Hindawi
Mathematical Problems in Engineering
Volume 2018, Article ID 3702808, 14 pages
https://doi.org/10.1155/2018/3702808

Research Article

On Detecting and Removing Superficial Redundancy in Vector Databases

Noemí DeCastro-García,[1] Ángel Luis Muñoz Castañeda,[2] Mario Fernández Rodríguez,[2] and Miguel V. Carriegos[1]

[1]*Departamento de Matemáticas, Universidad de León, Campus de Vegazana, s/n, ES-24071 León, Spain*
[2]*Research Institute on Applied Sciences in Cybersecurity, Universidad de León, Campus de Vegazana, s/n, ES-24071 León, Spain*

Correspondence should be addressed to Noemí DeCastro-García; ncasg@unileon.es

Received 10 December 2017; Revised 31 March 2018; Accepted 12 April 2018; Published 24 May 2018

Academic Editor: Emilio Insfran Pelozo

A mathematical model is proposed in order to obtain an automatized tool to remove any unnecessary data, to compute the level of the redundancy, and to recover the original and filtered database, at any time of the process, in a vector database. This type of database can be modeled as an oriented directed graph. Thus, the database is characterized by an adjacency matrix. Therefore, a record is no longer a row but a matrix. Then, the problem of cleaning redundancies is addressed from a theoretical point of view. Superficial redundancy is measured and filtered by using the 1-norm of a matrix. Algorithms are presented by Python and MapReduce, and a case study of a real cybersecurity database is performed.

1. Introduction

Current systems of knowledge extraction are based on the creation of the best models to solve a specific problem with particular data. In addition, the computational algorithms used are implemented and applied through different data management and processing architectures, from the most rudimentary to the most advanced analytical platforms using Big Data (BD) in real time.

In the most current cases, the creation of specific models, capable of analyzing, categorizing, and predicting different situations, such as anticipating trends or reacting to certain events, requires Big Data analytics. These techniques give rise to different challenges such as data inconsistency, incompleteness, scalability, timeliness, or data security [1, 2]. But, previously, data must be well-constructed [3].

On the other hand, good quality data is required to obtain good quality knowledge (in another case we fall in the well-known *garbage-in, garbage-out* constituted scenario; see [4]). Hence we should notice that not every datum is useful; for instance, it is expected that only 35% of considered data for analysis will be really useful by 2020 (see [5]).

In addition, motivated by the high increase of the number of incidents, the sensors, and the Internet of Things (IoT) devices, the rate of acquisition of data grows exponentially and, therefore, the volume of databases can become in a dangerous situation because the data obesity can be presented. Also, real-world databases are severely sensitive to be inconsistent, incomplete, and noisy. This fact turns out to be especially significant when several data sources need to be integrated. Working in a multisource system of acquisition of data generates high overlapping because new data are continuously included from different sources and thus increasing the probability of finding noise and dirty data. For instance, the above situation is typically in data warehouses, federated database systems, critical infrastructures, etc. (see [3, 6]). Thus, an appropriate strategy to remove unnecessary data (redundant data) in an automatized process is needed.

Data cleaning deals with detecting and removing errors from data and with eliminating the noise produced by the owned data collection procedure [7]. Hence, in a first approach, we state two types of redundancy: superficial and deep redundancy that could appear at instance or variables (features) level.

(1) Superficial redundancy refers to all variables that we do not need to take into account in our further analysis from a natural point of view (empty variables, constant, and identical cases). The study of superficial redundancy allows us to filter the database without advanced statistical analysis or previous transformation of the data in treatable variables. Moreover, this redundancy may be studied in any database.

(2) Deep redundancy collects all variables containing the same information, encoded in different ways as well as correlated variables, associated variables, or in general nondiscriminant variables to the fixed target. Note that in the first case of deep redundancy a simple frequency analysis could be enough to recognize the variables with the same information. However, the detection of the relation of correlation between variables, or the computation of the relevant features for a specific target, requires more advanced statistical analysis.

Note that it would be expected that redundancy appears in more than one type. The case of duplicated cases in a database is a special type of redundancy when the database is build up from several data sources, and it can show up in both types described above. In fact, removing the duplicate information is a very complex process in databases of cybersecurity reports, since identifying them is a difficult task that requires expert knowledge (deep redundancy). For example, we can have the same incident, reported by different sources, at different times, and by using a different lexical language. Or we can observe the same case reported twice by the same source because of defects in the collection procedure such as stuck-up of cases by updates.

Following [7], a data cleaning approach should satisfy several requirements:

(1) It should be able to remove all main errors and inconsistencies of data from individual and multiple sources.

(2) Manual inspection and programming effort should be limited.

(3) It should be flexible enough in order to integrate additional sources.

(4) Furthermore, data cleaning should be integrated together with schema-related data transformations.

(5) Data transformations along the cleaning procedure should be specified in a declarative way and be reusable.

There are several research works that develop different approaches to data cleaning of databases by as special data mining treatment, data transformations, or specific operators (see [8–14]). Also, some of them perform the data cleaning on a separate cluster that, later, we need to integrate into the data smart center. However, these works are focused on the study of duplicated cases, remove of typographical errors, or detecting inconsistencies or discrepancies. Thus, one of the remaining challenges of the data science is to design and propose efficient representations, access, and tools that let us preprocess and clean huge amount of variety of data before starting data analysis procedures [3]. Although there are a quite amount of commercial tools available to support these tasks, the cleaning and transformation work needs to be done manually or by low-level programs (see [7]).

Our goal in this paper is to give a mathematical model for detecting and removing superficial redundancies in an instance and variable level, for a single or multisource context, over certain kind of databases (vector databases). Our proposal is an oriented directed-graph which is theoretically based. Then, we address the problem of cleaning redundancies by using elementary algebraic tools. A matrix with entries in $B = \{0, 1\}$ is attached to a given database and removing redundancy operations arise as standard transformations on the matrix. Then we can give a concrete expression of the level of redundancy by using the 1-norm of a matrix. Thus we would be able to report redundancy in order to clean reports. Note that we do not intend just to delete all superficial redundancy data but to know the level in order to perform further actions in the design of statistical analysis. We remark that redundancy is not bad itself because some measures like the reputation of sources might be performed using redundant reports.

Moreover, in this work, we present a tool in open source that cleans the database in an automatized way and that computes the level of redundancy of the database. It also permits obtaining the original and the filtered database at any time of the process as well as the level of redundancy and the associated graph. The aim and procedures would be fully applied to any standard of reporting cases by means of formal language processing. The scripts mentioned above are available at a public GitHub repository (see [15, 16]).

In particular, this approach is applicable to clean up databases of cybersecurity reports (cyber databases). A cyber database contains a lot of unstructured information together with a high level of correlations where they are performed by human agents, that is, expert knowledge. The structural variety of the data of security reports is not unique (from machine-generated data to synthetic or artificial data). Moreover, the value of each feature could be structured, unstructured, or semistructured, and these typologies provide quantitative, (pseudo) qualitative, or string features. A security report usually is integrated, transformed, and combined with different data collection engines that provide only limited or null support for data cleaning, focusing on data transformations of management and schema integration. Since these engines receive information from different sources, in most cases we cannot modify the acquisition data process. Therefore, in order to extract knowledge from data, the best chance to get success is to optimize the different phases of the treatment and analysis of the data, and the first point is cleaning the database in an automatized way. Thus, we need to study the redundancy levels in order to detect superfluous reporters or optimize the resources. But, not every tool is useful. It must be noted here that some tools might be useless due to security constraints. In this case, it would not be possible to use online and private license software because sharing the data is not allowed. Therefore, in this situation, the tool to clean up the database would need to be integrated into an ecosystem with high levels of security.

In the final part of this paper, we also apply the developed tool to a real case of cleaning up a cyber database obtaining a 64% of superficial redundancy.

The paper is structured as follows: In Section 2 we give the model of a vector database from a matrix approach by graphs and our main results related to the computation of the level of redundancy. In Section 3, we develop the experimental section. This section includes the materials, the development of the tool that we have created to clean up databases as well as a comparison with some existent tools, and the case of study, in which we apply our tool to compute the level of the redundancy of a real fragment of a cyber database. Finally, our conclusions and references are given.

