IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 3, March 2017

Micro data Privacy Preserving Using Slicing

S. Bhuvanesh[1], V.N. Anushya[1], A.X. Suganya Gladies[1], M. Vignesh[1]

Assistant Professor, Department of Computer Science and Engineering, Dhaanish Ahmed Institute of Technology, Coimbatore[1]

Abstract: Several anonymization techniques, such as generalization and bucketization, have been designed for privacy preserving microdata publishing. Recent work has shown that generalization loses considerable amount of information, especially for high dimensional data. Bucketization, on the other hand, does not prevent membership disclosure and does not apply for data that do not have a clear separation between quasi identifying attributes and sensitive attributes. In this paper, we present a novel technique called slicing, which partitions the data both horizontally and vertically. We show that slicing preserves better data utility than generalization and can be used for membership disclosure protection. Another important advantage of slicing is that it can handle high-dimensional data. We show how slicing can be used for attribute disclosure protection and develop an efficient algorithm for computing the sliced data that obey the ℓ-diversity requirement. Our workload experiments confirm that slicing preserves better utility than generalization and is more effective than bucketization in workloads involving the sensitive attribute. Our experiments also demonstrate that slicing can be used to prevent membership disclosure.

Keywords: Microdata, Anonymization, Bucketization, Membership Disclosure, Slicing.

1. INTRODUCTION

Privacy-preserving publishing of microdata has been studied extensively in recent years. Microdata contains records each of which contains information about an individual entity, such as a person, a household, or an organization. Several microdata anonymization techniques have been pro- posed. The most popular ones are generalization for k-anonymity and bucketization for ℓ-diversity. In both approaches, attributes are partitioned into three categories: some attributes are identifiers that can uniquely identify an individual, such as Name or Social Security Number; some attributes are Quasi Identifiers (QI), which the adversary may already know (possibly from other publicly-available databases) and which, when taken together, can potentially identify an individual, e.g., Birth-date, Sex, and Zipcode; (3) some attributes are Sensitive Attributes (SAs), which are unknown to the adversary and are considered sensitive, such as Disease and Salary. In both generalization and bucketization, one first removes identifiers from the data and then partitions tuples into buckets. The two techniques differ in the next step. Generalization transforms the QI-values in each bucket into "less specific but semantically consistent" values so that tuples in the same bucket cannot be distinguished by their QI values.

1.1 Motivation of Slicing

It has been shown that generalization for k- anonymity losses considerable amount of information, especially for high-dimensional data. This is due to the following three reasons. First, generalization for k-anonymity suffers from the curse of dimensionality. In order for generalization to be effective, records in the same bucket must be close to each other so that generalizing the records would not lose too much information. However, in high-dimensional data, most data points have similar distances with each other, forcing a great amount of generalization to satisfy k-anonymity even for relative small k's. Second, in order to perform data analysis or data mining tasks on the generalized table, the data analyst has to make the uniform distribution assumption that every value in a generalized interval/set is equally possible, as no other distribution assumption can be justified. This significantly reduces the data utility of the generalized data. Third, because each attribute is generalized separately, correlations between different attributes are lost.

In order to study attribute correlations on the generalized table, the data analyst has to assume that every possible combination of attribute values is equally possible. This is an inherent problem of generalization that prevents effective analysis of attribute correlations.

While bucketization has better data utility than generalization, it has several limitations. First, bucketization does not prevent membership disclosure. Because bucketization publishes the QI values in their original forms, an adversary can find out whether an individual has a record in the published data or not. As shown in, 87% of the individuals in the United States can be uniquely identified using only three attributes (Birthdate, Sex, and Zipcode).

A microdata (e.g., census data) usually contains many other attributes besides those three attributes. This means that the membership information of most individual can be inferred from the bucketized table. Second, bucketization requires a clear separation between QIs and SAs. However, in many datasets, it is unclear which attributes

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 3, March 2017

are QIs and which are SAs. Third, by separating the sensitive attribute from the QI attributes, bucketization breaks the attribute correlations between the QIs and the SAs. In this paper, we introduce a novel data anonymization technique called slicing to improve the current state of the art. Slicing partitions the dataset both vertically and horizontally. Vertical partitioning is done by grouping attributes into columns based on the correlations among the attributes. Each column contains a subset of attributes that are highly correlated. Horizontal partitioning is done by grouping tuples into buckets.

