Taking advantage of highly-correlated attributes in similarity queries with missing values

Lucas S. Rodrigues, Mirela T. Cazzolato,
Agma J. M. Traina and Caetano Traina Jr.

Institute of Mathematics and Computer Sciences
University of São Paulo (USP), São Carlos, Brazil
{lucas_rodrigues, mirelac}@usp.br, {agma, caetano}@icmc.usp.br

Abstract. Incompleteness harms the quality of content-based retrieval and analysis in similarity queries. Missing data are usually evaluated using exclusion and imputation methods to infer possible values to complete gaps. However, such approaches can introduce bias into data and lose useful information. Similarity queries cannot perform over incomplete complex tuples, since distance functions are undefined over missing values. We propose the SOLID approach to allow similarity queries in complex databases without the need neither of data imputation nor deletion. First, SOLID finds highly-correlated metric spaces. Then, SOLID uses a weighted distance function to search by similarity over tuples of complex objects using compatibility factors among metric spaces. Experimental results show that SOLID outperforms imputation methods with different missing rates. SOLID was up to 7.3% better than the competitors in quality when querying over incomplete tuples, reducing 16.42% the error of similarity searches over incomplete data, and being up to 30.8 times faster than the closest competitor.

Keywords: Missing Data · Similarity Search · Complex · Metric Spaces

1 Introduction

Advances in data collection and sharing have substantially increased the amount of available data in the last decades. Datasets of complex data such as images, time series, and audios have been the target of several studies [3, 11]. Similarity queries explore complex objects using low-level data representations and compare them using distance functions to retrieve the relevant objects by their content [3]. However, problems in data acquisition, recording, and management may lead to missing values in the data. As data quality poses significant challenges for content-based retrieval, analysis, and management, missing data is a relevant issue [10]. A commom approach to query over incomplete datasets is to remove the tuples or attributes with missing values. Applications can also maintain the attribute and ignore only the missing values [2], but removing tuples from the dataset may also discard relevant information.

Figure 1 shows an example of a database of complex attributes extracted from drawn characters, with missing values. Distance functions can not measure the similarity of incomplete tuples. Imputation approaches replace missing values following predetermined criteria, filling the data, for traditional analyses. Mean

imputation [4] is one of the most common approaches, replacing missing values with the corresponding dimension's average or mode. Other techniques include the K-Nearest Neighbor (kNN) imputation [2], which estimates missing values based on the most similar objects considering the existing dimensions, fuzzy sets, interpolation functions, or regression methods [5]. However, ignoring or imputing values can introduce bias in the data and harm similarity searches [10].

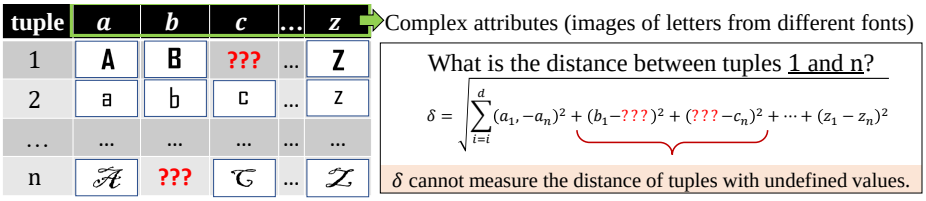

Fig. 1. Example of the problem of missing values in similarity searches.

In this work, we propose the `SOLID` method to retrieve the most similar tuples comparing complex attributes over incomplete databases. `SOLID` implements our `CorDiS` method to compute the correlation among metric spaces defined over complex attributes, and generates compatibility factors to yield more importance to the most correlated attributes. Experiments show that `SOLID` is three orders of magnitude better than the baseline approach when querying over incomplete tuples, being up to 7.3% better than the Decision Tree Imputation method, on average. In datasets with large amounts of missing values, `SOLID` reduced the error by 26.3% and 16.4% compared with the baseline approach and the closest competitor, respectively. Also, `SOLID` was faster than all competitors.

Paper outline. The remaining sections of this work are organized as follows: Section 2 describes the background and related work. Section 3 proposes the `SOLID` method. Section 4 shows the experiments. Section 5 concludes this work.