2. A Graph Approach to the Redundancy of a Database

A graph database is a database that can be structured in graph form so that the nodes of the graph contain the information and the edges contain properties and/or define relations of the information contained in the nodes. One of the main strengths of these kinds of databases is the capability to give answers in short time for questions regarding relations (see [17]).

In this section, we will define a graph structure on a database that conceptually differs from the usual one described above, and the motivation is based on the problem of detecting or cleaning redundancies in a database. In general, to show whether two columns or variables of our database are redundant or not, in some sense, one looks at the information contained in these variables and then decide. Although this will be our procedure eventually, we will cluster the set of variables by looking at the meaning they have and then we will consider the usual procedure. The point is that once the clustering is done the database and the clusters define a graph structure in a natural way where not all the nodes contain information.

2.1. A Graph Model for a Database. Observe that in the above discussion we started by considering a usual database and we finished with the database plus a clustering of the variables. Before defining the graph structure, we will formalize this situation, and we will use it as the starting point.

Definition 1. A vector database $\mathscr{V}\mathscr{D}$ is a tuple of databases $\mathscr{D}_i$, each one of them coming with a label, which satisfy the following properties:

(1) All the databases $\mathscr{D}_i$ have the same length; that it is, all of them have the same number of rows n.

(2) If a database has a unique column, the column name agrees with the database label.

(3) Two different databases must have different labels.

(4) To have same column names in different databases is allowed.

(5) The nature of features is of any type (strings, floats, integers, etc.).

Remark 2. We will state some notation for the sake of clarity.

(1) We will use the notation $\{I_1, \ldots, I_r\}$ for the labels of the databases that are one-column composed as well as their unique column name, and we will denote by I the above set.

(2) The set $X = \{x_1, \ldots, x_l\}$ is going to be the set of labels of the databases.

(3) The set $M^i = \{m_1^i, \ldots, m_{s_i}^i\}$ is the set of column names of each single database D_i. From $M = \{M^i\}$, we can construct the set $Y = \{y_1, \ldots, y_s\}$ and the different column labels collected from all the databases. We can reorder M according to Y, and then we use the following notation $\{y_1^i, \ldots, y_{s_i}^i\}$ for the ordered m_j^i.

(4) The kth row or report of a vector database is given by the vector constructed from the kth rows of each database D_i and is denoted by R_k.

With the notation described in Remark 2, a vector database has the following form:

$$\mathscr{V}\mathscr{D} = (D_1, \ldots, D_{l+r}), \tag{1}$$

where D_i could have the form

$$x_i \longrightarrow \begin{array}{|c|c|c|c|} \hline m_1^i & m_2^i & \ldots & m_{s_i}^i \\ \hline \text{Row 1} & & & \\ \hline \vdots & & & \\ \hline \text{Row n} & & & \\ \hline \end{array} \tag{2}$$

or

$$I_h \longrightarrow \begin{array}{|c|} \hline I_h \\ \hline \text{Row 1} \\ \hline \vdots \\ \hline \text{Row n} \\ \hline \end{array} \tag{3}$$

Remark 3. If we apply the ordering that the set Y provides, then we can understand a vector database as a unique table; see Table 1.

So, any database in the form of (1) or Table 1 verifies the condition of Definition 1.

Example 4. We consider the example shown in Table 2.

In this case, we can construct the following sets:

(1) $I = \{I_1 = \text{Time Stamp}\}$.

(2) $X = \{x_1 = \text{Attacker}, x_2 = \text{Source}, x_3 = \text{Target}\}$.

TABLE 1: Second form of a vector database.

$x_1(y_1^1)$	$\cdots$	$x_1(y_{s_1}^1)$	$\cdots$	$x_l(y_1^l)$	$\cdots$	$x_l(y_{s_l}^l)$	I_1	$\cdots$	I_h
$\vdots$									

Table 2: Example of a vector database.

Attacker Region Code	Attacker Port	Source Region Code	Source Port	Target Port	Target Region Code	Target Type IP	Time Stamp
35	137	35	137		45	static	23:56:24
56	1456	56	1456		45	static	23:56:39
67	1456	67	1456		45	dynamic	23:57:02
67	378	67	378		45	dynamic	23:57:43

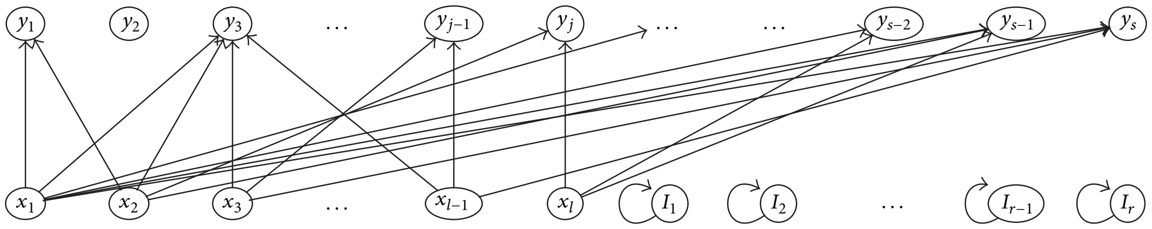


Figure 1: Example of the graph associated with a vector database.

(3) $M^1 = \{m_1^1 = \text{Region Code}, m_2^1 = \text{Port}\}$, $M^2 = \{m_1^2 = \text{Region Code}, m_2^2 = \text{Port}\}$, $M^3 = \{m_1^3 = \text{Port}, m_2^3 = \text{Region Code}, m_3^3 = \text{Type IP}\}$.

(4) From the last sets, we obtain the different column names, $Y = \{y_1 = \text{Region Code}, y_2 = \text{Port}, y_3 = \text{Type IP}\}$. Now, we transform m_j^i into y_j^i in order to normalize the vector database depending on Y. Then, $x_1(\{y_1^1 = \text{Region Code}, y_2^1 = \text{Port}\})$, $x_2(\{y_1^2 = \text{Region Code}, y_2^2 = \text{Port}\})$, $x_3(\{y_1^3 = \text{Region Code}, y_2^3 = \text{Port}, y_3^3 = \text{Type IP}\})$.

We can now give a graph structure to the object defined in Definition 1. In order to do so we have to define the set of nodes, N, and the set of arrows, A. The graph, $G = (N, A)$, is going to be composed of two layers of nodes in such a way that all the arrows have their source in one layer and target in the other one.

(1) Layer 1: the nodes are the elements of the sets X and I.

(2) Layer 2: the nodes are the elements of the set Y.

(3) We have an edge $x_i \to y_j$ if and only if y_j is a column name of the database x_i. From now on this relation will be expressed as $y_j \in x_i$.

(4) Every variable I_i has a cycle.

The look of such a graph is shown in Figure 1.

Example 5. The associated graph to the vector database of Example 4 is shown in Figure 2.

In the sequel, B will denote the set $\{0, 1\}$, and the set of matrices with entries in B with l rows and s columns will be denoted by $B^{l\times s}$.

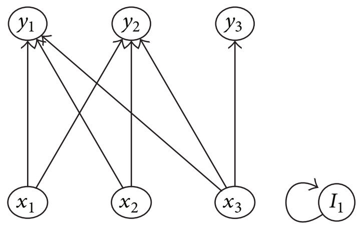


Figure 2: Associated graph with database shown in Table 2.

Definition 6. The adjacency matrix of a vector database is defined as the adjacency matrix, $A \in B^{(l+s+r)\times(l+s+r)}$, of the associated graph:

$$\begin{array}{c|c|c|c} & x_1\cdots x_l & y_1\cdots y_s & I_1\cdots I_r \\ \hline x_1,\ldots,x_l & 0 & \mathscr{C}=(c_{ij}) & 0 \\ \hline y_1,\ldots,y_s & 0 & 0 & 0 \\ \hline I_1,\ldots,I_r & 0 & 0 & \mathscr{I}=Id \end{array} \tag{4}$$

where

$$c_{ij} = \begin{cases} 1 & \text{if } y_j \in x_i \\ 0 & \text{if } y_j \notin x_i \end{cases} \tag{5}$$

for $i = 1, \ldots, l$, $j = 1, \ldots, s$.