Finally, within each bucket, values in each column are randomly permutated (or sorted) to break the linking between different columns. The basic idea of slicing is to break the association cross columns, but to preserve the association within each column. This reduces the dimensionality of the data and pre- serves better utility than generalization and bucketization.

Slicing preserves utility because it groups highly-correlated attributes together, and preserves the correlations between such attributes. Slicing protects privacy because it breaks the associations between uncorrelated attributes, which are infrequent and thus identifying is better than the following enhancement of the local recoding approach. Rather than using a generalized value to re- place more specific attribute values, one uses the multiset of exact values in each bucket. The multiset of exact values provides more information about the distribution of values in each attribute than the generalized interval. Therefore, using multisets of exact values preserves more information than generalization.

However, we observe that this multiset based generalization is equivalent to a trivial slicing scheme where each column contains exactly one attribute, because both approaches preserve the exact values in each attribute but break the association between them within one bucket. We observe that while one attribute per column slicing preserves attribute distributional information, it does not preserve attribute correlation, because each attribute is in its own column. In slicing, one groups correlated attributes together in one column and preserves their correlation.

2. SLICING

In this section, we first give an example to illustrate slicing. We then formalize slicing, compare it with generalization and bucketization, and discuss privacy threats that slicing can address. The original table is shown in Table 1. The three QI attributes are {Age, Sex , Zipcode}, and the sensitive attribute SA is Disease. For example, in the first bucket of the sliced table shown in Table 2, the values {(22, M), (22, F), (33, F), (52, F)} are randomly permutated and the values {(47906, dyspepsia), (47906, flu), (47905, flu), (47905, bronchitis)} are randomly permutated so that the linking between the two columns within one bucket is hidden.

Table 1. The Original Table

Age	Sex	Zip code	Disease
22	M	47906	dyspepsia
22	F	47906	flu
33	F	47905	flu
52	F	47905	bronchitis
54	M	47302	flu
60	M	47302	dyspepsia
60	M	47304	dyspepsia
64	F	47304	gastritis

Table 2. The Sliced Table

(Age, Sex)	(Zip code, Disease)
(22, M) (22, F) (33, F) (52, F)	(47905,flu) (47906,dysp.) (47905,bron.) (47906,flu)
(54, M) (60, M) (60, M) (64, F)	(47304,gast.) (47302,flu) (47302,dysp.) (47304,dysp.)

2.1 Comparison with Generalization

There are several types of recordings for generalization. The recoding that preserves the most information is local recoding. In local recoding, one first groups tuples into buckets and then for each bucket, one replaces all values of one attribute with a generalized value. Such a recoding is local because the same attribute value may be generalized differently when they appear in different buckets.

2.2 Comparison with Bucketization

To compare slicing with bucketization, we first note that bucketization can be viewed as a special case of slicing. There are exactly two columns: one column contains only the SA, and the other contains all the QIs. The advantages of slicing over bucketization can be understood as follows. First, by partitioning attributes into more than two columns, slicing can be used to prevent membership disclosure. Our empirical evaluation on a dataset shows bucketization does not prevent membership disclosure.

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 3, March 2017

Second, unlike bucketization, which requires a clear separation of QI attributes and the sensitive attribute, slicing can be used without such a separation. For dataset such as the census data, one often cannot clearly separate QIs from SAs because there is no single external public database that one can use to determine which attributes the adversary already knows. Slicing can be useful for such data.

Finally, by allowing a column to contain both some QI attributes and the sensitive attribute, attribute correlations between the sensitive attribute and the QI attributes are preserved. For example, Zipcode and Disease form one column, enabling inferences about their correlations. Attribute correlations are important utility in data publishing. For workloads that consider attributes in isolation, one can simply publish two tables, one containing all QI attributes and one containing the sensitive attribute.

