Learning Light-Weight Edge-Deployable Privacy Models

Yeon-sup Lim, Mudhakar Srivatsa, Supriyo Chakraborty
IBM T. J. Watson Research Center
Yorktown Heights, New York, USA
Email: y.lim@ibm.com, msrivats@us.ibm.com, supriyo@us.ibm.com

Ian Taylor
Cardiff University
CF10 3XQ, Cardiff, UK
Email: ian.j.taylor@cs.cardiff.ac.uk

***Abstract*—Privacy becomes one of the important issues in data-driven applications. The advent of non-PC devices such as Internet-of-Things (IoT) devices for data-driven applications leads to needs for light-weight data anonymization. In this paper, we develop an anonymization framework that expedites model learning in parallel and generates deployable models for devices with low computing capability. We evaluate our framework with various settings such as different data schema and characteristics. Our results exhibit that our framework learns anonymization models up to 16 times faster than a sequential anonymization approach and that it preserves enough information in anonymized data for data-driven applications.**

I. Introduction

Data-driven applications have proliferated in the last decade, e.g., recent Internet applications such as Amazon and Netflix provide customized services based on collected user data, AI machines like IBM Watson learn how to diagnose diseases based on medical records, and Internet-of-Things (IoT) applications such as smart home energy management utilize user profiles like household energy consumption patterns. The advent of these data-driven applications raises privacy concerns to protect personal identifiable or sensitive information such as purchasing history and healthcare records.

Data anonymization is the process that obfuscate data to ensure that released data does not disclose user identities, but does provide sufficient information for applications. To this end, several privacy metrics have been proposed such as k-anonymity [19], l-diversity [12], and t-closeness [10]. However, data anonymization using such privacy metrics introduces several concerns:

1) **Operating point of the anonymizer**: Who should perform anonymization process? Since statistical distributions of attributes are required to obtain a certain amount of privacy according to the privacy metrics, only data collectors such as service providers can properly apply anonymization. How to deploy an anonymization function close to users?
2) **Scalability**: Data anonymization needs to work over large volume of data in order to find data obfuscation rules to satisfy privacy metrics, incurring significant computation overhead.
3) **Model validity**: Data distributions can change over time. New collected data can be hard to be appropriately processed by a current anonymization model; changes in data characteristics incurs re-learning of the model.
4) **Utility of anonymized data**: Do anonymized data guarantee enough information for application providers? How worse applications perform with anonymized data?
5) **Discriminative attribute identification**: What attributes in records should be obfuscated? Each attribute can have discriminative power or not depending on its distribution. For example, if zip codes in all records have a same value, it does not provide any useful information to identify user or sensitive information. However, if a zip code is unique across records, the it can be used to distinguish a specific record.

In this work, we develop a data anonymization framework to address these concerns. Our framework utilizes Apache Spark [22] to quickly build an edge-deployable anonymization model for a huge amount of data. Once a model is built, our framework enables even users or IoT edge devices to obfuscate their own records without large computation overhead and knowledge on entire data. To reflect time-varying natures of data, our framework also provides a verification function to validate whether an anonymization model satisfies desired data privacy constraints. Thus, data administrators need not to continuously compute anonymization model for incoming records; they train an anonymization model again only when a current model fails to pass the validation procedure.

The rest of this paper is organized as follows: Section II provides the context for our work. We describe our framework in Section III. Section IV demonstrates the evaluations of our framework. Related work is reviewed in Section V, and we conclude in Section VI.

II. Background

A. Anonymization Techniques

Record attributes are usually divided into three types: *Personal Identifiable Information* (PII) that uniquely identifies a record (associates it with a specific user), *Quasi-Identifier* (QI) that does not individually identify a record but when combined with other QIs can still identify the record uniquely, such as age, sex, and marital status, and *Sensitive Information* (SI) that should not be linkable to any unique individual such as medical condition.

978-1-5386-5035-6/18/$31.00 ©2018 IEEE

To prevent user identity disclosure, several privacy metrics have been proposed. Our framework supports selected privacy metrics such as k-anonymity [19], l-diversity [12], and t-closeness [10].