Remark 7. Note if we are only interested in the database, without more relations than those that have been defined, the study of the adjacency matrix is reduced to the matrices $\mathscr{C}$

and $\mathcal{I}$. For this reason, in this paper, we will use the notation $A = (\mathcal{C}, \mathcal{I})$ to make reference to such adjacency matrix.

Example 8. The adjacency matrix associated with the vector database of Example 4 is

$$A = \begin{pmatrix} 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{0} & 0 \\ 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{0} & 0 \\ 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} \end{pmatrix} \quad (6)$$

where the matrices $\mathcal{C}$ and $\mathcal{I}$ are highlighted in bold characters.

Definition 9. From each row or report R_k, $k = 1, \dots, n$, we can construct the following weighted square matrix $W_k = (w_{pq}^k)$:

$$W_k = \begin{array}{c|c|c|c} & x_1 \cdots x_l & y_1 \cdots y_s & I_1 \cdots I_r \\ \hline x_1 \\ \vdots & 0 & \overline{\mathcal{C}}_k = (\overline{c}_{ij}^k) & 0 \\ x_l \\ \hline y_1 \\ \vdots & 0 & 0 & 0 \\ y_s \\ \hline I_1 \\ \vdots & 0 & 0 & \overline{\mathcal{I}}_k \\ I_r \end{array} \quad (7)$$

where

$$\left(\overline{c}_{ij}^k\right) = \begin{cases} \text{Value of } y_j^i \text{ in row } R_k & \text{if } c_{ij} = 1 \\ \text{empty} & \text{if } c_{ij} = 1 \wedge y_j^i = \emptyset \text{ in } R_k \\ 0 & \text{if } c_{ij} = 0 \end{cases} \quad (8)$$

for $i = 1, \dots, l$, $j = 1, \dots, s$, and

$$\overline{\mathcal{I}}_k = \begin{pmatrix} I_1^k & & \\ & \ddots & \\ & & I_r^k \end{pmatrix}_{r \times r} \quad (9)$$

where I_h^k represents the content of the column I_h in the report R_k for $1 \le h \le r$.

Remark 10. An added problem that we will have to take into account when carrying out advanced analysis and the cleaning of the data is the problem of removing missing values. We have two different missing values in our model; first, in some rows, we do not have the complete information on all the features. Secondly, we do not have all possible arguments in each feature subvector. The problem of missing data is solved, a priori, by substitution by two categories: sample zeroes = *empty* (lack of information in the sample or in the database) or structural zeroes = 0 (lack of information due to nonexistence).

Example 11. From the vector database of Example 4 we can construct the following weighted square matrices, one for each case of the database:

$$W_1 = \begin{pmatrix} 0 & 0 & 0 & \mathbf{35} & \mathbf{137} & \mathbf{0} & 0 \\ 0 & 0 & 0 & \mathbf{35} & \mathbf{137} & \mathbf{0} & 0 \\ 0 & 0 & 0 & \mathbf{45} & \mathbf{static} & \mathbf{empty} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{23{:}56{:}24} \end{pmatrix}$$

$$W_2 = \begin{pmatrix} 0 & 0 & 0 & \mathbf{56} & \mathbf{1456} & \mathbf{0} & 0 \\ 0 & 0 & 0 & \mathbf{56} & \mathbf{1456} & \mathbf{0} & 0 \\ 0 & 0 & 0 & \mathbf{45} & \mathbf{static} & \mathbf{empty} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{23{:}56{:}39} \end{pmatrix}$$

$$W_3 = \begin{pmatrix} 0 & 0 & 0 & \mathbf{67} & \mathbf{1456} & \mathbf{0} & 0 \\ 0 & 0 & 0 & \mathbf{67} & \mathbf{1456} & \mathbf{0} & 0 \\ 0 & 0 & 0 & \mathbf{45} & \mathbf{dynamic} & \mathbf{empty} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{23{:}57{:}02} \end{pmatrix} \quad (10)$$

$$W_4 = \begin{pmatrix} 0 & 0 & 0 & \mathbf{67} & \mathbf{378} & \mathbf{0} & 0 \\ 0 & 0 & 0 & \mathbf{67} & \mathbf{378} & \mathbf{0} & 0 \\ 0 & 0 & 0 & \mathbf{45} & \mathbf{dynamic} & \mathbf{empty} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{23{:}57{:}43} \end{pmatrix}$$

2.2. Cleaning the Database: Level of Redundancy. Once the graph structure is defined, our proposal is to use it to compute the level of redundancy of the database. The redundancies we are going to deal with are defined as follows:

`Type I`: it includes empty variables.

`Type II`: it includes constant variables.

`Type III`: it includes equivalences across the databases x_k.

$x_{k_1} \sim x_{k_2} \Leftrightarrow$ their variables contain the same information.

In order to give a close mathematical formula for the level of redundancy in terms of the graph structure we will have to define these redundancies properly.

Definition 12. Let $A = (\mathscr{C}, \mathscr{I})$ be the adjacency matrix of the database. One defines the following maps:

$$\begin{aligned} \mathscr{R}_{\mathrm{TI}}^{X(Y)} &: B^{l\times s} \longrightarrow B^{l\times s} \\ \mathscr{C} = \left(c_{ij}\right) &\longmapsto \mathscr{R}_{\mathrm{TI}}^{X(Y)}(\mathscr{C}) = \mathscr{C}' \end{aligned} \tag{11}$$

where $\mathscr{C}' = (c'_{ij}) \in B^{l\times s}$ is defined as

$$c'_{ij} := \begin{cases} 0 & \text{if } \left[c_{ij} = 0\right] \vee \left[c_{ij} = 1 \wedge \left(\bar{c}_{ij}\right)^k = \text{empty } \forall k = 1, \ldots, n\right] \\ c_{ij} & \text{otherwise} \end{cases} \tag{12}$$

We can define the analogous map $\mathscr{R}_{\mathrm{TI}}^{\mathscr{I}}$ acting over the submatrix $\mathscr{I}$.

Applying the maps $\mathscr{R}_{\mathrm{TI}}^{*}$ over the matrices $A = (\mathscr{C}, \mathscr{I})$, we obtain the filtered adjacency matrix $A' = (\mathscr{C}', \mathscr{I}')$, which contains the database once Type I redundant variables have been dropped.

Remark 13. In case the entries of the database were numerical, note that we could get the filtered database (without redundancy Type I), $\{R'_k\}$, constructing $W'_k = (\overline{\mathscr{C}}'_k, \overline{\mathscr{I}}'_k) = A' \cdot W_k$ from A', where $\cdot$ denotes the usual Hadamard (entry-wise) matrix product.

Definition 14. The level of redundancy of Type I, ρ_{TI}, is the rate of variables or columns that are always empty in the database and is computed by

$$\begin{aligned} \rho_{\mathrm{TI}}(A) &:= \frac{\|A\|_1 - \left\|A'\right\|_1}{\|A\|_1} \\ &= \frac{\left[\|\mathscr{C}\|_1 + \|\mathscr{I}\|_1\right] - \left[\left\|\mathscr{C}'\right\|_1 + \left\|\mathscr{I}'\right\|_1\right]}{\|\mathscr{C}\|_1 + \|\mathscr{I}\|_1} \end{aligned} \tag{13}$$

where $\|A\|_1$ denotes the 1-norm of the matrix A, that is, the number of $1's$ in matrix A.

Remark 15. Note that we can also obtain the redundancy of Type I in each subset of variables X and I separately:

$$\begin{aligned} \rho_{\mathrm{TI}}(\mathscr{C}) &= \frac{\|\mathscr{C}\|_1 - \left\|\mathscr{C}'\right\|_1}{\|\mathscr{C}\|_1}, \\ \rho_{\mathrm{TI}}(\mathscr{I}) &= \frac{\|\mathscr{I}\|_1 - \left\|\mathscr{I}'\right\|_1}{\|\mathscr{I}\|_1}. \end{aligned} \tag{14}$$

The next type of redundancy is also known (for numerical variables) as 0-variance redundancy, and it is constituted by the columns or variables that always take a constant value for all rows and so they do not provide us with relevant information. Note that redundancy of Type I is redundancy Type II when the constant value is zero.