2.3 Privacy Threats

When publishing microdata, there are three types of privacy disclosure threats. The first type is membership disclosure. When the dataset to be published is selected from a large population and the selection criteria are sensitive (e.g., only diabetes patients are selected). one needs to prevent adversaries from learning whether one's record is included in the published dataset. The second type is identity disclosure, which occurs when an individual is linked to a particular record in the released table.

In some situations, one wants to protect against identity disclosure when the adversary is uncertain of member-ship. In this case, protection against membership disclosure helps protect against identity disclosure. In other situations, some adversary may already know that an individual's record is in the published dataset, in which case, membership disclosure protection either does not apply or is insufficient.

The third type is attribute disclosure, which occurs when new information about some individuals is revealed, i.e., the released data makes it possible to infer the attributes of an individual more accurately than it would be possible before the release. Similar to the case of identity disclosure, we need to consider adversaries who already know the membership information. Identity disclosure leads to attribute disclosure. Once there is identity disclosure, an individual is re-identified and the corresponding sensitive value is revealed. Attribute disclosure can occur with or without identity disclosure, e.g., when the sensitive values of all matching tuples are the same. For slicing, we consider protection against membership disclosure and attribute disclosure.

It is a little unclear how identity disclosure should be defined for sliced data (or for data anonymized by bucketization), since each tuple resides within a bucket and within the bucket the association across different columns are hidden. In any case, because identity disclosure leads to attribute disclosure, protection against attribute disclosure is also sufficient protection against identity disclosure.

We would like to point out a nice property of slicing that is important for privacy protection. In slicing, a tuple can potentially match multiple buckets, i.e., each tuple can have more than one matching buckets.

This is different from previous work on generalization and bucketization. In fact, it has been recognized that restricting a tuple in a unique bucket helps the adversary but does not improve data utility. We will see that allowing a tuple to match multiple buckets is important for both attribute disclosure protection and attribute disclosure protection.

3. SLICING ALGORITHMS

We now present an efficient slicing algorithm to achieve ℓ-diverse slicing. Given a microdata table T and two parameters c and ℓ, the algorithm computes the sliced table that consists of c columns and satisfies the privacy requirement of ℓ-diversity. Our algorithm consists of three phases: attribute partitioning, column generalization, and tuple partitioning. We now describe the three phases.

3.1 Attribute Partitioning

Our algorithm partitions attributes so that highly-correlated attributes are in the same column. This is good for both utility and privacy. In terms of data utility, grouping highly-correlated attributes preserves the correlations among those attributes. In terms of privacy, the association of uncorrelated attributes presents higher identification risks than the association of highly-correlated attributes because the association of uncorrelated attribute values is much less frequent and thus more identifiable. Therefore, it is better to break the associations between uncorrelated attributes, in order to protect privacy. In this phase, we first compute the correlations between pairs of attributes and then cluster attributes based on their correlations.

3.1.1 Measures of Correlation

Two widely-used measures of association are Pearson correlation coefficient and mean-square contingency coefficient. Pearson correlation coefficient is used for measuring correlations between two continuous attributes while mean square contingency coefficient is a chi-square measure of correlation between two categorical attributes. We choose to use the mean-square contingency coefficient because most of our attributes are categorical.

3.1.2 Attribute Clustering

Having computed the correlations for each pair of attributes, we use clustering to partition attributes into columns. In our algorithm, each attribute is a point in the

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 3, March 2017

clustering space. Two attributes that are strongly correlated will have a smaller distance between the corresponding data points in our clustering space. First, many existing clustering algorithms (e.g., k- means) requires the calculation of the "centroids". But there is no notion of "centroids" in our setting where each attribute forms a data point in the clustering space. Second, k-medoid method is very robust to the existence of outliers (i.e., data points that are very far away from the rest of data points). Third, the order in which the data points are examined does not affect the clusters computed from the k-medoid method.

