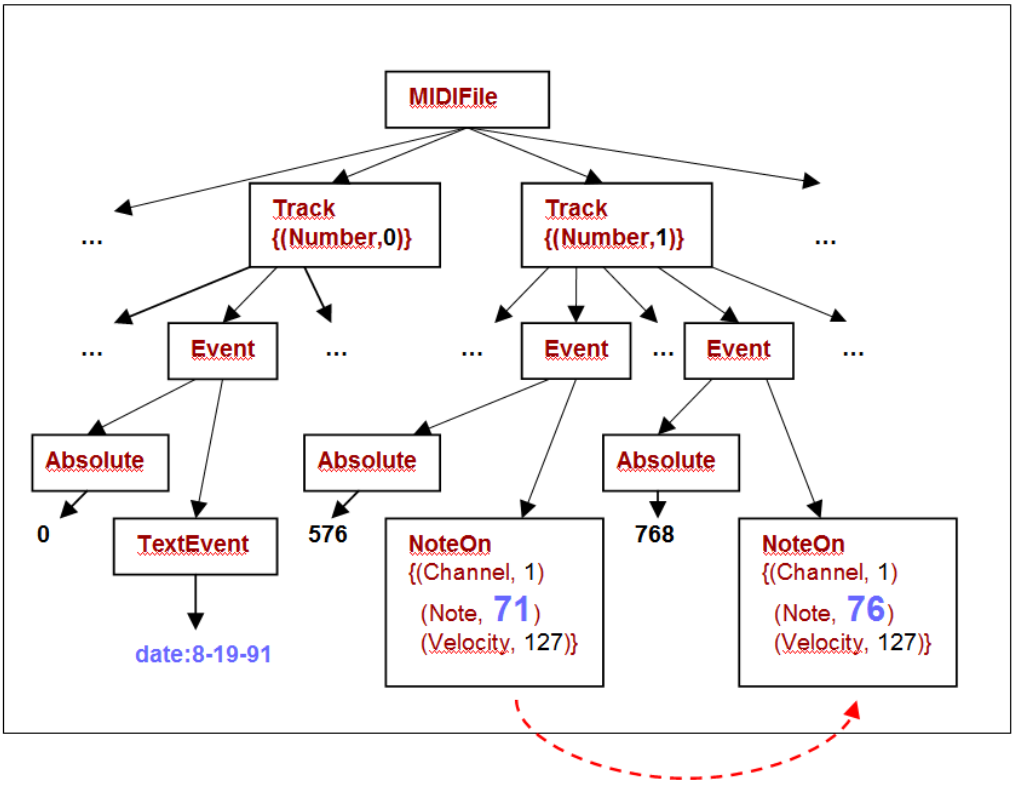


Figure 7.2: The XML tree representation for the MidiXML file from Figure 7.1 showing a document level sequence.

Sequence Definition. Formally, a sequence

$$S = s_0 s_1 ... s_i ... s_m$$

is defined relatively to a collection of XML documents as an ordered and finite non-empty set of symbols s_i selected from an alphabet Ω. An alphabet symbol s_i may be represented by:

- $v_j \in A(n_i)$ an attribute value,
- $t_i \in v_j \in A(n_i)$ a token composing an attribute value,
- USE_i an unstructured sub-element[2] of one of the XML nodes n_i,
- $t_i \in USE_i$ a token of an unstructured sub-element USE_i.

For instance, in the MidiXML file from Figure 7.1, a symbol value may indicate the "0" value of the *Number* attribute – for an attribute value type, or the *TextEvent* content – $date : 8 - 19 - 91$, for an USE type.

A sequence symbol s_i is linked with one of the unique indexed reference locators $rl_0 rl_1 ... rl_i ... rl_n$ of the XML collection set. A reference locator rl_i points to a unique position into the collection of XML documents and includes a reference to the XML context $p(n)$ of the symbol.

[2]In the case of a node having a mixed content, only the unstructured content is considered.

A regular sequence symbol s_i can be associated with an order key o_i (e.g. a timestamp or an id). The order key has the same characteristics as the sequence symbol itself. In order to be able to link a sequence symbol with an order key, they must be associated with the same root-to-leaf path of the XML tree. For instance we can associate the sequence *id* value *"01"* from Figure 7.3 as an order key for the attribute *name* value *"NUCLEAR RIBONUCLEOPROTEIN"* or with the content of the *residues* element *"SKSESPKEPEQLRKLFIGGLS..."*.

```
<?xml version="1.0" encoding="US-ASCII" ?>
<cluster>
   <note>
     Collection of sequences from SWISS- PROT
   </note>
   <seq id="01" name="NUCLEAR RIBONUCLEOPROTEIN">
    <dbxref>
      <database>SWISS-PROT</database>
      <unique_id>P09651</unique_id>
    </dbxref>
    <residues type="aa">
       SKSESPKEPEQLRKLFIGGLS...
    </residues>
  </seq>
  <seq id="02" name="NUCLEAR RIBONUCLEOPROTEIN">
    <dbxref>
      <database>SWISS-PROT</database>
      <unique_id>P04953</unique_id>
    </dbxref>
    <residues type="aa">
       QLRKLFIGGLSSKSESPKEPE...
    </residues>
   </seq>
</cluster>
```

Figure 7.3: An excerpt of an annotated protein sequence extracted from the Swiss-Prot database.

Sequence Structural Types. The above sequence definition allows to describe sequences of symbols associated with any arbitrary XML contexts from the collection. From a more practical point of view, several particular structural types of sequences frequently occur and prove to be of interest:

node level sequence: the whole sequence is contained in a single XML element or a single attribute value. The leafs nodes are favorites candidates in this case. This sequence representation is widely used in bioinformatics [191]. An example of an annotated protein sequence extracted from the Swiss-Prot database[3] is given in Figure 7.3 while its XML tree representation is shown in Figure 7.4.

[3]The Swiss-Prot database www.ebi.ac.uk/swissprot/

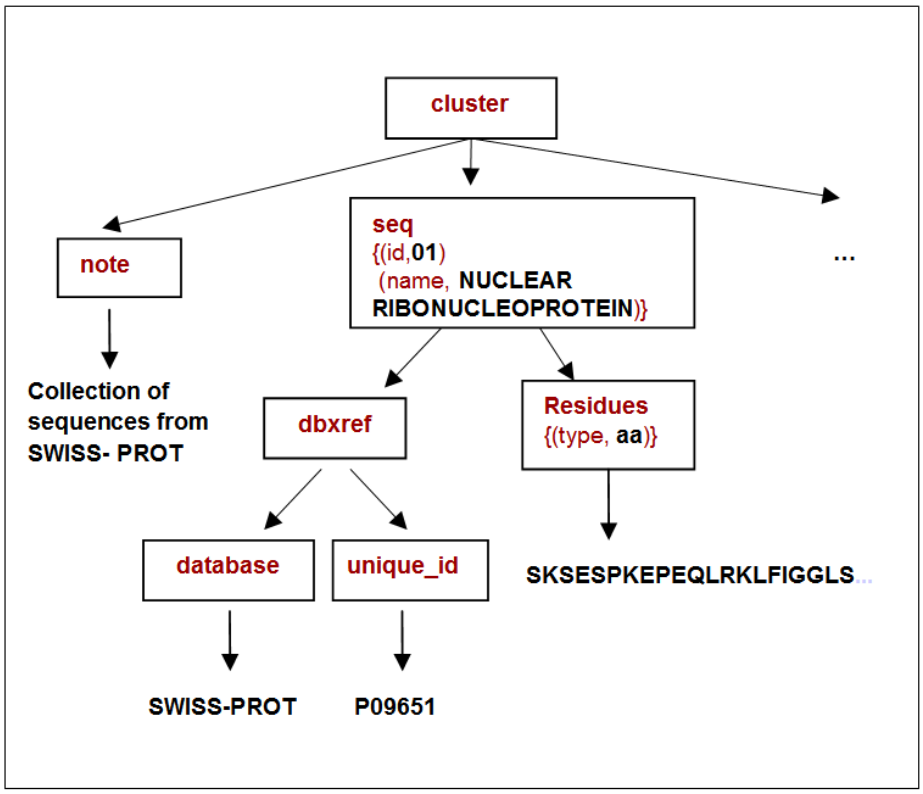


Figure 7.4: The XML tree representation of the annotated protein sequence from Figure 7.3.

document level sequence: these sequences are composed by the symbols associated to similar XML contexts of a single XML document – i.e. this includes perfect paths matches (e.g. see the link between the two *NoteOn* nodes of the XML tree from Figure 7.2), sibling nodes and nodes having a k-level common ancestor. For instance, this representation is imposed by the DTD's used in the musical field [4, 5],

collection level sequence: the sequences are composed with the symbols associated to similar XML contexts by crossing the physical boundaries of the XML documents. This sequence type may be of interest when searching sequences of information spread among several documents and that are not entirely dependent on the document order – e.g. for example information that is extracted from different medical records of a patient by using a social security ID number and ordered according to the available timestamps.

One of the main advantages of the first two sequence types is the possibility of maintaining the document order by default when two ordering keys o_i and o_j, with $i \neq j$ have identical values or the sequence order relation $\preceq$ is either:

- unspecified (no valid ordering key o_i has been extracted and associated with the sequence symbols s_i), or
- a partial order (indeterminate for certain cases, such as relationships between temporal information: durations, dateTime as defined in XML Schema Part 2:

"Datatypes Recomandation" [138].

For sequences constructed from symbols that are extracted from different XML documents, the document order can be locally applied within each XML document, but no global order of the symbols in the sequences can be inferred without the use of external information (i.e. order keys provided by the users) and/or heuristics.

A simple example will be the use of the document creation date for ordering the documents. We may also consider using the timestamp information associated to the symbols that are the closest ancestors into the XML trees. In our scheme we make no assumptions about the global order of the symbols extracted from different XML documents.

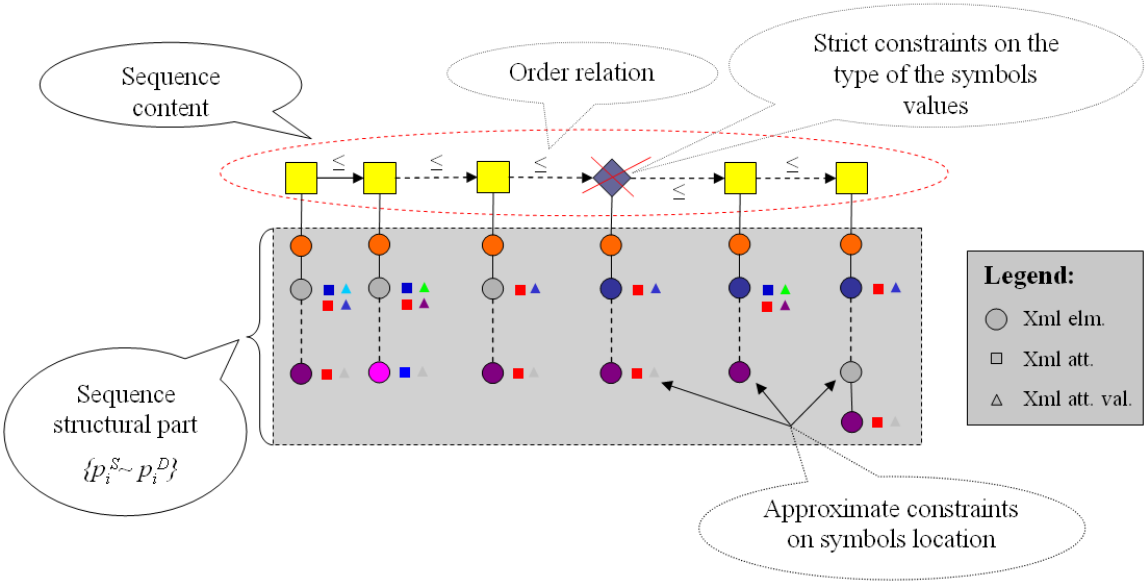


Figure 7.5: The sequence model.

Sequence Model. In this framework, we propose a sequence model – see Figure 7.5 – in which the symbols s_i are organized in sequences S by taking into account:

- similarities between their structural positions in the XML document tree(s) (i.e. using the XML context $p(n)$),
- type compatibility between their values (*numbers*, *dates* or *strings*) and
- a partial order relation $\preceq$.

A symbol s_i may occur in more than one sequence S and a sequence may contain symbols s_i with identical values.

7.5 Sequence Extraction

The users' interests in a heterogeneous XML environment can be highly diversified. Some users could search the chorus of a musical piece while someothers will seek similarities between the blood pressures curves of several patients' medical records.

In these conditions we will probably fail to index all the possible sequences that could match the user's subjective and time evolving interests.

We assume that the users or the system administrators detain at least an imprecise, incomplete or approximate knowledge of the particular underlying organization of the sequential data in which they are interested in. This assumption qualifies them to supervise a sequential data extraction process according to their specific needs – see Figure 7.6.

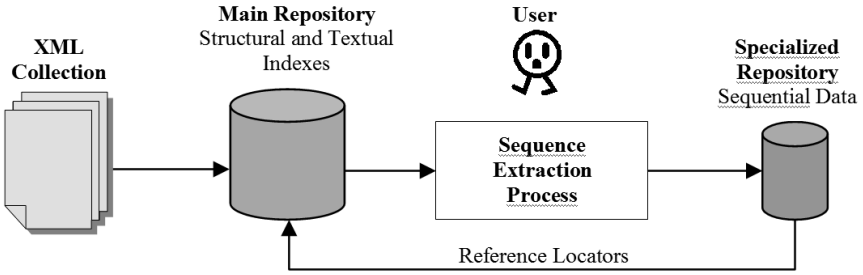


Figure 7.6: The sequence extraction process.

The Sequence Extraction Process. The sequence extraction process is based on a constructor operator $makeSeq$ (see Figure 7.7) that receives three arguments:

- A sequence extraction pattern represented as a path request $p^{R_{StrExtrPattern}}$ with the type (*number* – default value, *date*, or *string*) and position of the requested symbol s_i and (optionally) of an order key o_i (i.e. another symbol). The path request may be restricted by textual terminals $\{t_i\}$ and merging operators at element level (SEQ, AND, OR, WITHOUT, SAME, SAME+, etc).
- A minimum accepted threshold $\tau \in [0...1]$ for the relevance (both at structural and content levels) of the current XML context relative to the sequence extraction pattern provided by the user. The current context either is or includes the path location p_i^D of the symbol s_i in the XML tree of the document D . The threshold value $\tau \in [0...1]$ is used to decide if the current detected symbol s_i will be included or not within the extracted sequence.
- The sequence structural expected type: *node*, *document* (default value), or *collection*.

For instance, the Figure 7.7 shows an example that extracts all the notes from the twelve channel of the track ten of a MidiXML file and builds a strict – i.e. the structural threshold value is 1.0, no flexibility is allowed for the structural pattern match – sequence with numeric values at document level. The sequence preserves the symbols document order as no *orderBy* clause is specified.

During the extraction process, the compatibility constraints expressed on the symbols values are mandatory and treated strictly while the constraints on the structural localization of the symbols in the XML tree and their order relation are treated vaguely (see Figure 7.5).

```
(makeSEQ
  (OR [/midifile
      /track(== number 10)
      /event
      /noteon(and (RETURN @note NUMBER) (== channel 12))/])
  1.0 %threshold for the symbols relevance matching
  DOCUMENT %sequence structural type
)
```

Figure 7.7: $makeSEQ$ sequence extraction operator.

If the global matching score between the current XML context and the sequence extraction path request $p^{R_{StrExtrPattern}}$ using a customized editing distance is above a given threshold, the algorithm tries to detect the location of the required symbols within the current XML context.

The detection process of symbols values trace backwards the dynamic programming distance cumulative table. The purpose is to find one of the possible optimal paths that obeys the type compatibility constraints for both the symbol s_i and the order key o_i values (if an order key is specified).

The chosen alignment/transcript tries to concurrently match the positions of the requested symbol and of the ordering key with the elements of the current XML context. The matching of the two positions may take into consideration both the content type compatibility (this is mandatory) as well as the degree of satisfaction of a local structural match criterion at element level. I.e.:

- a "perfect" element match – matching both the XML element names and the attribute values constraints is necessary for a *return @attName* operator that retrieves an attribute value. For instance the query *(OR [/MIDIFile/Track-/Event/NoteOn(AND (RETURN @note NUMBER) (== channel 1))/])* requires that both the *NoteOn* element name as well as the attribute meet the matching conditions. That means that we had to extract only *@note* numeric attribute values from the first *channel*.
- only an "acceptable" match – XML element names matching – is necessary for a *return this* operator that retrieves an XML element content. For instance, *(OR [/MIDIFile/Track/Event/TextEvent(RETURN THIS STRING)/])* requires to match only the *TextEvent* element name and the symbol*'s* string type compatibility constraint.

The sequence extraction method searches in all the possible optimal path alignments of the two contexts until the type and structural conditions on the required symbol and order key are satisfied. The heuristic used to guide the search for the optimal alignments advantages the matching of the most specialized elements/attributes by allowing replacing/matching, inserting and deleting operations to occur in the traceback process in this particular order. At the end, if no optimal alignment satisfying simultaneously the required local structural constraints and the compatibility conditions specified on the value types was found, no answer is returned.

Therefore, applying the $makeSeq$ operator to the input XML data – i.e. to the collection of XML contexts – retrieves a set of (valid type) symbols associated with

XML contexts $\left\{s_i \rightarrow p_i^D\right\}$ that are relevant relative to the sequence extraction pattern provided by the user $p^{R_{StrExtrPattern}}$.

During this phase of the process, the sequences are built as indicated by the sequence expected type parameter and are eventually ordered by using the symbols order keys o_i. In the case of a mixture of symbols associated or not to an order key within the same sequence, the symbols without an order key are discarded.

The time complexity of such a sequence extraction algorithm includes a linear dependence with respect to the number of nodes of the indexed database – i.e. if no textual terminal is provided, we use a dummy textual terminal that linearly scans all the path structures found in the database. The dynamic programming table for computing a customized editing distance between the sequence extraction pattern and a single path associated with an indexed symbol can be filled out in

$$O\left(\overline{length(p^{R_{StrExtrPattern}})} \cdot \overline{depth(T^D)}\right)$$

time [83, page 220], where:

$\overline{length(p^{R_{StrExtrPattern}})}$ is the average length of the request path included in the sequence extraction pattern and

$\overline{depth(T^D)}$ the average depth of the document trees from the collection.

Once the dynamic programming table has been computed, an optimal edit script can be found in

$$O\left(\overline{length(p^{R_{StrExtrPattern}})} + \overline{depth(T^D)}\right)$$

time [83, page 222].

As statistical studies on XML documents collections [149] have shown, the average depth of the XML documents trees is 4. If we consider that the sequence structural extraction patterns have a comparable length, this time complexity remains reasonable enough.

By summing up, if no textual terminals are used to alleviate or to control the sequence extraction process, and if we consider that the product of the average depth of the XML documents trees (i.e. 4 levels) and of the requests length can be neglected, the time complexity necessary to extract a sequence may be considered to be linear with respect to the number of the indexed XML paths of the collection.

