Anonylitics: From a Small Data to a Big Data Anonymization System for Analytical Projects

Alexandra Pomares-Quimbaya[1,3], Alejandro Sierra-Múnera[1,3], Jaime Mendoza-Mendoza[1,3], Julián Malaver-Moreno[1,3], Hernán Carvajal[2,3] and Victor Moncayo[2,3]

[1]*Pontificia Universidad Javeriana, Bogotá, Colombia*

[2]*Pontificia Universidad Javeriana, Cali, Colombia*

[3]*Center of Excellence and Appropriation in Big Data and Data Analytics (CAOBA), Bogotá, Colombia*

Keywords: Anonymization, Analytics, Data Mining, Data Science, Big Data, K-anonymity, Data Privacy, Information Disclosure.

Abstract: When a company requires analytical capabilities using data that might include sensitive information, it is important to use a solution that protects those sensitive portions, while maintaining its usefulness. An analysis of existing anonymization approaches found out that some of them only permit to disclose aggregated information about large groups or require to know in advance the type of analysis to be performed, which is not viable in Big Data projects; others have low scalability which is not feasible with large data sets. Another group of works are only presented theoretically, without any evidence on evaluations or tests in real environments. To fill this gap this paper presents Anonylitics, an implementation of the k-anonymity principle for small and Big Data settings that is intended for contexts where it is necessary to disclose small or large data sets for applying supervised or non-supervised techniques. Anonylitics improves available implementations of k-anonymity using a hybrid approach during the creation of the anonymized blocks, maintaining the data types of the original attributes, and guaranteeing scalability when used with large data sets. Considering the diverse infrastructure and data volumes managed by current companies, Anonylitics was implemented in two versions, the first one uses a centralized approach, for companies that have small data sets, or large data sets, but good vertical infrastructure capabilities, and a Big Data version, for companies with large data sets and horizontal infrastructure capabilities. Evaluation on different data sets with diverse protection requirements demonstrates that our solution maintains the utility of the data, guarantees its privacy and has a good time-complexity performance.

1 INTRODUCTION

Confidentiality comprises a set of rules and mechanisms that ensure that information is not made available or disclosed to unauthorized individuals, entities, or processes. Although released information for analytical purposes should be as detailed as possible to produce accurate models, data utility is in conflict with the confidentiality guarantee (Lee, 2015). As for utility, it typically refers to two features of the processed data: the ease of use of such data for data mining and other analysis purposes; and the correctness of conclusions drawn from the processed data by such analysis (Clifton and Tassa, 2013). To protect the confidentiality and the privacy of the entities (e.g. clients, patients, citizens) to which information refers, data holders often remove or encrypt explicit identifiers such as ids, names, and phone numbers. However, this de-identification process, although necessary, is not enough for guaranteeing entities anonymity; the main risk is to disclose pseudo-identifiers (e.g. gender, age, number of children), which individually are not risky, but when they are combined and linked with available data sources can be easily used by an aggressor to re-identify an entity (Ciriani et al., 2007).

Many principles and algorithms have been proposed for protecting data released from improper disclosure of confidential data. Principles like k-anonymity (Samarati and Sweeney, 1998), (Sweeney, 2002) and ε-differential privacy (Dwork et al., 2006) have been widely used to produce anonymized data from original data sets. According to the k-anonymity principle, a data set can be published if every record is indistinguishable from at least other k-1 records (k≥

Pomares-Quimbaya, A., Sierra-Múnera, A., Mendoza-Mendoza, J., Malaver-Moreno, J., Carvajal, H. and Moncayo, V.
Anonylitics: From a Small Data to a Big Data Anonymization System for Analytical Projects.
DOI: 10.5220/0007685200610071
In *Proceedings of the 21st International Conference on Enterprise Information Systems (ICEIS 2019)*, pages 61-71
ISBN: 978-989-758-372-8

2), within the data set with respect to the combination of pseudo-identifiers (or Quasi-Identifiers). It means that for every combination of pseudo-identifiers there must be at least k records in the disclosed data sets. This guarantees that the entities cannot be re-identified by linking an external data source with one specific record. Alternatively, the ε differential privacy principle states that privacy can be preserved by calibrating the standard deviation of the noise according to the sensitivity of a function f, where the function f describes the query in which the user is interested on, for instance, given a database D a user may be interested in the total number of 1's in the database (Dwork et al., 2006),(Dwork and Roth, 2014) . The goal of this principle is to enable the user to learn properties of the population as a whole while protecting their privacy (Dwork and Naor, 2010).

In the context of Big Data projects, an analysis of existing implementations on both privacy principles was made, finding out some drawbacks on both principles. On the one hand, the differential privacy is intended to disclose aggregated information, which is not always the case on Big Data projects, where the complete data set must be available. In addition, its core requires noise addition, which seems to run into considerable resistance among specialists (Clifton and Tassa, 2013). Furthermore, in practice it requires an interactive implementation that demands prior knowledge of the analysis to be performed by the user in order to calibrate the level of noise to the global sensitivity of the query and to the targeted differential privacy parameter (Clifton and Tassa, 2013). On the other hand, we found two main drawbacks on the k-anonymity (and its variants) implementations; the first one is their low scalability for a Big Data set, and the second one is the quantity of information loss, when some of the classical generalization approaches are applied on the data set. Although both have problems when they are applied on Big Data problems, we decided to choose the k-anonymity as the principle to guarantee data privacy because it works well in non-interactive environments and its implementations can be enhanced to achieve scalability and to reduce information loss using non-perturbative methods.