We use the well-known k-medoid algorithm PAM (Partition Around Medoids). PAM starts by an arbitrary selection of k data points as the initial medoids. In each subsequent step, PAM chooses one medoid point and one non medoid point and swaps them as long as the cost of clustering decreases. Here, the clustering cost is measured as the sum of the cost of each cluster, which is in turn measured as the sum of the distance from each data point in the cluster to the medoid point of the cluster

3.1.3 Special Attribute Partitioning

The k-medoid method ensures that the attributes are clustered into k columns but does not have any guarantee on the size of the sensitive column Cc. In some cases, we may pre-determine the number of attributes in the sensitive column to be C . The parameter determines the size of the sensitive column Cc.

If S = 1, then Cc = 1, which means that Cc = S. And when c = 2, slicing in this case becomes equivalent to bucketization. If > 1, then Cc > 1, the sensitive column also contains some QI attributes. We adapt the above algorithm to partition attributes into c columns such that the sensitive column Cc contains attributes. We first calculate correlations between the sensitive attribute S and each QI attribute.

Algorithm tuple-partition(T, ℓ)
1. Q = {T}; SB= ϕ.
2. while Q is not empty
3. remove the first bucket B from Q; Q = Q-{B}
4 split B into two buckets B1 and B2, as in Mondrian.
5. if diversity-check(T,Q U {B1, B2} U SB , ℓ)
6. Q = QU {B1, B2}
7. else SB = SB U {B}
8. return SB.

Figure 1: The tuple partition algorithm

It provides the same level of privacy protection as generalization, with respect to attribute disclosure. Although column generalization is not a required phase, it can be useful in several aspects. First, column generalization may be required for identity/ membership disclosure protection. If a column value is unique in a column (i.e., the column value appears only once in the column), a tuple with this unique column value can only have one matching bucket. This is not good for privacy protection, as in the case of generalization/bucketization where each tuple can belong to only one equivalence-class/bucket.

The main problem is that this unique column value can be identifying. In this case, it would be useful to apply column generalization to ensure that each column value appears with at least some frequency.

Second, when column generalization is applied, to achieve the same level of privacy against attribute disclosure, bucket sizes can be smaller. While column generalization may result in information loss, smaller bucket-sizes allows better data utility. Therefore, there is a trade of between column generalization and tuple partitioning. In this paper, we mainly focus on the tuple partitioning algorithm.

The trade of between column generalization and tuple partitioning is the subject of future work. Existing anonymization algorithms can be used for column generalization, e.g., Mondrian. The algorithms can be applied on the sub table containing only attributes in one column to ensure the anonymity requirement

3.2 Tuple Partitioning

In the tuple partitioning phase, tuples are partitioned into buckets. We modify the Mondrian algorithm for tuple.

Algorithm diversity-check(T, T , ℓ)
1. for each tuple t € T , L[t] = ϕ.
2. for each bucket B in T
3. record f (v) for each column value v in bucket B.
4. for each tuple t € T
5. calculate p(t, B) and find D(t, B).
6. L[t] = L[t] U { p (t,B) , D (t, B)} .
7. for each tuple t € T
8. calculate p(t, s) for each s based on L[t].
9. if p(t, s) ≥ 1/ℓ, return false.
10. return true.

Figure 2: The diversity check algorithm

Each element in the list L[t] contains statistics about one matching bucket B: the matching probability p(t, B) and the distribution of candidate sensitive values D(t, B). The algorithm first takes one scan of each bucket B (line 2 to line 3) to record the frequency f (v) of each column value v in bucket B.

Then the algorithm takes one scan of each tuple t in the table T (line 4 to line 6) to find out all tuples that match B and record their matching probability p(t, B) and the distribution of candidate sensitive values D(t, B), which are added to the list L[t] (line 6). At the end of line 6, we have obtained, for each tuple t, the list of statistics L[t] about its matching buckets. A final scan of the tuples in T will compute the p(t, s) values based on the law of total probability.

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 3, March 2017

4. EXPERIMENTS

We conduct two experiments. In the first experiment, we evaluate the effectiveness of slicing in preserving data utility and protecting against attribute disclosure, as compared to generalization and bucketization. To allow direct comparison, we use the Mondrian algorithm and ℓ-diversity for all three anonymization techniques: generalization, bucketization, and slicing. This experiment demonstrates that: slicing is more effective than bucketization in workloads involving the sensitive attribute; and the sliced table can be computed efficiently.

