Costantino Thanos
Francesca Borri
Leonardo Candela (Eds.)

LNCS 4877

Digital Libraries: Research and Development

First International DELOS Conference
Pisa, Italy, February 2007
Revised Selected Papers

Springer

Costantino Thanos Francesca Borri
Leonardo Candela (Eds.)

Digital Libraries: Research and Development

First International DELOS Conference
Pisa, Italy, February 13-14, 2007
Revised Selected Papers

Springer

Volume Editors

Costantino Thanos
Francesca Borri
Leonardo Candela
Consiglio Nazionale delle Richerche
Istituto di Scienza e Tecnologie dell'Informazione
Via Moruzzi, 1, 56124, Pisa, Italy
E-mail: {costantino.thanos,francesca.borri,leonardo.candela}@isti.cnr.it

Library of Congress Control Number: 2007940368

CR Subject Classification (1998): H.2, H.4, H.3, H.2.4, H.5

LNCS Sublibrary: SL 3 – Information Systems and Application, incl. Internet/Web and HCI

ISSN 0302-9743
ISBN-10 3-540-77087-9 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-77087-9 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12199099 06/3180 5 4 3 2 1 0

Lecture Notes in Computer Science 4877

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Preface

Digital libraries represent the meeting point of a large number of technical areas within the field of informatics, i.e., information retrieval, document management, information systems, the Web, image processing, artificial intelligence, human – computer interaction, mass-storage systems, and others. Moreover, digital libraries draw upon other disciplines beyond informatics, such as library sciences, museum sciences, archives, sociology, psychology, etc. However, they constitute a relatively young scientific field, whose life spans roughly the last 15 years. During these years the DELOS Network of Excellence on Digital Libraries (`http://www.delos.info`) has represented a constant presence aiming to contribute to the consolidation of the field.

The activities of DELOS started many years ago, with the "DELOS Working Group" at the end of the 1990s, and the DELOS Thematic Network, under the Fifth Framework Program, from 2001 to 2003. Since the beginning, the main objective of DELOS has been to advance the state of the art in the field of digital libraries by coordinating the efforts of the major European research teams conducting activities in the major fields of interest.

Every year DELOS organizes the All-Tasks meeting, the annual appointment of the DELOS community where the scientific results achieved during the previous year are presented. Instead of the usual status reports from the various DELOS Tasks, in 2006 it was decided to issue a Call for Papers, soliciting papers reporting the scientific achievements of the DELOS members during 2006, and to organize a conference. The first DELOS conference was held during February 13–14, 2007, at the Grand Hotel Continental in Tirrenia, Pisa, Italy.

The conference represented a good coverage of the 27 research tasks in DELOS, with 38 papers being presented. In addition, two invited papers: "Semantic Digital Libraries" (John Mylopoulos, University of Toronto) and "Digital Libraries: From Proposals to Projects to Systems to Theory to Curricula" (Edward Fox, Virginia Tech) completed the program. The conference was open to the larger digital library community and not just to the DELOS partners. About 120 people attended , half of whom were not from DELOS partners. We believe that this is an indication of the increased interest in digital libraries, and the recognition that the DELOS research activities have played and are playing an important role in the European digital library scene.

This volume includes extended and revised versions of the papers presented at the DELOS Conference. We believe that it should be of interest to a broad audience potentially interested in the digital library research area. It has been structured into 10 sections, corresponding to the different sessions in which the

conference was structured, which in turn correspond to the major areas of research where DELOS has focussed its attention recently.

October 2007

Costantino Thanos
Francesca Borri
Leonardo Candela

Organization

General Chair

Costantino Thanos (Chair)	Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo" - Consiglio Nazionale delle Ricerche - Pisa, Italy

Scientific Committee

Vittore Casarosa	Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo" - Consiglio Nazionale delle Ricerche - Pisa, Italy
Tiziana Catarci	Dipartimento di Informatica e Sistemistica "A. Ruberti" - Università di Roma "La Sapienza" - Rome, Italy
Stavros Christodoulakis	Laboratory of Distributed Multimedia Information System - Technical University of Crete - Crete, Greece
Alberto del Bimbo	Dipartimento di Sistemi e Informatica - Facoltà di Ingegneria University of degli Studi di Firenze - Florence, Italy
Norbert Fuhr	Department of Computational and Cognitive Sciences - Faculty of Engineering Sciences of the University of Duisburg-Essen - Duisburg, Germany
Yannis Ioannidis	Department of Informatics - National and Kapodistrian University of Athens - Athens, Greece
Bruno Le Dantec	The European Research Consortium for Informatics and Mathematics - Sophia Antipolis, France
Liz Lyon	UKOLN - University of Bath - Bath, UK
Seamus Ross	Humanities Advanced Technology and Information Institute - University of Glasgow - Glasgow, UK
Hans-Jörg Schek	Database & Information Systems Group - Universität Konstanz - Konstanz, Germany