This paper presents Anonylitics, an implementation of the k-anonymity principle for Big Data settings. Anonylitics is intended for contexts where it is necessary to publish small or large data sets for applying supervised or non-supervised analytical techniques that require preserving the utility of the data. Anonylitics improved classical implementations of k-anonymity as Mondrian and Datafly using a hybrid approach during the creation of the blocks required to be of size at least k, and the replacement using a class representant. Throughout the design and implementation of Anonylitics to achieve k-anonymity in Big Data settings we have different aspects to deal with for transforming the version of Anonylitics that anonymizes small data sets (i.e. small enough to be processed in a single machine) into one version that can anonymize Big Data sets. These lessons learned are an important contribution of this paper.

The paper is structured as follows. Section 2 presents a summary of the main findings on the current anonymization approaches in Big Data. Then, Section 3 explains the strategy of Anonylitics to produce data sets compliant to the k-anonymity principle including its transformation from a centralized to a Big Data version. Section 4 includes the evaluation of our proposal demonstrating its good results on utility and privacy preservation. This sections also analyzes the performance results derived from different test setups. The final part of the paper presents in Section 5 the lessons learned, and in Section 4.4 a discussion of conclusions and future work.

2 RELATED WORK

When the confidential information is related to individuals, the right of privacy emerges. Privacy is the right that every individual must control or influence what information related to them can be collected and stored, by whom, and what information can be disclosed (Luisa Pfeiffer, 2008). This information is represented by attributes that can be classified into: (i) Identifiers: fields that uniquely identify a person (name, ids). (ii) Pseudo-identifiers: fields that do not directly identify, but combined with other pseudo-identifiers can identify an individual (age, sex, race). (iii) Sensitive data: fields with a certain value, on which conclusions are drawn in a subsequent analysis (medical condition, salary) (Lambert, 1993).

Preserving privacy in Big Data projects is one of the major concerns because it requires data to go outside the boundaries of the companies and involves pseudo-identifiers and sensitive data that need to be analyzed. With the aim of identifying the solutions to ensure privacy in Big Data contexts we analyzed related works on data anonymization in these environments.

The systematic review of the literature by B. Purcell (Purcell, 2013) presents four anonymization approaches for large volumes of data: conceptual, architectures, tools and algorithms. In the conceptual approach, anonymization is analyzed from a theoretical perspective where the objective is to evaluate the balance between information loss and data utility af-

ter anonymization. The architecture refers to the logical orchestration that allows generating anonymization systems, identifying two types of architectures: general purpose and specific applications. The tools refer to the practical application of principles, operations, algorithms and anonymization methodologies available to the general public, whether in open source or proprietary. It includes a commercial tool for anonymization purposes developed by a Japanese research group (Morisawa and Matsune, 2016). Finally, the algorithms are related to methodologies based on heuristics, which seek to find a balance between computational complexity and information privacy.

Regarding the classical algorithms developed for anonymization as Datafly (Sweeney, 1998), Incognito (LeFevre et al., 2005) and Mondrian (LeFevre et al., 2006), whose objective is to fulfill the principle of k-anonymity, we found that Mondrian is the most viable to be enhanced for Big Data problems. The Mondrian algorithm allows anonymizing tabular information through recursive partitioning of the data set applying the principle of k-anonymity. From the columns or dimensions that make up the data set, the user must define those that are considered as pseudo-identifiers. Recursively, the algorithm calculates its normalized range for each of the selected dimensions, which means that partitioning can only be done on numerical type dimensions. Once the ranges are calculated, the algorithm selects the dimension with the highest rank to partition. Partitioning is done by calculating the distribution median of the selected dimension, computed using the frequencies, and then creating two partitions, the first with values greater than or equal to the median and the second with values smaller than this same value. On these two new partitions, the algorithm is executed again until the partition is less than twice the size of k. When this occurs, it means that the algorithm will not be able to further partition the data, since the principle of k-anonymity would not be met. Once the partitions are created, the generalization process is carried out, in which for every partition, a representant of the class or data to be replaced in each pseudo-identifier is selected.