Definition 16. Let $A = (\mathscr{C}, \mathscr{I})$ be the adjacency matrix of the database. One defines the following maps:

$$\begin{aligned} \mathscr{R}_{\mathrm{TII}}^{X(Y)} &: B^{l\times s} \longrightarrow B^{l\times s} \\ \mathscr{C} = \left(c_{ij}\right) &\longmapsto \mathscr{R}_{\mathrm{TII}}^{X(Y)}(\mathscr{C}) = \mathscr{C}' \end{aligned} \tag{15}$$

where $\mathscr{C}'' = (c''_{ij}) \in B^{l\times s}$ is defined as

$$c''_{ij} := \begin{cases} 0 & \text{if } \left[c_{ij} = 0\right] \vee \left[c_{ij} = 1 \wedge \left(\bar{c}_{ij}\right)^k = M \; \forall k = 1, \ldots, n\right] \\ c_{ij} & \text{otherwise} \end{cases} \tag{16}$$

Note that the constant M is different for each position $\overline{c_{ij}}$, but its value in a position must be equal for all rows.

Again, we can define the analogous map $\mathscr{R}_{\mathrm{TII}}^{I}$ acting over the submatrices $\mathscr{I}$.

Remark 13 can be made in this case.

Definition 17. The level of redundancy of Type II, ρ_{TII}, is the rate of variables or columns that are always constant in the database and is computed by

$$\begin{aligned} \rho_{\mathrm{TII}}(A) &:= \frac{\|A\|_1 - \left\|A''\right\|_1}{\|A\|_1} \\ &= \frac{\|\mathscr{C}\|_1 + \|\mathscr{I}\|_1 - \left\|\mathscr{C}''\right\|_1 - \left\|\mathscr{I}''\right\|_1}{\|\mathscr{C}\|_1 + \|\mathscr{I}\|_1} \end{aligned} \tag{17}$$

Note that, in the same way as Remark 15, we can also compute redundancy Type II in both subsets of variables, X and I.

The redundancy of Type III has to be with the possible equivalences between the sets of variables of the different databases of the tuple, that is, between the elements of X. In this case, we make comparisons between the variables x_i that could have the same content in their arguments y_j.

Definition 18. Let $X = \{x_i\}$ and $Y_i = \{y^i_j\}$ be the set of databases and the set of variables for each x_i where $i = 1, \ldots, l$ and $j = 1, \ldots, s_i$. One defines the following equivalence relation:

$$\begin{aligned} &x_i \equiv x_t \quad \text{with } t \neq i \Longleftrightarrow \\ &\text{Content}\left(y^i_j\right) = \text{Content}\left(y^t_j\right) \end{aligned} \tag{18}$$

after reordering the variables if it were necessary

This equivalence relation introduces a partition in $x_i's$:

$$X = \coprod_{\theta=1}^{M} X_\theta, \tag{19}$$

in such a way that in any of those partitions X_θ included all $x_i's$ that are equivalent to each other.

Definition 19. Let $A = (\mathscr{C}, \mathscr{I})$ be the adjacency matrix of the database, and fix a representative x_{i_θ} for each part X_θ defined by the above equivalence relation. One defines the following map:

$$\begin{aligned} \mathscr{R}_{\mathrm{TIII}}^{X(Y)} : B^{l\times s} &\longrightarrow B^{l\times s} \\ \mathscr{C} = \left(c_{ij}\right) &\longmapsto \mathscr{R}_{\mathrm{TIII}}^{X(Y)}(\mathscr{C}) = \mathscr{C}' \end{aligned} \tag{20}$$

where $\mathscr{C}''' = (c_{ij}''') \in B^{l\times s}$ and

$$c_{i\bullet}''' := \begin{cases} 0 & \text{if for some } \theta, \overline{c}_{i\bullet}^{k} = \overline{c}_{i_\theta\bullet}^{k}\ \forall k = 1,\ldots,n \\ c_{i\bullet} & \text{otherwise,} \end{cases} \tag{21}$$

$i \neq i_\theta, \forall\theta$.

Applying $\mathscr{R}_{\mathrm{TIII}}$ over the matrices $A = (\mathscr{C}, \mathscr{I})$, we obtain the filtered adjacency matrix $A''' = (\mathscr{C}''', \mathscr{I}''')$. Note that $\mathscr{I}'' = \mathscr{I}'''$ because redundancy Type III does not concern individual features. Moreover, we can construct $W_k''' = (\overline{\mathscr{C}}_k''', \overline{\mathscr{I}}_k''') = A''' \cdot W_k$ from A''' and we get the row R_k filtered by redundancy Type III, R_k'''.

Definition 20. The level of redundancy of Type III, ρ_{TIII}, is the rate of $x_i's$ that are in the subsets X_θ once the representatives one has chosen have been taken out and is computed by

$$\begin{aligned} \rho_{\mathrm{TIII}}(A) &:= \frac{\|A\|_1 - \left\|A'''\right\|_1}{\|A\|_1} \\ &= \frac{\|\mathscr{C}\|_1 + \|\mathscr{I}\|_1 - \left\|\mathscr{C}'''\right\|_1 - \left\|\mathscr{I}'''\right\|_1}{\|\mathscr{C}\|_1 + \|\mathscr{I}\|_1} \end{aligned} \tag{22}$$

Definition 21. Let $A = (\mathscr{C}, \mathscr{I})$ be the adjacency matrix of the database. Let $A^{(*} = (\mathscr{C}^{(*}, \mathscr{I}^{(*})$ be the adjacency matrix of the database after applying all types of redundancy described above. Then, the redundancy of the database is defined as follows:

$$\begin{aligned} \rho_{\mathrm{Tot}}(A) &:= \frac{\|A\|_1 - \left\|A^{(*}\right\|_1}{\|A\|_1} \\ &= \rho_{\mathrm{I}}(A) + \rho_{\mathrm{II}}\left(A'\right) + \rho_{\mathrm{III}}\left(A''\right). \end{aligned} \tag{23}$$

Remark 22. At this point it is worth noticing that our procedure is able to write down superficial redundancy after each subset and filtered report may be recovered if it is necessary.

Table 3: The filtered database of Table 2 after removing the superficial redundancies Types I, II, and III.

Attacker Region Code	Attacker Port	Target Type IP	Time Stamp
35	137	static	23:56:24
56	1456	static	23:56:39
67	1456	dynamic	23:57:02
67	378	dynamic	23:57:43

Example 23. If we apply redundancy Type I over the adjacency matrix of Example 8, we obtain

$$A' = \begin{pmatrix} 0&0&0&\mathbf{1}&\mathbf{1}&\mathbf{0}&0 \\ 0&0&0&\mathbf{1}&\mathbf{1}&\mathbf{0}&0 \\ 0&0&0&\mathbf{1}&\mathbf{1}&\mathbf{0}&0 \\ 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&\mathbf{1} \end{pmatrix} \tag{24}$$

So, $\rho_{\mathrm{TI}}(A) = (8-7)/8 = 1/8 = 0{,}125$. Now, we apply over A' redundancy Type II and obtain

$$A'' = \begin{pmatrix} 0&0&0&\mathbf{1}&\mathbf{1}&\mathbf{0}&0 \\ 0&0&0&\mathbf{1}&\mathbf{1}&\mathbf{0}&0 \\ 0&0&0&\mathbf{0}&\mathbf{1}&\mathbf{0}&0 \\ 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&\mathbf{1} \end{pmatrix} \tag{25}$$

Then, $\rho_{\mathrm{TI+TII}}(A) = (8-6)/8 = 2/8 = 0{,}25$.

Finally, we apply redundancy Type III, getting the following adjacency matrix:

$$A''' = \begin{pmatrix} 0&0&0&\mathbf{1}&\mathbf{1}&\mathbf{0}&0 \\ 0&0&0&\mathbf{0}&\mathbf{0}&\mathbf{0}&0 \\ 0&0&0&\mathbf{0}&\mathbf{1}&\mathbf{0}&0 \\ 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&\mathbf{1} \end{pmatrix} \tag{26}$$

The superficial redundancy of the vector database is $\rho_{\mathrm{TI+TII+TIII}}(A) = (8-4)/8 = 4/8 = 0{,}5$, meaning that the half database is mathematically redundant. The filtered database would be as shown in Table 3.