7.6 Indexing Scheme

We propose a hybrid index model (see Figure 7.8) designed to merge both types of data: semi-structured and sequential data.

7.6.1 Main Repository

The main repository uses inverted lists as basic structures to encode textual and structural information. For this model, the entries of the inverted lists are of three kinds:

- structural entries, i.e. nodes n_i of the XML tree,
- tokens of the unstructured sub-elements of the XML tree nodes $t_{i,j} \in USE_i \in n_i$,

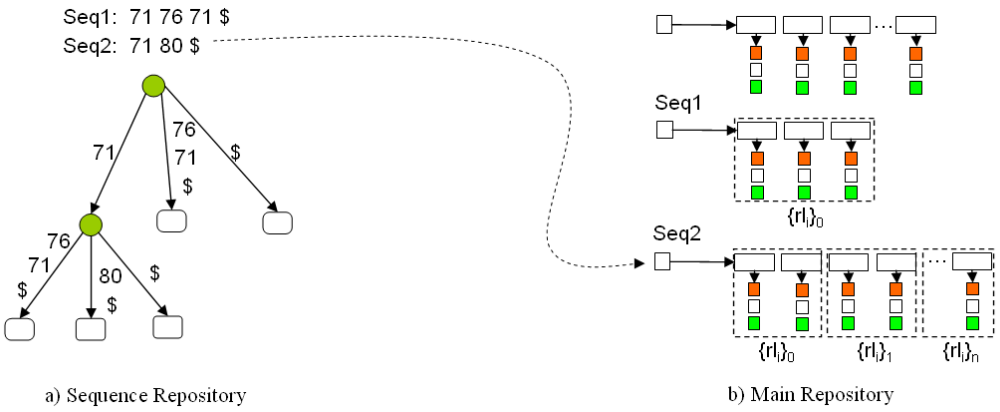


Figure 7.8: The index model.

- sequential entries, i.e. unique sequence IDs. – $USIDs$.

Each entry is associated to a list of reference locators rl_i pointing to a unique node position in the collection of XML documents. A reference locator rl_i refers to three pieces of information:

- a link to the URI of the document,
- a link toward the XML context $p(n_i)$, and
- an index specifying the location of the token $t_{i,j}$ within the XML node n_i.

For the structural entries (i.e a node n of the XML tree), only the link to the URI of the document and a link to the XML context $p(n)$ are required.

The inverted list resulting from the indexing process is encoded using binary randomized search trees, namely TREAPS as defined in [140], associated to hashtable structures. TREAP structures can balance the search tree according to the frequency of requested items. This provides an access speed-up over the use of regular hashtables [148]. The inverted lists are implemented as disk-resident index structures.

7.6.2 Sequence Repository

A supplementary index structure is used to optimize the management of sequential data. A well known and efficient data structure for indexing and searching sequential data in text processing [154] and bioinformatics applications [146, 94] is the generalized suffix tree (GST).

In a nutshell, the *suffix tree* is an indexing structure for all the suffixes of a string and it can be constructed in linear time and linear space [144]. A suffix-tree is a data structure that exposes the internal structure of a sequence in a deeper way than any other data structure such as inverted index. Suffix tree is an important data structure for indexing text strings and sequences since we can search for patterns in the data efficiently and the search complexity is independent of the original sequence size. There exist many practical applications that rely on suffix tree, especially for processing biological sequence data [83].

A *generalized suffix tree* (GST) is a suffix tree for representing all the suffixes of a set of strings [83]. In our implementation we use the Ukkonen's on-line construction algorithm [227] for building a GST.

For each extracted sequence S, all its symbols values s_i are indexed in the generalized suffix tree structure with respect to the sequence order relation $\preceq$ – see Figure 7.8.

The same sequence of indexed values S can have associated multiple sets of reference locators as it may occurs in different locations in the collection of XML documents:

$$S \rightarrow \{rl_i\}_0\{rl_i\}_1...\{rl_i\}_n$$

We consider a sequence to occur in two different locations in the XML collection if:

- all its symbols values $\{s_i\}$ are equal with the ones of a previously indexed sequence with respect to the sequence order relation $\preceq$, and
- at least one of the reference locators rl_i associated with the sequence symbols s_i are not matching.

Consequently, for each indexed sequence we receive a unique sequence id $USID$, from the generalized suffix tree. The $USID$ is further used as an entry key in the inverted lists to link the multiple sets of reference locators to the referred sequence – see Figure 7.8. The sets of reference locators have all the same cardinality $|S|$ and are stored contiguously in the inverted list. The order of the reference locators from each set complies with the order relation $\preceq$ used to build the sequence.

7.7 Searching Scheme

We introduce a searching scheme designed to manage unstructured sequential / time series data in an XML context based on two levels of approximation:

- on the structural localization/organization of the sequential data, and
- on its content.

7.7.1 Sequence Structural Approximate Matching

Since the user cannot be aware of the exact unstructured content or the complete structure of the XML documents due to their heterogeneity, the searching process should involve exact and approximate matching mechanisms. The format of the indexed documents being XML, it is natural to consider that the structural query itself complies, at least partially, with the XML standard. If so, the structural searching mechanism can be handled through approximate tree matching algorithms that try to match the query XML tree to the XML trees for the corresponding indexed documents.

Tree matching algorithms exist based on editing distance and mappings. This kind of tree matching is not suited to the task we intend to perform. First of all, the matching complexity is too high – $O(n^3)$ [58] – considering the size of documents and databases we want to handle. Secondly a sequence may involve paths of more than one XML tree of the collection. Therefore we have chosen to perform an approximate search based on the matching of $p(n)$ sub-structures of the indexed XML trees. In other

words, we are rather dealing with approximate root–leaf or root–node path alignment rather than complete tree matching algorithm.

In particular, a sequence S is defined as an ordered non-empty set of finite symbols s_i each of them having attached a path p_i^D from the root of the XML tree of the document D to the symbol node location in the tree. The ordered set of p_i^D sub-structures of the XML trees represents the structural part of the sequence, while the symbol values its content – see Figure 7.5. Thus, approximate matching the structural part of an indexed sequence is based on approximate path alignments.

We choose to use the matching scheme implemented in the SIRIUS XML IR system (see Section 3.4.1 on page 47) to perform the approximate matching of the structural part of the sequences.

We evaluate the structural similarity σ_{str} between a path request p^R (expressing an elementary structural query coding a path with constraints) and the set of root-leaf paths $\{p_i^S\}$ corresponding to an indexed sequence S as follows:

$$\sigma_{str}(p^R,\ \{p_i^S\}) = Max_i \left(\frac{1}{1+\delta_L(p^R, p_i^S)} \right)$$

where $\delta_L$is a dedicated editing distance (see [239]).

Thus, the sequences which lead to the higher structural matching values are the sequences pointing to a candidate path that best matches the structural request p^R. Note that the matching value $\sigma_{str}(p^R, \{p_i^S\})$ belongs to $[0..1]$, ranging from 0 (not acceptable – no matching) to 1 (perfect matching).

The search complexity for such an algorithm – see [239] for justification – is:

$$O(length(p^R) \cdot Max_{j=0}^{|D_j|}(depth(T^{D_j})) \cdot \mid \{p_i^S\} \mid)$$

where:

- $\mid \{p_i^S\} \mid$ is the size of the set $\{p_i^S\}$ – i.e. the number of paths associated with the sequence S, where a sequence may be linked with paths from different documents D_j. The paths start at the root of the document D_j, and lead to the last element of the path request p^R (i.e. the sequence symbol value or the textual terminal),
- $length(p^R)$ stands for the length of the request path p^R and
- $depth(T^{D_j})$ represents the deepest level of the XML tree T associated with one of the documents D_j linked to the sequence S.

This complexity remains acceptable for this application as 99% of the XML documents available on the web have fewer than 8 levels and their average depth $depth(T^D)$ is 4 [149]. Conversely, the user's structural queries are even shallower. As a consequence, the term that will have the most influence on the response time is the number of paths associated with the extracted sequences.

7.7.2 Sequence Approximate Matching

We presume that a user has an imprecise, incomplete, or approximate knowledge of the sequential content extracted from the collection of XML documents. As a direct consequence, a sequential request S^R will represent only a short, probably inaccurate, fragment of an indexed sequence. These types of queries are usually found in query

by humming systems [252] or in biological data processing [146, 94, 83]. Under these conditions, an exact searching scheme for sequence retrieval will fail to respond to the user information needs. Thus, we choose a retrieval scheme that approximately matches a sequential request S^R with any subsequence S_i of the set of indexed sequences.

The sequence similarity is based on a distance – obtained by applying a dynamic programming technique: an editing Levenshtein distance [131] or Dynamic Time Warping[4] (DTW) distance [165]. Both distances have the same computational complexities $O(|S^R| \cdot |S|)$ and allow the approximate matching of two sequences or time series of different lengths.

We want to retrieve all the similar subsequences S_i from the database with a user request S^R, having the distance δ less than a specified threshold ξ – the *P-against-all problem* [83].

The sequential scan complexity for achieving this goal is expressed as:

$$O\left(m \cdot \overline{|S|}^2 \cdot \left|S^R\right|\right)$$

where m is the number of data sequences whose average length is $\overline{|S|}$.

The use of a hybrid method based on a suffix tree as an index structure and a dynamic programming method can reduce the complexity problem and efficiently solves the similar subsequences retrieval task [83, 146, 165]. The performance gain of the method comes from:

- the branch-pruning method that reduces the search space using the threshold value ξ, and
- the suffixes with common prefixes that share the cumulative distance tables during the index traversal. The time complexity of this approach is:

$$O\left(\frac{m \cdot \overline{|S|}^2 \cdot \left|S^R\right|}{R_d \cdot R_p}\right)$$

where $R_d(\geq 1)$ is the reduction factor saved by sharing the cumulative distance tables, and the $R_p(\geq 1)$ is the reduction factor gained from the branch-pruning. In the worst case where there is no common subsequence and the branch-pruning cannot help, both values of R_d and R_p are 1, and therefore the complexity becomes the same as that of the sequential scan [165].

The similarity between a sequential request S^R and an indexed subsequence S_i is given by their normalized distance δ and is computed as follows:

$$\sigma_{seq}\left(S^R,\ S_i\right) = b^{-\delta\left(S^R,\ S_i\right)}$$

where $b > 1$, usually e and $\sigma_{seq}\left(S^R,\ S_i\right) \in [0..1]$.

The value of the b parameter sets the sensitivity of the sequence similarity indicator. It specifies the distribution of the possible distance values δ in the $[0..1]$ interval and boost the best matches. The sequence similarity takes values between 0 – for no correspondence between sequences, and 1 – for a perfect matching.

[4]DTW is a pseudo-distance as it is not respecting the triangular inequality, see [248] for demonstration.

The match of a sequential request S^R with an index sequence S is defined as the best match between the query and any subsequence S_i of S :

$$\sigma_{seq}\left(S^R,\ S\right) = Max_i\left(\sigma_{seq}\left(S^R,\ S_i\right)\right)$$

7.7.3 The Fusion of Structural and Sequential Approximate Matching Scores

The information fusion is defined as the fusion of complementary information provided by different sources with the scope of obtaining an information gain due to the utilization of multiple sources of information vs. a single source.

In our scheme we have chosen the weighted geometric mean to fusion the two levels of approximation: on the structural localization/organization of the sequential data and on its content. The geometric mean is a way to construct an aggregate measure between different indicators that is sensitive to small values. This is appropriate for our purpose of retrieving sequences with highly similar content and being related to highly relevant structures to the user query. The weighted geometric mean is defined as:

$$\Phi\left(\sigma_{seq}\left(S^R,\ S\right),\ \sigma_{str}\left(p^R,\ \{p_i^S\}\right)\right) =^{\alpha_1+\alpha_2} \sqrt{\sigma_{seq}\left(S^R,\ S\right)^{\alpha_2} \cdot \sigma_{str}\left(p^R,\ \{p_i^S\}\right)^{\alpha_1}}$$

where $\frac{\alpha_1}{\alpha_2} = \nu$ is a parameter allowing to specify the relative importance of the indicators to the final score.

We rewrite the above formula in order to transform the logarithmic scale of the ν parameter to a linear scale that is more suited to the user common-sense understanding:

$$\Phi\left(\sigma_{seq}\left(S^R,\ S\right),\ \sigma_{str}\left(p^R,\ \{p_i^S\}\right)\right) =^{1+\nu} \sqrt{\sigma_{seq}\left(S^R,\ S\right) \cdot \sigma_{str}\left(p^R,\ \{p_i^S\}\right)^{\nu}}$$

where $\nu = -log_2(1-\gamma)$ and $\gamma \in [0..1)$.

- The γ parameter is application dependent: it is used for specifying the degree of penalty applied to the final score with respect to the structural matching indicator. A $\gamma = 0$ value will discard the sequence structural factor from the calculus of the overall score, while a $\gamma \to 1$ value will boost its importance at maximum. At $\gamma = \frac{1}{2}$ the fusion process will equally take in consideration the two similarity indicators.

7.8 Extracting and Querying Sequential Data by Examples

We extend the SIRIUS query language introduced in Section 3.5 by adding the following sequence based operators, as well as some temporal operators (not described here but used in an implicit manner) to perform temporal ranking of the retrieved results:

makeSEQ operator – it is equivalent with a contextual p^R request including at most two return statements. One return statement is mandatory and is used to retrieve a sequence symbol and the second one is optional and used to associate

an order key to the extracted symbol. By processing the *makeSEQ* operator we extract and index the sequences into a generalized suffix tree. An example is given in Figure 7.7 and the significance of the required parameters is described together with the sequence extraction process in Section 7.6 .

sameSEQ operator – searches similar subsequences in the generalized suffix tree. Indexing more sequences, will provide more redundancy and better overall performance for the suffix tree. Of course this is done at the expense of larger storing (especially memory) space. As shown in Figure 7.9, the operator receives 3 arguments:

- on the 1^{st} line: the sequential query — here the *$* symbol is used to create a new sequence with the order and the values given as arguments between the parenthesis - i.e. numerical sequence by default
- on the 2^{nd} line we have the structural request with constraints on attributes and on the attributes value,
- on the 3^{rd} line we have the minimum threshold value ξ used for selecting the similar subsequences with respect to both the sequential and structural patterns.

The most important modification besides the addition of the sequential operators consists in the fact that a path request p^R may now include a "return" statement that retrieves either an attribute value (using $return\ @attname$) or an XML element content using the $return\ this$ operator.

The $return\ this$ operator requires auxiliary indexing structures to store the XML nodes content for fast retrieval; – otherwise we will have to parse the XML documents content each time we had to perform this kind of operation. As the XML documents from the indexed databases may be large (as this is the case for the MidiXML files), the second option is obviously inefficient. We implemented the first solution by using a region repository file in which, for each unique $(documentId,\ nodeId)$ pair, we have indexed their textual content and associate a reference to their $parentId$ node. The $parentId$ value is necessary to get access to their ancestors nodes in the XML path. The retrieved content supported by this additional structure may be further parsed to extract the sequence symbols or used on an "as is" basis to build a sequence symbol by itself.

For instance in Figure 7.10 we have a context and content query with an attribute value statement (lines 1–6). In this case we use an entry of the inverted list selected by the request term $'country'$ (line 5) to process the request – i.e. we note here that the requests paths that do not contain content conditions are always terminated with the path delimiter $'/'$ (see line 12) . In Figure 7.10, lines 8-14, we use the $return\ this$ operator to retrieve all the contents of the $'textevents'$ nodes, where the structural constraint is interpreted vaguely.

We note here that a structural extraction pattern can specify a maximum of two return values: one is the value to be returned by the $return$ operator and one is the value retrieved by the $orderBy$ operator – this configuration extracts pairs of *(values, orderKeys)* that are used later on by the sorting procedure.

More complex requests can be formulated by using boolean or fuzzy merging operators (AND, OR, etc.) and by integrating textual, structural or temporal operators with

```
(sameSEQ ($ 62 59 55 66 66 67 62 50 67 62 69 69 71 67 )
       [/jazz/Event/NoteOn(= channel 10)/] % str. constraint
       0.5 % threshold for the similarity value
)
```

Figure 7.9: $sameSEQ$ sequence similarity search operator.

```
(OR [/midifile
    /track(and (RETURN @number NUMBER) (== number 0))
    /event
    /textevent
    /country ]
)

(OR [/midifile
    /track(== number 0)
    /event
    /textevent(RETURN THIS STRING) %retrieve node content
    / % no textual terminal
    ]
)
```

Figure 7.10: $RETURN$ operator.

different granularities. For instace, in Figure 7.11 we show a complex query which processes two sequence extractions patterns including textual and temporal constraints and then performs approximate subsequence similarity search on the extracted sequences.

7.9 Evaluation

We present some preliminary experimental results dedicated to midi files retrieval in a heterogeneous XML MIDI library.

7.9.1 Prototype

We have implemented the approximate sequential matching operators and the fusion method based on the presented index model in the SIRIUS XML information retrieval engine [148, 177, 179]. The prototype is entirely developed in Java and uses the *dynamic time warping* [165] algorithm to compute sequence similarity scores. The implementation of the similarity search for sequence retrieval follows the algorithms introduced in [165].

SIRIUS provides approximate operators for managing textual/sequential/time series data in a heterogeneous XML environment. The graphical user interface of the SIRIUS XML IR engine showing a subsequence match is shown in Figure 7.12.

```
(AND
  concert %textual keyword
  (sameSEQ %sequence similarity search
    ($ 83 76 81 ) %sample sequence
    (makeSEQ
      (OR [/midifile/track(== number 1)/event
         /noteon(AND (== channel 1)(RETURN @note NUMBER))/])
      1.0 %threshold for the extracted symbols relevance
      DOCUMENT %sequence structural type
    ) %end_makeSEQ
    (makeSEQ
      (AND russia %text keyword
        (NEAR '2000-01-01' MONTH) %temporal constraints
        (OR [/midifile/track(== number 2)/event
            /noteon(AND (== channel 2) (RETURN @note NUMBER) )/])
      ) %end_AND
      1.0 %threshold for the extracted symbols relevance
      DOCUMENT %sequence structural type
   ) %end_makeSEQ
   0.5 %similarity threshold for the_sameSEQ operator
  ) %end_sameSEQ
) %end_AND
```

Figure 7.11: Complex request.

