Improved Slicing Algorithm For Greater Utility In Privacy Preserving Data Publishing

Ajinkya A. Dhaigude *ajinkya.a@learner.manipal.edu*
Department of Information and Communication Technology, Manipal Institute of Technology, Manipal University, Manipal, 576104, India

Preetham Kumar *preetham.kumar@manipal.edu*
Head of the Department, Department of Information and Communication Technology, Manipal Institute of Technology, Manipal University, Manipal, 576104, India

Abstract

Several algorithms and techniques have been proposed in recent years for the publication of sensitive microdata. However, there is a trade-off to be considered between the level of privacy offered and the usefulness of the published data. Recently, slicing was proposed as a novel technique for increasing the utility of an anonymized published dataset by partitioning the dataset vertically and horizontally. This work proposes a novel technique to increase the utility of a sliced dataset even further by allowing overlapped clustering while maintaining the prevention of membership disclosure. It is further shown that using an alternative algorithm to Mondrian increases the efficiency of slicing. This paper shows though workload experiments that these improvements help preserve data utility better than traditional slicing.

Keywords: Data Anonymization, Privacy Preservation, Data Mining, Slicing.

1. INTRODUCTION

Nowadays, data is being collected through our everyday activities such as using credit cards, surfing the web, using emails etc. This data can be very useful to corporations and service providers and mining it may give them a competitive edge in the market. However, the data is usually collected and utilized without the consent of the data subjects. Since this data may contain unaggregated and person specific information, sensitive individual information may get exposed to the various parties involved in its data mining. Thus there is a need to anonymize the dataset before its publication to avoid a privacy breach.

Microdata contains information on an individual level and may reveal specific sensitive attributes about a subject. Microdata attributes can be divided broadly into three categories [1]:

1) *Identifiers* (ID) which can uniquely identify an individual such as SSN or Passport No.

2) *Quasi Identifiers* (QI) which can be can be used in combination with other publicly available records to uniquely identify an individual such as Birthdate and Zipcode.

3) *Sensitive Attributes* (SA) are attributes that an individual seeks to protect and the linking of this attribute to a unique individual could be considered a privacy breach. Eg. Disease, Salary.

Several microdata anonymization techniques have been proposed to prevent the exposure of SAs such as generalization [2], bucketization [3], and more recently, slicing [4].

1.1 Generalization

Generalization [2] works by first removing identifiers from the data and then partitioning tuples into buckets and then transforming the QI values in each bucket into less specific but semantically consistent values such that the tuples in the same bucket cannot be distinguished by their QI values. However, this technique fails for high-dimensional data [5] and forces a large amount of generalization which greatly reduces the utility of the published dataset. Also, since the specific value of a generalized interval cannot be determined, the data analyst has to assume a uniform distribution for each value in the interval. This further reduces the utility of the anonymized dataset.

1.2 Bucketization

Bucketization [3] too works by first removing identifiers from the data and then partitioning tuples into buckets but then it separates the SAs from the QIs by randomly permuting the SA values in each bucket. The anonymized dataset then consists of a set of buckets with randomly permuted sensitive attribute values. This technique does not provide protection against membership disclosure and an adversary can find out whether an individual has a record in the published dataset or not because the QI values are published in their original forms. Also, bucketization requires a clear distinction between SAs and QIs which may not be possible in every dataset.

1.3 Slicing

Slicing [4], as proposed by Tiancheng Li et al., works by removing revealing identifiers from the data and then grouping highly correlated attributes together. This is done by finding the correlation between each attribute and then clustering on the basis of these correlation coefficient values using a k-medoid clustering algorithm. The dataset is then partitioned vertically in accordance with the attribute clusters. The dataset is then partitioned horizontally into buckets using the Mondrian algorithm [6] after which the column values in each bucket are randomly permuted to give the anonymized dataset.

Slicing's major contribution is an increase in the data utility of the published dataset which is achieved by preserving associations between correlated attributes and breaking the associations between uncorrelated attributes. However, since the attribute clusters are not overlapping, an attribute can be mined for knowledge only from within its own cluster. This vastly hampers the utility of the dataset. Note that due to the high number of fake tuples generated in slicing, data mining the whole dataset is also not feasible. Another shortcoming of slicing is that the attribute clustering phase often produces lone columns i.e. a column with only one (or relatively very few) attributes. Such columns may not lend to the utility of the published dataset. Lastly, the Mondrian algorithm that slicing employs for its bucketization phase causes a high overhead in the computation time due to its sorting phase but experiments show that it fails to provide a better result than other random partition algorithms.