Concerning the scalable implementation of these classical algorithms, we have found different proposals. However, they are more focused on large data sets that can be processed in a single machine (e.g. thousands of records). In the centralized approach, LeFevre, DeWitt and Ramakrishnan propose two (2) algorithms Top-Down Specialization (TDS) Greedy Algorithms based on LeFreve's Mondrian algorithm (LeFevre and DeWitt, 2007): Rothko-T or Rothko-Tree and Rothko-S or Rothko-Sampling. Rothko-T is a variation of the Mondrian algorithm that gives special consideration when a data set is larger than the size of the memory. Rothko-S is a variation in the sampling of the Rothko-T algorithm applied to data sets aimed at obtaining greater performance. In the distributed approach, Zhang, Yang, Liu and Chen present two works (Zhang et al., 2014b), (Zhang et al., 2014a): Map Reduce Top-Down Specialization (MRTDS), and Map Reduce Bottom- Up Generalization (MRBUG). MRTDS anonymizes data partitions to generate a level of intermediate anonymization, where it first initializes the data utility metrics, and then, with an iterative algorithm, selects the best specialization according to the level of anonymization and does so, updating the anonymization level. MRBUG initializes a data utility metric and makes multiple generalizations in an iteration round, to improve the degree of parallelization and efficiency, until the anonymized data set complies with the principle of k-anonymity. In addition, Zhang et al. presents the Single-Objective Proximity-Aware Clustering algorithm (SPAC) (Zhang et al., 2015). It is a technique adapted to a distributed system based on files such as HDFS; and, therefore, can be applied to a large volume of data. On the other hand, Sowmya and Nagaratna introduce the Parallelizing k-anonymity Algorithm (Pk-a) (Sowmya and Nagaratna, 2016), where the implementation of the principle of k-anonymity is redefined using distributed programming, MapReduce and the Hadoop ecosystem. Then, Zhang et al. propose Locality Sensitive Hashing based on local-recoding (LSH) (Zhang et al., 2016), which is a scalable local recoding approach using distance metrics for categorical data (Jaccard distance) and numerical data (Euclidean distance) for measuring the similarity between the registers. Zhang et al. propose MRMondrian (Map-Reduce jobs on Mondrian) (Zhang et al., 2016), where the objective is to partition the data iteratively, into smaller subsets, until they all fit into the memory at each processing node. The work presented in (El Ouazzani and El Bakkali, 2018) proposes a k-anonymity approach without prior value of the threshold k. However, its complexity is not calculated. Eyupoglu et al. propose an algorithm based on chaos and perturbation techniques (Eyupoglu et al., 2018) to preserve the privacy and utility in large data independent of the type of data set, and can be applied to both numerical and categorical attributes, but so far has not been used in real applications. Finally, Sopaoglu and Abul developed an ascending and scalable anonymization algorithm for Apache Spark (Sopaoglu and Abul, 2017). Basically they reimplemented the TDS (top-down spe-

cialization) algorithm for the Apache Spark platform, being an experimental evaluation that suggests that it would be highly scalable for TDS.

From the review it was evidenced that some of the developments in Big Data do not delve into the details on the algorithm and/or code, this is understandable when dealing with proprietary software, however, it does not allow to measure the confidence in the results(Morisawa and Matsune, 2016). Besides, most of the described algorithms were tested with a data set that does not correspond to large volumes of information, which does not comply with the Volume characteristic of Big Data projects (Zhang et al., 2015) (Sowmya and Nagaratna, 2016). Finally, some of the algorithms mentioned above are presented theoretically, and there is no evidence on evaluations or tests that they can be used in real environments (Sopaoglu and Abul, 2017).

3 ANONYLITICS

Taking into account the need of anonymizing information while preserving its utility for analytical purposes, an algorithm capable of anonymizing numerical and categorical data maintaining the principle of k-anonymity in Small and Big Data environments was developed. This algorithm, called Anonylitics, is based on the Mondrian algorithm (see Section 2), but incorporates a series of improvements aimed to preserve the usefulness of the anonymized information and to guarantee its scalability. Two approaches were carried out, the first one for processing low volume data sets, and the second one providing scalability qualities for Big Data problems.

3.1 Small Data Version: Centralized Implementation

The centralized version of Anonylitics guarantees the anonymization using the k-anonymity principle, at the same time that it keeps the usefulness of data for analytical purposes. An important aspect of our proposal is that it does not change the data type of the original pseudo-identifier, as most of the existing approaches do, and aims to preserve the distribution of numerical values. The main phases of Anonylitics are presented in Figure 1. As it can be seen, our proposal includes a series of improvements that allow to guarantee the privacy and utility of the input data. The main improvements are:

1. Partitioning. The partition is made calculating the median of the cut dimension whenever it is possible to partition it (step A). In addition, when it has checked all the dimensions and can not find a partition that can be divided by the median, it starts over to review from the first dimension with greater entropy, in order to create the partition using the k-first elements that comply with the anonymity k (step D). This hybrid partitioning approach led us to reduce the information loss, because it reduces the size of the blocks, at the same time that it guarantees the K-anonymity principle.

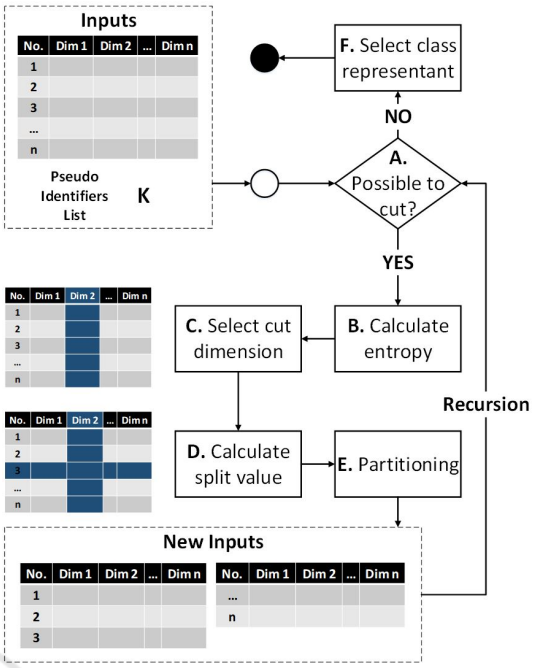

Figure 1: Anonylitics Centralized Implementation.