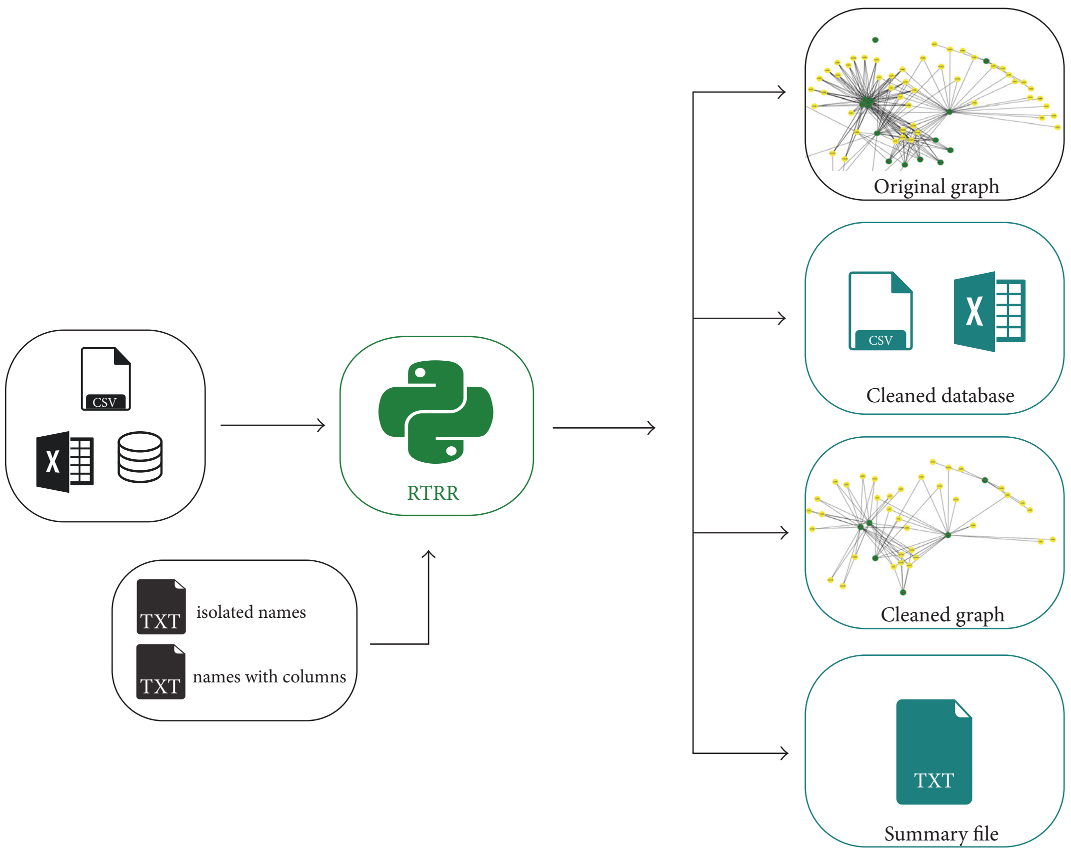


FIGURE 3: General working flow of RTRR.

3. Experimental Section

This section is divided into three different subsections. In the first part, the materials that have been used in the experimental section are described. The second part is devoted to developing the tool *RIASC Tool for Removing Redundancies* (RTRR). It has been created in Python language to apply what we have developed in Section 2. Finally, we summarize the results obtained from the application of this tool to a real database.

3.1. Materials. The *RIASC Tool for Removing Redundancies* (RTRR) is available at public git repository (see [15, 16]). It has been coded using Python 2.7 (version 1, v1), and there is also a version for MapReduce (version 2, v2). Both versions allow the user to clean a database to prepare it for processing the data. The computation of time and complexities of the presented algorithms had been performed by using Cloudera's virtual machine with OS Red Hat (64-bit). The CPU specifications are as follows: 4 GB of RAM with 4 cores of which only 1.5 GB of RAM was available for the MapReduce task.

In the case of study, the database is a real fragment with cybersecurity reports with 363 variables or columns and 2600 rows. It has been anonymized due to confidentiality constraints.

3.2. RTRR: Development and Comparative. The execution flow of the RTRR (see Figure 3) begins with the input data source. It is assumed that the input database is taken in a nonstructured way. Therefore the graph structure has to be discovered and introduced later or has to be given by the user as a second input. These are the two modes the RTRR can work with. The way the RTRR clusters the variables or column names in order to get the graph structure is by looking at the lexical structure of these column names. It is assumed that the column names have the structure

$$\begin{aligned}&\texttt{column name i}\\&\quad=\texttt{word}-1\ \ \texttt{word}-2\ \cdots\ \texttt{word}-\texttt{j}\,(\texttt{i}),\end{aligned}\tag{27}$$

where the empty space that separates each word can be changed by any other character (delimiter character), and it has to be included as an input.

In the first case mentioned above, the user distinguishes between the three types of nodes of the eventual graph structure, providing two different text files (.txt). One of them is with the list of names of the single-column-tables (I), chosen among the column names. Another one is with the list of names of the rest of the tables (X), chosen among the rest of the column names. In this case, an element $x \in X$ is not formed by the complete chain of words of one of the column names (27) but by that chain of words cut off at some

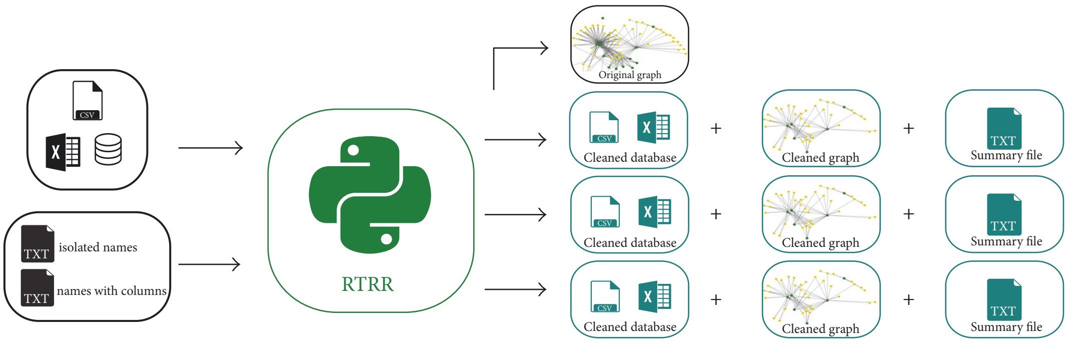


Figure 4: Single mode of RTRR.

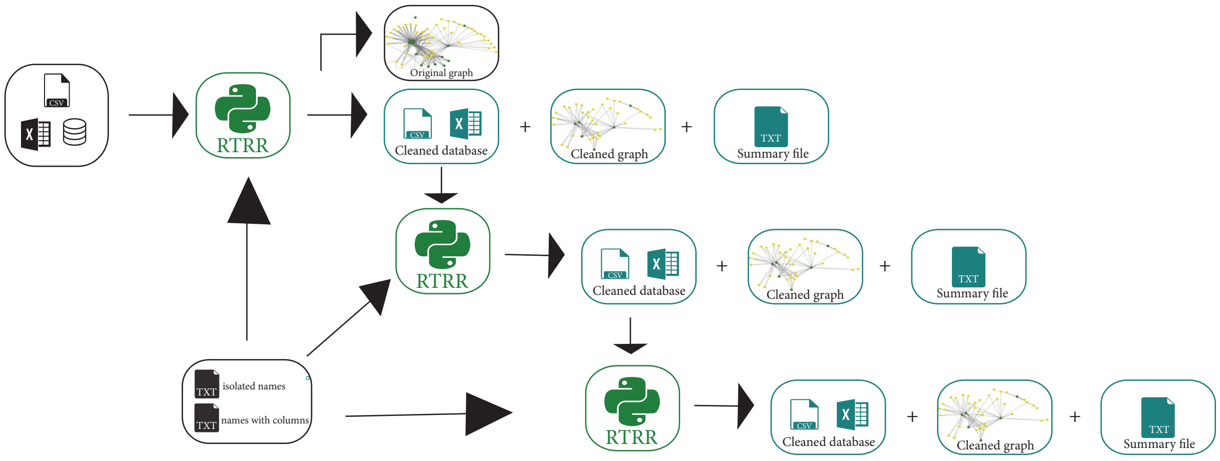


Figure 5: Accumulative mode of RTRR.

point. With these two files, the tool automatically constructs the set Y with the word chains that complement the strings that make up the names of the tables $x \in X$. In the second case, RTRR distinguishes between table names and the names of the columns of them using only the delimiter character between words in (27), given by the user. In this case, the first word is taken always as a table name and the rest of the chain of words make up a column name of that table.