Organizing Committee

Francesca Borri	Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo" - Consiglio Nazionale delle Ricerche - Pisa, Italy
Alessandro Launaro	Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo" - Consiglio Nazionale delle Ricerche - Pisa, Italy

Table of Contents

Similarity Search

Architectures

Personalization

Interoperability

Evaluation

Miscellaneous

Preservation

Video Data Management

3D Objects

Peer to Peer

MESSIF: Metric Similarity Search Implementation Framework*

Michal Batko, David Novak, and Pavel Zezula

Masaryk University, Brno, Czech Republic
{xbatko,xnovak8,zezula}@fi.muni.cz

Abstract. The similarity search has become a fundamental computational task in many applications. One of the mathematical models of the similarity – the metric space – has drawn attention of many researchers resulting in several sophisticated metric-indexing techniques. An important part of a research in this area is typically a prototype implementation and subsequent experimental evaluation of the proposed data structure. This paper describes an implementation framework called MESSIF that eases the task of building such prototypes. It provides a number of modules from basic storage management, over a wide support for distributed processing, to automatic collecting of performance statistics. Due to its open and modular design it is also easy to implement additional modules, if necessary. The MESSIF also offers several ready-to-use generic clients that allow to control and test the index structures.

1 Introduction

The mass usage of computer technology in a wide spectrum of human activities brings the need of effective searching of many novel data types. The traditional strict attribute-based search paradigm is not suitable for some of these types since they exhibit complex relationships and cannot be meaningfully classified and sorted according to simple attributes. A more suitable search model to be used in this case is *similarity search* based directly on the very data content.

This research topic has been recently addressed using various approaches. Some similarity search techniques are tailored to a specific data type and application, others are based on general data models and are applicable to a variety of data types. The *metric space* is a very general model of similarity which seems to be suitable for various data and which is the only model applicable to some important data types, e.g. in multimedia processing. This concept treats the dataset as unstructured objects together with a *distance* (or *dissimilarity*) measure computable for every pair of objects.

Number of researchers have recently focused on indexing and searching using the metric space model of data. The effort resulted in general indexing principles

* This research has been funded by the following projects: Network of Excellence on Digital Libraries (DELOS), national research project 1ET100300419, and Czech Science Foundation grant No. 102/05/H050.

C. Thanos, F. Borri, and L. Candela (Eds.): Digital Libraries: R&D, LNCS 4877, pp. 1–10, 2007.

and fundamental main-memory structures, continued with designs of disk-based structures, and also of distributed data-structures for efficient management of very large data collections. An important part of research in this area is typically a prototype implementation and subsequent experimental evaluation of the proposed data structure. Individual structures are often based on very similar underlying principles or even exploit some existing structures on lower levels. Therefore, the implementation calls for a uniform development platform that would support a straightforward reusability of code. Such a framework would also simplify the experimental evaluation and make the comparison more fair.

This reasoning led us to a development of MESSIF – The Metric Similarity Search Implementation Framework, which brings the above mentioned benefits. It is a purely modular system providing a basic support for indexing of metric spaces, for building both centralized and distributed data structures and for automatic measurement and collecting of various statistics.

The rest of the paper maps individual MESSIF components from basic management of metric data in Sections 2 and 3, over the support for the distributed processing in Section 4, to the user interfaces in Section 5. Description of each area which MESSIF supports is subdivided into three parts – the theoretical background, specific assignment for the framework (MESSIF Specification), and description of the currently available modules which provide the required functionality (MESSIF Modules). The architecture of the framework is completely open – new modules can be integrated into the system in a straightforward way.

2 Metric Space

The metric space is defined as a pair $\mathcal{M} = (\mathcal{D}, d)$, where $\mathcal{D}$ is the domain of objects and d is the total distance function $d : \mathcal{D} \times \mathcal{D} \longrightarrow \mathbb{R}$ satisfying the following conditions for all objects $x, y, z \in \mathcal{D}$: $d(x, y) \geq 0$, $d(x, y) = 0$ iff $x = y$ (*non-negativity*), $d(x, y) = d(y, x)$ (*symmetry*), and $d(x, z) \leq d(x, y) + d(y, z)$ (*triangle inequality*). No additional information about the objects' internal structure or properties are required. For any algorithm, the function d is a black-box that simply measures the (dis)similarity of any two objects and the algorithm can rely only on the four metric postulates above.

MESSIF Specification. Our implementation framework is designed to work with a generic metric space objects. The internal structure of the objects is hidden and not used in any way except for the purposes of evaluation of the metric function. In particular, every class of objects contains an implementation of the metric function applicable to the class data.