The key idea behind anonymization is to obfuscate QIs such that different records can be grouped together into a same equivalence class. Thus, records within an equivalence class are indistinguishable with respect to their QIs, i.e., it is not possible to uniquely link their corresponding sensitive information to a specific user. To this end, data anonymization algorithms employ "data generalization" and "data suppression" [18].

- **Data generalization** of an attribute value is an operation in which specific value taken by the attribute is reported as an interval containing the value to increase its uncertainty and provide privacy.
- **Data suppression** is the process of removing some records from the released data. Typically, the suppressed records are ones with outlier attribute values; including such records results in unnecessarily large equivalence classes.

However, both of data generalization and suppression reduce overall data utility, i.e., loss of data effectiveness [2]. Therefore, the goal of anonymization algorithms is to find the best data obfuscation which guarantees a particular level of the privacy metrics while maximizing data utility. In our framework, we use application-independent utility measures such as:

- Entropy-based utility: Entropy is a measure of average information in the dataset. Entropy is highest when none of the records are grouped together and decreases as equivalence classes are formed. Denote the total number of records as N and the number of records in equivalence class i as n_i. The entropy-based utility of anonymized records is defined as $\sum_i -\frac{n_i}{N} \log \frac{n_i}{N}$
- Height-based utility: Assume that the obfuscation levels for quasi-identifiers are step-wise and each has a maximum level. The loss of utility is given by how higher obfuscation levels are applied to QIs. Let L_j and l_j be the maximum obfuscation level and the applied level of QI j. The height-based utility is defined as: $\sum_j (1 - l_j/L_j)$.

B. Differential Privacy

Differential privacy is a framework to protect user identities in statistical databases, which release global and statistical information about data. A database satisfies differential privacy if the presence or absence of any individual in the database cannot be inferred from statistical query outputs. To provide privacy-preserving data-driven services, differential privacy must be guaranteed in released information. Let D_1 be a database in which the individual is included and D_2 be one in which not. Denote the query outputs from the database i as $Q(D_i)$ and privacy budget as ϵ. D_1 and D_2 is ϵ-differentially private when:

$$\frac{Pr\left(Q(D_2) = R\right)}{Pr\left(Q(D_1) = R\right)} \quad \leq \quad e^{\epsilon}, \tag{1}$$

where R is the output space of Q.

III. Framework

A. Overview

Our framework consists of four major components: **Learn** that runs in high-performance devices and **Encode**, **Verify**, and **Metrics** that can be deployed on any type of computing devices such as mobile handsets and IoT edge devices.

• **Learn** - Learn component generates a deployable encoding model for obfuscating QIs in JSON [5] format. To load structured data according to their types (PII, QI, and SI), Learn component first reads a schema, which describes data attribute types and declares several parameters for the anonymization algorithms such as privacy metric, utility measure, and suppression threshold (maximum number of raw data records that can be suppressed).

To find the best obfuscation level for QIs, Learn component traverses over a problem space, which is determined by the number of QI obfuscation level combinations. To this end, our framework provides two types of space representations: a lattice of which nodes represent the obfuscation levels of QIs, or continuous numeric domains that can be recursively split to represent the domains of attributes as smaller partitions that satisfy a privacy metric. The learning procedure that relies on the lattice is called lattice-based learning and the recursive splitting based one as Mondrian[9]-based learning. For traversal on a problem space, our framework supports Flash [7] and breadth/depth first search with pruning.

• **Encode** - Encode component transforms data to the generalization level specified in a model that defines the set of anonymization rules. This allows greater flexibility and model reuse. The model reuse is appropriate when input data have similar characteristic (e.g., same table schema, and similar distribution of values taken by the records) to training data. In that case, a same model can be used to encode input data without any new learning process. However, since the data characteristics are not be exactly same and hence utility degradation might occur.

To enable multiple types of data inputs (online and batched), our framework supports two types of encode operations: *encodeSafe* and *encode*.

- The *encodeSafe* operation, runs on an entire batch of data and thus guarantees that the output data meets a privacy metric. When training and input datasets have significantly different characteristics, it is possible that encodeSafe leads to suppression of a large number of records to satisfy privacy metrics.
- The *encode* operation, processes individual records and does not check whether released data satisfies the privacy metrics. In fact, the privacy metrics are valid only over aggregate multi-user datasets, these metrics are not guaranteed when applied to an individual record (from a user). A model based encoding simply provides a meaningful heuristic for anonymization if data characteristics have not changed much; in this case, encoded records will be likely to meet a privacy metric.