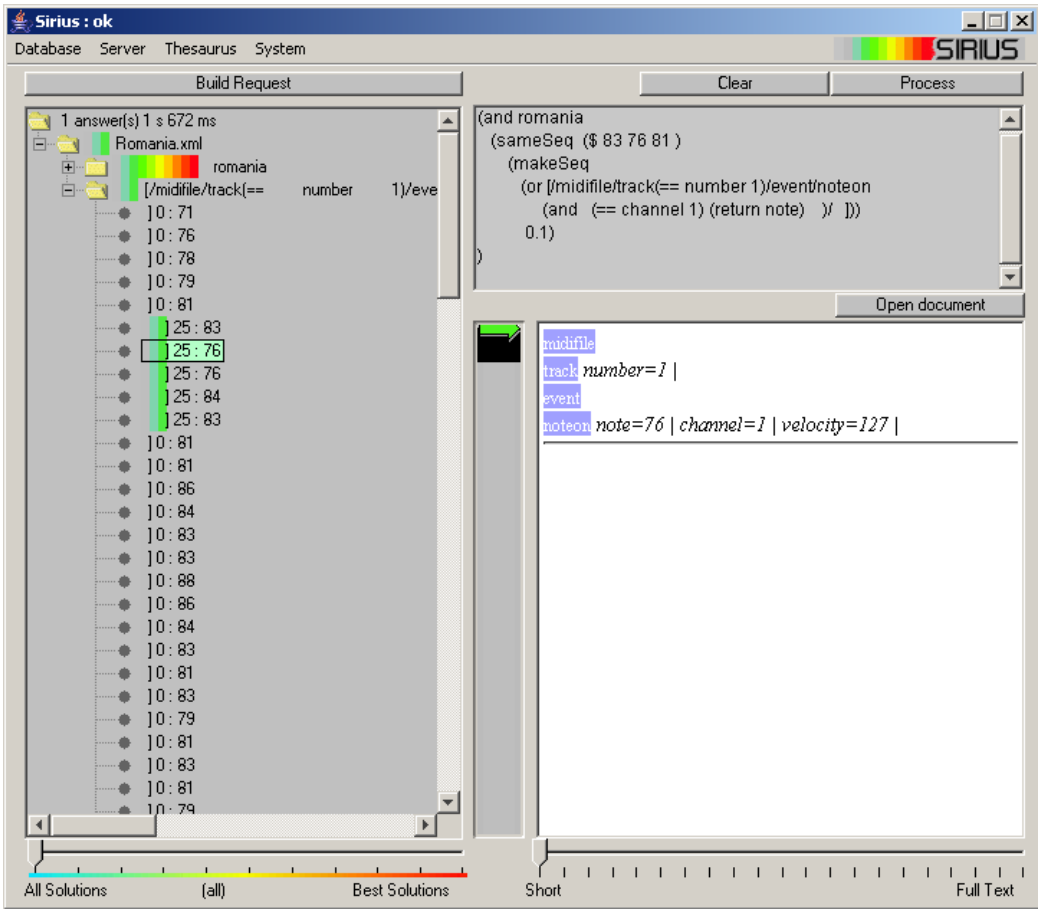


Figure 7.12: Subsequence match in SIRIUS GUI.

7.9.2 Experimental Dataset

The experimental dataset is formed by a MIDI file collection (32 Disney themes) downloaded from the public domain[5]. The files are transformed in the XML format according to the Standard MIDI File DTD [4] version 0.9.

In a MIDI file the sequences of notes are organized in tracks (having a maximum of 16 channels) and events (see Figure 7.2). The "human readable" information – i.e. the meta-events – such as lyrics, copyrights, tempo indications, time and key signatures, markers, etc. are attached to the first track of the file – *track 0* (see the Standard MIDI Files 1.0 specification (SMF) published by the MIDI Manufacturer's Association[6] for details).

To simulate the heterogeneity of the collection and to validate the approximate structural localization of the sequential data, we generate and append a meta-structure for each standard MidiXML file. The meta-structure for each MidiXML file is generated by randomly selecting structural contexts from a structural pattern that encode the structure of a website providing MIDI files on the public domain – see Figure 7.13.

[5] `http://themes.mididb.com/anthems/`
[6] MIDI Manufacturers Association (MMA) `http://www.midi.org`

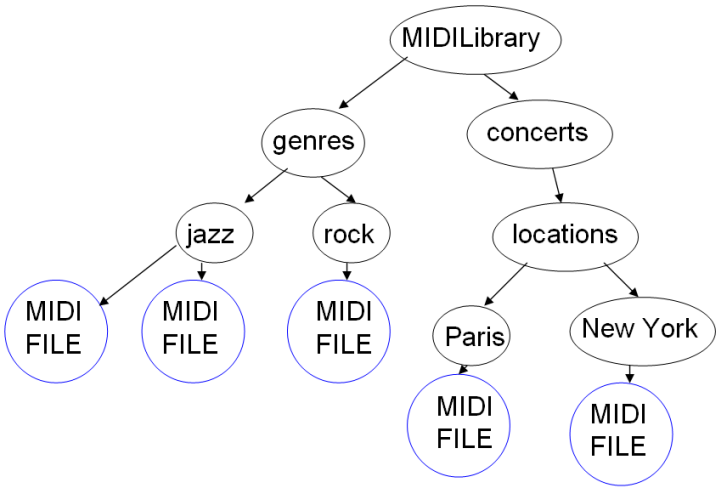


Figure 7.13: Basic scheme used to randomly generate meta-structures for the MidiXML test collection.

7.9.3 Early Evaluations

We present some early experiments of the XML data indexing and sequence extraction algorithms on datasets with sizes ranging from 1MB to 15MB. The system used for experimentation has a 2.4 GHz processor and 512 RAM. The XML data indexing time represents the elapsed time for the creation of the inverted lists without taking into consideration the sequential data. The sequence extraction time stands for the time spent in the process of approximate matching the XML contexts and the time spent to index the extracted sequences.

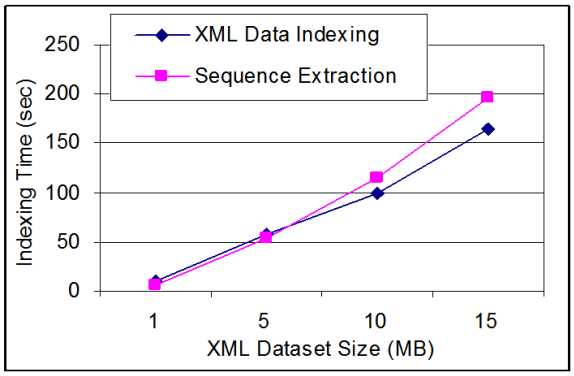


Figure 7.14: XML data indexing time / Sequence extraction and indexing time as the size of the index dataset.

We can observe in Figure 7.14 quasi linear indexing and extraction times with the size of the indexed datasets (i.e. the total length of the indexed sequences), which is quite encouraging. The average response time for 90 randomly generated request (30 with structural constraints only, 30 with sequential constraints only, and 30 requests with constraints on both the structure and the sequence content) are shown in Figure

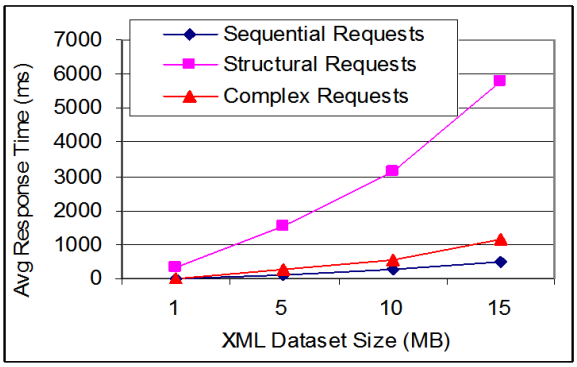


Figure 7.15: Average response time for structural, sequential, and complex requests.

7.15. The structural requests seem to have a polynomial behavior. The sequential queries are less sensitive to the size of the indexed dataset than the structural ones due to the organization of the GST index structure. A GST scales well to the dataset size as it uses the common prefixes of the index sequences to reduce the research space (see Section 7.7.2).

This fact raises interesting perspectives for the optimization of the overall response time of the complex requests involving both the structure and the sequential content of the XML documents. We use the sequential queries as a first filter when answering to complex requests with mixed constraints on both the content and structure. This improves significantly the overall response time as shown in Figure 7.15. Finally, the complexity of the alignment algorithms is maintained as low as possible to preserve the capability of indexing large datasets.

As for the quality of the retrieved results, we could not evaluate it completely, as the current evaluation framework for the retrieval of multimedia structured document fragments [232, 243] focuses on images and is not yet adapted to evaluate sequential data. A general opinion [61] and also our belief is that using similarity operators adapted to the document content types and to the XML structure in the retrieval process should improve the precision of the results.

7.10 Conclusions

In this chapter we have described approximate searching mechanisms to extract, index and query sequential data from semi-structured information embedded into XML datasets. Such mechanisms are based on the alignment of root-node paths that are sub-structures of XML trees. The proposed mechanisms allow to merge structured data ($< attribute, value >$ pairs) or structural organization of documents ($< MIDIFile >< Track >< Event >< NoteOn >$ ...) with unstructured data such as textual (free text) or sequential/time series data.

At the current author knowledge, there is no existing integrated method for approximate querying specific sequential data in a heterogeneous semi-structured environment. Even if each part of the problem was extensively studied and beneficiated of strong research efforts of well established scientific communities, the fusion of the methods developed in this two research areas (sequential similarity search and XML

information retrieval) was not yet deeply considered. The proposed scheme was designed to cover this gap and to highlight extended and useful querying capabilities for the user.

7.10.1 Main Contributions

Our main contributions to the field of XML multimedia information retrieval concerns the introduction of a set of specialized operators designed for the extraction, indexing and retrieval of sequential/time series data embedded in heterogeneous XML documents. More precisely:

- We have investigated sequential/time series data and its organization within the XML documents;
- We have described an approximate sequence extraction scheme guided by structural patterns for extracting the sequence symbols and contextual information from heterogeneous XML documents;
- We have proposed a hybrid index model for the indexing of textual, structural and sequence/time series data;
- We have defined a model for retrieving similar sequences extracted from XML documents. The matching mechanism allows to aggregate two levels of approximation: on the structural localization/organization of the sequential data and on its unstructured content;
- We have developed a complete prototype to evaluate the benefits and the drawbacks of the proposed approach. Our main experimental contribution so far, shows that the fusion of structural and sequential search criteria drastically improves the global retrieval performances of the similarity search mechanisms when exploiting heterogeneous XML databases.

7.10.2 Future Work

The work presented in this chapter can be extended in several directions:

- A first enhancement for the set of the sequential operators implemented in the system is to make them aware of the temporal aspects of the data;
- Second, we want to experiment with a flexible organization relative to the sequence's symbols order relation, and to measure the impact (both on the efficiency and the effectiveness of the system) for such relaxation;
- Another line of interest is the design of a weighted model for sequential data in which each symbol of the sequence can be associated with a weight indicating its contribution/relevance to the current sequence;
- Both, the disk resident organization of the index structures and the parallelization of the research algorithms will be a straight forward research direction in order to validate our approach on larger amount of data;

- Finally, designing an evaluation framework for the retrieval of sequential data embedded in heterogeneous XML documents represents a major research challenge.

Part IV

XML IR on Specialized Hardware

Chapter 8

ReMIX – Reconfigurable Memory for Indexing Mass of Data

In this chapter we make a short overview of the main characteristics of the ReMIX[1] project. ReMIX stands for *Reconfigurable Memory for Indexing Mass of Data*. The project aimed to design an original memory architecture for both storing very large indexed data structures, and allowing fast information retrieval. It follows the concept *on-the-fly filtering* of huge amounts of data by combining two technologies: FLASH memories and FPGA components.

Contents

[1] Reconfigurable Memory for Indexing Mass of Data (ReMIX) `http://www.irisa.fr/remix`

8.1 Introduction

Indexing is a well-known technique that accelerates searches within large volumes of data, such as the ones needed by applications related to genomics, to content-based image or text retrieval. An index is a data structure designed to organize the data by mapping a key value onto one or more records containing the key value, thus providing a mechanism to efficiently locate the storage location of records.

General-purpose computer systems usually feature a hierarchy of memory levels, each level with its own cost and performance characteristics. At the lowest level, CPU registers and caches are built with the fastest but most expensive memory. For internal main memory, dynamic random access memory (DRAM) is typical. At a higher level, inexpensive but slower magnetic disks are used for external mass storage, and even slower but larger-capacity devices such as tapes and optical disks are used for archival storage. Therefore, not all memory references require the same access time. Large address spaces span multiple levels of the memory hierarchy, and accessing the data in the lowest levels of memory is orders of magnitude faster than accessing the data at the higher level [238]. For example, loading a register takes on the order of a nanosecond ($^{-9}$ seconds), and accessing internal memory takes tens of nanoseconds, but the latency of accessing data from a disk is several milliseconds ($^{-3}$ seconds), which is about one million times slower! In applications that process massive amounts of data, the input/output communication (or simply I/O) between the memory levels is often the bottleneck.

For massive data sets, the index structures may not fit completely inside the computer's internal memory and are generally stored on magnetic disks. In this case, the design of indexes is fully disk-oriented, since minimizing disk I/Os is the key point to reduce response times. Therefore such indexes are concerned with low level notions such as pages, fill-factors, tracks, cylinders, etc. In addition, such disk-oriented design indirectly impacts the search algorithms that navigate within the index since they have to favor sequential patterns both for processing data in individual disk pages and for fetching disk pages, avoiding as much as possible any random access to data.

To solve the scalability problem for applications managing massive data sets several solutions exist. Among them we may cite the use of dedicated high-end multiprocessor servers or distributed computing using clusters of regular computers. The last approach is a more cost-effective solution. Some of the benefits of the network of workstations model are as follow [18]. First, network of workstations has become extraordinarily powerful and offer a better price-performance than parallel computers. Second, most networks of workstations have a huge amount of memory and very fast processors, both of which sit idle most of the time. Third, switched networks allow bandwidth to scale with the number of processors and low overhead communications protocols have made it possible to do very fast communications among workstations. Among the drawbacks we may cite significant increases in the costs associated with the maintenance and administrations tasks such as the required physical space, the required human resources, the repair cost, the power consumption and the cooling issues. A successful implementation of this model is the Google cluster architecture based on a farm of several thousands commodity-class computers using the Linux operating system [27].

Nevertheless, any of the previous approaches bring solutions for the latency limitations of traditional on-disk indexes. One approach is to ignore the disk memory level

completely and to consider only application cases where data management relies exclusively on the main memory – i.e. Main Memory Database Systems MMDBS [73, 39]. A different approach is the design of "intelligent disks". In this view several projects like RDISK [84], *Smart Disk* [147], *Active Disk* [189], IDISK [111] or [242] used reconfigurable hardware closely connected to a hard disk in order to overcome the I/O bus bottleneck of conventional systems. [242] applies specialized hardware for processing unstructured data. Data are sniffed on the IDE bus out of one disk drive so that they can be filtered, compressed or encrypted on-the-fly into a FPGA component.

Inspired by the later approaches, the ReMIX project addresses the data access problem. The idea is to propose a hardware mechanism allowing fast random accesses to Gbytes of data. In the ReMIX architecture, hard drives are replaced by FLASH memories whose access times are 2 or 3 orders of magnitude faster. In the same way, data bandwidth is increased by accessing simultaneously a large number of FLASH memories. Data are processed on-the-fly by reconfigurable hardware directly connected to the memory.

8.2 ReMIX Project Objectives

The ReMIX project (*Mémoire reconfigurable pour l'indexation des masses de données*) started in 2003 granted by the French initiative *ACI Masses de Données*[2] (ACI MD). The project involved four research groups:

- *Symbiose*, Bioinformatic Group, IRISA, Rennes
- *R2D2*, Computer Architecture Group, IRISA, Rennes
- *TexMex*, Multimedia Document Group, IRISA, Rennes
- *Aprim*, HyperText and Database Group, VALORIA, Vannes

and had two main targets:

- first, to identify the necessary features for a dedicated hardware system designed to manage the querying of a large amount of data. The project tackled content-based search for different applications domains [74] with focus on the genomics [129, 128, 155, 167, 182], images [17, 29] and semi-structured text processing fields [148, 175]. The aim was to propose a conceptual framework able to manage as efficiently as possible content-based retrieval applications using indexes.
- second, to design, build and test a specilalized hardware/software framework to speed up this model.

8.3 ReMIX Idea

The ReMIX project proposes the design of a dedicated and very large index memory (512 Gbytes) which is big enough to entirely store huge indexes. The use of an almost

[2]Action Incitative Masse de Données `http://www.enseignementsup-recherche.gouv.fr/appel/2003/acimd.htm`

unlimited memory raises completely new issues when designing indexes. Here, within this scheme, direct access to data, massive parallel processing, huge data redundancy, pre-computed structures, etc., can be advantageously promoted to speed-up the search.

8.3.1 Reconfigurable Resources

The index memory includes reconfigurable hardware resources to tailor at a hardware level the memory management to best support the specific properties of each indexing scheme. It also offers the opportunity to implement at a hardware level application dependent filtering algorithms having interesting potential parallelism for processing data directly from the output of the index memory.

8.3.2 FLASH Technology

Characteristics of the index to manage features both a large volume and a relative stability. Indexing huge amount of data (several hundred Gbytes) takes time and we make the assumption that is not performed continuously. Consequently, the storage device only need to support an acceptable number of write operations, while allowing quasi unlimited read accesses. The FLASH memory technology fit these requirements. In addition, the memory capacity is high (more than 1 Gbytes per chip) and the access time is low (20 microseconds, 10^{-6} seconds) compared to magnetic disks (several miliseconds, 10^{-3} seconds).

8.4 ReMIX Architecture

8.4.1 ReMIX System

The ReMIX system (see Figure 8.1) is composed of a small cluster of five PCs: 4 slaves and 1 host. The host (or master) acts as a front-end machine and the four others are the processing nodes, each one housing two PCI boards containing a FPGA and 64 Gbytes of FLASH memory each. The whole system holds 512 Gbytes of FLASH memory. The nodes are interconnected through an Ethernet switchbox.

8.4.2 RMEM Board

The RMEM (ReMIX MEMory) board (see Figure 8.2) is the core component of the ReMIX system. It is designed to associate on the same support both a large memory and reconfigurable components in order to allow on-the-fly filtering of data as soon as it is read from the memory. An RMEM board is composed of:

- 64 GBytes of FLASH memory
- 1 Xilinx Virtex 2 Pro - XC 2VP30

Depending of the type of query, an adequate hardware filter is first downloaded to the FPGA component before scanning the banks. The filtering occurs locally and results are sent back to the front-end computer for further post-processing.

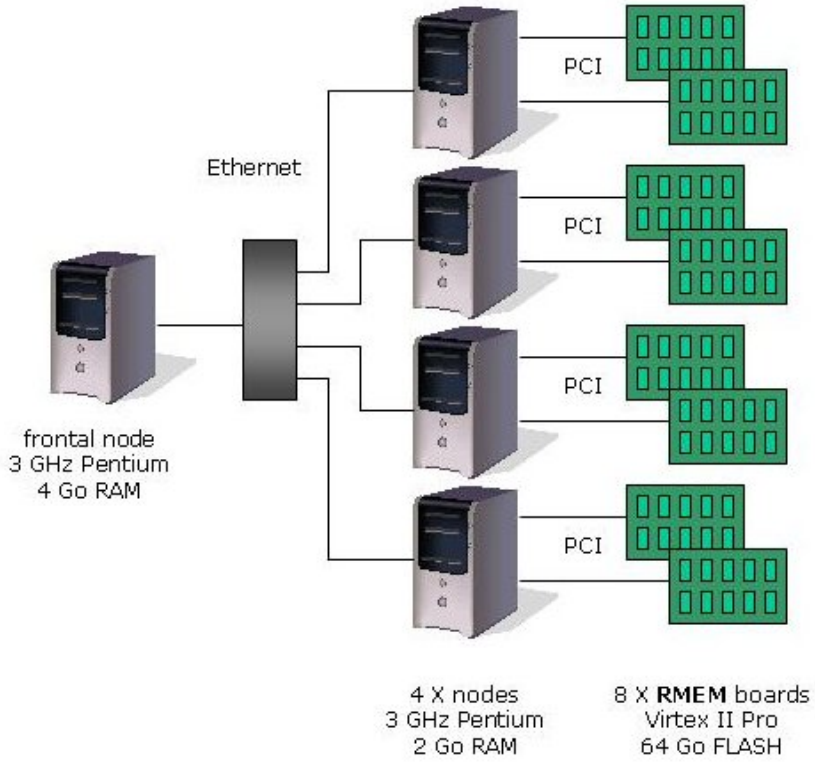


Figure 8.1: ReMIX System Architecture.