2. Selection of the cutting dimension. Our algorithm uses the entropy (step B) to find the dimension on which the data set is recursively partitioned. It means that, by each recursion, entropy is calculated for each pseudo identifier and the partition is made by the one with highest entropy value. When the partition can not be made in the selected dimension, the algorithm selects the second pseudo-identifier with highest entropy and try to partition the data using that dimension. This is done until all dimensions are evaluated, but, if partitioning can not be achieved, it means that the k-first principle is fulfilled. Using entropy allows us to partition not only using numerical dimensions, but also those of categorical type.

3. Categorical data hierarchy. Our proposal allows to partition taking into account numerical and categorical data. Unlike related works, which concatenate the different values in the partition and establish them as class representants, a hierarchy is associated with each categorical attribute. This

hierarchy allows having a tree structure, where the leaves are the original categorical values and the nodes values group together different levels of the categorical data as shown in Figure 2. When carrying out the generalization process (Step F), a class representant is not calculated, but the categorical value is replaced by the one that precedes it in the hierarchy until the principle of k-anonymity is fulfilled. This aspect ensures the utility of the anonymized categorical values.

Level 0	Level 1	Level 2
Miami	Florida	United States
Fort Lauderdale	Florida	United States
Boston	Massachusetts	United States
Springfield	Massachusetts	United States
Harburgo	Hamburgo	Germany
Altona	Hamburgo	Germany
Munich	Baviera	Germany
Augsburgo	Baviera	Germany

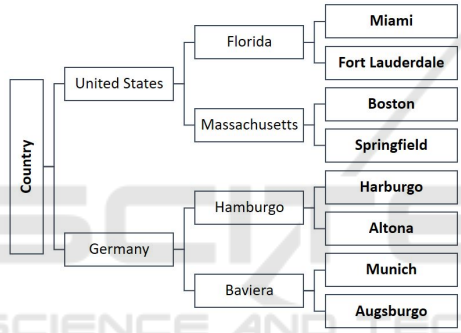

Figure 2: Cities Hierarchies Implementation Example.

3.2 From a Centralized to a Distributed Implementation: Challenges

As stated by Katsogridakis et al., Spark, and in general Map Reduce based systems, are not suited for recursive algorithms, mainly due to the distributed nature of the code and the data (Katsogridakis et al., 2017). Data resides in RDDs (Resilient Distributed Data set) (Zaharia et al., 2012) or dataframes (RDDs with named columns) distributed across the cluster while the code and the metadata reside on the drive node. Therefore, recursive algorithms need to either be rewritten as iterative algorithms or the job scheduler needs to be changed as proposed in (Katsogridakis et al., 2017). In our case we aim to write an algorithm that can be executed on a standard Big Data environment and thus we opt for switching to an iterative algorithm.

This decision brought additional challenges because iterative algorithms could lead to space complexities. In our case, after each iteration, the partitions defined by the previous step are not represented in the original data set because dataframes and RDDs in Spark are immutable. The consequence is having twice the data after each iteration and forcing the algorithm to constantly create checkpoints or caching intermediate resources. In the case of Spark jobs, transformations are not executed until an action is called and therefore this lazy execution, if not managed correctly, could result in big RDDs or Dataframes created when an action is executed.

As we convert the algorithm to an iterative algorithm, after each iteration data has to be split in different logical partitions. We could either create one new dataframe containing all the data and using an attribute value to store the logical partition, or splitting the dataframe into multiple dataframes, each one containing a partition. The second approach has the inconvenience of building the new dataframes with the help of filters over the original dataframe, but having to iterate over the data multiple times, one for each new partition.

Another challenge related to the algorithm involves sorting inside a partition or calculating row numbers within a partition. For instance, finding the first k elements of a partition requires ordering within the partition all the rows given an ordering attribute. In Spark, with window functions, such action requires that the whole partition is loaded in one worker's memory and locally sorting the rows, but in the case of a large data set partitions could also be very large, particularly in the first iterations, where partitions could potentially be half the entire data set and therefore the whole partition could not fit in one worker's memory. One example of such action is computing the median of a partition. This requires sorting the partition rows and finding the middle point of the data set. In the last phase of the algorithm a class representant is chosen. One of the original approaches in the centralized algorithm is choosing the representant given a truncated distribution, sampling from the partition given the global distribution. Such process, which is easily implemented in a centralized python process, has issues in a distributed Spark environment. This could be achieved by computing global distributions, collecting all the data from a partition and then choosing inside the driver node the representant for each class. This has the drawback of processing the complete data set in one node and scalability is compromised.

3.3 Big Data Version: Distributed Implementation

All the challenges presented in Section 3.2 were solved adapting the centralized anonymization version to distributed environments managing large data sets stored in a Big Data environment. In particular, the environment on which this solution was developed was a Hadoop cluster with a distributed HDFS storage and an Apache Spark processing component in which the algorithm was written. This technology restricts us in the use of recursion in the process of anonymization, therefore, we develop a series of changes with respect to the centralized version that are illustrated in Figure 3.