• **Verify** - The Verify component checks if the anonymized data set satisfies a privacy metric. Through this function, an data administrator is able to decide whether learning for newly collected data is required or not.
• **Metrics** - The Metrics component provides computed metrics on anonymized data, such as data utility and number of records that have been suppressed to satisfy a privacy metric, as a feedback to trigger a model re-training.

B. Parallelization in Learn Component

In this section, we describe parallelization schemes applied to **Learn** component using the lattice-based learning as an example. Recall that a lattice represents a problem space comprising of possible obfuscation choices for QIs; each node represents the combination of obfuscation levels for QIs. The algorithm evaluates all nodes in terms of privacy metric and data utility measure to find the best node that maximizes obfuscation levels with feasible data utility. As a dataset contains more number of records and attributes, computation overhead at each node evaluation and problem space become larger.
• Fine-grained (in the unit of records) parallelized node evaluation - We apply parallelization to three procedures of node evaluation: encoding according to obfuscation levels, checking privacy violations of encoded results, and computing corresponding data utility. To this end, our framework applies MapReduce operations by using distributed key-value pairs in which the encoded QIs and the number of records in the corresponding equivalence class are keys and values. This enables QI encoding, privacy metric computation, and violation check to be done in parallel through multiple executors.

Then, the algorithm traverses a lattice as follows: If a node satisfies privacy requirements (desired levels of privacy metric and number of suppressed records), the node is chosen as a candidate for the best node and the algorithm continues to evaluate its siblings. Recall that increasing an obfuscation level of any attribute from the current level degrades data utility (i.e. no children of candidates has better data utility). Therefore, the algorithm stops going further toward children of the candidates.
• Coarse-grained (in the unit of record clusters) parallelized node evaluation - On large data sets, it is often useful to divide data into coarse grained clusters before an anonymization algorithm is applied; by clustering datasets with similar characteristics, a solution satisfying privacy requirements becomes easier to find. Therefore, parallelization in the unit of record clusters can be faster than in the unit of records. To exploit this, our framework supports a function to partition datasets into multiple sub-datasets. To properly partition datasets, our framework utilizes the k-means clustering [6] of which seed is chosen by the k-means++ algorithm [1]. After the clustering completes, our framework simultaneously runs multiple non-parallelized node evaluations, i.e., sequential over records but parallel by clusters, called coarse-grained parallelization.

Note that although partitioning a dataset improves performance, a larger number of clusters is likely to result in clusters with fewer records that intrinsically fail to satisfy privacy requirements, e.g., it cannot satisfy $(k = 3)$-anonymity if each partitioned dataset has only two records. Also, while the coarse-grained parallelization can quickly complete learning, it may incur lower data utility of anonymized data since it locally finds the best node for each cluster instead of an entire dataset. In Section IV-F, we will compare the performance of fine and coarse-grained parallelizations in terms of learning time and data utility.

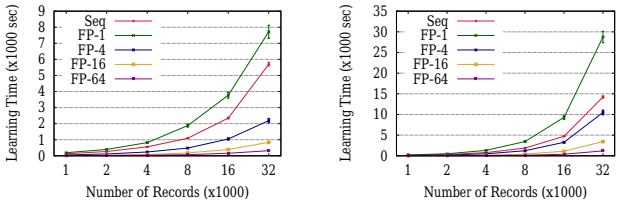
C. Inferring Quasi-Identifiers

Our framework also provides a function (*inferQuasiIdentifiers*) in order for data-administrators to correctly select QIs among record attributes. *inferQuasiIdentifiers* evaluates entropies of obfuscated dataset according to possible combinations of attributes. Note that if each record is classified into a separate equivalence class, the entropy is maximized. Define a residual entropy as the difference between the maximum and generated equivalence classes' entropies. As generating equivalence classes with more discriminative attributes, the equivalence classes are likely to include less records, resulting in a smaller residual entropy, i.e., those attributes are likely to be QIs. *inferQuasiIdentifiers* continues to include an attribute combination with a residual entropy smaller than a given threshold into candidates for QIs.