2 Background and Related Work

Database Management Systems (DBMS) compare scalar data using identity and order operators, but those are not adequate to compare complex data. Similarity-based comparisons of complex data rely on the data representation obtained by **Feature Extraction Methods (FEM)**, and object comparisons carried out by **Distance Functions (δ)**. Table 1 shows the symbols employed in this work.

FEMs extract discriminative features from complex data, often represented by numerical feature vectors. There are several FEMs dedicated to images, such as the ones defined by the MPEG-7 [9]. Distance functions (δ) measure the similarity between pairs of feature vectors, and output a real value R^+, where the smaller the distance, the greater the similarity. Euclidean and Manhattan are examples of widely used distance functions [11]. A distance function is called a metric when it meets the properties of metric spaces, following Definition 1.

Definition 1. ***Metric Space*** **($\mathbb{M}$).** *A Metric Space is a pair $\mathbb{M} = \langle \mathbb{S}, \delta \rangle$, where $\mathbb{S}$ is the domain of complex data and $\delta : \mathbb{R}^m \times \mathbb{R}^m$ is a distance function. A δ*

is a metric if, for any complex objects $s_i, s_j, s_k \in \mathbb{S}$, it meets the following properties: (i) ***Non-Negativity****: $\delta(s_i, s_j) \geqslant 0$; (ii)* ***Identity of the indiscernible****: $\delta(s_i, s_i) = 0 \Rightarrow s_i = s_j$; (iii)* ***Symmetry****: $\delta(s_i, s_j) = \delta(s_j, s_i)$; (iv)* ***Triangular Inequality****: $\delta(s_i, s_j) \leqslant \delta(s_i, s_k) + \delta(s_k, s_j)$.*

Table 1. Main symbols employed in this work.

Symbol	Description	Symbol	Description
$\mathbb{M}$	Metric space	$\mathbb{S}$	Domain of complex objects
$\mathfrak{M}$	Set of metric spaces	d	Dimensionality of $\mathcal{D}$
δ	Distance Function	ω	Compatibility factor of a metric space
s_i	Complex attribute	Ω	Set of Compatibility Factors
ϕ	Correlation function	ψ	SOLID's distance function
S	Feature vector set	γ	Threshold for the compatibility factors
$\mathcal{D}$	Dataset	Υ	Set of consolidated compatibility factors
n	Cardinality of $\mathcal{D}$	ϖ	Consolidated compatibility factors

Let $\mathcal{D}$ be a dataset of complex objects $S \in \mathbb{S}$, and δ be a metric distance function. The **Range** and ***k*-Nearest Neighbor** are the most employed similarity queries. Let s_q and s_i be two complex attributes in domain $\mathbb{S}$, where s_q is the query center. A Range Query (R_q) retrieves every element $s_i \in S$ whose similarity to s_q is less or equal than a similarity threshold ξ, that is $\delta(s_q, s_j) \leqslant \xi$. Given an amount k of objects and a query center s_q, a k-NN Query (Knn_q) retrieves the k objects $s_i \in S$ most similar to s_q measured by δ.

Traditional distance functions cannot measure the similarity between incomplete complex objects, drastically reducing both the available data and the efficiency of similarity queries. Many existing methods treat incompleteness based on data deletions or imputation [5]. The Mean Imputation is the simplest approach, which infers an average value of the available attributes in a tuple. Regression Imputation involves incorporating knowledge, such as data correlation, for inferring missing values [5]. The kNN Imputation searches for the k-nearest neighbors to the missing value, computing and inserting the average of the candidates into the missing spot [2]. Imputation methods based on Decision Trees Imputation partition the data based on its correlation, looking for the best candidates for each partition to infer values [7]. However, imputation methods may lead to biased results when it increases the rate of missing values [10].

3 The Proposed Method

SOLID (***S****earch* ***O****ver Correl****a****ted and* ***I****ncomplete* ***D****ata*) allows similarity queries over incomplete databases by taking advantage of *highly correlated search spaces*. The method does not discard nor replace missing values. Instead, it weights compatible metric spaces to give an approximation of the missing complex object location in each search space. We describe each step of SOLID next.