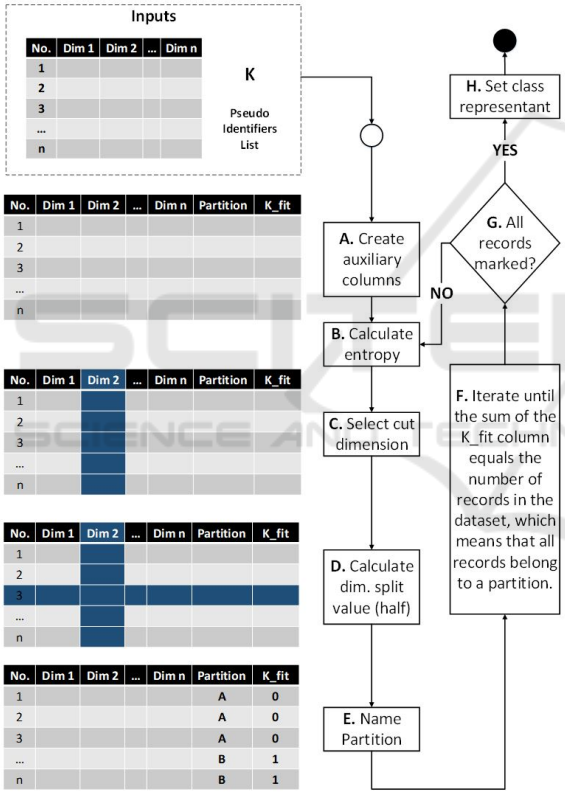

Figure 3: Anonylitics Distributed Implementation.

1. Anonymization strategy. Since the use of recursion over Spark is not recommended or efficient, it was necessary to implement a proposal that performs the anonymization process iteratively (F). The algorithm uses the Spark's data structures RDDs and dataframes on which all operations are performed.
2. Identification of partitions. In each iteration of the process a new dataframe is created with a new column, which indicates the partition to which each record belongs (A). It is always necessary to create new dataframes since the data stored in the RDDs is immutable.
3. Entropy calculation. The entropy is calculated by each pseudo-identifier in each partition creating auxiliary data frames, which allow grouping and aggregation operations on the data (B).
4. Partitioning. The partition process is done by selecting the highest entropy (B, C) pseudo-identifier and dividing it in 2 partitions using the mid-point (D).
5. Calculating the value of the median is a computationally expensive task that Spark must perform. In contrast to the centralized version of the algorithm, the data set is not divided into partitions, but in each iteration the records are marked with an identifier corresponding to the partition to which it belongs (E).

The architecture of Anonylitics is based on a client server model as shown in Figure 4. It consists of three main components that allow Anonylitics to be used both, in its centralized version and in its distributed version.

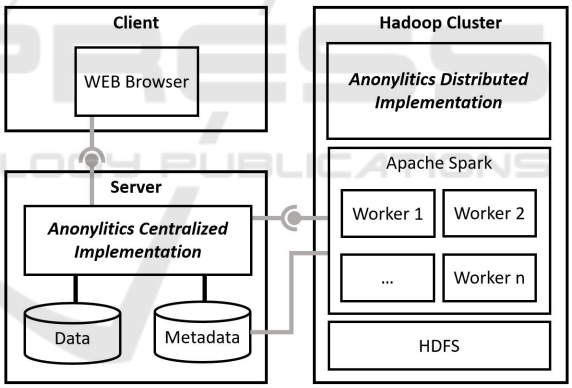

Figure 4: Anonylitics Architecture.

The web client contains the front end of Anonylitics and where the users parameterize and execute the anonymization processes. The Anonylitics server is where the anonymization processes are orchestrated. In the centralized version, the anonymization is executed in this same server. It also stores the anonymized data in the centralized version, and the metadata of the anonymization projects. Finally, the Hadoop Cluster allows us to execute the distributed anonymization component and provides a distributed file system and the Spark processing component.

4 ANONYLITICS EVALUATION

In order to compare the big data and the small data implementations, we ran two sets of tests. One set evaluates the quality of the anonymization process and a second set was intended to evaluate the performance by measuring the running time under different configurations.

This section is divided into three parts: the experimental setup description, the evaluation of the anonymization process, and the performance evaluation.

4.1 Experimental Setup

The centralized version presented in section 3.1 was implemented in Python 3.5 with a MySQL 5.7 database. For performing Big Data experiments, a Hadoop Distributed File System (HDFS) was used. The code was implemented in Python 3.5, using Apache Spark 2.2.

All the experiments were conducted on a virtualized cluster that consists of 9 machines. Each machine had four 2.30GHz Intel Xeon processors, 32GB of memory, 1TB IDE disks, and a gigabit Ethernet link.

Tests were run on multiple sized synthetic data sets, all of them composed by one categorical and five numerical columns. The categorical attribute contains a random sample with cities. And the numerical quasi-identifiers were generated following a normal distribution.

4.1.1 Anonymization Evaluation Setup

Multiple tests were performed varying the k value and using the same data set and cluster configuration. The values of k ranged from 24 to 29 and the table with data contained 1 million records.

4.1.2 Performance Evaluation Setup

In order to measure the performance of the Big Data approach, three different setup variations were tested: changing the k-anonymity constant, changing the dataset size and changing the cluster configuration:

1. K-anonymity: the K-anonymity parameter was changed in powers of 2 from 2 to 512 and the dataset sized was fixed to a 1.000.000 rows dataset.
2. Data Size: the K-anonymity was maintained constant in the worst performance case K=2 (higher number of iterations) and the data set size was increased from 10.000 to 10.000.000 registers.

Table 1: Performance evaluation parameters setup.