For efficient pruning of attribute combinations, *inferQuasiIdentifiers* constructs a lattice of which the dimensionality is equal to the number of attributes in dataset. For each dimension, there are two levels $\{1, 2\}$. A node in the lattice encodes a subset of the attributes: $node[i] = 1 \rightarrow$ attribute i is excluded, and $node[i] = 2 \rightarrow$ attribute i is included. *inferQuasiIdentifiers* performs breadth-first-search in this lattice starting with node $\{1, ..., 1\}$. When a node's residual entropy falls under a given threshold, *inferQuasiIdentifiers* adds the attributes represented by the node to QI candidates and stops traversing over its children, i.e., no node that strictly dominates this node will be considered for evaluation.

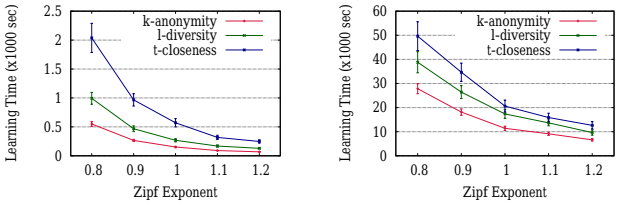
D. Statistics with Differential Privacy

Our framework supports functions to provide responses with differential privacy to statistical queries such as count, max, min, sum, mean, and nth moments. Taking data and ϵ (privacy budget) as input parameters, these functions return answers to which noises are added in order to satisfy the equation (1). Each statistical query is processed in parallel by Apache Spark. By appending the functions to a query response pipeline, a database administrator can defend attacks that try to reveal user identities through statistical queries. Note that differential privacy concerns user identities while retrieving statistical information over an entire database, which is not likely to happen at IoT or edge devices. In the rest of this paper, we will focus on evaluation of our framework with general privacy metrics such as k-anonymity.



(a) k-anonymity (b) t-closeness

Fig. 1. Learning Time over Adult Datasets Sampled with Different Number of Records



(a) 32K records (b) 1M records

Fig. 2. Learning Time of Synthetic Dataset with Zipf Distribution

IV. Evaluation

A. Setup

We examine the performance of following anonymization schemes:

- *Seq*: anonymization scheme that sequentially evaluates datasets.
- *FP-n*: fine-grained parallelization that uses n Spark executors for parallelizing anonymization jobs.
- *CP-m*: course-grained parallelization that divides the dataset with m partitions.

We conduct experiments using servers equipped with an Intel Xeon CPU and 128 GB of memory. We evaluate the schemes using the Adult dataset (Census Income dataset) from the UC Irvine Machine Learning Repository [11], which has been used heavily in previous work on data anonymization research. For experiments with synthetic datasets, we utilize a random dataset of which distribution follows Zipf distribution [13] with an exponent of various values. To examine the utility of dataset anonymized by our framework, we investigate the performance of logistic regression using example data from [16].

We set k to three for k-anonymity, l to three for l-diversity, and t to 0.1 for t-closeness using suppression thresholds of which values is zero. For the data utility measure, we use the Entropy-based utility described in Section II. We use the Chi-Square distance to calculate the distribution distance for t-closeness. More extensive experiments using various parameters for the privacy metrics will be our future work.

B. Effect of Problem Size

The goal of our first set of experiments is to investigate the performance of anonymization schemes with/without parallelization according to problem size (i.e., the number of records and the number of attributes). To this end, we compare Seq and FP-n varying the number of records from 1K to 32K by randomly sampling the Adult dataset. Figure 1 presents the average learning time (time to complete learning anonymization models on the datasets) as a function of the number of records. The numbers are averages over five runs (same for the rest of experiments). As shown in Figure 1, the parallelized schemes (FP-n) yield shorter learning times for the all privacy metrics than Seq except for the case of $n = 1$. FP-1 performs worst because of overheads to construct and maintain distributed data structures for parallelization, which cannot be properly utilized with a single executor. In contrast, FP with $n \geq 4$ exhibits significantly shorter learning times as n and the size of dataset becomes larger. Comparing privacy metrics, we observe that t-closeness is hardest to be computed followed by l-diversity and k-anonymity; l-diversity adds more constraints to k-anonymity and t-closeness does to l-diversity. Note that the performance gain does not increase in multiples of n, e.g., FP-64 performs approximately 15 times faster than Seq even with 64 executors. This is because of parallelization overheads such as synchronization and data communications between executors.