The input data source is introduced in the tool to be processed and when the process finishes, the user obtains four outputs for each request of erasing each redundancy type (the user can request more than one type in the same process as will be described later):

(i) The first output is the graph that represents the input data source, that is to say, the graph with all the variables before removing redundancies.

(ii) The second output is the cleaned input data source, the database without redundancies.

(iii) The third output is the graph linked to the cleaned input data source, a new graph without redundant variables.

(iv) The last output is a text file that gathers the information about the level of redundancy and the list of the variables removed throughout the execution process.

There are three key features in the tool: the first one is the cleaning of the input source by removing redundant variables. The second one is the two different ways that the tool provides for erasing redundant variables, that is to say, the working modes: the single mode and the accumulative mode, which will be described below. Finally, the generated graphs based on the recognition of variables and the later elimination of redundancies.

The tool has two modes for removing redundancies: single mode and accumulative mode.

The single mode allows removing only one type of redundancy at once because it processes always the original input database. If the user specifies more than one redundancy type to be erased, those types of redundancies are eliminated individually, according to Figure 4, where each path corresponds to the erasing of one type of redundancy (three paths in the worst case).

The accumulative mode allows concatenating the removal of several redundancies in the same execution, that is to say, to remove multiple redundancies at once. The diagram is radically different because the cleaned database obtained in a previous execution is used as input database in the next execution as shown in Figure 5.

Finally, the tool generates oriented graphs with the isolated names and the relationships between the names and the columns. The isolated names and names are green nodes and the columns are yellow nodes; those two different colors allow the user to identify the direction of the relationship. When we introduce an input data source in the tool, an initial graph representation of the initial database is generated, as well as a new graph representation for each type of redundancy that the user decides to remove. The user obtains two graphs (the original one and another without one of the types of redundancy) in the basic case and four graphs (the original one and three more, one for each type of redundancy removed) in the worst case.

In order to make a comparison between RTRR and other tools of cleaning databases, we have proposed the following assessment indicators for an eventual tool that integrates the cleaning procedure:

(1) Required computational resources—for example, we should take into account the minimal required RAM needed for the tool to work properly and also the operative systems that can support it, such as Windows (W), Linux (L), and MacOS (M)

(2) Types of cleaning redundancies tasks, that is, what kind of redundancies the tool is able to remove

(3) Provided services—for example, the possibility of getting back to the original database, the cleaned database, the removed entries, or variables at each stage of the cleaning process as well as graphical representations of it

(4) Formats and sizes of datasets that the tool is able to manage

(5) The platform features, such as graphical interface and online version

(6) License.

Note that the assessment of each indicator could be considered positive or negative depending on the context the tool is working in. For instance, we can find cleaning tools that work only online without existing any version working in local hosts. This could be positive, a priori, but in a confidentiality context this tool could not be taken into account.

In Table 5, we show a comparison among different tools that deal, in some way, with at least one of the redundancies defined above. These tools are *Data Wrangler* [T1], *Open Refine* [T2], *Trifacta Wrangler* [T3], *Data Cleaner* [T4], *WinPure Data Cleaning Tool* [T5], *Excel* [T6], and *Database Cleaner* [T7] (see [18–24]). Although there exist more tools for cleaning data, most of them are focused on removing duplicated data as identical cases; see the references [25–29]. Most of the analyzed tools are designed to clean at the row level, so the way to proceed when removing empty variables is by transposing the database (that is what the symbol $*$ means in Table 5). This fact, however, demands a high level of computation resources. Also, with some of above tools we are not allowed to include as many variables as we want, so transposing the database is not possible when the amount of rows is large. Even when transposing the database is possible, in some of these tools, the allowed type of variables is just numerical, and this imposes a strong restriction. All of them let us recover the original dataset as well as the cleaned dataset. However, none of them compute the level of redundancy.

Table 4: Analysis of RTRR.

Indicator	RTRR Tool
Minimal Required RAM	NO
Redundancy type I	√
Redundancy type II	√
Redundancy type III	√
Representations	Graphs
Allowed Input Format	v1: CSV, PostgreSQL, MySQL, MS Excel v2: CSV, MS Excel
Output Format	CSV
Size restrictions	NO
Operative systems	W/L/M
Online version	X
Local version	√
Interface	Console
License	GPL (free)
Company	RIASC

In Table 4 we establish the analysis of the proposed assessment indicators for RTRR tool.

Although the tool RTRR covers different needs regarding cleaning data, there are still certain limitations that make this work open for future research.

In the first place, it would be necessary to enlarge the possible types of inputs and outputs that the tool can deal with. It is important to highlight that RTRR has been designed to work in a localhost, so it would be necessary to adapt it to different frameworks; for instance, it would be important to have another version being able to work in the cloud.

Note also that there is no graphic interface for RTRR, and it would be important to design it in order to bring the tool to nonexpertise users.

As the last limitation, it is worth highlighting the fact that RTRR cleans redundancies of three different types, although more types can be detected in a database. As future work, the introduction of more types of redundancies will be considered.

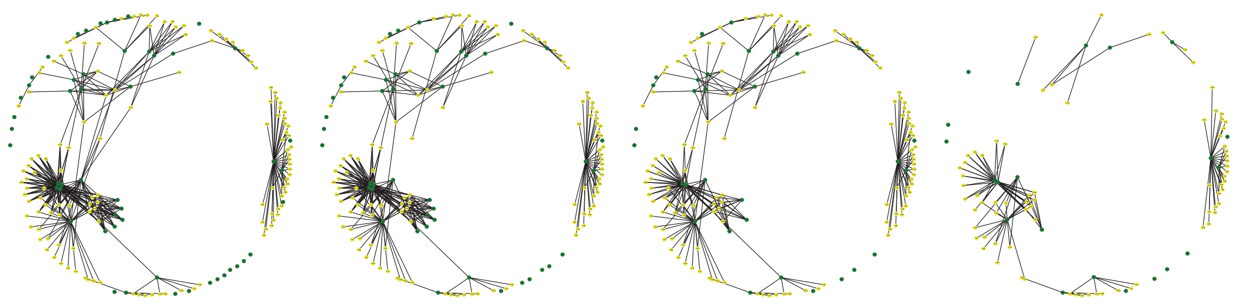
3.3. A Case Study

Definition 24. A cyber database is formed by a huge amount of security reports ($R_\bullet$) as the row described in a vector database, that is to say, vectors with n states or features whose confidential information is about a security incident.

Usually, a cyber database has the following properties: regarding the volume of data, a cyber database has a large amount of data every day. The structural variety of the data of security reports depends on how we acquire the data, from machine-generated data (acquired by engines), correlated

Table 5: Analysis of cleaning dataset tools. In the notation •/•, the first argument represents a feature of the free version of the corresponding tool, and the second one represents the same feature but in the enterprise version. Also, the symbol $*$ means that the tool cleans by row level instead of column level.