	Experiments	Parameter settings	Dataset size
1	K-anonymity	K = 2, 4, 8, 16, 32, 64, 128, 256 and 512	10^6
2	Data Size	K = 2	10^4, 50^4, 10^5, 50^5, 10^6, 50^6, 10^7
3	Cluster Configuration	K = 2 Executors = 8 Cores per Executor=1-4	10^6

3. Cluster Configuration: there are mainly three parameters related to the configuration of the cluster, the number of YARN containers or executors to allocate for this application, the number of processor cores to allocate on each executor, and the maximum heap or memory size to allocate to each executor. In this experiment, we used one executor for each container node (8 in total), and the number of cores were modified for each Spark executor increasing it from 1 to 4. The 1.000.000 observations data set and a value of K equal to 2 were maintained constant.

Table 1 shows the configuration of the cluster used to measure the scalability of the algorithm.

4.2 Anonymization Evaluation

The goal of this section is to evaluate whether the Big Data approach affects the quality of the anonymization process. To do so, the Information Loss metric (IL from now on), introduced at (Byun et al., 2007) was used to evaluate the quality of the anonymized table. This metric estimates the data utility by quantifying the amount of data values that have been generalized.

Let E be the set of clusters produced by an anonymization algorithm, and for each record x in each equivalent class (or cluster) e, let $x_1, ..., x_D$ denote the values of the attributes of x. Then the amount of IL introduced by e, is defined as:

$$IL(e) = |e| \left(\sum_{i=1}^{D} \frac{max\{x_i : x \in e\} - min\{x_i : x \in e\}}{|\{x_i : x \in e\}|} \right) \quad (1)$$

and the total information loss of the anonymized table as:

$$Total - IL(E) = \sum_{e \in E} IL(e) \quad (2)$$

Figure 5 shows the information loss obtained using different k values. As expected, it can be observed how the information loss score tends to increase, since the attributes were generalized further, which reduced the utility of the data set. This result confirms that the Big Data approach does not affect the quality of the anonymization process.

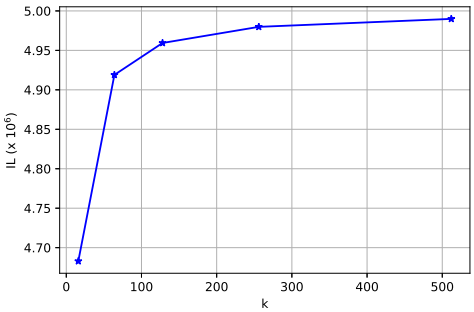

Figure 5: Information loss evaluation.

4.3 Performance Evaluation

Figures 6, 7 and 8 present the time performance of Anonylitics based on k-anonymity, dataset size and cluster configuration, respectively. As expected, a visible relationship between run time and K-anonymity was found. Higher values of K influence directly over the total number of iterations. We got an execution time reduction of 46%, from 69 min (k=2, 20 iterations) to 37 min (k=512, 12 iterations).

In the second experiment (Figure 7), we present the time complexity performance in terms of dataset size. As the number of rows in the data increases, so does the time, but the results show that final behavior is better than quadratic $O(n^2)$, exponential $O(2^n)$ and linear $O(n)$ functions. When data size grows 10 times from 10.000 to 100.000, the run time grows 2,01 times; but, when the number of observations increases from 10.000 to 10.000.000 (1000 times bigger) the run time is only 14,7 times higher. This shows that the Anonylitics algorithm has a good time-complexity performance, providing an acceptable behavior with big datasets.

Finally, our third experiment shows the scalability capabilities of distributed computation in Anonylitics (Figure 8). While the core number in each executor is increased to its maximum value (4 cores in each node for our assembly), the final time is reduced in 59%, from 86 to 35 min. We can improve the performance results not only by increasing the number of cores by executor, but, adding more nodes into the cluster and optimizing the execution configuration.

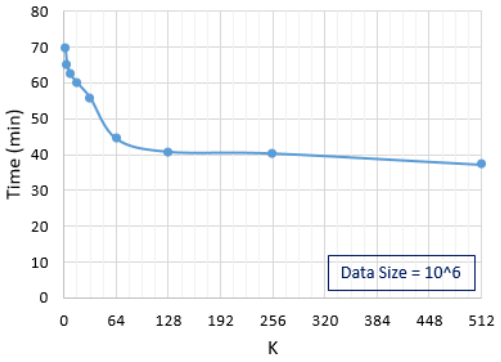

Figure 6: K-Anonymity performance evaluation.

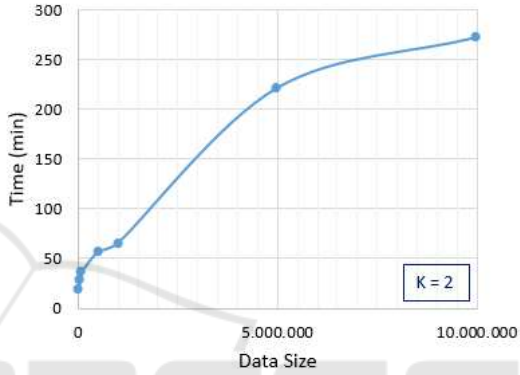

Figure 7: Dataset size performance evaluation.