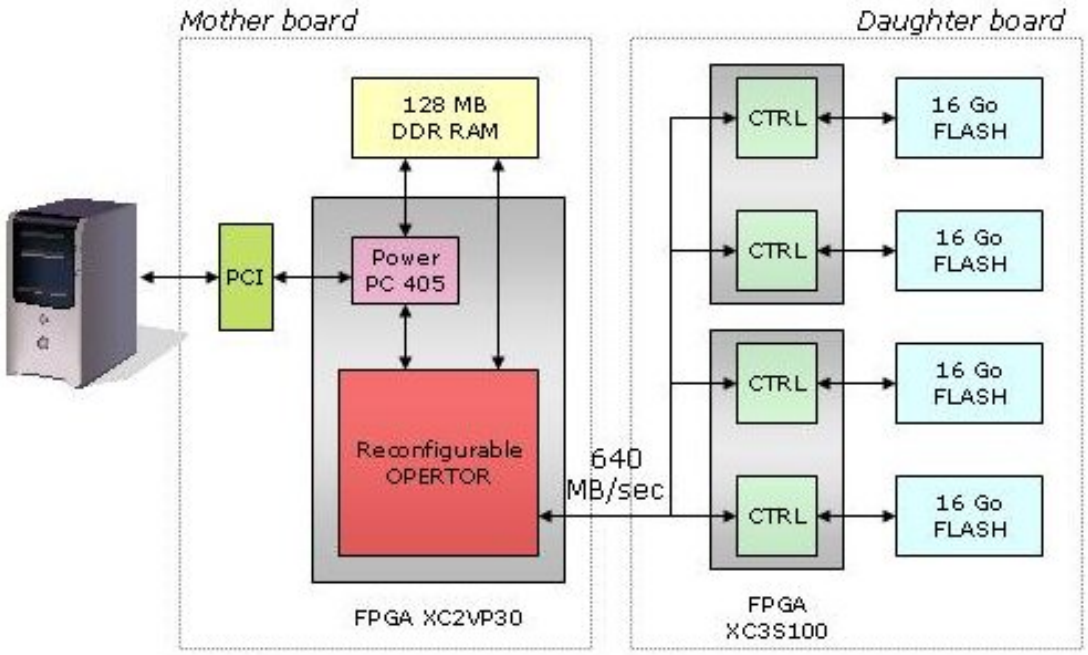


Figure 8.2: ReMIX Memory Board.

8.4.3 ReMIX Memory Specificity

It is important to point out that the ReMIX memory architecture is not only a simple memory extension to substantially increase the memory capacity of a standard computer. The reasons are the following:

- The reconfigurable index memory is not a simple storage device. It is enhanced with additional reconfigurable hardware resources for tailoring its use according to the index characteristics and to the data it manipulates.
- The reconfigurable index memory does not fit in the addressing space of the processor. It is indirectly accessed by specific queries submitted by the processor in order to execute crucial and costly indexing subroutines.
- The reconfigurable index memory does not hold any cache hierarchy, and therefore memory accesses do not have to worry about the data locality. Memory read operations have a unique cost, whatever the memory address, and whatever the previous memory accesses.
- Due to the FLASH technology, writing operation are limited. It only aims to periodicaly store huge volume of data while allowing quasi unlimited read access.

8.5 Programming the ReMIX cluster

8.5.1 Framework

The ReMIX cluster is programmed through a framework. The goal is to simplify the development of indexing algorithms and to ease the deployment of the application on the hardware solution. It has been designed with two major requirements: simplicity and reutilisability [74].

Simplicity comes from a simple master/slave execution model. Furthermore, the data distribution is explicit and static. Execution, communication and synchronization are fully handled by the programming environment.

Reutilisability both takes place at a software and hardware level. The programming environment is not linked to any specific application and doesn't depend of a dedicated platform. Programs can be executed either on a single PC, a parallel machine (cluster) or a dedicated platform (ReMIX). This is particularly useful during the debugging phase.

8.5.2 Operator Synthesis

Most of the time, the critical part of a reconfigurable architecture concerns its programming facility. The remix FLASH memory is tightly connected to FPGA components housing application dependant hardware filters.

Specifying the filter functionality can be seen as a *data flow* description: data come as a regular flow from the ReMIX memory, and are processed on-the-fly. Only data meeting some requirements are pushed to the RMEM board output. Unfortunately, the task of specifying an adequate hardware filter for the FPGA processors is quite complex and currently exclusively reserved to VHDL specialists.

8.5.3 ReMIX Query Processing Model

In this section we take a general look at the query processing model and present the role and the interaction between the different components of the ReMIX architecture.

In the ReMIX architecture, all the applications follow the client/server of requests paradigm (see Figure 8.3). In this view we accept that the index and the requests can be distributed/duplicated on the nodes of the ReMIX cluster. Only the frontal node (i.e. the host) detains the complete knowledge of the index and requests distributions on the nodes. The distribution of the index is recorded into a `MemoryMap` structure. Each `FullQuery` `submitted` to the the host is processed by a `QueryServer` that splits the `FullQuery` in `SubQueries`. The `SubQueries` are `dispatched` and `processed` independently on the nodes. Generally the `processing` phase consist in fetching from the `LocalMemory` – i.e. `getAt()` – the indexed information associated with an index key entry and computing a similarity distance between the request and all the objects from the retrieved list. The `SubQueries` may retrieve none, a single or a list of partial `SubResults`. A retrieved result may influence (i.e. stop the process, or change the threshold minimum value for the top-k retrieved results for instance) the current processing of the request by using a `feedback` mechanism. The partial lists of `SubResults` are collected and `merged` by the `SubResultServer`. The `SubResultServer` is running on the host and delivers the `FullResult` to the user. The whole system operates in parallel and is able to pipeline a flow of queries.

8.5.4 ReMIX API

For this generic client/server of requests application model, the programmer has to specify:

- how the index is constructed and distributed on the ReMIX nodes,
- the content of the complex request and how the request can be split in elementary requests,
- the content of an elementary request and how this is processed on the nodes,
- the content and the processing of a partial result list,
- the feedback of a partial result to a request,
- how the partial result lists are merged,
- the content and the processing of the complete result.

Programming with the ReMIX framework API (see Figure 8.4) consists in implementing seven Java abstract classes and interfaces – namely `FullQuery`, `SubQuery`, `SubResult`, `FullResult`, `GlobalMemory`, `MemoryEntry` and `MemoryValue` – in order to adapt the ReMIX runtime to the targeted application domain and to control the query processing.

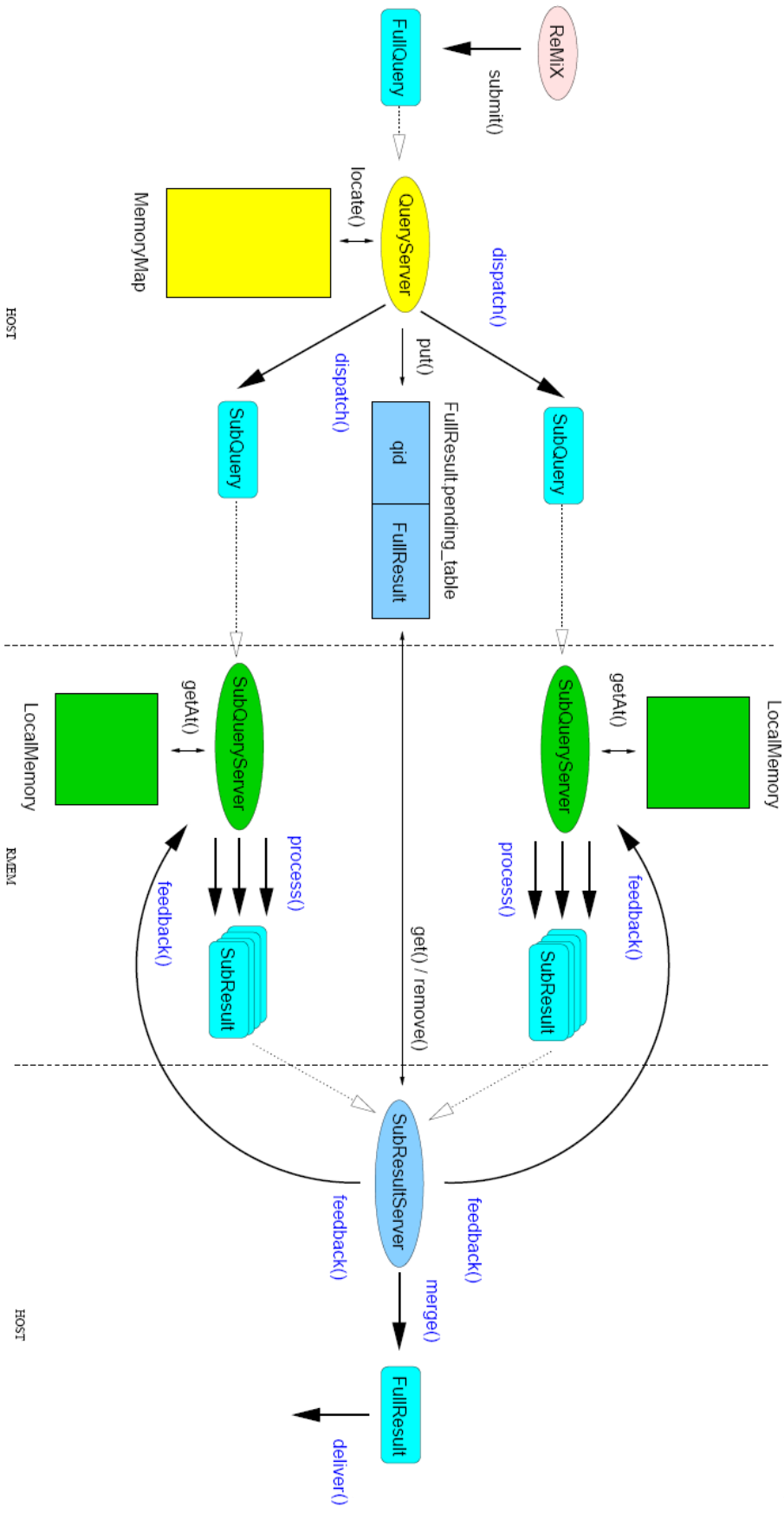


Figure 8.3: ReMIX generic query processing model [182].

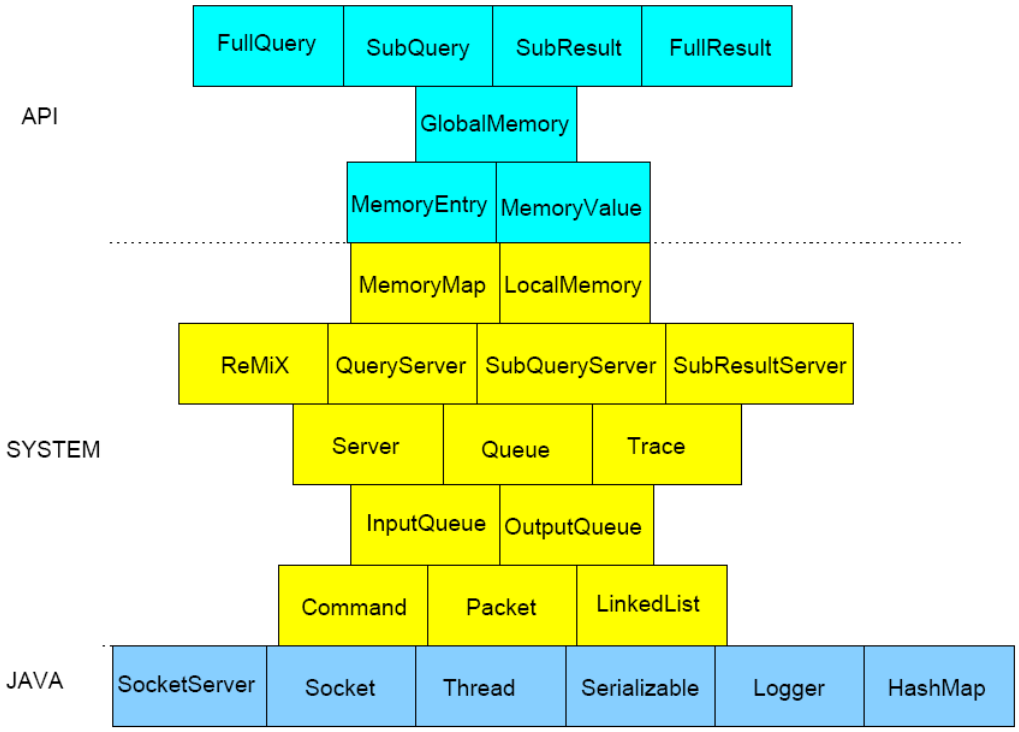


Figure 8.4: The ReMIX programming framework API [182].

8.6 Conclusion

In this chapter we have introduced the ReMIX project. We have presented the motivation, the main idea, the technology used, the system architecture, and the programming paradigm. An implementation of an XML retrieval application using the ReMIX architecture is described in the next chapter.

Chapter 9

Approximate Search of Semi-Structured Documents Using Dedicated FLASH Memory and FPGA Components

In this chapter we adapt a subset of our XML IR operators to the application model developed in the context of the ReMIX project. The implementation described allows on-the-fly approximate structural filtering (and relevance ranking) as support for searching relevant information in heterogeneous semi-structured databases. A preliminary version of the approach proposed in this chapter was presented in [175].

Contents

9.1 Introduction

Larger and ever increasing volumes of data in XML format are accessible on the Web through the use of standard search engines. The users drowned in the information

mass stressed for the development of new enhanced research capabilities allowing for more focused access to the sought information. XML information retrieval aims to consider the additional information and rich annotations provided by the structure of XML documents and their element names to improve the precision of the retrieved results.

XML IR applications require both adapted storage and access spaces for managing and searching important volumes of XML data and powerful computing capabilities to perform approximate structural matches. In this chapter, we propose to investigate the use of a specialized hardware architecture to perform the expensive computational steps of the approximate structure matching algorithms introduced in [148]. We used for this purpose the hardware memory developed by the ReMIX project.

9.2 General Notes on ReMIX Programming Philosophy

In the previous chapter, we have introduced the ReMIX project, its main motivations and requirements, the technology used and the proposed programming paradigm.

We summarize here some of the characteristics of the ReMIX project that will have an essential impact in the design of our semi-structured application:

- High (e.g. 512 GB) limit for memory usage – i.e. suited for large redundant index structures,
- Fast random access to the index key entries,
- The flow of data is processed and filtered. No additional information may be fetched at retrieval time unless it was specifically associated with the data entry in the index.
- Fast data filtering and computational capabilities performed in parallel by the components of the FPGA processors and for each FPGA processor associated with one of the cluster nodes.
- Joins and ranking of partial results lists can be performed only on the front end.

Summing up, the ReMIX project considers the database as a data stream and performs fast hardware filtering to select regions of interest for further processing.

9.3 Approximate Structural Filtering for XML IR using the ReMIX Architecture

Further, we present an application that allows approximate structural filtering and relevance ranking for semi-structured documents adapted to the ReMIX application framework.

The application provides structure and content ranking scores at element level as support for XML information retrieval in heterogeneous collections of XML documents.

The content score use a variant of *tf*idf* [195] ranking scheme and basic terms statistics at document, element and collection level to estimate the relevance of an XML element, while the structural score computes an editing distance [131] between

the path specified in the request and the path of the retrieved XML element. The structural and content scores are further merged by using a weighted linear approach. The output of the application consist in a list of XML elements ranked according to their relevance to the user request, either in a list of documents ranked according to their relevance to the user request and containing pointers and scores to the relevant elements inside each document. The system can answer either content only queries or content and structure queries.

The optimization occurs for the content only queries at the data access level – i.e. fast random access at the index entries and efficient retrieval of the associated information from the FLASH memory. For the content and structure queries, the optimization includes besides the efficient access and retrieval of the index data, a filtering operation that computes an approximate structural match between the structural pattern specified in the request and the structure of the retrieved information.This is processed in parallel by the components of the FPGA processors as soon as the data are fetched from the FLASH memory. Only the data having a structural similarity above a given threshold is allowed to pass the filter condition and is sent to the host for further processing.

9.4 Specifying the Application Characteristics

ReMIX programming framework requires the specification of the following characteristics for a given application:

Index Construction: The model used is the inverted list model. For each entry in the posting list we store the document id, the XML path and the term position in the XML element.

Index Partitioning: We use a global index partitioning equally balanced between the nodes of the cluster. For each index entry key, the posting list is distributed in a round-robin strategy on all the nodes of the cluster.

FullQueries: A complex request may be composed of multiple elementary requests and merging operators at both element and document level – i.e. terms occurring in the same XML element or results that must occur in the same document.

SubQueries: An elementary request consists of either a simple textual terminal, either of an (eventually incomplete) path with conditions on attributes and attributes values ending with a textual terminal. The elementary requests are considered as independent and therefore they are suitable candidates to be processed in parallel.

SubQueries Processing: In the first phase, the set of matches is selected using the terminal node of the request. If this is equal to an index entry key we get access to the list of postings. Further, each posting is analyzed and the structural distance between the request path and the path pointed by the post entry is computed. If the structural score is above a given threshold the current posting entry is send to the host for further processing, if not it is discarded.

SubResult: A subresult consist of a list of unsorted elements associated with their approximate structural matching score. A result specifies its position within the

XML element (a number), its position within the XML tree (using an XML path), and its position within the indexed collection (document id).

Feedback Mechanism: The feedback of a partial result to a request may optionally change the minimum threshold value used to select the valid structural matches in the case of top-k results processing.

Merging Partial SubResults Lists: The partial result lists are merged by using the union of the their sets values. In case of a redundant distributed index, the same result may be fetched from different nodes or even several times from the same node. The duplicate values are eliminated in this step.

FullResult Processing: Finally the list of results is sorted either by the relevance of the XML elements, either by the global relevance of the documents containing relevant elements. The first top-k results are returned to the user.

9.5 Indexing

```
<document>
   <title> Slant </title>
   <author> G.Bear </author>
   <text>
      <chapter num="1" >
         Omphalos dominates ...
      </chapter>
      <chapter num="2" >
         Alice Grale believes ...
      </chapter>
   </text>
</document>
```

Figure 9.1: A simple XML document.