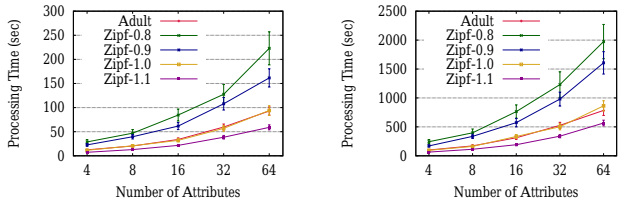
We also conduct experiments while varying the number of attributes (figures are omitted due to the lack of space). In these experiments, we observe that the learning time for the all privacy metrics grows linearly as the number of attributes increases due to a larger search space.

C. Effect of Data Distribution

The purpose of these experiments is to explore how data characteristics affect the anonymization performance. For these experiments, we generate synthetic datasets with 16 attributes, of which distribution follows Zipf distribution with different exponent parameters: a small parameter value results in a distribution similar to an uniform distribution and as the parameter value increases, the resulting distribution becomes heavy-tailed (i.e., there exist many records that has a same value for an attribute). Using the generated datasets with 32K and 1M records, we examine the performance of FP-64. Figure 2 presents the learning time of FP-64 as a function of Zipf parameters. As shown in Figure 2, FP-64 takes shorter time to complete learning as the Zipf exponent parameter is larger. This is because the privacy metrics become easier to be satisfied as a dataset becomes heavy-tailed: for example, if there are more than three records with same attribute values when $(k = 3)$-anonymity is applied, the first node with minimum obfuscation can be chosen as the best node without more traversal, resulting in quick completion of learning process. We observe that it consistently happens across the all privacy metrics regardless with the number of records.

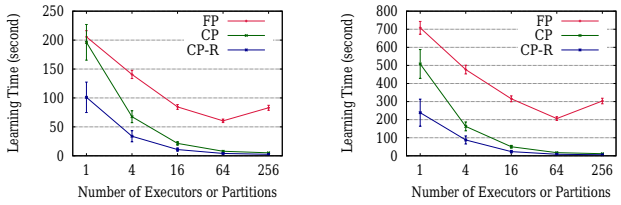
D. QI Inference Performance

Next, we investigate the performance of QI Inference function of our framework. Similar to the learning process, the QI inference performance is affected by dataset size, the



(a) 4K records (b) 32K records

Fig. 3. QI Inference Time according to Number of Attributes



(a) k-anonymity (b) t-closeness

Fig. 4. Learning Time with FP, CP, and CP-R

TABLE I
PERFORMANCE OF LOGISTIC REGRESSION ON RAW AND ANONYMIZED DATASET

Data		Accuracy (%)
Raw		69.3
Anonymized	$k = 2$	63.5
	$k = 3$	62.5
	$k = 4$	59.5

number of attributes, and dataset characteristics. Figure 3 exhibits the QI inference time (time to complete QI inference on datasets) as a function of the number of attributes using the Adult and synthetic datasets with 4K and 32K records. It is straightforward that the more records the dataset has, the longer time QI inference takes as with the result of Section IV-B. Also, as the problem space becomes larger (the dataset has more attributes), QI inference takes longer time. QI inference requires more processing time as datasets becomes heavy-tailed (larger exponent for Zipf distribution), which is consistent with the result of Section IV-C. We observe that Adult dataset fits a Zipf distribution with the exponent between 0.98 and 1.04. Therefore, in Figure 3, QI inference on the Adult dataset yields similar trends of performance to the Zipf-1.0 dataset.