Defining Metric Spaces. SOLID extracts features from the d complex attributes in a dataset $\mathcal{D}$ with the chosen FEM. SOLID defines a metric space $\mathbb{M}_i$ for every complex attribute, *i.e* every image, using the corresponding distance function. Let $\mathfrak{M} = \mathbb{M}_1, \mathbb{M}_2, \cdots, \mathbb{M}_d$ be the set d of metric spaces defined

over the attributes of $\mathcal{D}$. We represent the ith metric space defined over $\mathcal{D}$ as $\mathbb{M}_i = \langle \mathbb{S}, \delta \rangle \in \mathfrak{M}$, $1 \leq i \leq d$, and δ is the corresponding distance function.

Measuring the Correlation. `SOLID` implements `CorDiS` (Algorithm 1) to map the correlation between pairs of metric spaces, working with a sample $\mathcal{D}'$ from $\mathcal{D}$ and the set of extracted features (Line 2). Function *GetSetOfAverageDistances* computes the average distances (Lines 3–4), comparing each object to every other element in the feature set (Lines 8–10), and returning the set of mean distances of each element in the feature set (Lines 7–12). After obtaining the distance of every object to all elements in $\mathcal{D}'$, `CorDiS` computes and returns the correlation $\mathcal{F}_{\text{corr}}$ between both metric spaces (Line 5–6).

Algorithm 1: `CorDiS` (***Cor****relation of* ***Di****stance* ***S****paces*)

Input : A pair of metric spaces ($< \mathbb{M}_i, \mathbb{M}_j >$), the distance function (δ), the correlation function (ϕ)
Output: The correlation map $\mathcal{F}_{\text{corr}}$

```
1  begin
       // Generate sets of average distances 𝔸_i and 𝔸_j
2      Let S_i and S_j be the feature vector sets from 𝕄_i and 𝕄_j;
3      𝔸_i ← GetSetOfAverageDistances(S_i, δ);
4      𝔸_j ← GetSetOfAverageDistances(S_j, δ);
5      F_corr ← φ(𝔸_i, 𝔸_j); // Compute the correlation
6      return F_corr;
7  Function GetSetOfAverageDistances(S, δ)
8      foreach object s_i in S do
9          foreach object s_j in S do
10             D_i ← D_i ∪ [ δ(s_i, s_j) ]; // Get distances to s_i
11         𝔸 ← 𝔸 ∪ MEAN(D_i); // Compute the mean distance
12     return 𝔸;
```

Generating Compatibility Factors. `SOLID`'s mapping step obtains a correlation matrix relating all metric spaces. `SOLID` normalizes each row of the matrix's correlation values, which corresponds to the compatibility factors of that attribute. The compatibility factor is given by Definition 2.

Definition 2. ***Compatibility Factor (ω).*** *Let a and b be two attributes of $\mathcal{D}$, such that $0 \leq a, b \leq d$. The compatibility factor $\omega_{a,b}$ is the complement to one of the normalized correlation considering the metric spaces $< \mathbb{M}_a, \mathbb{M}_b >$.*

Accordingly, Ω_a is the set of compatibility factors of attribute a over each attribute from $\mathcal{D}$, and $\omega_{a,b} \in \Omega_a$ is the compatibility factor of a over attribute b. A minimum threshold parameter γ allows removing low correlation values.

Querying over Incomplete Tuples. To compare a pair of tuples $< t_q, t_i >$, $1 \leq i \leq n$, `SOLID` considers that zero or more attributes in each tuple may have missing values. Algorithm 2 looks for the attributes with missing values in each tuple. It accumulates the compatibility factors ω corresponding to the missing dimensions in t_i (Lines 3 to 5), adding it to the consolidated set of factors Υ_i. The algorithm returns the set of consolidated compatibility values Υ_i of t_i (Line

7). Let v be a feature vector. SOLID compares each pair of tuples using the SOLID-dist distance ψ, according to Definition 3.

