Information Systems 45 (2014) 37–47

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys

Analyzing and dynamically indexing the query set

Juan Manuel Barrios [a,b,*,1], Benjamin Bustos [b,2], Tomáš Skopal [c,3]

[a] ORAND S.A., Chile
[b] PRISMA, Department of Computer Science, University of Chile, Chile
[c] SIRET Research Group, Faculty of Mathematics and Physics, Charles University in Prague, Czech Republic

ARTICLE INFO

Available online 13 June 2013

Keywords:
Similarity search
Metric indexing
Multimedia information retrieval
Content-based multimedia retrieval

ABSTRACT

Most of the current metric indexes focus on indexing the collection of reference. In this work we study the problem of indexing the query set by exploiting some property that query objects may have. Thereafter, we present the Snake Table, which is an index structure designed for supporting streams of *k*-NN searches within a content-based similarity search framework. The index is created and updated in the online phase while resolving the queries, thus it does not need a preprocessing step. This index is intended to be used when the stream of query objects fits a snake distribution, that is, when the distance between two consecutive query objects is small. In particular, this kind of distribution is present in content-based video retrieval systems, image classification based on local descriptors, rotation-invariant shape matching, and others. We show that the Snake Table improves the efficiency of *k*-NN searches in these systems, avoiding the building of a static index in the offline phase.

1. Introduction

In the metric approach for similarity search, the most common search methods require an example object as query. As result, the search system returns the similar objects to the query. For example, in a *k*-NN query, the search returns the *k* closest objects to the query in the data collection. For improving the efficiency of the search, the standard approach is to preprocess the data collection for building a metric access method. While the preprocessing cost for building the index may be high, this cost is amortized during the processing of the queries. Normally, each query is processed in an isolated way with respect to previous or future queries. However in some cases, it is possible to improve the efficiency by taking advantage of inherent properties of the stream of queries.

Among the possible properties of a stream of queries, an interesting one is when consecutive queries are similar to each other. This property naturally arises in applications like similarity search for videos [1], where consecutive frames of a video may be used as query object. If the query frames are taken from the same shot, it follows that consecutive queries are similar to each other. Another application is time series for shape retrieval [2], where consecutive queries correspond to the same time series but shifted according to the temporal dimension. This results in consecutive queries that are also similar. Therefore, researching techniques that exploit the similarity between query objects to increase the efficiency becomes relevant, as it may have a large impact in the aforementioned applications.

In this paper, we study the approach of preprocessing the query set and indexing it dynamically. Specifically, we present the Snake Table, which is a dynamic indexing

* Corresponding author at: ORAND S.A. Tel.: +56 222474691.
E-mail addresses: juan.barrios@orand.cl, jbarrios@dcc.uchile.cl (J.M. Barrios), bebustos@dcc.uchile.cl (B. Bustos), skopal@ksi.mff.cuni.cz (T. Skopal).

[1] This research has been supported by CONICYT Project PAI-78120426.
[2] This research has been supported by FONDEF Project D09I1185.
[3] This research has been supported in part by Czech Science Foundation project GA CR 202/11/0968.

0306-4379/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.is.2013.05.010

structure designed for supporting streams of k-NN searches. Unlike most of the metric access methods, the Snake Table is short-lived and query-object oriented. The index is intended to be used when the stream of query objects fits a "snake distribution", which we define formally in this work. We show that the snake distribution for query objects arises naturally in some problems (like in the example for video similarity search) and also in other problems can be "artificially" forced by reordering the query set. We show experimentally that processing a stream of queries with snake distribution using the Snake Table can outperform a static metric access method.

An existing indexing structure with similar objectives and properties is called the D-file [3]. In this work, we show that the D-file suffers from high internal realtime complexity making it unviable to use it in metric spaces with computationally inexpensive (i.e., fast) distance functions (like Manhattan distance or Euclidean distance). We compare the Snake Table with D-file and LAESA index, showing that Snake Table achieves the best performance when the data follows a snake distribution.

The structure of the paper is as follows. Section 2 gives a background of metric spaces and efficiency issues. Section 3 analyses the properties of a query set and presents a taxonomy for query sets. Section 4 reviews the related work, focusing on the techniques that preprocess and index the query set. Section 5 defines a snake distribution and presents the Snake Table. Section 6 evaluates the performance achieved by indexing the query sets using different scenarios. Finally, Section 7 summarizes the contributions of this work.

2. Background

Let $\mathcal{M}=(\mathcal{D},d)$ be a metric space [4]. Given a collection $\mathcal{R}\subseteq\mathcal{D}$, and a query object $q\in\mathcal{D}$, a range search returns all the objects in $\mathcal{R}$ that are closer than a distance threshold ϵ to q, and a nearest neighbor search (k-NN) returns the k closest objects to q in $\mathcal{R}$.