E. Preserving Data Utility

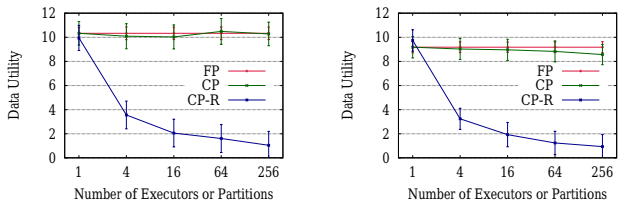
To validate whether our framework obfuscates data while preserving feasible data utility, we investigate the performance of a machine learning algorithm on raw and anonymized dataset while varying k from two to four for k-anonymity. For these experiments, we use data from [16], which consists of four fields: GRE, GPA, rank (prestige of the undergraduate institution; institutions with the first rank have the highest prestige), and corresponding admission status. We examine logistic regression algorithm, which is one of popular machine learning algorithms for binary classification. We train logistic regression on the raw data to predict the admission status given GRE, GPA, and rank and we investigate its performance on raw and anonymized testing data.

Table I presents the test accuracy of predictions on the raw and anonymized dataset. As shown in Table I, logistic regression on the anonymized datasets yields lower accuracy than on the raw data: 5.8% lower accuracy with $k = 2$ and 9.8% with $k = 4$. Note that 95.75% of records in the raw data are unique instances, thus, those records are linkable to a single identity; only 4.25% of records can be already anonymized as-is. We observe that our framework protects 95.75% of the identities only with 5.8% lower accuracy (for the case of $k = 2$). Also, we see that the accuracy degrades more as the privacy metric becomes more strict (larger k). This shows that data obfuscated by our framework can be used for machine learning applications by choosing a proper parameter setting for privacy metric; data administrators need to carefully choose a level of privacy considering trade-off with application performance.

F. Fine vs. Coarse-grained Parallelization

Lastly, we compare the performance of fine and coarse-grained parallelizations. To use a same level of parallelization for both, we apply same number of executors for the fine-grained (FP) and partitions for the coarse-grained parallelization (CP). Recall that our framework supports the k-means clustering for partitioning datasets. To examine the effectiveness of this partitioning scheme, we also investigate the performance of CP with random clustering, which randomly categorizes records into partitions, referred to as CP-R. Figure 4 presents the learning times of FP, CP, and CP-R on the sampled Adult dataset, which consists of 1K records with 16 attributes, according to the number of executors or partitions. We omit figures of l-diversity due to the lack of space. As shown in Figure 4, for the all privacy metrics, both of CPs yield significantly shorter learning times than FP. Interestingly, we observe that the performance of FP degrades with 256 executors compared to FP with 64 executors; its performance is similar to FP with 16 executors. This might be because the parallelization overheads for maintaining 256 executors is too large to obtain performance gain for the dataset with 1K records. This shows that the degree of parallelization for FP needs to be carefully selected with respect to problem size. In contrast, CPs do not experience performance degradation with a large number of partitions since CPs separately run multiple threads that sequentially execute learning process on each partition.

However, since CP tries to find optimal obfuscation levels not in the entire dataset but in local partitions, it is likely to decrease data utilities in obfuscated datasets. Figure 5 shows the obtained data utility of FP and CPs according to the number of executors for FP or partitions for CPs. As shown in Figure 5, CP with k-means clustering exhibits similar data utility to FP while CP-R yields significantly lower data utility



(a) k-anonymity (b) t-closeness

Fig. 5. Data Utility with FP, CP, and CP-R

as the number of partitions becomes larger. This proves that CP is able to significantly shorten anonymization learning time while preserving optimal data utility by applying an appropriate partitioning algorithm (e.g., k-means clustering) rather than random clustering.

V. Related Work

Several anonymization applications using the privacy algorithms discussed in the previous sections have been developed: UTD Anonymization Toolbox is a command-line tool, of which configuration is defined in an XML file, that uses anonymization methods such as DataFly [17], Incognito [8], and Mondrian [9]. It is known that it has a scalability issue with a large dataset [15]. Cornell Anonymization Tool (CAT) [20] is an application for interactive data anonymization. CAT allows data administrators to interactively define record attributes, e.g., selecting which attributes are quasi-identifiers. Although CAT supports a procedure of anonymization and utility/risk evaluations, it has scalability issues such as difficulty in handling a large dataset and requirements for manual configuration of input data. TIAMAT [4] only supports the k-anonymity metric implementing Mondrian and k-Member [3] algorithms, while providing anonymization quality evaluation with the normalized certainty penalty [21]. As with CAT, TIAMAT also provides a visual interface to define data attributes, but similarly it still has limited scalability. SECRETA [14] is a stand-alone system to run several anonymization algorithms such as Mondrian with a graphical user interface. Since SECRETA does not provide anonymization API, it is hard to be extended for developing other applications. ARX [15] is an open source data anonymization tool for health data, implementing several privacy algorithms such as Flash [9]. But it only relies on a global search on entire dataset without parallelization supporting up to about 15 quasi-identifiers while our framework provides parallelized search on either of entire (FP) and partitioned (CP) without limiting number of quasi-identifiers.