Definition 3. ***SOLID-dist*** *(ψ). Let $< v_{a,q}, v_{a,i} >$ be a pair of feature vectors from attribute a, $1 \leq a \leq d$. Let Υ_q, Υ_i be the sets of consolidated compatibility factors of t_q and t_i, respectively. The distance between $< t_q, t_i >$ is given by:*

$$\psi(t_q, t_i) = \sum_{a=1}^{d} \varpi_{q,a} \times \varpi_{i,a} \times \delta(v_{a,q}, v_{a,i}), \tag{1}$$

where $\varpi_{q,a} \in \Upsilon_q$ and $\varpi_{i,a} \in \Upsilon_i$ are the consolidated compatibility factors for a.

Algorithm 2: SOLID (Consolidated Compatibility Factors of a Tuple)

```
Input  : t_i: A tuple
Output: Υ: The set of consolidated compatibility factors of t_i.
1 begin
2     Let P_i be the set of missing attributes in t_i;
3     foreach attribute a in the t_i do
4         foreach missing attribute p in p_i do
5             W_p[a] += ω_{a,p};
6         Υ_i.add(W_p[a]);
7     return Υ_i;
```

4 Experimental Analysis

Material and Methods. We employed the four image datasets described in Table 2, composed of tuples of complex attributes, ensuring alignment of features when comparing tuples. We explored the Euclidean distance and the FEMs Local Binary Pattern, Edge Histogram, Haralick, Zernike, Color Layout, Scalable Color, Color Structure, Texture Spectrum, and Color Histogram. CorDiS ran over ten random combinations of FEMs, and choose those with the highest correlation sum as the features of every complex attribute. We employed the Pearson coefficient, since it presented higher correlation values than the Spearman coefficient. The source code, datasets and the detailed description of features are provided in the Git repository `github.com/lsrusp/SOLID-Method`.

Our competitors are: **Deletion** (baseline), which removes missing attributes; **Mean** Imputation; k**NN** Imputation [2]; kNN-**Regression** Imputation [5]; and the **Decision-Tree** Imputation [7]. We split the datasets into train and test: 70%/30% split proportion for DS-LibraGestures, and 50%/50% for the remaining datasets. The correlation threshold was $\gamma = 0.5$ for all experiments.

Quantitative Analysis. First, we analyzed *how well SOLID retrieves information from complete datasets using a query tuple with missing values.* We randomly placed missing values into the query center, with 10%, 20%, 30%, 40%, and 50% missing rates. We measured the query quality using the Jaccard coefficient between the Knn_q results posed over the complete tuples and the results of each approach. Figure 2 shows the results for different k values, where the

higher the values, the better. SOLID presented the best results in most cases, being up to 3 orders of magnitude better than the Deletion approach, and up to 7.3% better than the Decision Tree Imputation, on average. SOLID is better and faster at reducing the missingness impact when performing queries.

Table 2. Datasets used in the experiments

Dataset	Imgs	n	d	**Description**
DS-MSTSpine [8]	540	54	10	Lumbar muscles and vertebral bodies MRIs.
DS-HandPD [6]	594	66	9	Handwritten image to detect Parkinsons diseases.
DS-LibraGestures [1]	4800	120	40	Images of hand gestures.
DS-Letters [1]	15340	295	52	Font types of alphabetic letters.