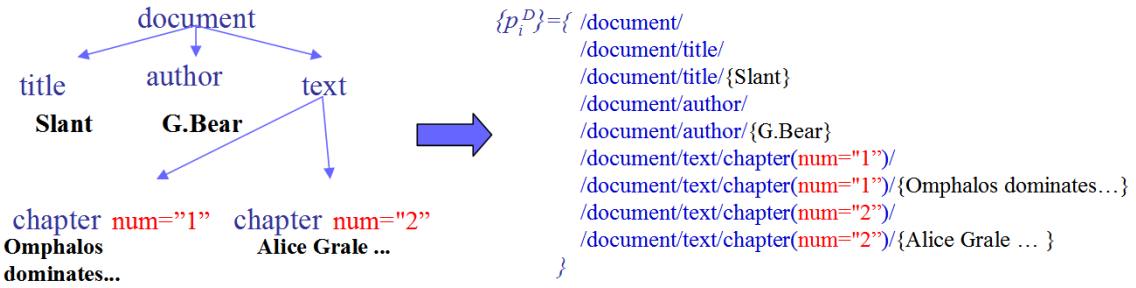


Figure 9.2: XML document tree and set of root-to-leaf paths representation for the XML excerpt from Figure 9.1.

We use an equivalent representation of the XML documents (see Figure 9.1) as for the one previously described in Chapter 3.

An XML document D can be represented by the tree T^D of its XML elements. Each XML tree T^D can be seen as the set of XML paths $\{p_i^D\}$ starting from the root of the tree to the leafs (see Figure 9.2).

We use a simplified indexing model based on inverted lists. We attach to each index term its `DocID`, its XML `path` in the document tree , and its word position `pos` within the XML element.

At implementation level we adapt this model for the ReMIX framework – i.e. we index all the data in clear and use no encoding for element names or XML paths. Therefore no secondary access is necessary and more flexibility can be allowed when computing the path editing distances, both at path level – using the editing distance – and at element name level – for instance to correct a tag name with typing errors. An example of a reference locator associated with the 'alice' entry key from the XML document from Figure 9.1 is shown below.

$$'alice' \rightarrow < "doc.xml" >< /document/text/chapter(num = "2")/ >< 1 >$$

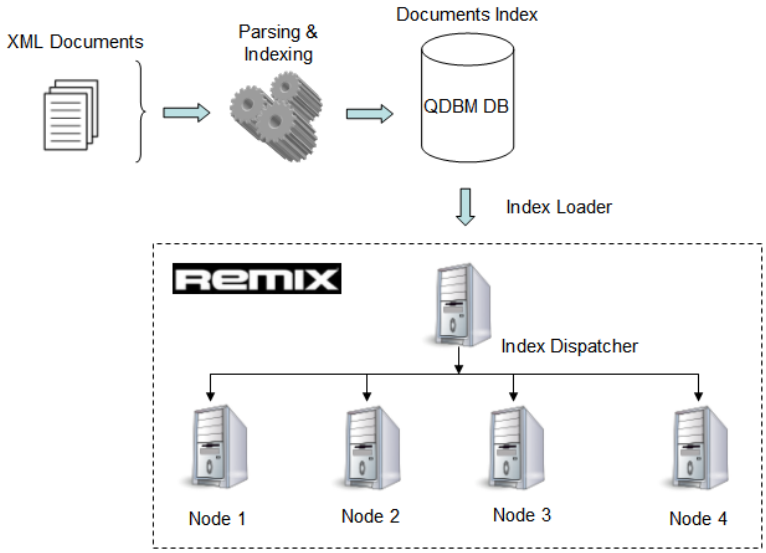


Figure 9.3: Index construction and loading on the ReMIX architecture.

Since ReMIX is not designed to perform the indexing, we have to provide the set of inverted list by a custom external program. The indexing process is shown in Figure 9.3. The XML documents are parsed with a SAX parser and inverted lists structures that include path locations are created. This process is completely independent of the ReMIX architecture and requires the creation of a dedicated index. We use the CURIA distributed hashtables of the QDBM[1] database to store the inverted lists. After the index creation phase, we load and dispatch the information on the ReMIX cluster nodes.

[1]QDBM: Quick Database Manager `http://qdbm.sourceforge.net/`

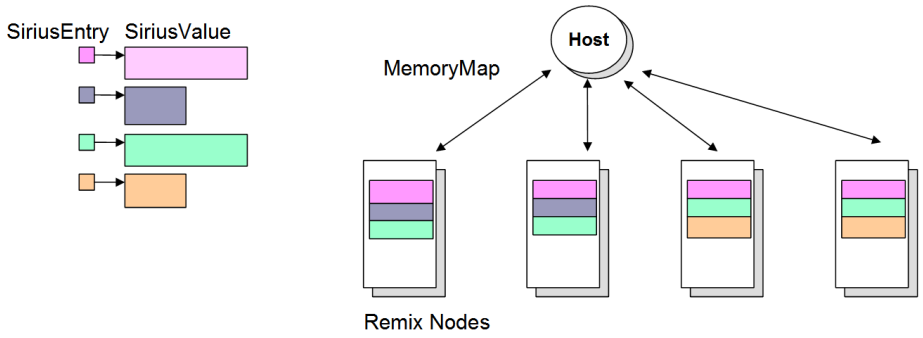


Figure 9.4: Index partitioning.

Index Partitioning

We take a global index partitioning approach as the complete index is available in its totality before loading it on the ReMIX nodes – see Figure 9.3. One possible alternative to partition the documents index is to equally load all the nodes – see Figure 9.4. This ensure that we have the same amount of workload at query processing phase on each processor. The indexed data is distributed by using a round-robin strategy on the inverted lists posting lists. The nodes may contain a different number of entries for an entry list, but they are globally balanced. For very short posting lists some nodes may not feature the whole entry key list. This is recorded in the `MemoryMap` index and further used in the query distribution phase.

9.6 Searching

A complex request is made of multiple elementary requests and merging operators at both element and document level – i.e. terms occurring in the same XML element or path, or results that must occur in the same document. An elementary request consists of an (eventually incomplete) path with conditions on attributes and attributes values ending with a textual terminal – see Figure 9.5. The elementary requests are considered as independent and therefore they are suitable candidates to be processed in parallel.

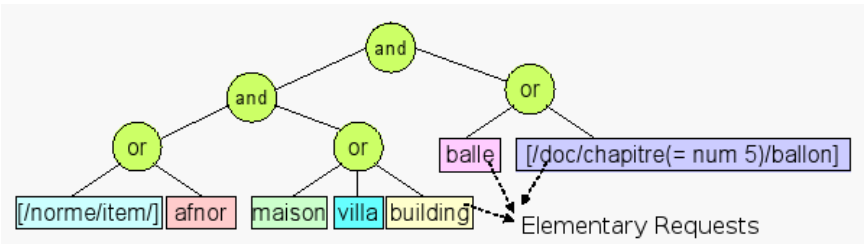


Figure 9.5: Complex request build of elementary requests and Boolean operators.

The complex request is parsed and the elementary requests are sent to the nodes of the ReMIX cluster. The requests dispatching process uses a `MemoryMap` structure that records all the mappings between the entry keys and the nodes of the cluster having

a non null posting list for that entry. However, the possibility that all the elementary requests to be sent to all the nodes is the most likely as the inverted lists postings are well distributed among the different nodes in the load-balancing process. This is a consequence of the fact that the loading process aims to have a relatively equally amount of data charged on each node of the cluster.

The search process for an elementary request consists of two distinct phases:

1. Access and filter the preliminary results on the ReMIX nodes.
2. Fine grained computation and relevance ranking on the server side after merging the preliminary results. Attribute conditions may be eventually computed at this step.

In the first phase, the set of matches is selected using the terminal node of the request. If this is equal to an index entry key we get access to the list of postings. Further, each posting is analyzed and the structural distance between the request path and the path pointed by the post entry is computed in parallel by the components included in the FPGA processors. In our implementation we use a customized editing distance [148] to compute the path similarity score. If the score is beneath a given threshold, the result is discarded. Therefore, only a reduced part of the postings are returned to the server host.

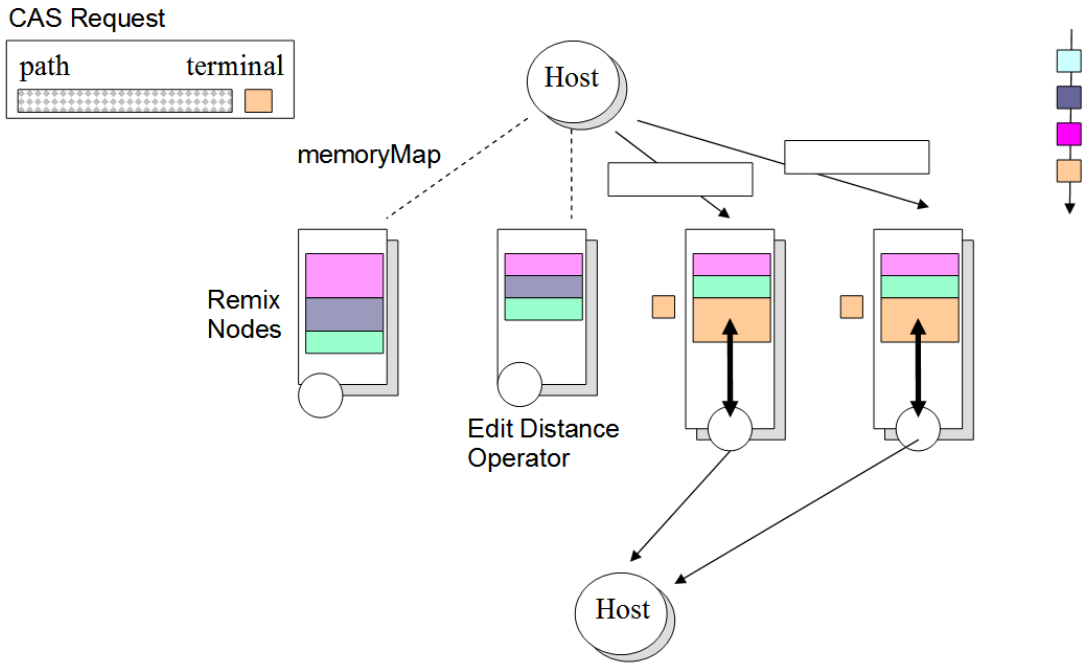


Figure 9.6: Dispatching and processing elementary requests.

In the second phase, we aggregate the lists of sub-results for the elementary requests on the host according to the complex operators specified in the request at element level. Next we can optionally compute further refinements for attributes and attributes values matching. Finally we perform the relevance score computation at element level, rank the results by relevance, merge them at document level (if required) and return the top-k answers to the user.

requests.txt

```
article/author/model
article/author
```

answers.txt

```
The request "?:  /article/author/model" gives 198 answer(s)
Node(0) with a score of (0.25) -> ./data/e0019.xml:  line(94)
/article/bdy/sec/p/
Node(1) with a score of (0.25) -> ./data/e0032.xml:  line(226)
/article/bdy/sec/p/
Node(2) with a score of (0.25) -> ./data/e0050.xml:  line(15)
/article/fm/abs/p/
Node(0) with a score of (0.25) -> ./data/e0050.xml:  line(15)
/article/fm/abs/p/
Node(1) with a score of (0.25) -> ./data/e0050.xml:  line(20)
/article/bdy/sec/p/
...
 The request "?:  /article/author" gives 81 answer(s)
Node(0) with a score of (0.25) -> ./data/e0019.xml:  line(187)
/article/bdy/sec/p/
Node(1) with a score of (0.25) -> ./data/e0061.xml:  line(30)
/article/bdy/sec/p/
Node(2) with a score of (0.16) -> ./data/e0061.xml:  line(4)
/article/bm/vt/fig/fgc/p/
Node(0) with a score of (0.2) -> ./data/e0061.xml:  line(43)
/article/bm/vt/fig/fgc/
...
```

Figure 9.7: Input and output files for the XML IR application adapted for the ReMIX architecture.

The steps performed with the current index and search model configurations to retrieve a list of ranked results using the ReMIX architecture are shown in Algorithm 1.

9.7 Current Implementation Status

A simplified prototype of the SIRIUS query engine [148] has been implemented and validated by using the ReMIX API. The prototype manage content only and content and structure requests for XML contexts without attributes conditions. The similarity between the context specified in the request and the indexed contexts is computed using a Levenshtein distance [131] operator.

A dedicated indexing program independently of the ReMIX platform was developed based on distributed hashtables from the QDBM database. The program produces an index file compatible with the ReMIX API.

The retrieval system takes as input a file containing the indexed database as discussed above, and a file containing a list of elementary requests. The output is given as a file containing the list of documents and XML elements paths and positions associated with their relevance scores. The list of results is not sorted.

Since the ReMIX hardware was shared between three different research teams, it was uneasy to get a full-time access to it. Therefore, we focused on the implementation of a first subset of SIRIUS operators on the ReMIX API which emulates the hardware. We show that this subset work on this API but, of course, since this API emulates ReMIX, the execution results (speed, bandwidth, etc.) cannot be accurate and do not reflect the hardware performance. For instance, input and output files obtained by our prototype when using the ReMIX API are given in Figure 9.7.

9.8 Early Experimental Results

We used a subset of the INEX IEEE XML corpus to validate our prototype. We indexed 4455 XML documents having a total size of 108 MB. We index only XML contexts ended by valid textual terminals where a textual terminal is valid if it does not occurs in a stop list of common words. The indexing time took 20 Mn on a Pentium IV processor at 3.2 GHz. The indexing process used exclusively the hard disk as storage support and the memory use was minimized as far as possible.

The obtained index was used to make a first series of tests with the XML IR application based on the ReMIX programming API. We simulated three cluster nodes by using threads on a single computer. The loading time using the ReMIX API (i.e. index distribution on the simulated nodes) took less than 4 Mn for 108 MB of indexed data.

9.9 Discussion

The proposed scheme is a straightforward parallelization of the indexing and approximate structural research scheme introduced in the SIRIUS XML IR engine [148]. This scheme was based on the inverted list model and path editing distance computations. The main objective was to validate the general schema proposed by the ReMIX project in the context of semi-structured information retrieval. In the current status of our work, a simplified index model and the approximate structure matching scheme was implemented and validated by using the ReMIX API.

For future works, a better organization of the indexed data is required in order to make a more efficient use of the capabilities offered by the ReMIX hardware memory. This can be achieved by constraining the FPGA processors to perform some of the merging operations at element level (AND,OR, IN, IN+, SEQ, etc.) , including the content relevance ranking algorithms (SAME+). This should allow:

- to improve the usage percentage of the FPGA accelerating capacities outside the standard editing distance computation for the structural match, and
- to drastically reduce the amount of information transferred from the FLASH memory to the host server.

To achieve this, all the information required for the merging and ranking operations must be available/indexed with the posting entry in order to be processed into a single pass by the FPGA processor. This is due to the fact that the FPGA processors use the data-flow paradigm and have no (or very low) capacities (i.e. internal memory) to fetch supplementary data from the main (FLASH) memory in order to perform a join operation or to make a ranking of the results list for example.

An index structure that may be suited for this purpose is the *signature files* [26, page 205]. Their search complexity is linear with the indexed information and not sublinear as in the case of the inverted lists. This is one of the reasons for which the signature files were shown to be outperformed by the inverted list files for most applications cases [26, page 205]. However, in the particular case of the ReMIX aggressive filtering architecture the signature files may represent an interesting choice that deserves to be evaluated.

We propose thus to investigate the use of a full redundant index for all the elements of the XML document tree where each XML element is associated with:

- a bit mask signature representing its textual content and
- a path representing its structural position in the tree.

The relevance ranking computation and the merging operators at element level can thus be performed or at least estimated by using the signature associated with the XML element content. This may be seen as a more aggressive and efficient filtering condition that combined with the structural one can be performed in parallel by the components included in each FPGA processor.

9.10 Conclusion

In this chapter we have described a straightforward implementation of an XML IR application on the application model developed in the context of the ReMIX project. The implementation provides on-the-fly approximate structural filtering and textual relevance ranking for the content of XML elements as support for searching relevant information in heterogeneous XML databases. We also provide some directions to further improve the performances of our implementation. The objective was to make a better usage of the possibilities offered by the the simultaneous use of FPGA components and FLASH memories.

Algorithm 1 Content and structure ranking using inverted lists on the ReMIX architecture.

Input :

- complex request: i.e. elementary requests with merging operators at both element and document level
- *StructuralThreshold*, *ContentThreshold*, minimum threshold values for structure and content matching scores

Output :

- ranked list of XML elements/documents with respect to both content and structure conditions

---

1: (Host) parse the complex request and extract the elementary requests -- i.e. a path request p^R ended with a textual terminal t

2: (Host) dispatch the elementary requests to the slave nodes

3: (Node 1-4) fetch the list of reference locators for the textual terminal

$t \rightarrow rl_0, rl_1, ..., rl_n$

4: (Node 1-4) for each path associated with a reference locator p^{rl_i}

5: (Node 1-4) $StructuralScore = EdDist(p^R, p^{rl_i})$

6: (Node 1-4) if $(StructuralScore \geq StructuralThreshold)$

then send the result to (Host),
else discard the result

7: (Host) merge the results at element level

8: (Host) compute the content relevance scores at element level

9: (Host) if $(ContentScore \leq ContentThreshold)$

then discard the result

9: (Host) merge content and structure scores at element level

10: (Host) merge elements scores at document level

10. (Host) final ranking of the results list

11: (Host) return top-k results

Part V

Conclusions

Chapter 10

Conclusions

Contents

10.1 Conclusions

In this dissertation we have introduced methods for managing and searching collections of heterogeneous multimedia XML documents. More precisely, we have focused on the flexible searching of structure, text, and sequential data embedded in heterogeneous XML document databases [173]. We have also designed and evaluated methods for i) approximate match of structural constraints in an XML IR framework [177, 176, 179, 178], and ii) detecting best entry points for starting to read an XML document on a given topic [179, 174]. Finally, we have adapted and validated the proposed approach for approximate search of semi-structured documents on a dedicated hardware memory developed in the context of the ReMIX[1] project [175].

The following sections summarize the main contributions of the thesis, and point out open problems and possible directions for future work.

10.2 Summary of Contributions

In this section we summarize the contributions of this thesis with respect to three main application domains: XML information retrieval, XML multimedia IR and XML IR on specialized hardware.

XML Information Retrieval

In the second part of this thesis we have tackled the ”XML Information Retrieval” field. We have mainly focused on indexing, retrieving and evaluating retrieval approaches for text-rich semi-structured documents with heterogeneous structures.

[1] Reconfigurable Memory for Indexing Mass of Data (ReMIX) `http://www.irisa.fr/remix`

We have proposed, implemented and evaluated a new search mechanism based on a set of tree paths matching that involves a modified Levenshtein [131] editing distance and information fusion heuristics.

We have proposed specific data structures dedicated to the indexing and retrieval of information elements embedded within heterogeneous XML data bases. The indexing scheme was found to be well suited for the characterization of various contextual searches, expressed either at a structural level or at an information content level.