VI. Conclusion

This paper proposes, implements, and evaluate a scalable and light-weight data anonymization framework for the flexible deployment of anonymization function. Our framework expedites learning process for annonymiztion rules by applying parallel computation and generates models feasibly deployable to the state-of-art devices. Using our framework, we investigate several factors that affect anonymization performance as well as parallelization. Our experimental results show that our framework is able to shorten the time to build anonymization models.

Acknowledgement

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001.

References

[1] D. Arthur and S. Vassilvitskii. K-means++: The advantages of careful seeding. In *Proc. of ACM-SIAM SODA'07*, pages 1027–1035.
[2] J. Brickell and V. Shmatikov. The cost of privacy: Destruction of data-mining utility in anonymized data publishing. In *Proc. of ACM KDD '08*, pages 70–78.
[3] J.-W. Byun, Y. Sohn, E. Bertino, and N. Li. *Secure Anonymization for Incremental Datasets*, pages 48–63. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.
[4] C. Dai, G. Ghinita, E. Bertino, J.-W. Byun, and N. Li. Tiamat: a tool for interactive analysis of microdata anonymization techniques. *PVLDB*, 2(2):1618–1621, 2009.
[5] JavaScript Object Notation. https://www.json.org. [Online; accessed 11-August-2017].
[6] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. An efficient k-means clustering algorithm: analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):881–892, Jul 2002.
[7] F. Kohlmayer, F. Prasser, C. Eckert, A. Kemper, and K. A. Kuhn. Flash: Efficient, stable and optimal k-anonymity. In *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*, pages 708–717.
[8] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Incognito: Efficient full-domain k-anonymity. In *Proc. of ACM SIGMOD'05*, pages 49–60.
[9] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Mondrian multidimensional k-anonymity. In *Proc. of IEEE ICDE'06*, pages 25–.
[10] N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *Proc. of IEEE ICDE'07*, pages 106–115.
[11] M. Lichman. UCI machine learning repository, 2013.
[12] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. L-diversity: Privacy beyond k-anonymity. *ACM Trans. Knowl. Discov. Data*, 1(1), Mar. 2007.
[13] M. E. J. Newman. Power laws, pareto distributions and zipfs law. *CONTEMPORARY PHYSICS*, 2005.
[14] G. Poulis, A. Gkoulalas-Divanis, G. Loukides, S. Skiadopoulos, and C. Tryfonopoulos. *SECRETA: A Tool for Anonymizing Relational, Transaction and RT-Datasets*, pages 83–109. Springer International Publishing, Cham, 2015.
[15] F. Prasser and F. Kohlmayer. *Putting Statistical Disclosure Control into Practice: The ARX Data Anonymization Tool*, pages 111–148. Springer International Publishing, Cham, 2015.
[16] R Data Analysis Examples. https://stats.idre.ucla.edu/r/dae/logit-regression/.
[17] L. Sweeney. *Datafly: a system for providing anonymity in medical data*, pages 356–381. Springer US, Boston, MA, 1998.
[18] L. Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):571–588, Oct. 2002.
[19] L. Sweeney. K-anonymity: A model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):557–570, Oct. 2002.
[20] X. Xiao, G. Wang, and J. Gehrke. Interactive anonymization of sensitive data. In *Proc. of ACM SIGMOD'09*, pages 1051–1054.
[21] J. Xu, W. Wang, J. Pei, X. Wang, B. Shi, and A. W.-C. Fu. Utility-based anonymization using local recoding. In *Proc. of ACM SIGKDD'06*, pages 785–790.
[22] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, A. Ghodsi, J. Gonzalez, S. Shenker, and I. Stoica. Apache spark: A unified engine for big data processing. *Commun. ACM*, 59(11):56–65, Oct. 2016.