Information Systems 52 (2015) 130–148

Contents lists available at ScienceDirect

Information Systems

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

Similarity sets: A new concept of sets to seamlessly handle similarity in database management systems

I.R.V. Pola *, R.L.F. Cordeiro, C. Traina, A.J.M. Traina

Department of Computer Science – Institute of Mathematics and Computer Science, Trabalhador São-Carlense Avenue, 400, University of São Paulo, São Carlos, Brazil

ARTICLE INFO

Article history:
Received 13 January 2014
Received in revised form
21 October 2014
Accepted 26 January 2015
Available online 13 February 2015

Keywords:
Metric space
Similarity sets
Independent dominating sets
Graphs

ABSTRACT

Besides integers and short character strings, current applications require storing photos, videos, genetic sequences, multidimensional time series, large texts and several other types of "complex" data elements. Comparing them by equality is a fundamental concept for the Relational Model, the data model currently employed in most Database Management Systems. However, evaluating the equality of complex elements is usually senseless, since exact match almost never happens. Instead, it is much more significant to evaluate their similarity. Consequently, the set theoretical concept that a set cannot include the same element twice also becomes senseless for data "sets" that must be handled by similarity, suggesting that a new, equivalent definition must take place for them. Targeting this problem, this paper introduces the new concept of "similarity sets", or *SimSets* for short. Specifically, our main contributions are: (i) we highlight the central properties of SimSets; (ii) we develop the basic algorithm required to create SimSets from metric datasets, ensuring that they always meet the required properties; (iii) we formally define unary and binary operations over SimSets; and (iv) we develop an efficient algorithm to perform these operations. To validate our proposals, the SimSet concept and the tools developed for them were employed over both real world and synthetic datasets. We report results on the adequacy, stability, speed and scalability of our algorithms with regard to increasing data sizes.

1. Introduction

Traditional Database Management Systems (DBMS) are severely dependent on the concept that a set never includes the same element twice. In fact, the Set Theory is the main mathematical foundation for most existing data models, including the Relational Model [1]. However, current applications are much more demanding on the DBMS, requiring it to handle increasingly complex data, such as images, audio, long texts, multidimensional time series, large graphs and genetic sequences.

Complex data elements must be compared by similarity, since it is generally senseless to use identity to compare them [2]. A typical example is the decision making process that physicians do when analyzing images of medical exams. Exact match provides very little help here, as it is highly unlikely that identical images exist, even within exams from the same patient. On the other hand, it is often important to retrieve related past cases to take advantage of their analysis and treatment, by searching for *similar* images from previous patients' exams. Thus, similarity search is

* Corresponding author.
E-mail addresses: ives@icmc.usp.br (I.R.V. Pola), robson@icmc.usp.br (R.L.F. Cordeiro), caetano@icmc.usp.br (C. Traina), agma@icmc.usp.br (A.J.M. Traina).

http://dx.doi.org/10.1016/j.is.2015.01.011
0306-4379/

almost mandatory, for this example and also for many others, such as in data mining [3], duplicate document detection [4], whether forecast and complex data analysis in general.

As exact match on pairs of complex elements seldom occurs or makes sense, the usual concept of "sets" also blurs for these data, suggesting that a new, equivalent concept must take place. With that in mind, the main questions we answer here are:

1. *What concept equivalent to "sets" is appropriate for complex data?*
2. *How to design novel operations and algorithms that allow it to be naturally embedded into existing DBMS?*

The problem stems from the need to compare complex data elements by similarity, as opposed to comparing them by equality, which is the standard strategy used in traditional data (i.e., data in numeric or in short character string domains). To support similarity comparisons, it is common to represent the dataset in a metric space. This space is formally defined as $M = \langle \mathbb{S}, d \rangle$, in which $\mathbb{S}$ is the data domain and $d: \mathbb{S} \times \mathbb{S} \to \mathbb{R}^+$ is a distance function that meets the following properties for any $s_1, s_2, s_3 \in \mathbb{S}$: Identity: $d(s_1, s_2) = 0 \to s_1 = s_2$; Non-negativity: $0 < d(s_1, s_2) < \infty$, $s_1 \neq s_2$; Symmetry: $d(s_1, s_2) = d(s_2, s_1)$ and Triangular inequality: $d(s_1, s_2) \leq d(s_1, s_3) + d(s_3, s_2)$.