We have developed a fully functional XML IR system. The implementation described highlights the mixing of structured information presented as field value instances and free text elements.

We have experimentally evaluated the proposed approach and the system within the INEX 2005 and INEX 2006 evaluation campaigns [177, 179]. During these evaluations we have obtained encouraging results compared with current state of the art XML IR systems. We have also shown that despite the lightweight characteristics of SIRIUS we were able to retrieve highly relevant non overlapping XML elements and to obtain quite good results for low values of recall.

The INEX 2005 and INEX 2006 test collections allowed us to evaluate the relevance gain to information access brought by the use of structural approximate matching mechanisms in an XML IR framework. The experimental results showed that a vague interpretation of the structural constraints can highly improve the quality of the retrieved results versus a strict interpretation [176, 178]. We have also shown that taking the structural constraints into account in an XML IR process may increase in some cases the effectiveness of the returned answers. This is although dependent of the quality and the discriminant power of the structural requests relative to the content and the structure of the test collection being used.

Finally, we have presented and evaluated a simple, efficient and effective approach for detecting the best entry points (BEPs) that suggest where to start reading a semi-structured document for a given topic [174]. Experiments conducted within the framework of INEX 2006 evaluation campaign have ranked the proposed approach on the 1^{st} place out of 77 official submissions for the *Best In Context* task. Furthermore, we have compared the effectiveness of the approach with a standard 'flat' document retrieval system that returns document snippets as BEPs based on the Google search engine. The experimental results seem to indicate that the current Web search engines technology could be profitably mixed with BEPs detection strategy in order to help the users to obtain better access to relevant information inside XML documents.

XML Multimedia IR

In the third part of the thesis, entitled "XML Multimedia IR", our main contributions concerned the introduction of a set of specialized operators designed for the extraction, indexing and retrieval of sequential/time series data embedded in heterogeneous XML documents. More precisely:

We have investigated sequential/time series data and their organization within the XML documents. This allowed us to observe that the XML document structure and the document order may encode useful and potentially (semantically) rich information about the sequential organization of the data. Starting from this observation we have proposed a structural sequential data types classification based on the structural properties of the XML documents and of the collection: *node level sequence, document*

level sequence and *collection level sequence.* We have also discussed how the implicit and explicit order relations could be applied to the various sequential structural types.

We have described and formalized a model that allows to represent sequences of symbols associated with any arbitrary XML context from the collection. The symbols are organized in sequences by taking into account: the type compatibility between their values, the similarities between their structural positions in the XML document trees, and an order relation.

We have also introduced an approximate sequence extraction scheme guided by structural patterns for extracting the sequence symbols and their corresponding contextual information from heterogeneous XML documents.

Further, we have devised and proposed a hybrid index model based on inverted lists and suffix trees index structures in order to support the storage of textual, structural and sequential data.

Furthermore, we have defined a model for retrieving similar sequences extracted from XML documents. The matching mechanism allows to aggregate two levels of approximation: on the structural localization/organization of the sequential data and on its unstructured content.

Finally, we have implemented the sequential operators within the SIRIUS XML IR system in order to evaluate the benefits and the drawbacks of the proposed approach.

XML IR on Specialized Hardware

In the fourth part of this thesis dedicated to "XML IR on Specialized Hardware" our main contribution was to explore the use of an original memory architecture for both storing very large indexed data structures, and allowing fast information retrieval as support for XML IR. More precisely we have adapted a simplified model of our XML IR approach to the application model developed in the context of the ReMIX project. The implementation presented allows on-the-fly approximate structural filtering as support for searching relevant information in heterogeneous semi-structured databases.

10.3 Future Research

> *"Success is the ability to go from one failure to another with no loss of enthusiasm."*
>
> *— Sir Winston Churchill*

In this section we point out open problems and suggest interesting directions for future research.

XML Information Retrieval

We have proposed and evaluated structural approximate matching mechanisms based on a modified Levenshtein [131] editing distance and information fusion heuristics to the problem of path matching. Our experiments at INEX showed the pertinence of the proposed approach for the vague structural match and for focused access to relevant elements.

When applying this scheme for flexible path matching, we use statically predefined costs for all the tag names – i.e. no difference is made between the relative importance

of the tag names within the collection. Nonetheless, some of the tag names may occur more or less frequent in the collection, and therefore, be more or less discriminant for the retrieval process. A straight research perspective to this work concerns the extension of the tag weights in order to integrate their discriminant power within the test collection. For example, we can adapt term frequency models like the well-known TF-IDF-based [195] weighting schemes to XML to derive the weights. The tag names weighting schemes should be adapted for both heterogeneous and homogeneous contexts. The different relationships between elements, and their rareness could also be taken into consideration. Another possible approach is to automatically adapt the tag weights during the query evaluation process by taking into account the query results (or a subset of it) within a structural relevance feedback mechanism.

The experimental results have also showed that the approximate search inside XML elements implemented using the *same+* operator improves the overall performance of the ranking, compared to a strict sequence search (*seq* operator), except for low recall values. The complementarity of the two operators call for the design of a new matching operator based on their combination to further improve the retrieval performance.

Another important direction for future research is to extensively evaluate the behavior of the proposed approximate structural matching approach across different documents and collections.

We have described, implemented and evaluated a simple and effective method to automatically identify BEPs in semi-structured documents. As a perspective to this work, we can explore the impact of aggregating BEPs heuristics and document ranking heuristics in order to better support the users' information-seeking behavior and to improve the overall performance of the search strategy.

Finally, future work should also address the optimization of the search algorithms and index structures in the context of the *top-k* query processing paradigm, and the development of a rich graphical interface for browsing and querying large collections of heterogeneous XML documents. The GUI should be able to assist the user in the query formulation process by combining the use of structural summaries [9], semantic concepts[2], and the automatic schema matching approaches [181].

XML Multimedia IR

The work presented in this part can be extended in several directions.

One future research direction is to experiment with a flexible organization relative to the order relation of the symbols in the sequences and to measure the impact on the efficiency and effectiveness of the system for such relaxation.

Another line of interest is the design of a weighted model for sequential data in which each symbol of the sequence can be associated with a weight indicating its contribution/relevance to the current sequence.

A different research direction, concerns the design of an integrated index model that allows precomputed access for the most likely sequential extraction patterns and sequential similarity queries. The model could extend the nested relational sequence model presented in [127]. We make here the observation that this index model had to be extended to allow arbitrary ordering patterns for the sequence symbols (other than

[2]WordNet `http://wordnet.princeton.edu/`

the document order currently supported) and to represent sequences that do not fit a crisp structural pattern.

Both, the disk resident organization of the current index structures and the parallelization of the search algorithms will be a straight forward research direction in order to validate our approach on important volumes of data.

Finally, designing an evaluation framework for the retrieval of sequential data embedded in heterogeneous XML documents represents a major research challenge. We consider the possibility of reusing and integrating the assessed datasets of two existing test collections: the first provided by *The Music Information Retrieval Evaluation eXchange* (MIREX)[3] (more precisely, the collection used in the *Symbolic Melodic Similarity Task*[4]), and the second collection provided by the *INEX XML Multimedia Track*[5] [232, 243].

XML IR on Specialized Hardware

One direction to further improve the performances of our implementation is to make a better usage of the possibilities offered by the simultaneous use of the FPGA components and of the FLASH memories – i.e. devise an indexing and retrieval model that can aggressively filter the data as soon as it leaves the FLASH memory. One possibility is to use a kind of a bitmask signature for the content of the selected elements and to attach this signature to each entry in the posting list. The influence of these signatures in terms of storage space requirements as well as their impact in: i) reducing the volume of transferred data between the FLASH memory and the ReMIX nodes during the retrieval process and ii) their influence on the quality of the retrieved results, also deserves to be explored.

A second perspective concerns the study of sequence based tree indexing structures [241, 185] in the context of on-the-fly filtering paradigm.

In addition, the investigation of the highly parallel computing power of the general purpose graphics processing units (GPGPU) as support for approximate structural matching algorithms within an XML IR context represent another appealing future research direction.

[3]The Music Information Retrieval Evaluation eXchange (MIREX) `http://www.music-ir.org/mirex2007/index.php/Main_Page`

[4]The MIREX Symbolic Melodic Similarity Task `http://www.music-ir.org/mirex2007/index.php/Symbolic_Melodic_Similarity`

[5]INEX Multimedia Track `http://inex.is.informatik.uni-duisburg.de/2007/mmtrack.html`

Bibliography

[1] *SIGIR '98: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 24-28 1998, Melbourne, Australia*. ACM, 1998.

[2] Synchronized Multimedia Integration Language (SMIL) 1.0 Specification, W3C Recommendation. Technical report, W3C - The World Wide Web Consortium, 1998.

[3] Mathematical Markup Language (MathML) Version 2.0 (Second Edition), W3C Recommendation. Technical report, W3C - The World Wide Web Consortium, 21 October 2003.

[4] MidiXML, Standard MIDI File DTD: MIDI XML, Version 1.0 - 13 January 2004. http://www.recordare.com/dtds/midixml.html, 2004.

[5] MusicXML, MusicXML Definition, Version 1.0, January 2004. http://www.recordare.com/xml.html, 2004.

[6] *Proceedings of ACM SIGIR 2006 Workshop on XML Element Retrieval Methodology, Seattle, WA, USA*. ACM Press, Aug. 2006.

[7] *SIGIR 2006: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, Washington, USA, August 6-11, 2006*. ACM, 2006.

[8] M. Abolhassani and N. Fuhr. Applying the Divergence from Randomness Approach for Content-Only Search in XML Documents. In S. McDonald and J. Tait, editors, *ECIR*, volume 2997 of *Lecture Notes in Computer Science*, pages 409–419. Springer, 2004.

[9] M. S. Ali, M. P. Consens, X. Gu, Y. Kanza, F. Rizzolo, and R. K. Stasiu. Efficient, Effective and Flexible XML Retrieval Using Summaries. In Fuhr et al. [71], pages 89–103.

[10] S. Amer-Yahia, R. Baeza-Yates, M. Consens, and M. Lalmas. XML Retrieval: Integrated IR-DB Challenges and Solutions. SIGIR 07 Tutorial, July 2007.

[11] S. Amer-Yahia and P. Case. XQuery 1.0 and XPath 2.0 Full-Text Use Cases. W3C working draft, W3C, May 2006. http://www.w3.org/TR/2006/WD-xmlquery-full-text-use-cases-20060501/.

[12] S. Amer-Yahia, N. Koudas, A. Marian, D. Srivastava, and D. Toman. Structure and Content Scoring for XML. In Böhm et al. [30], pages 361–372.

[13] S. Amer-Yahia, N. Koudas, and D. Srivastava. Approximate Matching in XML. In U. Dayal, K. Ramamritham, and T. M. Vijayaraman, editors, *Proceedings of the 19th International Conference on Data Engineering (ICDE), March 5-8, 2003, Bangalore, India*, page 803. IEEE Computer Society, 2003.

[14] S. Amer-Yahia, N. Koudas, and D. Srivastava. Approximate Matching in XML. *ICDE*, 00:803, 2003.

[15] S. Amer-Yahia, L. V. S. Lakshmanan, and S. Pandit. FleXPath: Flexible Structure and Full-Text Querying for XML. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 83–94, New York, NY, USA, 2004. ACM Press.

[16] S. Amer-Yahia and M. Lalmas. XML Search: Languages, INEX and Scoring. *SIGMOD Record*, 35(4):16–23, 2006.

[17] L. Amsaleg, P. Gros, and S.-A. Berrani. Robust Object Recognition in Images and the Related Database Problems. *Multimedia Tools Appl.*, 23(3):221–235, 2004.

[18] T. E. Anderson, D. E. Culler, D. A. Patterson, , and the NOW team. A Case for NOW (Networks of Workstations). *IEEE Micro*, 15(1):54–64, 1995.

[19] P. Arvola, J. Kekäläinen, and M. Junkkari. Query Evaluation with Structural Indices. In Fuhr et al. [69], pages 134–145.

[20] E. Ashoori and M. Lalmas. Using Topic Shifts for Focussed Access to XML Repositories. In *ECIR*, pages 444–455, 2007.

[21] E. Ashoori, M. Lalmas, and T. Tsikrika. Examining Topic Shifts in Content-Oriented XML Retrieval. *International Journal on Digital Libraries*, 2007. to appear.

[22] R. Baeza-Yates, N. Fuhr, and Y. S. Maarek, editors. *Proceedings of the SIGIR 2002 Workshop on XML and Information Retrieval*, 2002.

[23] R. Baeza-Yates, N. Fuhr, R. Sacks-Davis, and R. Wilkinson, editors. *Proceedings of the SIGIR 2000 Workshop on XML and Information Retrieval*, 2000.

[24] R. Baeza-Yates and M. Lalmas. XML Information Retrieval. SIGIR 06 Tutorial, August 2006.

[25] R. Baeza-Yates and G. Navarro. XQL and Proximal Nodes. *J. Am. Soc. Inf. Sci. Technol.*, 53(6):504–514, 2002.

[26] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press. Addison-Wesley, New-York, 1999.

[27] L. A. Barroso, J. Dean, and U. Hölzle. Web Search for a Planet: The Google Cluster Architecture. *IEEE Micro*, 23(2):22–28, 2003.

[28] D. Beech, N. Mendelsohn, H. S. Thompson, and M. Maloney. XML Schema Part 1: Structures Second Edition. W3C recommendation, W3C, Oct. 2004. http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/.

[29] S.-A. Berrani, L. Amsaleg, and P. Gros. Application de la recherche approximative de plus proches voisins à la recherche d'images par le contenu pour la détection des copies. In J. L. Maitre, editor, *BDA*, pages 197–218, 2004.

[30] K. Böhm, C. S. Jensen, L. M. Haas, M. L. Kersten, P.-Å. Larson, and B. C. Ooi, editors. *Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005*. ACM, 2005.

[31] G. Bordogna and G. Pasi. Flexible Querying of WEB Documents. In *SAC '02: Proceedings of the 2002 ACM symposium on Applied computing*, pages 675–680, New York, NY, USA, 2002. ACM.

[32] D. Braga, A. Campi, E. Damiani, P. Lanzi, and G. Pasi. FXPath: Flexible Querying of XML Documents. In *EuroFuse Workshop on Information Systems*, Sep. 2002.

[33] J.-M. Bremer. *Next-Generation Information Retrieval: Integrating Document and Data Retrieval Based on XML*. PhD thesis, Univerity of California, Davis, USA, 2003.

[34] S. Brin and L. Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[35] A. Broder and M. R. Henzinger. Information Retrieval on the Web. In *FOCS '98: Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, page 6, Washington, DC, USA, 1998. IEEE Computer Society.

[36] N. Bruno, N. Koudas, and D. Srivastava. Holistic Twig Joins: Optimal XML Pattern Matching. In *SIGMOD '02: Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 310–321, New York, NY, USA, 2002. ACM Press.

[37] D. Butler. A Short Survey of Document Structure Similarity Algorithms. In *International Conference on Internet Computing*, 2004.

[38] J. P. Callan. Passage-level Evidence in Document Retrieval. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 302–310, New York, NY, USA, 1994. Springer-Verlag New York, Inc.

[39] L. Camargos, F. Pedone, and M. Wieloch. Sprint: A Middleware for High-Performance Transaction Processing. In *2nd European Conference on Systems Research (EuroSys'2007)*, 2007.

[40] G. R. Camps. *Structural Features in XML Retrieval*. PhD thesis, University of Amsterdam, 2007.

[41] D. Carmel, Y. S. Maarek, M. Mandelbrod, Y. Mass, and A. Soffer. Searching XML Documents via XML Fragments. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 151–158, New York, NY, USA, 2003. ACM Press.

[42] P. Case, S. Amer-Yahia, M. Holstege, J. Doerre, D. McBeath, J. Shanmugasundaram, C. Botev, M. Rys, and S. Buxton. XQuery 1.0 and XPath 2.0 Full-Text. W3C working draft, W3C, May 2006. http://www.w3.org/TR/2006/WD-xquery-full-text-20060501/.

[43] B. Catania, A. Maddalena, and A. Vakali. XML Document Indexes: A Classification. *IEEE Internet Computing*, 9(5):64–71, 2005.

[44] S. Chakrabarti, M. van den Berg, and B. Dom. Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery. In *WWW '99: Proceeding of the eighth international conference on World Wide Web*, pages 1623–1640, New York, NY, USA, 1999. Elsevier North-Holland, Inc.

[45] D. Chamberlin, M. F. Fernández, S. Boag, J. Siméon, J. Robie, A. Berglund, and M. Kay. XML Path Language XPath 2.0. Candidate recommendation, W3C, June 2006. http://www.w3.org/TR/2006/CR-xpath20-20060608/.

[46] N. Chen. A Survey of Indexing and Retrieval of Multimodal Documents: Text and Images. Technical Report 2006-505, Scholl Of Computing Queen's University, Kingston, Ontario, Canada, February 2006.

[47] Y. Chiaramella, P. Mulhem, and F. Fourel. A Model for Multimedia Information Retrieval. Technical Report FERMI ESPRIT BRA 8134, University of Glasgow, July 1996.

[48] C. Clarke. Range Results in XML Retrieval. In *Proceedings of the INEX 2005 Workshop on Element Retrieval Methodology*, 2005.

[49] C. Clarke, J. Kamps, and M. Lalmas. INEX 2006 Retrieval Task and Result Submission Specification. In *INEX 2006 Pre-proceedings*, pages 381–388, 2006.

[50] C. L. A. Clarke. Controlling Overlap in Content-Oriented XML Retrieval. In *SIGIR*, pages 314–321, 2005.

[51] E. Cohen, H. Kaplan, and T. Milo. Labeling Dynamic XML Trees. In *PODS '02: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 271–281, New York, NY, USA, 2002. ACM Press.

[52] S. Cohen, J. Mamou, Y. Kanza, and Y. Sagiv. XSEarch: A Semantic Search Engine for XML. In *VLDB*, pages 45–56, 2003.

[53] F. Crestani, P. de la Fuente, and J. Vegas. Design of a Graphical User Interface for Structured Documents Retrieval. In *SPIRE*, pages 246–249, 2001.

[54] C. J. Crouch, D. B. Crouch, M. Ganapathibhotla, and V. Bakshi. Dynamic Element Retrieval in a Semi-structured Collection. In Fuhr et al. [71], pages 82–88.

[55] E. Damiani and L. Tanca. Blind Queries to XML Data. In *Database and Expert Systems Applications*, pages 345–356, 2000.

[56] L. M. de Campos, J. M. Fernández-Luna, and J. F. Huete. Using Context Information in Structured Document Retrieval: An Approach Based on Influence Diagrams. volume 40, pages 829–847, Tarrytown, NY, USA, 2004. Pergamon Press, Inc.

[57] C. de Loupy. *Évaluation de l'Apport de Connaissances Linguistiques en Recherche Documentaire et Désambigusation Sémantique*. PhD thesis, Université d'Avignon et des Pays de Vaucluse, 2000.