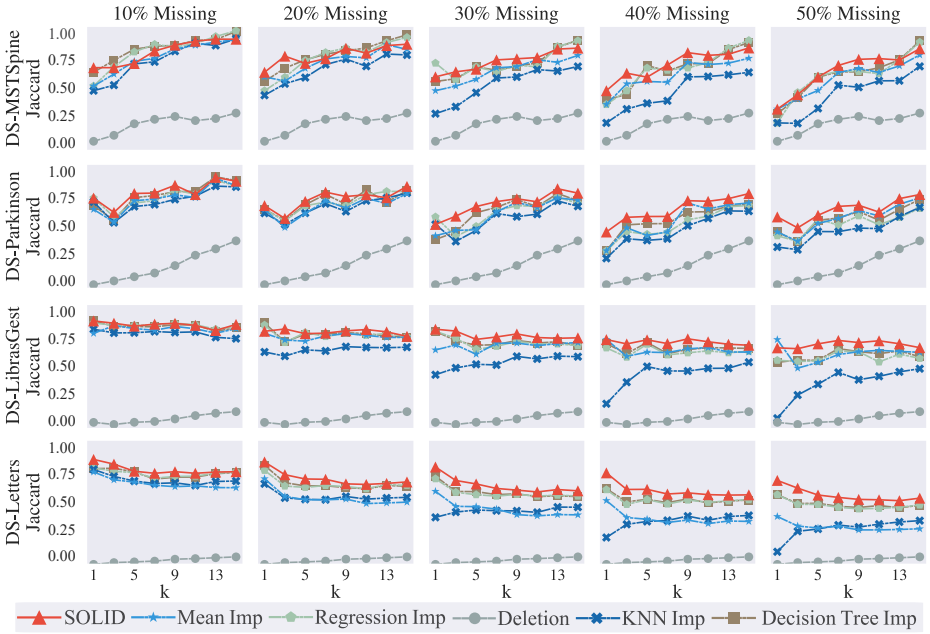

Fig. 2. Incomplete query tuple (**the higher, the better**): SOLID ties or outperforms the competitors in most experiments, for every percentage of missing values.

Next, we evaluated *whether SOLID reduces the error of similarity queries posed over databases containing high amounts of missing data.* We randomly inserted missing values into the database, with 10%, 20%, 30%, 40%, and 50% of missing rates. The error was computed as one minus the Jaccard coefficient between query results over the complete and the incomplete databases for each approach. Figure 3 shows the results, where the lower the values, the better. SOLID reduced the error in most of the analyzed scenarios because it takes advantage of correlated attributes given by the metric spaces' compatibility factors.

[1] https://www.kaggle.com/killen/bw-font-typefaces?select=BRLNSR

Our method outperformed its competitors for high values of k, reducing the error by 26.3% on average when compared to the Deletion approach, and being 16.4% more precise than the Decision Tree Imputation.

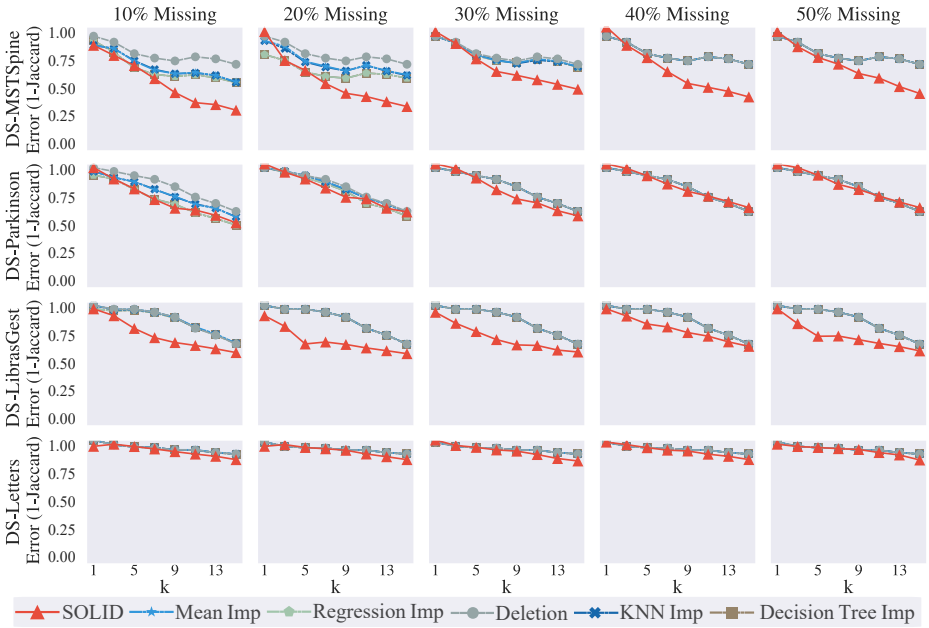

Fig. 3. Incomplete databases (**the lower the error, the better**): `SOLID` presented the lowest error rates in most experiments.

Performance Analysis. We computed the average time of 100 runs of each approach over a dataset with 15% of missing values. Figure 4 shows the execution time results of Knn_q with $k = 15$, with a 50/50 train and test division. The fastest approach was the Deletion, which drops the objects with missing values. `SOLID` was the fastest approach among the remaining ones. It was up to 30.8 times faster than its direct competitor, the Decision Tree Imputation.