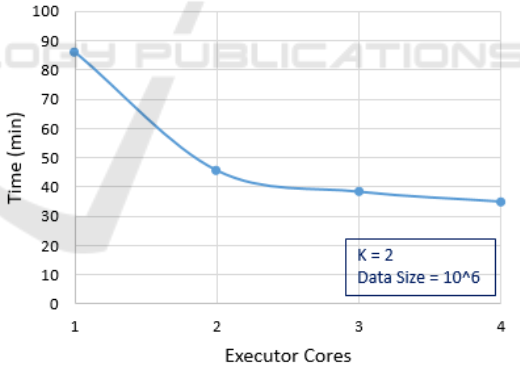

Figure 8: Cluster configuration performance evaluation.

4.4 Comparison with Related Works

A qualitative comparison was made with the Top-Down Specialization algorithm (TDS) (Fung et al., 2005) and its Big Data implementations: Two Phase TDS (Zhang et al., 2014b) for Hadoop (Map-Reduce), and TDS for Apache Spark (Sopaoglu and Abul, 2017).

The main difference between the two algorithms is the partitioning process, TDS approaches generalizes the best attribute within each iteration and over the

entire column until satisfying k-anonymity. On the other hand, our implementation first builds iteratively a group of partitions composed by at least k elements, and later, it implements what we defined as local and global generalization process. Local means, that we could generalize each partition individually and for that reason, we could have observations at different hierarchy levels.

TDS proposals and our algorithm support both types of input data: numerical and categorical, however, TDS transforms the continuous pseudo-identifiers into discrete values by grouping them into numerical ranges. Changing its output data type could impact the later use of analytical and data mining techniques, especially when we want to evaluate the information loss and the utility of the anonymized data. Further transformations will be necessary if we want to compare initial and final data distributions, or measuring the information change with techniques like Kullback-Leibler divergence.

On the contrary, our algorithm is capable of maintaining the data type, keeping all the variables together in the partitioning process, but carrying out a different generalization approach depending on the variable type. The categorical variables are anonymized based on a hierarchical approach, thus a domain hierarchy or tree is needed. Instead, numerical values in each partition are changed by a class representant, like the average, the median or a random element within the group.

Both anonymization techniques use information theory or entropy in order to select the best pseudo-identifier, nevertheless TDS is much more computationally intensive because it calculates the information loss or the entropy difference before and after the variable is generalized. On the other hand, our approach only requires the initial entropy among all the pseudo-identifiers.

5 LESSONS LEARNED: FROM SMALL TO BIG DATA ANONYMIZATION APPROACH

Converting an algorithm that was designed to run on a single node into a scalable algorithm introduced some challenges to our anonymization system. This demonstrates that not all centralized approaches are easily implemented in a distributed environment. Distributed implementations of originally centralized algorithms may involve an overhead that has to be managed when deciding which implementation to use. Such decision depends on the data volume and the vertical or horizontal scalability capabilities of the infrastructure. Companies must profile and understand their data and determine whether a single machine could process all the data, considering its growth over time. If not, horizontal scalability is needed, and a distributed algorithm is the right solution.

Additionally, regarding anonymization approaches, we proposed a way of taking into account both, numerical and categorical variables when deciding how to partition a data set. This is valuable because real world data sets have many types of variables, which must be considered in an anonymization process, to produce useful, but at the same time confidential datasets. The nature of the data is important in our system because the utility of the anonymized data is essential, therefore we preserve the data types and domains in all the process.

From the point of view of the infrastructure, Big Data software like Hadoop and Spark are complex platforms that could be parametrized in many different ways depending on the characteristics of the data to be processed and the physical infrastructure available. Building software with such technologies does not only involve the process of writing and testing the code but also understanding their architecture and adjusting it according to the algorithms and data.

Finally, it is important to remark that when an organization requires analytic capabilities that take advantage of vast amounts of data, and such data might include sensitive information, it is vital to understand all the variables contained in the data set and protect the information before making it available to external analysts.

Likewise, data analysis processes require useful information, representing real patterns and real populations in order to provide good insights to be used by decision makers. Using an anonymization system that protects while maintaining the data utility is essential, especially when it comes to information about clients, patients, citizens, etc. or information about other companies, which need to be analyzed externally or even internally by people who do not have the right to see the original data.

6 CONCLUSIONS AND FUTURE WORK

We proposed an implementation of the k-anonymity principle called Anonylitics with two versions, a centralized and a distributed version, designed for small and Big Data scenarios. Our algorithm not only works for numerical data but also is designed to consider categorical data as pseudo-identifiers. Related works

described different anonymization approaches that focus on specific algorithms or specific platforms. Unfortunately, some of them have no implementation or have not been tested with large datasets.

Converting the originally centralized algorithm to a distributed algorithm using the Apache Spark platform, brought to light some challenges related to the recursive nature of the algorithm and the specific data transformation capabilities of Spark. Although an alternative is to make changes on the platform like Katsogridakis et al. did in (Katsogridakis et al., 2017), this is not possible in most of the companies. Instead, we proposed, implemented and tested solutions to these challenges, analyzing the final performance and data utility. Our implementation allows companies with huge data-sets to anonymize them in order to perform analysis tasks with the anonymized dataset protecting confidentiality.

As future work it is important to test our algorithm in a more controlled environment with dedicated machines instead of virtual machines, to avoid sharing resources between virtual machines that may impact the overall execution performance. Those tests should also vary the cluster size and the size of the files used, in order to have better understanding of the scalability and the overall performance of the approach proposed. Additionally, it will provide more concrete results that will serve us to further compare in depth the different approaches.