[58] E. Demaine, S. Mozes, B. Rossman, and O. Weimann. An Optimal Decomposition Algorithm for Tree Edit Distance. In *Proceedings of the 34th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 146–157, 2007.

[59] L. Denoyer and P. Gallinari. The Wikipedia XML Corpus. *SIGIR Forum*, 2006.

[60] S. DeRose and J. Clark. XML Path Language (XPath) Version 1.0. W3C recommendation, W3C, Nov. 1999. http://www.w3.org/TR/1999/REC-xpath-19991116.

[61] C. F. Dorneles, C. A. Heuser, A. E. N. Lima, A. S. da Silva, and E. S. de Moura. Measuring Similarity Between Collection of Values. In A. H. F. Laender, D. Lee, and M. Ronthaler, editors, *WIDM*, pages 56–63. ACM, 2004.

[62] D. Egnor and R. Lord. Structured Information Retrieval using XML. In *ACM SIGIR 2000 Workshop On XML and Information Retrieval*, Athens, Greece, July 28 2000.

[63] M. F. Fernández, J. Robie, S. Boag, D. Chamberlin, D. Florescu, and J. Siméon. XQuery 1.0: An XML Query Language. Candidate recommendation, W3C, June 2006. http://www.w3.org/TR/2006/CR-xquery-20060608/.

[64] N. Fuhr, N. Gövert, G. Kazai, and M. Lalmas, editors. *Proceedings of the First Workshop of the INitiative for the Evaluation of XML Retrieval (INEX), Schloss Dagstuhl, Germany, December 9-11, 2002*, 2002.

[65] N. Fuhr and K. Großjohann. XIRQL: A Query Language for Information Retrieval in XML Documents. In *Research and Development in Information Retrieval*, pages 172–180, 2001.

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

[67] N. Fuhr, C.-P. Klas, A. Schaefer, and P. Mutschke. Daffodil: An Integrated Desktop for Supporting High-Level Search Activities in Federated Digital Libraries. In *Research and Advanced Technology for Digital Libraries. 6th European Conference, ECDL 2002*, pages 597–612, Heidelberg et al., 2002. Springer.

[68] N. Fuhr, M. Lalmas, and S. Malik, editors. *Proceedings of the Second Workshop of the INitiative for the Evaluation of XML Retrieval, INEX 2002, Schloss Dagstuhl, Germany, December 15-17 2003*, 2004.

[69] N. Fuhr, M. Lalmas, S. Malik, and G. Kazai, editors. *Advances in XML Information Retrieval and Evaluation, 4th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2005, Dagstuhl Castle, Germany, November 28-30, 2005, Revised Selected Papers*, volume 3977 of *Lecture Notes in Computer Science*. Springer, 2006.

[70] N. Fuhr, M. Lalmas, S. Malik, and Z. Szlávik, editors. *Advances in XML Information Retrieval, Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004, Dagstuhl Castle, Germany, December 6-8, 2004, Revised Selected Papers*, volume 3493 of *Lecture Notes in Computer Science*. Springer, 2005.

[71] N. Fuhr, M. Lalmas, and A. Trotman, editors. *Comparative Evaluation of XML Information Retrieval Systems, 5th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2006, Dagstuhl Castle, Germany, December 18-20, 2006, Revised Selected Papers*, volume 4518 of *Lecture Notes in Computer Science*. Springer, 2007.

[72] M. Fuller, E. Mackie, R. Sacks-Davis, and R. Wilkinson. Structured Answers for a Large Structured Document Collection. In *SIGIR '93: Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 204–213, New York, NY, USA, 1993. ACM Press.

[73] H. Garcia-Molina and K. Salem. Main Memory Database Systems: An Overview. *IEEE Transactions on Knowledge and Data Engineering*, 4(6):509–516, 1992.

[74] G. Georges, S. Derrien, S. Rubini, F. Raimbault, L. Amsaleg, and D. Lavenier. ReMIX: une architecture pour la recherche dans les masses de donn'ees indexées. In *Symposium en Architecture de Machines*, Perpignan, France, 2006.

[75] M. Géry. Indexing "Reading Paths" for a Structured Information Retrieval at INEX 2006. In Fuhr et al. [71], pages 160–164.

[76] S. Geva. GPX - Gardens Point XML Information Retrieval at INEX 2004. In Fuhr et al. [70], pages 211–223.

[77] S. Geva. GPX - Gardens Point XML IR at INEX 2006. In Fuhr et al. [71], pages 137–150.

[78] N. Gövert, M. Abolhassani, N. Fuhr, and K. Großjohann. Content-Oriented XML Retrieval with HyRex. In Fuhr et al. [64], pages 26–32.

[79] T. Grabs and H.-J. Schek. ETH Zürich at INEX: Flexible Information Retrieval from XML with PowerDB-XML. In Fuhr et al. [64], pages 141–148.

[80] J. Graupmann, R. Schenkel, and G. Weikum. The SphereSearch Engine for Unified Ranked Retrieval of Heterogeneous XML and Web Documents. In *VLDB '05: Proceedings of the 31st International Conference on Very Large Data Bases*, pages 529–540. VLDB Endowment, 2005.

[81] K. Grobjohann, N. Fuhr, D. Effing, and S. Kriewel. Query Formulation and Result Visualization for XML Retrieval. In *Proceedings ACM SIGIR 2002 Workshop on XML and Information Retrieval*. ACM, 2002.

[82] T. Grust. Accelerating XPath Location Steps. In *SIGMOD '02: Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 109–120, New York, NY, USA, 2002. ACM Press.

[83] D. Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, New York, NY, USA, 1997.

[84] S. Guyetant, M. Giraud, L. L'Hours, S. Derrien, S. Rubini, D. Lavenier, and F. Raimbault. Cluster of Re-Configurable Nodes for Scanning Large Genomic Banks. *Parallel Comput.*, 31(1):73–96, 2005.

[85] T. Härder, M. Haustein, C. Mathis, and M. Wagner. Node Labeling Schemes for Dynamic XML Documents Reconsidered. *Data Knowl. Eng.*, 60(1):126–149, 2007.

[86] M. Hassler and A. Bouchachia. Searching XML Documents - Preliminary Work. In Fuhr et al. [69], pages 119–133.

[87] M. A. Hearst. TileBars: Visualization of Term Distribution Information in Full Text Information Access. In *CHI*, pages 59–66, 1995.

[88] M. A. Hearst. TextTiling: Segmenting Text into Multi-Paragraph Subtopic Passages. volume 23, pages 33–64, Cambridge, MA, USA, 1997. MIT Press.

[89] A. Henrich and G. Robbert. POQL$^{\mathrm{mm}}$: A query language for structured multimedia documents. In *Proceedings 1st International Workshop on Multimedia Data and Document Engineering (MDDE'01)*, 2001.

[90] M. L. Hetland. *Data Mining in Time Series Databases*, chapter A Survey of Recent Methods for Efficient Retrieval of Similar Time Sequences. World Scientific Press, 2004.

[91] D. Hiemstra. *Using Language Models for Information Retrieval*. PhD thesis, Centre for Telematics and Information Technology, University of Twente, 2001.

[92] L. Hlaoua, M. Torjmen, K. Pinel-Sauvagnat, and M. Boughanem. XFIRM at INEX 2006. Ad-Hoc, Relevance Feedback and MultiMedia Tracks. In Fuhr et al. [71], pages 373–386.

[93] G. Hubert. Tuning and Evolving Retrieval Engine by Training on Previous INEX Testbeds. In Fuhr et al. [71], pages 243–252.

[94] E. Hunt, M. P. Atkinson, and R. W. Irving. Database Indexing for Large DNA and Protein Sequence Collections. volume 11, pages 256–271, Secaucus, NJ, USA, 2002. Springer-Verlag New York, Inc.

[95] D. N. F. Iskandar, J. Pehcevski, J. A. Thom, and S. M. M. Tahaghoghi. Combining Image and Structured Text Retrieval. In Fuhr et al. [69], pages 525–539.

[96] D. N. F. A. Iskandar, J. Pehcevski, J. A. Thom, and S. M. M. Tahaghoghi. Social Media Retrieval Using Image Features and Structured Text. In Fuhr et al. [71], pages 358–372.

[97] K. Itakura and C. Clarke. From Passages into Elements in XML Retrieval. In *SIGIR 2007 Workshop on Focused Retrieval*, pages 17–22. University of Otago, Dunedin New Zealand, 2007.

[98] K. Järvelin and J. Kekäläinen. Cumulated Gain-Based Evaluation of IR Techniques. *ACM Transactions on Information Systems*, 20(4):422–446, 2002.

[99] J. Jiang and C. Zhai. Extraction of Coherent Relevant Passages using Hidden Markov Models. *ACM Trans. Inf. Syst.*, 24(3):295–319, 2006.

[100] J. Kamps, M. de Rijke, and B. Sigurbjörnsson. Length Normalization in XML Retrieval. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 80–87. ACM Press, 2004.

[101] J. Kamps and M. Koolen. On the Relation between Relevant Passages and XML Document Structure. In *SIGIR 2007 Workshop on Focused Retrieval*, pages 28–32. University of Otago, Dunedin New Zealand, 2007.

[102] J. Kamps, M. Koolen, and M. Lalmas. Where to Start Reading a Textual XML Document? In *SIGIR '07: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 723–724, New York, NY, USA, 2007. ACM.

[103] J. Kamps, M. D. Rijke, and B. Sigurbjörnsson. The Importance of Length Normalization for XML Retrieval. *Inf. Retr.*, 8(4):631–654, 2005.

[104] J. Kamps and B. Sigurbjörnsson. What Do Users Think of an XML Element Retrieval System? In Fuhr et al. [69], pages 411–421.

[105] M. Kaszkiel, J. Zobel, and R. Sacks-Davis. Efficient Passage Ranking for Document Databases. volume 17, pages 406–439, New York, NY, USA, 1999. ACM Press.

[106] R. Kaushik, R. Krishnamurthy, J. F. Naughton, and R. Ramakrishnan. On the Integration of Structure Indexes and Inverted Lists. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 779–790, New York, NY, USA, 2004. ACM Press.

[107] G. Kazai and E. Ashoori. What does Shakespeare have to do with INEX? In *SIGIR 2006 Workshop on XML Element Retrieval Methodology*, pages 20–26, 2006.

[108] G. Kazai and M. Lalmas. INEX 2005 Evaluation Measures. In Fuhr et al. [69], pages 16–29.

[109] G. Kazai, M. Lalmas, and A. P. de Vries. The Overlap Problem in Content-Oriented XML Retrieval Evaluation. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 72–79, New York, NY, USA, 2004. ACM Press.

[110] G. Kazai, M. Lalmas, and T. Rölleke. Focussed Structured Document Retrieval. In A. H. F. Laender and A. L. Oliveira, editors, *SPIRE*, volume 2476 of *Lecture Notes in Computer Science*, pages 241–247. Springer, 2002.

[111] K. Keeton, D. A. Patterson, and J. M. Hellerstein. A Case for Intelligent Disks (IDISKs). *SIGMOD Rec.*, 27(3):42–52, 1998.

[112] J. M. Kleinberg. Authoritative Sources in a Hyperlinked Environment. *Journal of the ACM*, 46(5):604–632, 1999.

[113] Z. Kong and M. Lalmas. Integrating XLink and XPath to Retrieve Structured Multimedia Documents in Digital Libraries. In *RIAO 2004 Conference on Coupling approaches, coupling media and coupling languages for information retrieval*, 2004.

[114] Z. Kong and M. Lalmas. XML Multimedia Retrieval. In *Proceedings of String Processing and Information Retrieval (SPIRE'05)*. Argentinean Computer Science Society (SADIIO), Buenos Aires, Argentina, 2005. Short Paper.

[115] Z. Kong and M. Lalmas. Combining Multiple Sources of Evidence in XML Multimedia Documents: An Inference Network Incorporating Element Language Models. In *29th European Conference on Information Retrieval (ECIR'07)*, pages 716–719, 2007.

[116] Z. Kong and M. Lalmas. Using XML Logical Structure to Retrieve (Multimedia) Objects. In *11th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2007)*, pages 100–111, Budapest, Hungary, 2007.

[117] M. Lalmas. Dempster-Shafer's Theory of Evidence Applied to Structured Documents: Modelling Uncertainty. In *SIGIR*, pages 110–118. ACM, 1997.

[118] M. Lalmas and G. Kazai. Evaluating XML Retrieval Effectiveness at INEX. *ACM SIGIR Forum*, 40(1):49–57, June 2006.

[119] M. Lalmas and G. Kazai. Report on the Ad-hoc Track of the INEX 2005 Workshop. *SIGIR Forum*, 40(1):49–57, 2006.

[120] M. Lalmas, G. Kazai, J. Kamps, J. Pehcevski, B. Piwowarski, and S. Robertson. INEX 2006 Evaluation Measures. In Fuhr et al. [71], pages 20–34.

[121] M. Lalmas and J. Reid. Automatic Identification of Best Entry Points for Focused Structured Document Retrieval. In *CIKM*, pages 540–543, NY, USA, 2003.

[122] M. Lalmas and A. Tombros. Evaluating XML Retrieval Effectiveness at INEX. *ACM SIGIR Forum*, 41(1):40–57, June 2007.

[123] A. N. Langville and C. D. Meyer. A Survey of Eigenvector Methods for Web Information Retrieval. volume 47, pages 135–161, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics.

[124] R. R. Larson. Probabilistic Retrieval, Component Fusion and Blind Feedback for XML Retrieval. In Fuhr et al. [69], pages 225–239.

[125] R. R. Larson. Probabilistic Retrieval Approaches for Thorough and Heterogeneous XML Retrieval. In Fuhr et al. [71], pages 318–330.

[126] C. Lau, D. Tjondronegoro, J. Zhang, S. Geva, and Y. Liu. Fusing Visual and Textual Retrieval Techniques to Effectively Search Large Collections of Wikipedia Images. In Fuhr et al. [71], pages 345–357.

[127] H. L. Lau and W. Ng. Querying XML Data by the Nested Relational Sequence Database System. In *IDEAS*, pages 236–241. IEEE Computer Society, 2003.

[128] D. Lavenier, G. Georges, and X. Liu. A Reconfigurable Index FLASH Memory tailored to Seed-Based Genomic Sequence Comparison Algorithms. *The Journal of VLSI signal processing systems. Special issue on Computing Architectures and Acceleration for Bioinformatics Algorithms*, 43(4), September 2007.

[129] D. Lavenier, L. Xinchun, and G. Georges. Seed-Based Genomic Sequence Comparison using a FPGA/FLASH Accelerator. *Field Programmable Technology, 2006. FPT 2006. IEEE International Conference on*, pages 41–48, Dec. 2006.

[130] M. Lehtonen and A. Doucet. EXTIRP: Baseline Retrieval from Wikipedia. In Fuhr et al. [71], pages 115–120.

[131] V. I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady.*, 10(8):707–710, February 1966.

[132] Q. Li and B. Moon. Indexing and Querying XML Data for Regular Path Expressions. In *VLDB '01: Proceedings of the 27th International Conference on Very Large Data Bases*, pages 361–370, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[133] X. Liu and W. B. Croft. Passage Retrieval Based on Language Models. In *CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management*, pages 375–382, New York, NY, USA, 2002. ACM Press.

[134] W. Lu, S. E. Robertson, and A. MacFarlane. Field-Weighted XML Retrieval Based on BM25. In Fuhr et al. [69], pages 161–171.

[135] H. P. Luhn. A Statistical Approach to Mechanized Encoding and Searching of Literary Information. *IBM Journal of Research and Development*, 1:309–317, 1957.

[136] R. W. P. Luk, H. V. Leong, T. S. Dillon, A. T. S. Chan, W. B. Croft, and J. Allan. A Survey in Indexing and Searching XML Documents. *Journal of the American Society for Information Science & Technology (JASIST)*, 53(6):415–437, 2002.

[137] P. Lyman and H. R. Varian. How Much Information, 2003. Retrieved from http://www.sims.berkeley.edu/how-much-info-2003.

[138] A. Malhotra and P. V. Biron. XML Schema Part 2: Datatypes Second Edition. W3C recommendation, W3C, 2004. http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/.

[139] F. Mandreoli, R. Martoglia, and P. Tiberio. Approximate Query Answering for a Heterogeneous XML Document Base. In X. Zhou, S. Y. W. Su, M. P. Papazoglou, M. E. Orlowska, and K. G. Jeffery, editors, *WISE*, volume 3306 of *Lecture Notes in Computer Science*, pages 337–351. Springer, 2004.

[140] C. Martínez and S. Roura. Randomized binary search trees. *J. ACM*, 45(2):288–323, 1998.

[141] Y. Mass. IBM HRL at INEX 06. In Fuhr et al. [71], pages 151–159.

[142] Y. Mass and M. Mandelbrod. Component Ranking and Automatic Query Refinement for XML Retrieval. In Fuhr et al. [70], pages 73–84.

[143] Y. Mass and M. Mandelbrod. Using the INEX Environment as a Test Bed for Various User Models for XML Retrieval. In Fuhr et al. [69], pages 187–195.

[144] E. M. McCreight. A Space-Economical Suffix Tree Construction Algorithm. *J. ACM*, 23(2):262–272, 1976.

[145] J. McHugh, S. Abiteboul, R. Goldman, D. Quass, and J. Widom. Lore: A Database Management System for Semistructured Data. *SIGMOD Rec.*, 26(3):54–66, 1997.

[146] C. Meek, J. M. Patel, and S. Kasetty. OASIS: An Online and Accurate Technique for Local-alignment Searches on Biological Sequences. In *VLDB*, pages 910–921, 2003.

[147] G. Memik, M. T. Kandemir, and A. Choudhary. Design and Evaluation of a Smart Disk Cluster for DSS Commercial Workloads. *J. Parallel Distrib. Comput.*, 61(11):1633–1664, 2001.

[148] G. Ménier and P.-F. Marteau. Information Retrieval in Heterogeneous XML Knowledge Bases. In *The 9th International Conference on Information Processing and Magement of Uncertainty in Knowledge-Based Systems (IPMU)*, pages 1399–1405, Annecy, France, July 2002.

[149] L. Mignet, D. Barbosa, and P. Veltri. The XML Web: A First Study. In *Proc. of 12th International World Wide Web Conf.*, pages 500–510, Budapest, Hungary, May 2003.

[150] V. Mihajlovic, H. E. Blok, D. Hiemstra, and P. M. G. Apers. Score Region Algebra: Building a Transparent XML-R Database. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 12–19, New York, NY, USA, 2005. ACM Press.

[151] V. Mihajlovic, G. Ramírez, T. Westerveld, D. Hiemstra, H. E. Blok, and A. P. de Vries. TIJAH Scratches INEX 2005: Vague Element Selection, Image Search, Overlap, and Relevance Feedback. In Fuhr et al. [69], pages 72–87.

[152] A. Moffat, R. Sacks-Davis, R. Wilkinson, and J. Zobel. Retrieval of Partial Documents. In *Text REtrieval Conference*, pages 181–190, 1993.

[153] S.-H. Myaeng, D.-H. Jang, M.-S. Kim, and Z.-C. Zhoo. A Flexible Model for Retrieval of SGML Documents. In *SIGIR* [1], pages 138–145.