In the second experiment, we show the effectiveness of slicing in membership disclosure protection. For this purpose, we count the number of fake tuples in the sliced data. We also compare the number of matching buckets for original tuples and that for fake tuples. Our experiment results show that bucketization does not prevent membership disclosure as almost every tuple is uniquely identified. Slicing provides better protection against membership disclosure: The number of fake tuples in the sliced data is very large, as compared to the number of original tuples and the number of matching buckets for fake tuples and that for original tuples are close enough, which makes it difficult for the adversary to distinguish fake tuples from original tuples.

5. RELATED WORK

Two popular anonymization techniques are generalization and bucketization. Generalization replaces a value with a "less-specific but semantically consistent" value. Three types of encoding schemes have been proposed for generalization: global recoding, regional recoding, and local recoding. Global recoding has the property that multiple occurrences of the same value are always replaced by the same generalized value. Regional record is also called multi-dimensional recoding (the Mondrian algorithm) which partitions the domain space into non-intersect regions and data points in the same region are represented by the region they are in. Local recoding does not have the above constraints and allows different occurrences of the same value to be generalized differently. Bucketization first partitions tuples in the table into buckets and then separates the quasi-identifiers with the sensitive attribute by randomly permuting the sensitive attribute values in each bucket. The anonymized data consists of a set of buckets with permuted sensitive attribute values.

6. DISCUSSIONS AND FUTURE WORK

This paper presents a new approach called slicing to privacy preserving microdata publishing. Slicing overcomes the limitations of generalization and bucketization and preserves better utility while protecting against privacy threats. We illustrate how to use slicing to prevent attribute disclosure and membership disclosure. Our experiments show that slicing preserves better data utility than generalization and is more effective than bucketization in workloads involving the sensitive attribute. The general methodology proposed by this work is that before anonymizing the data, one can analyze the data characteristics and use these characteristics in data anonymization.

The rationale is that one can design better data anonymization techniques when we know the data better. This work motivates several directions for future research. First, in this paper, we consider slicing where each attribute is in exactly one column. An extension is the notion of overlapping slicing, which duplicates an attribute in more than one columns. This releases more attribute correlations. For example, one could choose to include the Disease attribute also in the first column. That is, the two columns are Age, Sex, Disease and Zipcode, Disease. This could provide better data utility, but the privacy implications need to be carefully studied and understood. It is interesting to study the trade of between privacy and utility. Second, we plan to study membership disclosure protection in more details. Our experiments show that random grouping is not very effective. We plan to design more effective tuple grouping algorithms.Third, slicing is a promising technique for handling high dimensional data. By partitioning attributes into columns, we protect privacy by breaking the association of uncorrelated attributes and preserve data utility by preserving the association between highly correlated attributes.

For example, slicing can be used for anonymizing transaction databases. Finally, while a number of anonymization techniques have been designed, it remains an open problem on how to use the anonymized data. In our experiments, we randomly generate the associations between column values of a bucket. This may lose data utility.

REFERENCES

[1] Asuncion and D. Newman. 2007. UCI machine learning repository.
[2] Blum, C. Dwork, F. McSherry, and K. Nissim. 2005. Practical privacy: the sulq framework. In PODS, pages 128-138.
[3] J. Brickell and V. Shmatikov. 2008. The cost of privacy: destruction of data-mining utility in anonymized data publishing. In KDD, pages 70-78.
[4] B.C. Chen, R. Ramakrishnan, and K. LeFevre. 2007. Privacy skyline: Privacy with multidimensional adversarial knowledge. In VLDB, pages 770-781, [5] H. Cramt'er. Mathematical Methods of Statistics.Princeton, 1948.
[5] Dinur and K. Nissim. 2003. Revealing information while preserving privacy. In PODS, pages 202-210.
[6] Dwork. Differential privacy.2006. In ICALP, pages 1-12.