5 Conclusions

In this work, we proposed the `SOLID` approach to answer similarity queries over complex attributes with missing values, without discarding elements. `SOLID` computes the correlation of metric spaces by implementing our `CorDiS` correlation method. The approach gives higher importance to attributes that more likely present correlated spatial distribution of data concerning the missing attribute. Experimental results over four representative datasets show that `SOLID` improved the similarity search quality by 7.3% regarding its best competitor. Even in datasets with missing rates as high as 50%, `SOLID` reduced the error by up to 16.4%, and was up to 30.8 times faster than other approaches. Thus, `SOLID` has proved to be well-fitted for similarity search over incomplete data.

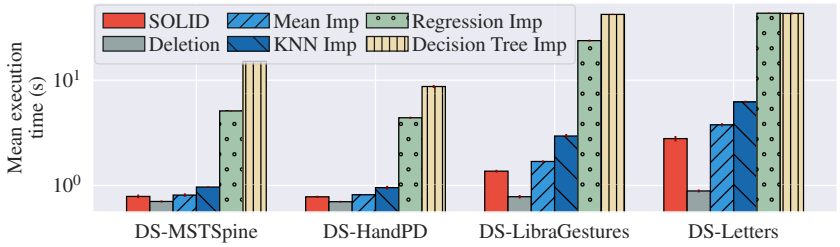

Fig. 4. Execution time of all approaches, for each dataset.

Acknowledgments

This research was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, by the São Paulo Research Foundation (FAPESP, grants No. 2016/17078-0, 2018/24414-2, 2020/10902-5, 2020/07200-9), and the National Council for Scientific and Technological Development (CNPq).

References

1. Bastos, I.L.O., Angelo, M.F., Loula, A.C.: Recognition of static gestures applied to brazilian sign language (libras). In: 28th SIBGRAPI (2015). https://doi.org/10.1109/SIBGRAPI.2015.26
2. Batista, G.E.A.P.A., Monard, M.C.: A study of k-nearest neighbour as an imputation method. His **87**(251-260), 48 (2002)
3. Figueroa, K., Reyes, N.: Permutation's signatures for proximity searching in metric spaces. In: International Conference on Similarity Search and Applications. pp. 151–159. Springer (2019). https://doi.org/10.1007/978-3-030-32047-8_14
4. Hunt, L.A.: Missing data imputation and its effect on the accuracy of classification. In: Data Science, pp. 3–14. Springer (2017). https://doi.org/10.1007/978-3-319-55723-6_1
5. Little, R.J., Rubin, D.B.: Statistical analysis with missing data, vol. 793. John Wiley & Sons (2019)
6. Pereira, C.R., et al.: Deep learning-aided parkinson's disease diagnosis from handwritten dynamics. In: 29th SIBGRAPI (2016). https://doi.org/10.1109/SIBGRAPI.2016.054
7. Rahman, M.G., Islam, M.Z.: Missing value imputation using decision trees and decision forests by splitting and merging records: Two novel techniques. Knowledge-Based Systems **53**, 51 – 65 (2013). https://doi.org/10.1016/j.knosys.2013.08.023
8. Rohrmeier, A., et al.: Lumbar muscle and vertebral bodies segmentation of chemical shift encoding-based water-fat mri: the reference database myosegmentum spine. BMC musculoskeletal disorders **20**, 152 (2019). https://doi.org/10.1186/s12891-019-2528-x
9. Salembier, P., Sikora, T., Manjunath, B.: Introduction to MPEG-7: Multimedia Content Description Interface. John Wiley & Sons, USA (2002)
10. Traina, A.J., et al.: Querying on large and complex databases by content: Challenges on variety and veracity regarding real applications. Information Systems **86**, 10 – 27 (2019). https://doi.org/10.1016/j.is.2019.03.012
11. Zabot, G.F., Cazzolato, M.T., Scabora, L.C., Traina, A.J.M., Traina-Jr., C.: Efficient indexing of multiple metric spaces with spectra. In: 2019 IEEE ISM. pp. 169–1697 (2019). https://doi.org/10.1109/ISM46123.2019.00038