For the purposes of quick addressing, every object is automatically assigned a unique identifier *OID*. Since the metric objects are sometimes only simplified representations of real objects (e.g. a color histogram of an image), the objects also contain a URI locator address pointing to the original object – a web address of an image for instance.

MESSIF Modules. Currently, there are two basic data types with different metric functions: **Vectors** with L_p metric function, quadratic form distance and some functions from MPEG7 standard; and **Strings** with (weighted) edit distance and protein distance functions.

2.1 Collections and Queries

Let us have a collection of objects $\mathcal{X} \subseteq \mathcal{D}$ that form the database. This collection is dynamic – it can grow as new objects $o \in \mathcal{D}$ are inserted and it can shrink by deletions. Our task is to evaluate queries over such a database, i.e. select objects from the collection that meet some specified similarity criteria. There are several types of similarity queries, but the two basic ones are the range query $\mathbf{Range}(q, r)$ and the k-nearest neighbors query $\mathbf{kNN}(q, k)$.

Given an object $q \in \mathcal{D}$ and a maximal search radius r, *range query* $\mathbf{Range}(q, r)$ selects a set $S_A \subseteq \mathcal{X}$ of indexed objects: $S_A = \{x \in \mathcal{X} \mid d(q, x) \leq r\}$.

Given an object $q \in \mathcal{D}$ and an integer $k \geq 1$, *k-nearest neighbors query* $\mathbf{kNN}(q, k)$ retrieves a set $S_A \subseteq \mathcal{X} : |S_A| = k, \forall x \in S_A, \forall y \in \mathcal{X} \setminus S_A : d(q, x) \leq d(q, y)$.

MESSIF Specification. In MESSIF, we introduce concept of *operations* to encapsulate manipulations with a collection. An operation can either modify the collection – insert or delete objects – or retrieve particular objects from it. Every operation carries the necessary information for its execution (e.g. an object to be inserted) and after its successful evaluation on the collection it provides the results (e.g. a list of objects matching a range query). If the operation is a query, it also provides an implementation of its basic evaluation algorithm – the *sequential scan*. It is a straightforward application of the particular query definition: given a collection of objects, the operation inspect them one by one updating the result according to that particular query instance.

MESSIF Modules. At present time, MESSIF supports **insert** and **delete operations** that allow addition or removal of objects from collections. To retrieve similar objects, the basic metric-space **range**, **kNN** and **incremental kNN query operations** are available.

3 Metric Data Management

We have explained the concept of the metric-based similarity search. In this section, we will focus on efficient management and searching of metric data collections. So far, we can use the aforementioned framework modules to design a primitive data structure – it would execute the sequential scan implementation of a query on the whole collection of generic metric space objects. This works for small and static data sets, but when the data is dynamic and its volume can grow, more sophisticated effectiveness-aimed structures are needed. The framework offers additional modules to simplify the task of implementing

such structures – namely the data management support, reference objects choosing (including partitioning) and the encapsulation envelope for algorithms that provides support for operation execution.

A vital part of every implementation is its performance assessment. Without any additional effort required, the framework automatically gathers many statistic values from the summarizing information about the whole structure to the details about local operation execution. In addition, every structure can define its own statistics, which can take advantage of other framework modules.

3.1 Storing the Collections

Above, we have defined the collection as the finite subset of the metric domain $\mathcal{X} \subseteq \mathcal{D}$. Practically, the collection is any list of objects of arbitrary length, which is stored somewhere, e.g. the result of any query is a collection too. Moreover, a union of two collections is also a collection and also its subset is a collection.

MESSIF Specification. The collections of objects can be stored in data areas called *buckets*. A bucket represents a metric space partition or it is used just as a generic object storage. The bucket provides methods for inserting one or more objects, deleting them, retrieving all objects or just a particular one (providing its *OID*). It also has a method for evaluating queries, which pushes all objects from the bucket to the sequential scan implementation of the respective query. Every bucket is also automatically assigned a unique identifier *BID* used for addressing the bucket. An example of a bucket is shown in Figure 1b. The buckets have

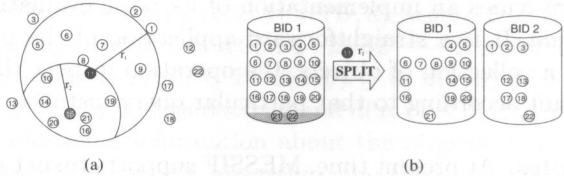

Fig. 1. Ball partitioning (a) and a bucket split (b)

usually limited capacity and MESSIF offers methods for splitting them if they overflow as depicted by the figure.

MESSIF Modules. To physically store objects, MESSIF offers **main memory** and **disk storage buckets**. The former is implemented as a linked list of objects while the latter uses block organization with a cached directory.