Another fact distinguishes complex data even more from traditional data: ordering properties does not hold among complex data elements. One can say only that two elements are equal or different, as in a general case there is no rule that allows sorting the elements. As a consequence, the relational operators $<$, $\leq$, $>$ and $\geq$ cannot be used for comparisons. Moreover, since exact match rarely occurs or makes sense here, the identity-based operators $=$ and $\neq$ are also almost useless. Therefore, only similarity-based operations can be performed over complex data.

The main similarity-based comparison operations are the similarity range and k-nearest neighbor. In this paper we represent similarity-based selection using the same traditional notation of selections based on identity and relationship ordering [1], that is: $\sigma_{(S \theta s_q)} \mathsf{T}$, in which T is a relation, S is an attribute from T in domain $\mathbb{S}$, $s_q \in \mathbb{S}$ is a constant, called the query center, and θ is a comparison operator valid in $\mathbb{S}$. The similarity range selection uses $\theta = Rq(d, \xi)$ and retrieves the elements $s_i \in S$, such that $d(s_i, s_q) \leq \xi$. Likewise, the k-nearest neighbor selection uses $\theta = k\text{–}NN(d, k)$ and retrieves the k elements in S nearest to s_q.

This paper defines a concept equivalent to "sets" for complex data: the "similarity sets", or *SimSets* for short. In a nutshell, a SimSet is a set of complex elements without any two elements "too similar" to each other, mimicking the definition that a set of scalar elements does not contain the same element twice. Although intuitive, this concept brings some interesting issues, such as:

- Among many "too similar" elements, how to select the SimSet that best represents the original set?
- What properties hold for SimSets, and how to maintain them along with dataset modifications?

Table 1
Table of symbols.

Symbol	Description
$\mathsf{T}, \mathsf{T}_1, \mathsf{T}_2$	Database relations
$\mathbb{R}, \mathbb{S}, \mathbb{T}$	Data domain
R, S, T	Sets of elements, from $\mathbb{R}, \mathbb{S}, \mathbb{T}$ respectively
r_i, s_i, t_i	Elements in a set ($r_i \in R$, $s_i \in S$, $t_i \in T$)
σ	Relational selection operator
$\ddot{\sigma}$	Similarity selection operator
$\hat{\delta}$	Similarity extraction operator
ξ	Similarity threshold
d	Distance function (metric)
$\hat{=}^{\xi}$	Sufficiently similar operator
$\triangleq$	Equivalent expression
$\hat{\neq}$	Not equivalent expression
$\hat{R}^{\xi}, \hat{S}^{\xi}, \hat{T}^{\xi}$	ξ-Similarity-sets
$\hat{G}^{\xi}$	A ξ-similarity graph
$P\hat{S}^{\xi}(S)$	The ξ-similarity cover of a set S
λ	A extraction policy
$\hat{\bigcup}^{\hat{\lambda}}$	The similarity union operator
$\hat{\bigcap}^{\hat{\lambda}}$	The similarity intersection operator
$\hat{-}$	The similarity difference operator

- What operations are available for SimSets?
- How to handle SimSets on real data?

In this paper we formally define the concept of SimSets and develop basic resources to handle them in a database environment, as well as binary operators to manipulate SimSets. Specifically, our main contributions are as follows:

1. *Concept and properties*: We introduce the concept of SimSets, its terminology and its most important properties.
2. *Binary operations*: We define binary operations to be used with SimSets, such as union, intersection and difference. We also derive useful operations to handle SimSets in DBMS, such as insert, delete and query.
3. *Algorithms*: We propose the algorithm *Distinct* to extract SimSets from any metric dataset, and the algorithm *BinOp* to execute the proposed binary operations with SimSets. Both were carefully designed to be naturally embedded into existing DBMS.
4. *Evaluation on real and synthetic data*: We evaluate SimSets on three real world applications to show that they can improve analysis processes, besides providing a better conceptual underpinning for similarity search operations. Experiments on synthetic data are also reported to demonstrate that our algorithms are stable, fast and scalable to increasing dataset sizes.

The rest of this paper is organized as follows. Section 2 discusses our driving applications as well as some of the many other real world scenarios that may benefit from our SimSets. Section 3 reviews background concepts. Section 4 discusses related work. A precise definition of SimSets is given in Section 5, while Sections 6 and 7 respectively present algorithms to extract SimSets from metric datasets