Indicator	T1	T2	T3	T4	T5	T6	T7
Minimal Required RAM	NO	NO	4 GB	NO	NO	2 GB	128 MB
Redundancy type I	NO*	NO*	NO*	NO*	NO*	√$*$	NO
Redundancy type II	NO	NO	NO*	NO*	NO*	NO	NO
Redundancy type III	NO	NO	NO	NO	NO	NO	NO
Representations	NO	NO	statistical graphics	statistical graphics	statistical graphics	NO	NO
Allowed Input Format	text	CSV, text JSON, XML Google Format, RDF	csv, text, MS Excel, JSV, LOG, MySQL, JSON, , SQL Server	CSV, MS Excel, SQLServer, Oracle DB XML, PotsgreSQL, Apache Derby, IBM DB2, HSQL DB MySQL, Mongo DB	CSV, text, MS Excel, MySQL, Oracle DB SQLServer, MS Access	CSV, text, MS Excel, XML, DIF, SYLK, DBASE	CSV, text, MS Access, SQL Server MySQL,
Output Format	CSV, text tsv, JSON Lookup Table	CSV, TSV MS Excel HTML, Template	CSV, JSON TDE/ /MS Access, MySQL, SQLServer	CSV, MS Excel MySQL, SQLServer Oracle DB	CSV, text MS Access	CSV, text Excel, DIF, XML, SYLK, PDF, XPS HTML, Open XML Open Document	CSV, text MS Excel
Size restrictions	1000 cases and 40 variables	NO	NO	NO	NO	1.048.576 cases and 16.384 variables	1000 cases/ /without limit
Operative systems	W/L/M	W/L/M	W/M	W/L/M	W	W/M	W
Online version	√	X	X	cloud	X	X	Cloud
Local version	X	√	√	√	√	√	√
Interface	Web	Web	Desktop	Desktop	Console/desktop	Desktop	Desktop
License	Free	BSD	Free /Enterprise	Free/ /Paying	Free/ /Paying	Free/ /Paying	Free/ /Paying
Company	Stanford/ /Berkeley	Open Source	Trifacta	Human Interface	WinPure	Microsoft	Ashisoft



(a) Original graph

(b) Graph without T2

(c) Graph without T2 and T3

(d) Cleaned without T1, T2, and T3

Figure 6: Graphs.

Table 6: Level of redundancy in the case of study.

Adjacency matrix	$\#X$	$\#I$	$\#Y$	Redundancy level
$\Vert A\Vert_1 = 363$	28	24	131	– –
$\Vert A'\Vert_1 = 337$	28	15	129	$\rho_{\mathrm{TII}} = 0.07$
$\Vert A''\Vert_1 = 253$	23	9	129	$\rho_{\mathrm{II}} + \rho_{\mathrm{III}} = 0.34$
$\Vert A'''\Vert_1 = 131$	13	8	76	$\rho_{\mathrm{Tot}} = 0.64$

data (correlated data by engines based on different rules), and synthetic or artificial data (added data by expert agents). The types of variables are not uniform. They could be structured, unstructured, or semistructured. These typologies provide quantitative, qualitative, string, or pseudo-qualitative features. Cyber databases are dynamical, and features can attach new values, a priori unknown, every single day, even those features that are categorical. The rate of creation, storage, and analysis of the data implies a very high velocity. Also, it is not constant because the sources do not periodically update the reported information. With respect to veracity, cyber databases are volatile because of their short lifetime. Data quality is measured bearing in mind certain white noise in the transmission. Moreover, the data validity depends strongly on the accuracy of information and the reliability of data source. Regarding the valence of a cyber database, this is a dense set of data because we usually find high valence. Data are related to each other because they rely on real events reported by several agents. But these relations are not explicit and there are links among sources, types of attacks, reports, and incidents of the same type of attacks. Finally, the value is precisely the actionable knowledge that we can get from the cyber database from analyzing the quality of the data, automatized process, prediction of incidents, or detecting intrusion in different networks (see [30–32]).

Remark 25. In a cybersecurity context, we usually cannot design the whole acquisition process of data. Then, the task of cleaning data is always the first available stage of the procedure in which we can try to improve the efficiency. Usually, superficial redundancy is presented because the common data integration systems are not designed for cybersecurity reports.

Now we will analyze the superficial redundancy of the real cyber database we started describing at the beginning of this section, by applying the tool developed in Section 3.2. Recall that the database has 363 variables. Recall also that the clustering of the set of variables that defines the graph structure was done manually because we have improved the procedure with expert knowledge, leading to $\#X = 28$, $\#I = 24$, and $\#Y = 131$.

Table 7: Time needed to complete the tasks in Python and MapReduce.

Redundancy	Python	MapReduce
TypeI	13 sec	6 min 10 sec
TypeII	20 sec	6 min 3 sec
TypeIII	6 sec	6 min 21 sec

After applying the redundancy maps described in Section 2 on our sample, we obtain the redundancy levels that are shown in Table 6. The computation has been done in an accumulative way.

Results on Table 6 show that the most common type of redundancy in the sample is Type I. Also, the second type of redundancy more frequent in the sample is Type III. Both facts could be because the data integration engines are multisource and they are not designed specifically for cybersecurity reports.

The evolution of the associated graphs in the different cleaning stages is given in Figure 6, where connected green nodes represent the elements in X, the isolated green nodes represent the elements in I, and the yellow nodes represent the elements in Y.

The time from obtaining the initial matrices, the graph, and the level of redundancies that is taken for both, Python mode and MapReduce mode, is shown in Table 7

Result 1 (computational gain). After removing redundancies of types I, II, and III, we drop off about 64% of the rows of the original database. Hence the filtered database would be, roughly speaking, a third part of the original one.

4. Conclusions and Future Work

In this work, we have developed a novel graph approach for certain databases that allows computing the level of redundancy as the 1-norm of some adjacency and weighted matrices.

Furthermore, a tool (RTRR) to detect and remove some kind of redundancies has been presented and described, making use of the above theory.

Finally, this tool has been applied to a real cyber database made up by several sources, presenting a level of redundancy quite high and showing that, approximately, two-thirds of the data could be unuseful for further analysis.

As future work, we propose to model more types of redundancies, in particular, to face deep redundancies. Also, we will focus on improving the tool RTRR by creating a graphical interface that makes the tool more accessible to nonexpert users to create an online version.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors would like to acknowledge the Spanish National Cybersecurity Institute (INCIBE) that partially supported this work.

References

[1] D. Agrawal, P. Bernstein, E. Bertino, S. Davidson, and U. Dayal, "Challenges and Opportunities with Big Data," Tech. Rep., Cyber Center, Purdue University, West Lafayette: Purdue e-Pubs, 2011.

[2] R. T. Kouzes, G. A. Anderson, S. T. Elbert, I. Gorton, and D. K. Gracio, "The changing paradigm of data-intensive computing," *IEEE Computer Society*, vol. 42, no. 1, pp. 26–34, 2009.

[3] C. L. P. Chen and C. Y. Zhang, "Data-intensive applications, challenges, techniques and technologies: A survey on Big Data," *Information Sciences*, vol. 275, pp. 314–347, 2014.

[4] R. Baeza-Yates, "Big Data or Right Data," in *Proceedings of the 7th Alberto Mendelzon International Workshop on Foundations of Data Managements, (CEUR Workshop)*, p. 14, 2013.

[5] V. N. Gudivada, R. Baeza-Yates, and V. V. Raghavan, "Big data: Promises and problems," *The Computer Journal*, vol. 48, no. 3, pp. 20–23, 2015.

[6] I. Abaker, "The rise of big data on cloud computing," *Information Systems*, vol. 47, no. C, pp. 98–115, 2015.

[7] E. Rahm and H. H. Do, "Data cleaning: problems and current approaches," *IEEE Data Engineering Bulletin*, vol. 23, pp. 3–13, 2000.

[8] R. Srikant and R. Agrawal, "Mining Generalized Association Rules," in *Proceedings of the 21th International Very Large Data Bases (VLDB) Conference*, pp. 407–419, 1995.

[9] A. Savasere, E. Omiecinski, and S. Navathe, "An Efficient Algorithm for Mining Association Rules in Large Data- bases," in *Proceedings of the 21th International Very Large Data Bases (VLDB) Conference*, pp. 432–444, 1995.

[10] T. Milo and S. Zohar, "Using Schema Matching to Simplify Heterogeneous Data Translation," in *Proceedings of the 24th International Very Large Data Bases (VLDB) Conference*, pp. 122–133, 1998.

[11] S. Abiteboul, S. Cluet, T. Milo, P. Mogilevsky, J. Simeon, and S. Zohar, "Tools for data translation and integration," *IEEE Data Engineering Bulletin*, vol. 22, no. 1, pp. 3–8, 1999.

[12] H. Galhardas, D. Florescu, D. Shasha, and E. Simon, "Declaratively Cleaning your Data using AJAX," in *Proceedings of Journées Bases de Données Avancées (BDA) Conference*, 2000.