3.2 Partitioning the Collections

As the data volume grows, the time needed to go through all objects becomes unacceptable. Thus, we need to partition the data and access only the relevant

partitions at query time. To do this in a generic metric space, we need to select some objects – we call them *pivots* – and using the distance between the pivots and the objects, we divide the collection. The two basic partitioning principles are called the *ball partitioning* and the *generalized hyperplane partitioning* [1] and they can divide a set of objects into two parts – see an example of ball partitioning in Figure 1a. Since the resulting partitions can be still too large, the partitioning can be applied recursively until all the partitions are small enough.

At query time, the metric's triangular inequality property is exploited to avoid accessing some partitions completely. All the remaining partitions are searched by the sequential scan. Even then, some distance-function evaluations can be avoided provided we have stored some distances computed during object insertion. We usually refer to this technique as the *pivot filtering* [2].

MESSIF Specification. One of the issues in the metric-space partitioning is the selection of pivots, since it strongly affects the performance of the query evaluation. There are several techniques [1] that suggests how to do the job and the framework provides a generic interface allowing to choose an arbitrary number of pivots from a particular collection (usually a bucket or a set of buckets). These pivots are usually selected so that effectiveness of a specific partitioning or filtering is maximized.

MESSIF Modules. Automatic selection of reference objects can be currently done by **random**, **incremental** or **on-fly pivot choosers**. The first select pivots randomly while the second uses a sophisticated and good but time-consuming method. The third is a low-cost chooser with slightly worse results.

3.3 Metric Index Structures

The previous sections provide the background necessary for building an efficient metric index structure. We have the metric space objects with the distance function abstraction, we can process and store dynamic collections of objects using operations and we have tools for partitioning the space into smaller parts. Thus, to implement a working metric index structure we only need to put all these things together. Practically all algorithms proposed in the literature, see for example surveys [3,1], can be easily built using MESSIF.

MESSIF Specification. The building of an index technique in MESSIF means to implement the necessary internal structures (e.g. the navigation tree) and create the operation evaluation algorithms. Since the buckets can evaluate operations themselves, the index must only pick the correct buckets according to the technique used and the actual internal state of the index. A MESSIF internal mechanism also automatically detect the operations implemented by an algorithm (the algorithms do not necessarily implement available operations) and also supports their parallel processing in threads.

To demonstrate the simplicity of the implementation, we provide an example of a basic Vantage Point Tree (VPT) algorithm [1]. The structure builds a binary

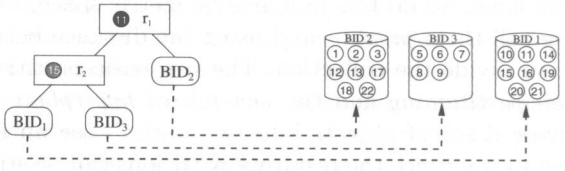

Fig. 2. Example of Vantage Point Tree structure

tree (see Figure 2), where every internal node of the tree divides the indexed data into two partitions – specifically, the ball-partitioning depicted in Figure 1a is used – and objects are stored in leaves. In MESSIF, we need to implement the inner nodes, i.e. a data structure holding a pivot and a radius. Leaf nodes are the MESSIF buckets, so no additional implementation effort is needed. Then, the insert and range query operations are implemented, but this only involves a simple condition-based traversal of the inner tree nodes. Once we reach the leaf nodes, the MESSIF bucket's processing takes over and provides the results.

MESSIF Modules. Several centralized metric indexing algorithms are implemented using MESSIF: **M-Tree** [4], **D-Index** [5], **aD-Index** and **VPT**, that can serve as an implementation tutorial.

3.4 Performance Measurement and Statistics

We have described the potential and the building blocks provided by the framework for creating index structures. However, essential part of every index is the performance statistics gathering. Statistics allow either automatic or manual tuning of the index and that can also serve during the operation-cost estimation (e.g. for a query optimizer). In the metric space, computation of the distances can be quite time demanding. Therefore, the time necessary to complete a query can vary significantly and it is also not comparable between different metric spaces. Thus, not only the time statistics should be gathered, but also the distance computations of various operations should be counted.

MESSIF Specification. Framework provides an automatic collecting of various statistics during the lifetime of an index structure – no additional implementation effort is needed. Any other statistics required by a particular index structure can be easily added. However, their querying interface is the same as for the automatic ones and they are accessible in the same way.

Specifically, every MESSIF module contains several global statistical measures. These are usually counters that are incremented whenever a certain condition occurs. For example, the *distance computations* counter is incremented when a metric function is evaluated. Moreover, other statistics can be based on the already defined ones – they can bind to an existing measure and then they will be updated every time the parent statistic is modified.