[154] G. Navarro. A Guided Tour to Approximate String Matching. *ACM Comput. Surv.*, 33(1):31–88, 2001.

[155] V. Nguyen and D. Lavenier. Recherche dans les banques d'ADN par indexation parall'ele. In *4th International Conference on Research, Innovation & Vision for the Future*, Ho Chi Minh Ville, Vietnam, 2006.

[156] P. Ogilvie and J. Callan. Using Language Models for Flat Text Queries in XML Retrieval. In N. Fuhr, S. Malik, and M. Lalmas, editors, *INEX 2003 Workshop Proceedings*, 2003.

[157] P. Ogilvie and J. Callan. Hierarchical Language Models for XML Component Retrieval. In Fuhr et al. [70], pages 224–237.

[158] P. Ogilvie and J. Callan. Parameter Estimation for a Simple Hierarchical Generative Model for XML Retrieval. In Fuhr et al. [69], pages 211–224.

[159] R. A. O'Keefe and A. Trotman. The Simplest Query Language That Could Possibly Work. In Fuhr et al. [68], pages 167–175.

[160] P. O'Neil, E. O'Neil, S. Pal, I. Cseri, G. Schaller, and N. Westbury. ORDPATHs: Insert-Friendly XML Node Labels. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 903–908, New York, NY, USA, 2004. ACM Press.

[161] N. Orio. Music Retrieval: A Tutorial and Review. *Foundations and Trends in Information Retrieval*, 1(2):1–90, November 2006.

[162] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford Digital Library Technologies Project, 1998.

[163] S. Pal. XML Retrieval: A Survey. Technical Report TR/ISI/CVPR/IR07-01, CVPR, June 2007.

[164] J. Paoli, T. Bray, E. Maler, C. M. Sperberg-McQueen, and F. Yergeau. Extensible Markup Language (XML) 1.0 (Fourth Edition). W3C recommendation, W3C, Aug. 2006. http://www.w3.org/TR/2006/REC-xml-20060816.

[165] S. Park, W. W. Chu, J. Yoon, and J. Won. Similarity Search of Time-Warped Subsequences via a Suffix Tree. *Inf. Syst.*, 28(7):867–883, 2003.

[166] F. Pereira and R. Koenen. MPEG-7: A Standard for Multimedia Content Description. *Int. J. Image Graphics*, 1(3):527–546, 2001.

[167] P. Peterlongo, L. Noé, D. Lavenier, G. Georges, J. Jacques, G. Kucherov, and M. Giraud. Protein Similarity Search with Subset Seeds on a Dedicated Reconfigurable Hardware. In *Parallel Bio-Computing, Workshop on Parallel Computational Biology*, Gdansk, Poland, September 2007.

[168] K. Pinel-Sauvagnat and M. Boughanem. A survey on XML Focussed Component Retrieval. In *Large-Scale Semantic Access to Content (Text, Image, Video and Sound) (RIAO), Pittsburgh, USA, 30/05/07-01/06/07*, http://www.le-cid.org, June 2007. Centre de hautes études internationales d'informatique documentaire (C.I.D.).

[169] B. Piwowarski, G.-E. Faure, and P. Gallinari. Bayesian Networks and INEX. In Fuhr et al. [64], pages 149–154.

[170] B. Piwowarski and P. Gallinari. A Bayesian Framework for XML Information Retrieval: Searching and Learning with the INEX Collection. *Inf. Retr.*, 8(4):655–681, 2005.

[171] B. Piwowarski, H.-T. Vu, and P. Gallinari. Bayesian Networks and INEX'03. In N. Fuhr, M. Lalmas, and S. Malik, editors, *INitiative for the Evaluation of XML Retrieval (INEX). Proceedings of the Second INEX Workshop*, Dagstuhl, Germany, Dec. 2003.

[172] J. M. Ponte and W. B. Croft. A Language Modeling Approach to Information Retrieval. In *SIGIR* [1], pages 275–281.

[173] E. Popovici, P.-F. Marteau, and G. Ménier. Information Retrieval of Sequential Data in Heterogeneous XML Databases. In *Adaptive Multimedia Retrieval: User, Context, and Feedback: Third International Workshop, AMR 2005, Glasgow, UK, July 28-29, 2005, Revised Selected Papers*, volume 3877 of *Lecture Notes in Computer Science*, pages 236–250. Springer-Verlag, 2006.

[174] E. Popovici, P.-F. Marteau, and G. Ménier. An Effective Method for Finding Best Entry Points in Semi-Structured Documents. In *SIGIR '07: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 851–852, New York, NY, USA, 2007. ACM Press.

[175] E. Popovici, G. Ménier, and P.-F. Marteau. Recherche approchée d'information dans une base de documents semi-structurés : une application ReMIX. In *Panorama des Recherches Incitatives en STIC (PaRISTIC)*, LaBRI, Bordeaux, France, November 2005.

[176] E. Popovici, G. Ménier, and P.-F. Marteau. Recherche approchée d'information dans une base de documents semi-structurés. In *3éme Conférence en Recherche d'Informations et Applications (CORIA'06)*, pages 53–64, Lyon, France, Mars 2006.

[177] E. Popovici, G. Ménier, and P.-F. Marteau. SIRIUS: A Lightweight XML Indexing and Approximate Search System at INEX 2005. In Fuhr et al. [69], pages 321–335.

[178] E. Popovici, G. Ménier, and P.-F. Marteau. Interprétation vague des contraintes structurelles pour la RI dans des corpus de documents XML - Évaluation d'une méthode approchée de RI structurée. *Document Numérique*, 10(1):63–88, 2007. Special Issue on "Recherche d'information dans les documents structurés".

[179] E. Popovici, G. Ménier, and P.-F. Marteau. SIRIUS XML IR System at INEX 2006: Approximate Matching of Structure and Textual Content. In Fuhr et al. [71], pages 185–199.

[180] M. F. Porter. An Algorithm for Suffix Stripping. *Program*, 13(3):130–137, 1980.

[181] E. Rahm and P. A. Bernstein. A Survey of Approaches to Automatic Schema Matching. *VLDB J.*, 10(4):334–350, 2001.

[182] F. Raimbault and D. Lavenier. Le projet ReMIX et ses applications en génomique. In *Journées de l'Action Spécifique "Indexation de texte et découverte de motifs" (ASIM), Réunion thématique*, Faculté des Sciences, Nantes, 27-28 mai 2004.

[183] G. Ramírez, T. Westerveld, and A. P. de Vries. Structural Features in Content Oriented XML Retrieval. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 291–292, New York, NY, USA, 2005. ACM Press.

[184] G. Ramírez, T. Westerveld, and A. P. de Vries. Using Structural Relationships for Focused XML Retrieval. In *Proceedings of the Seventh International Conference on Flexible Query Answering Systems (FQAS 2006)*. Springer, 2006.

[185] P. Rao and B. Moon. PRIX: Indexing And Querying XML Using Prüfer Sequences. page 288, 2004.

[186] J. Reid, M. Lalmas, K. Finesilver, and M. Hertzum. Best Entry Points for Structured Document Retrieval: Part I: Characteristics. *Inf. Process. Manage.*, 42(1):74–88, 2006.

[187] J. Reid, M. Lalmas, K. Finesilver, and M. Hertzum. Best Entry Points for Structured Document Retrieval: Part II: Types, Usage and Effectiveness. *Inf. Process. Manage.*, 42(1):89–105, 2006.

[188] J. Revault. Propagation of Pertinence Indicator using Distance Models. In *The 9th International Conference on Information Processing and Magement of Uncertainty in Knowledge-Based Systems (IPMU)*. IEEE, 1-5 July 2002.

[189] E. Riedel, C. Faloutsos, G. A. Gibson, and D. Nagle. Active Disks for Large-Scale Data Processing. *Computer*, 34(6):68–74, 2001.

[190] S. Robertson and K. S. Jones. Relevance Weighting of Search Terms. *Journal of the American Society for Information Science*, 27:129–146, 1977.

[191] A. Robinson. "XML's and DTD's for Biology", An XML Workshop for Biologists and Bioinformaticians. http://industry.ebi.ac.uk/ alan/XMLWorkshop/, 2004.

[192] T. Roelleke and A. P. de Vries, editors. *Proceedings of the first SIGIR Workshop on the Integration of Information Retrieval and Databases (WIRD'04)*, Sheffield, UK, 2004.

[193] M. Rys and S. Buxton. XQuery and XPath Full-Text Requirements. W3C working draft, W3C, May 2003. http://www.w3.org/TR/2003/WD-xquery-full-text-requirements-20030502/.

[194] G. Salton, J. Allan, and C. Buckley. Approaches to Passage Retrieval in Full Text Information Systems. In *SIGIR '93: Proceedings of the 16th annual international ACM SIGIR conference on research and development in information retrieval*, pages 49–58, New York, NY, USA, 1993. ACM Press.

[195] G. Salton and C. Buckley. Term Weighting Approaches in Automatic Text Retrieval. Technical report, Ithaca, NY, USA, 1987.

[196] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.

[197] G. Salton, A. Wong, and C. S. Yang. A Vector Space Model for Automatic Indexing. *Commun. ACM*, 18(11):613–620, November 1975.

[198] K. Sauvagnat. *Modèle flexible pour la Recherche d'Information dans des corpus de documents semi-structurés*. PhD thesis, Université Paul Sabatier de Toulouse, France, Juin 2005.

[199] K. Sauvagnat, M. Boughanem, and C. Chrisment. Searching XML Documents Using Relevance Propagation. In A. Apostolico and M. Melucci, editors, *SPIRE*, volume 3246 of *Lecture Notes in Computer Science*, pages 242–254. Springer, 2004.

[200] K. Sauvagnat, M. Boughanem, and C. Chrisment. Why Using Structural Hints in XML Retrieval? In H. L. Larsen, G. Pasi, D. O. Arroyo, T. Andreasen, and H. Christiansen, editors, *FQAS*, volume 4027 of *Lecture Notes in Computer Science*, pages 197–209. Springer, 2006.

[201] K. Sauvagnat, L. Hlaoua, and M. Boughanem. XFIRM at INEX 2005: Ad-Hoc and Relevance Feedback Tracks. In Fuhr et al. [69], pages 88–103.

[202] T. Schlieder. *Fast Similarity Search in XML Data*. PhD thesis, Freien Universität Berlin, Germany, December 2002.

[203] T. Schlieder and H. Meuss. Querying and Ranking XML Documents. *Journal of the American Society for Information Science and Technology (JASIST)*, 53(6):489–503, 2002.

[204] G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

[205] D. Shin, H. Jang, and H. Jin. BUS: An Effective Indexing and Retrieval Scheme in Structured Documents. In *DL '98: Proceedings of the third ACM conference on Digital libraries*, pages 235–243, New York, NY, USA, 1998. ACM Press.

[206] B. Sigurbjörnsson. *Focused Information Access using XML Element Retrieval*. PhD thesis, University of Amsterdam, 2006.

[207] B. Sigurbjörnsson and J. Kamps. The Effect of Structured Queries and Selective Indexing on XML Retrieval. In Fuhr et al. [69], pages 104–118.

[208] B. Sigurbjörnsson, J. Kamps, and M. de Rijke. An Element-Based Approach to XML Retrieval. In Fuhr et al. [68].

[209] B. Sigurbjörnsson, J. Kamps, and M. de Rijke. Focused Access to Wikipedia. In *Proceedings of the 6th Dutch-Belgian Information Retrieval Workshop (DIR 2006)*, 2006.

[210] B. Sigurbjörnsson, A. Trotman, S. Geva, M. Lalmas, B. Larsen, and S. Malik. INEX 2005 Guidelines for Topic Development. 2005.

[211] M. Springmann. A Novel Approach for Compound Document Matching. *IEEE TCDL Bulletin*, 2(2), 2006.

[212] K.-C. Tai. The Tree-to-Tree Correction Problem. *J. ACM*, 26(3):422–433, 1979.

[213] X. Tannier. Dealing with XML structure through "Reading Contexts". Technical Report 2005-400-007, Ecole Nationale Supérieure des Mines de Saint-Etienne, apr 2005.

[214] X. Tannier. *Extraction et recherche d'information en langage naturel dans les documents semi-structurés*. PhD thesis, Ecole Nationale Supérieure des Mines de Saint-Etienne, 2006.

[215] X. Tannier, J.-J. Girardot, and M. Mathieu. Classifying XML Tags through "Reading Contexts". In P. R. King, editor, *Proceedings of the 2005 ACM Symposium on Document Engineering*, pages 143–145, Bristol, United Kingdom, Nov. 2005. ACM Press, New York City, NY, USA.

[216] A. Theobald and G. Weikum. The Index-Based XXL Search Engine for Querying XML Data with Relevance Ranking. In C. S. Jensen et al., editor, *8th Int. Conference on Extending Database Technology*, volume 2287 of *Lecture Notes in Computer Science*, pages 477–495. Springer, 2002.

[217] M. Theobald, A. Broschart, R. Schenkel, S. Solomon, and G. Weikum. TopX - AdHoc Track and Feedback Task. In Fuhr et al. [71], pages 233–242.

[218] M. Theobald, R. Schenkel, and G. Weikum. TopX and XXL at INEX 2005. In Fuhr et al. [69], pages 282–295.

[219] M. Theobald, R. Schenkel, and G. Weikum. The TopX DB&IR engine. In *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 1141–1143, New York, NY, USA, 2007. ACM Press.

[220] A. Tombros, S. Malik, and B. Larsen. Report on the INEX 2004 Interactive Track. *SIGIR Forum*, 39(1):43–49, 2005.

[221] A. Trotman and S. Geva. Passage Retrieval and Other XML-Retrieval Tasks. In *Proceedings of the SIGIR 2006 Workshop on XML Element Retrieval Methodology*, pages 43–50, 2006.

[222] A. Trotman, S. Geva, and J. Kamps. Report on the SIGIR 2007 Workshop on Focused Retrieval. *SIGIR Forum*, 41(2):97–103, 2007.

[223] A. Trotman and M. Lalmas. Strict and Vague Interpretation of XML-Retrieval Queries. In *SIGIR* [7], pages 709–710.

[224] A. Trotman and M. Lalmas. Why Structural Hints in Queries do not Help XML retrieval. In *SIGIR* [7], pages 711–712.

[225] A. Trotman, M. Lalmas, and N. Fuhr, editors. *Proceedings of the INEX 2005 Workshop on Element Retrieval Methodology*, Glasgow, Scotland, 30 July 2005.

[226] A. Trotman and B. Sigurbjörnsson. Narrowed Extended XPath I (NEXI). In Fuhr et al. [70], pages 16–40.

[227] E. Ukkonen. On-Line Construction of Suffix Trees. *Algorithmica*, 14(3):249–260, 1995.

[228] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, U.K., 1979.

[229] R. van Zwol. B^3-SDR and Effective Use of Structural Hints. In Fuhr et al. [69], pages 146–160.

[230] R. van Zwol. Multimedia Strategies for B^3-SDR, Based on Principal Component Analysis. In Fuhr et al. [69], pages 540–553.

[231] R. van Zwol, J. Baas, H. van Oostendorp, and F. Wiering. Query Formulation for XML Retrieval with Bricks. In *Proceedings of the INEX 2005 Workshop on Element Retrieval Methodology*, Glasgow, Scotland, July 2005. University of Glasgow, Glasgow, Scotland.

[232] R. van Zwol, G. Kazai, and M. Lalmas. INEX 2005 Multimedia Track. In Fuhr et al. [69], pages 497–510.

[233] R. van Zwol and T. van Loosbroek. Effective Use of Semantic Structure in XML Retrieval. In G. Amati, C. Carpineto, and G. Romano, editors, *ECIR*, volume 4425 of *Lecture Notes in Computer Science*, pages 621–628. Springer, 2007.

[234] R. van Zwol and W. Weerkamp. XSee: Structure Xposed. In Fuhr et al. [71], pages 271–283.

[235] J.-N. Vittaut and P. Gallinari. Machine Learning Ranking and INEX'05. In Fuhr et al. [69], pages 336–343.

[236] J.-N. Vittaut and P. Gallinari. Supervised and Semi-supervised Machine Learning Ranking. In Fuhr et al. [71], pages 213–222.

[237] J.-N. Vittaut, B. Piwowarski, and P. Gallinari. An Algebra for Structured Queries in Bayesian Networks. In Fuhr et al. [70], pages 100–112.

[238] J. S. Vitter. External Memory Algorithms and Data Structures: Dealing with Massive Data. In *ACM Computing Surveys*, volume 33, pages 209–271, June 2001.

[239] R. A. Wagner and M. J. Fischer. The String-to-String Correction Problem. *J. ACM*, 21(1):168–173, 1974.

[240] H. Wang and X. Meng. On the Sequencing of Tree Structures for XML Indexing. In *ICDE '05: Proceedings of the 21st International Conference on Data Engineering (ICDE'05)*, pages 372–383, Washington, DC, USA, 2005. IEEE Computer Society.

[241] H. Wang, S. Park, W. Fan, and P. S. Yu. ViST: A Dynamic Index Method for Querying XML Data by Tree Structures. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 110–121, New York, NY, USA, 2003. ACM Press.

[242] B. West, R. D. Chamberlain, R. S. Indeck, and Q. Zhang. An FPGA-based Search Engine for Unstructured Database. In *Proc. of 2nd Workshop on Application Specific Processors*, 2003.

[243] T. Westerveld and R. van Zwol. The INEX 2006 Multimedia Track. In Fuhr et al. [71], pages 331–344.

[244] R. Wilkinson. Effective Retrieval of Structured Documents. In *Research and Development in Information Retrieval*, pages 311–317, 1994.

[245] I. H. Witten, A. Moffat, and T. C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images, second edition*. Morgan Kaufmann Publishing, May 1999.

[246] F. G. S. S. W. Wong, R.K.; Lam. An XML Repository for Molecular Sequence Data. *Bio-Informatics and Biomedical Engineering, 2000. Proceedings. IEEE International Symposium on*, pages 35–42, 2000.

[247] A. Woodley and S. Geva. NLPX at INEX 2006. In Fuhr et al. [71], pages 302–311.

[248] B.-K. Yi, H. V. Jagadish, and C. Faloutsos. Efficient Retrieval of Similar Time Sequences Under Time Warping. In *ICDE '98: Proceedings of the Fourteenth International Conference on Data Engineering*, pages 201–208, Washington, DC, USA, 1998. IEEE Computer Society.

[249] S. Yoo. An XML Retrieval Model based on Structural Proximities. In Fuhr et al. [64], pages 125–132.

[250] L. Zadeh. Fuzzy Sets. *Information and Control*, 3(8):338–353, 1965.

[251] K. Zhang, R. Statman, and D. Shasha. On the Editing Distance between Unordered Labeled Trees. *Inf. Process. Lett.*, 42(3):133–139, 1992.

[252] Y. Zhu and D. Shasha. Warping Indexes with Envelope Transforms for Query by Humming. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 181–192, New York, NY, USA, 2003. ACM Press.

[253] G. K. Zipf. *Human Behaviour and the Principle of Least-Effort*. 1949.