[13] M. L. Lee, H. Lu, T. W. Ling, and Y. T. Ko, "Cleansing Data for Mining and Warehousing," in *Lecture Notes in Computer Science*, T. J. Bench-Capon, G. Soda, and A. M. Tjoa, Eds., vol. 1677 of *Database and Expert Systems Applications. DEXA 1999*, pp. 751–760, Springer, Berlin, Germany, 1999.

[14] V. Raman and J. M. Hellerstein, "Potter's Wheel: An Interactive Data Cleaning System," in *Proceedings of the 27th Int Conference of Very Large Databases (VLDB)*, pp. 381–390, September 2001.

[15] Public GitHub repository of RIASC Tool for Removing Redundancies (RTRR), Version 1. Available in https://github.com/amunc/RIASC_Removing_Redundancies_Tools/tree/Python.

[16] Public GitHub repository of RIASC Tool for Removing Redundancies (RTRR), Version 2. Available in https://github.com/amunc/RIASC_Removing_Redundancies_Tools/tree/MapReduce.

[17] I. Robinson, J. Webber, and E. Eifrem, *Graph Databases. New Opportunities for Connected Data*, O'Reilly Media, 2nd edition, 2015.

[18] DataWrangler$^{\text{alpha}}$, Retrieved from http://vis.stanford.edu/wrangler/, 2013.

[19] OPEN Refine, Retrieved from http://openrefine.org/, 2016.

[20] TRIFACTA, Retrieved from https://www.trifacta.com/products/wrangler/, 2017.

[21] quadient DataCleaner, Retrieved from https://datacleaner.org, 2017.

[22] Winpure, Retrieved from http://www.winpure.com/cleanmatch.html.

[23] Microsoft Office Excel 2016, 2017 Retrieved from https://products.office.com/es-es/excel.

[24] Database Cleaner, Retrieved from http://www.database-cleaner.com, 2014.

[25] dedupeio/csvdedupe, Public GitHub repository, Retrieved from https://github.com/dedupeio/csvdedupe, 2017.

[26] CSVed, Retrieved from http://csved.sjfrancke.nl, 2017.

[27] Dedupe.io, Retrieved from https://dedupe.io/.

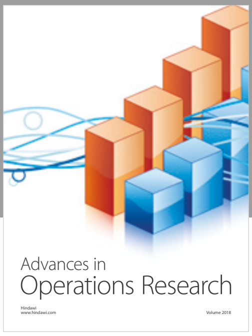
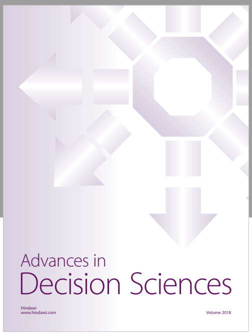
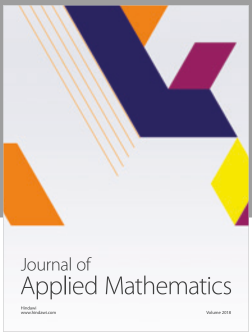
[28] BAYCASTLE SOFTWARE, DATASLAVE, Retrieved from http://www.baycastle.co.uk/v2/DataSlave/DataSlave.htm, 2010.

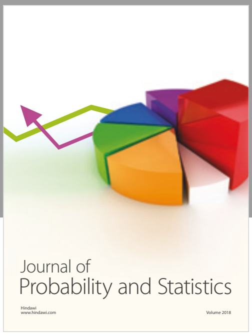
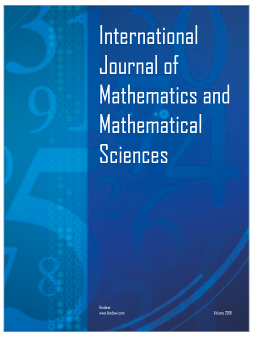
[29] datamartist, Retrieved from http://www.datamartist.com, 2017.

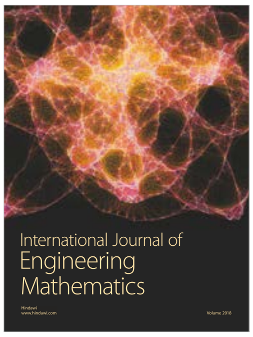
[30] R. Chen, W. Niu, X. Zhang, Z. Zhuo, and F. Lv, "An Effective Conversation-Based Botnet Detection Method," *Mathematical Problems in Engineering*, vol. 2017, Article ID 4934082, 9 pages, 2017.

[31] H. Zhu, W. Liu, M. Sun, and Y. Xin, "A Universal High-Performance Correlation Analysis Detection Model and Algorithm for Network Intrusion Detection System," *Mathematical Problems in Engineering*, vol. 2017, Article ID 8439706, 9 pages, 2017.

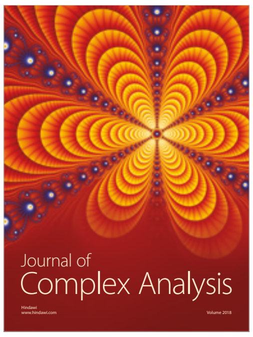
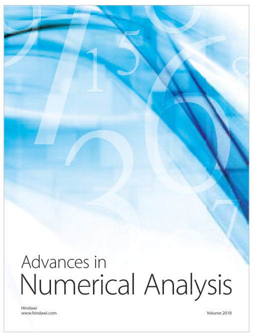
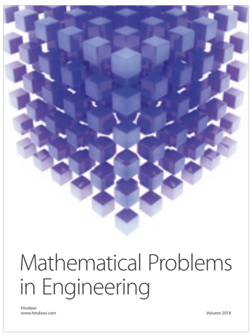
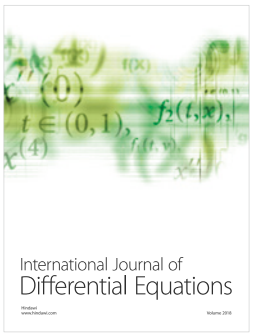
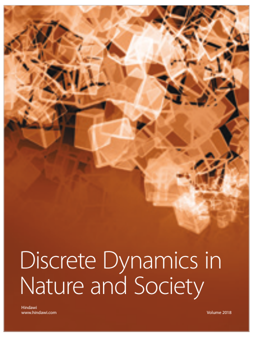
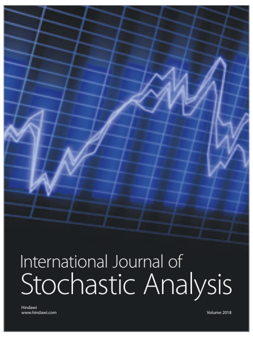
[32] J. R. Moya, N. DeCastro-García, R. A. Fernández-Díaz, and J. Lorenzana Tamargo, "Expert knowledge and data analysis for detecting advanced persistent threats," *Open Mathematics*, vol. 15, no. 1, pp. 1108–1122, 2017.


Advances in
Operations Research

Advances in
Decision Sciences

Journal of
Applied Mathematics

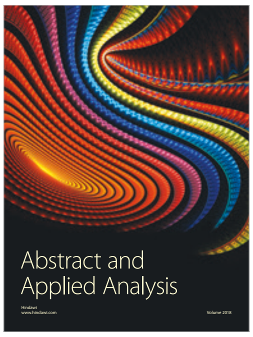
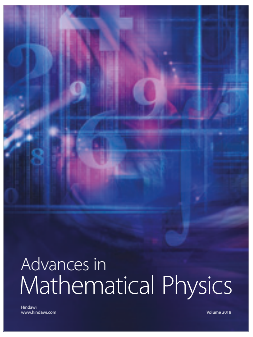
The Scientific
World Journal

Journal of
Probability and Statistics

International
Journal of
Mathematics and
Mathematical
Sciences

Journal of
Optimization

International Journal of
Engineering
Mathematics

International Journal of
Analysis


Journal of
Complex Analysis

Advances in
Numerical Analysis

Mathematical Problems
in Engineering

International Journal of
Differential Equations

Discrete Dynamics in
Nature and Society

International Journal of
Stochastic Analysis

Journal of
Mathematics

Journal of
Function Spaces

Abstract and
Applied Analysis

Advances in
Mathematical Physics