For improving efficiency in metric spaces, Metric Access Methods (MAMs) [5] are index structures designed to efficiently perform similarity search queries. MAMs avoid a linear scan over the whole database by using the metric properties to save distance evaluations. Given the metric space $\mathcal{M}$, the object-pivot distance constraint [4] guarantees that $\forall a,b,p\in\mathcal{D}$:

$$|d(a,p)-d(p,b)|\le d(a,b)\le d(a,p)+d(p,b) \tag{1}$$

One index structure that uses pivots for indexing is the Approximating and Eliminating Search Algorithm (AESA) [6]. It first computes a matrix of distances between every pair of objects $x,y\in\mathcal{R}$. The structure is simply an $|\mathcal{R}|\times|\mathcal{R}|$ distance matrix. In fact, only a half of the matrix needs to be stored, due to symmetry of d. The main drawback of the AESA approach is the quadratic space of the matrix. Linear AESA (LAESA) [7] gets around this problem by selecting a set of pivots $\mathcal{P}\subseteq\mathcal{R}$. The distance between each pivot to every object is calculated and stored in a $|\mathcal{R}|\times|\mathcal{P}|$ distance matrix, also known as the *pivot table*. LAESA reduces the required space compared to AESA, however an algorithm for selecting a good set of pivots is required [8].

Given a query object q (not necessarily in $\mathcal{R}$), the similarity search algorithm first evaluates the distance $d(q,p)$ for each pivot $p\in\mathcal{P}$, then scans $\mathcal{R}$, and for each $r\in\mathcal{R}$ it evaluates the lower bound function $\mathrm{LB}_{\mathcal{P}}$:

$$\mathrm{LB}_{\mathcal{P}}(q,r)=\max_{p\in\mathcal{P}}\{|d(q,p)-d(r,p)|\} \tag{2}$$

Note that $\mathrm{LB}_{\mathcal{P}}$ can be evaluated efficiently because $d(q,p)$ is already calculated and $d(r,p)$ resides in the pivot table. In the case of range searches, if $\mathrm{LB}_{\mathcal{P}}(q,r)>\epsilon$ then r can be safely discarded because r cannot be part of the search result. In the case of k-NN searches, if $\mathrm{LB}_{\mathcal{P}}(q,r)\ge d(q,o^k)$ then r can be safely discarded, where o^k is the current kth nearest neighbor candidate to q. If r is not discarded, the actual distance $d(q,r)$ must be evaluated to decide whether or not r is part of the search result.

The efficiency of some MAM in $\mathcal{M}$ is related to: (1) the number of distance evaluations that are discarded when it performs a similarity search; and (2) the internal cost for deciding whether some distance can be discarded or not. A similarity search using any MAM will be faster than a linear scan when the time saved due to the discarded distances is greater than the time spent due to the internal cost. For example, in the case of LAESA, the internal cost for a similarity search comprises the evaluation of $d(q,p)$ for each pivot p in $\mathcal{P}$, and the evaluation of $\mathrm{LB}_{\mathcal{P}}(q,r)$ for each object r in $\mathcal{R}$, thus it increases linearly with $|\mathcal{P}|$. The amount of distances discarded by LAESA depends on the metric space itself and on the size and quality of $\mathcal{P}$.

In order to analyze the efficiency that any MAM can achieve in a collection $\mathcal{R}$, Chávez et al. [5] propose to analyze the histogram of distances of d. A histogram of distances is constructed by evaluating $d(a,b)$ for a random sample of objects $a,b\in\mathcal{R}$. The histogram of distances reveals information about the distribution of objects in $\mathcal{M}$. Given a histogram of distances for $\mathcal{M}$, the intrinsic dimensionality ρ is defined as $\rho(\mathcal{M})=\mu^2/2\sigma^2$, where μ and σ^2 are the mean and the variance of histogram of distances for $\mathcal{M}$. The intrinsic dimensionality estimates the efficiency that any MAM can achieve in $\mathcal{M}$, therefore it tries to quantify the difficulty in indexing a metric space. A histogram of distances with small variance (i.e., a high value of ρ) means that the distance between any two objects $d(a,b)$ with high probability will be near μ, thus the difference between any two distances with high probability will be a small value. In that case, for most of the pivots the lower bound from Eq. (1) will become ineffective at discarding objects. Increasing the number of pivots will improve the value of the lower bounds, however the internal cost of the MAM will also increase.

3. Streams of k-NN searches

MAMs can be classified as static or dynamic depending on how they manage the insertion or deletion of objects in $\mathcal{R}$ during the online phase. A dynamic MAM can update its structures to add or remove any object, hence it can remain online even for growing collections. Usually, the tree-based MAMs are dynamic, like the M-Tree [9]. A static MAM cannot manage large updates in its structures, thus after many modifications of $\mathcal{R}$ the whole indexing