Further future work includes integrating the algorithm in streaming data for velocity and also including unstructured data anonymization, which are also important aspects of Big Data strategies.

ACKNOWLEDGEMENTS

This research was carried out by the Center of Excellence and Appropriation in Big Data and Data Analytics (CAOBA). It was funded partially by the Ministry of Information Technologies and Telecommunications of the Republic of Colombia (MinTIC) through the Colombian Administrative Department of Science, Technology and Innovation (COLCIENCIAS) contract No. FP44842- anex46-2015.

REFERENCES

Byun, J.-W., Kamra, A., Bertino, E., and Li, N. (2007). Efficient k-anonymization using clustering techniques. In *International Conference on Database Systems for Advanced Applications*, pages 188–200. Springer.

Ciriani, V., di Vimercati, S. D. C., Foresti, S., and Samarati, P. (2007). Microdata protection. In Yu, T. and Jajodia, S., editors, *Secure Data Management in Decentralized Systems*, volume 33 of *Advances in Information Security*, pages 291–321. Springer.

Clifton, C. and Tassa, T. (2013). On syntactic anonymity and differential privacy. *Trans. Data Privacy*, 6(2):161–183.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. D. (2006). Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography, Third Theory of Cryptography Conference, TCC*, pages 265–284.

Dwork, C. and Naor, M. (2010). On the difficulties of disclosure prevention in statistical databases or the case for differential privacy. *Journal of Privacy and Confidentiality*, 2:93–107.

Dwork, C. and Roth, A. (2014). The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3–4):211–407.

El Ouazzani, Z. and El Bakkali, H. (2018). A new technique ensuring privacy in big data: K-anonymity without prior value of the threshold k. *Procedia Computer Science*, 127:52–59.

Eyupoglu, C., Aydin, M. A., Zaim, A. H., and Sertbas, A. (2018). An efficient big data anonymization algorithm based on chaos and perturbation techniques. *Entropy*, 20(5):373.

Fung, B. C., Wang, K., and Yu, P. S. (2005). Top-down specialization for information and privacy preservation. In *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on*, pages 205–216. IEEE.

Katsogridakis, P., Papagiannaki, S., and Pratikakis, P. (2017). Execution of recursive queries in apache spark. In *European Conference on Parallel Processing*, pages 289–302. Springer.

Lambert, D. (1993). Measures of disclosure risk and harm. *JOURNAL OF OFFICIAL STATISTICS-STOCKHOLM-*, 9:313–313.

Lee, C. (2015). Security in telecommunications and informationtechnology. Technical report, ITU-T – Telecommunication Standardization Bureau (TSB).

LeFevre, K. and DeWitt, D. (2007). Scalable anonymization algorithms for large data sets. *Age*, 40:40.

LeFevre, K., DeWitt, D. J., and Ramakrishnan, R. (2005). Incognito: Efficient full-domain k-anonymity. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 49–60. ACM.

LeFevre, K., DeWitt, D. J., and Ramakrishnan, R. (2006). Mondrian multidimensional k-anonymity. In *Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on*, pages 25–25. IEEE.

Luisa Pfeiffer, M. (2008). The right to privacy. protecting the sensitive data. *REVISTA COLOMBIANA DE BIOETICA*, 3(1):11–36.

Morisawa, Y. and Matsune, S. (2016). Nestgate—realizing personal data protection with k-anonymization technology. *FUJITSU Sci. Tech. J*, 52(3):37–42.

Purcell, B. (2013). The emergence of" big data" technology and analytics. *Journal of technology research*, 4:1.

Samarati, P. and Sweeney, L. (1998). Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical report.

Sopaoglu, U. and Abul, O. (2017). A top-down k-anonymization implementation for apache spark. In *Big Data (Big Data), 2017 IEEE International Conference on*, pages 4513–4521. IEEE.

Sowmya, Y. and Nagaratna, M. (2016). Parallelizing k-anonymity algorithm for privacy preserving knowledge discovery from big data. *International Journal of Applied Engineering Research*, 11(2):1314–1321.

Sweeney, L. (1998). Datafly: A system for providing anonymity in medical data. In *Database Security XI*, pages 356–381. Springer.

Sweeney, L. (2002). k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570.

Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., Franklin, M. J., Shenker, S., and Stoica, I. (2012). Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pages 2–2. USENIX Association.

Zhang, X., Dou, W., Pei, J., Nepal, S., Yang, C., Liu, C., and Chen, J. (2015). Proximity-aware local-recoding anonymization with mapreduce for scalable big data privacy preservation in cloud. *IEEE transactions on computers*, 64(8):2293–2307.

Zhang, X., Leckie, C., Dou, W., Chen, J., Kotagiri, R., and Salcic, Z. (2016). Scalable local-recoding anonymization using locality sensitive hashing for big data privacy preservation. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 1793–1802. ACM.

Zhang, X., Liu, C., Nepal, S., Yang, C., Dou, W., and Chen, J. (2014a). A hybrid approach for scalable sub-tree anonymization over big data using mapreduce on cloud. *Journal of Computer and System Sciences*, 80(5):1008–1020.

Zhang, X., Yang, L. T., Liu, C., and Chen, J. (2014b). A scalable two-phase top-down specialization approach for data anonymization using mapreduce on cloud. *IEEE Transactions on Parallel and Distributed Systems*, 25(2):363–373.