2. IMPROVED SLICING

In this section, a novel data anonymization model is introduced that improves upon the shortcomings of slicing. The major contributions of this model are the use of an overlapped clustering technique [7] in the attribute partitioning phase and the use of an alternative tuple partitioning algorithm in lieu of Mondrian.

Age	Sex	Zip	Occupation	Education	Disease
20	**F**	**12578**	**Student**	**12th**	**Flu**
41	**M**	**12589**	**Government**	**Post-Graduate**	**Dyspepsia**
26	**M**	**12460**	**Sales**	**10th**	**Dyspepsia**
23	**F**	**12216**	**Student**	**Graduate**	**Flu**
29	**M**	**12903**	**Agriculture**	**12th**	**Gastritis**
32	**M**	**12093**	**Army**	**Graduate**	**Bronchitis**

TABLE 1: Sample Database.

Sex	Occupation	Zip	Education	Age	Disease
M	**Sales**	**12460**	**10th**	**32**	**Bronchitis**
M	**Army**	**12578**	**12th**	**26**	**Dyspepsia**
F	**Student**	**12093**	**Graduate**	**20**	**Flu**
M	**Agriculture**	**12216**	**Graduate**	**29**	**Gastritis**
F	**Student**	**12589**	**Post-Graduate**	**23**	**Flu**
M	**Government**	**12903**	**12th**	**41**	**Dyspepsia**

TABLE 2: Sliced Database.

Sex	Occupation	Zip	Education	Age	Disease	Disease	Occupation
M	**Sales**	**12460**	**10th**	**32**	**Bronchitis**	**Dyspepsia**	**Sales**
M	**Army**	**12578**	**12th**	**26**	**Dyspepsia**	**Flu**	**Student**
F	**Student**	**12093**	**Graduate**	**20**	**Flu**	**Bronchitis**	**Army**
M	**Agriculture**	**12216**	**Graduate**	**29**	**Gastritis**	**Gastritis**	**Agriculture**
F	**Student**	**12589**	**PG**	**23**	**Flu**	**Dyspepsia**	**Government**
M	**Government**	**12903**	**12th**	**41**	**Dyspepsia**	**Flu**	**Student**

TABLE 3: Improved Sliced Database.

Improved slicing works by first finding the correlations between each pair of attributes and then clustering these attributes into columns by overlapped clustering on the basis of their correlation coefficients. The dataset is then horizontally partitioned into buckets satisfying *l*-diversity [8] using a novel tuple partitioning algorithm. The columns within each bucket are then randomly permuted with respect to one another to give an improved sliced dataset.

2.1 Overlapped Clustering

As mentioned above, restricting an attribute to only one column hampers the data utility of the published dataset. The whole idea behind slicing is to release correlated attributes together which then lends to the usefulness of the anonymized dataset. Thus, permitting an attribute to belong to more than one column would release more attribute correlations and thus enhance the utility of the published dataset.

Tables 2 and 3 show the anonymized tables after applying slicing and improved slicing techniques respectively. In Table 2, *Disease* is grouped with *Age* and *Sex* is grouped with *Occupation*. Even if *Occupation* also had a reasonably high correlation with *Disease* but *Sex* did not, they could not be combined into a bigger group and thus the data utility due to the correlation between *Disease* and *Occupation* is lost. In Table 3, the attributes *Occupation* and *Disease* are present in more than one column i.e. they are overlapping. This allows highly correlated attributes to group together. This also solves the problem of lone columns by merging correlated attributes into a new column instead of just leaving out an attribute with a low correlation.

The notion of *Overlapping Correlation Clustering* [7] was proposed by F. Bonchi et al. and can be employed to the attribute partitioning phase of the slicing algorithm. In this technique, a set of non-overlapping clusters is converted to overlapped clusters by allowing an attribute to belong to

more than one cluster by examining the similarity function between the attribute and each cluster. The algorithm works by finding a multi-labeling function that preserves the similarities between objects. Given a set of n objects $V = \{v_1, ..., v_n\}$, a similarity function s over $V \times V$, and a similarity function H between sets, it finds a multi-labeling function b that minimizes the cost:

$$C_{OCC}(V, b) = \sum_{(u,v) \in V \times V} \left| H(l(u), l(v)) - s(u, v) \right|$$

The similarity function s can be defined by the Pearson's correlation coefficient and H is usually defined in the following two ways [7]:

- **Jaccard coefficient:** This is a natural set-similarity function defined as

$$J(E, F) = \left| \frac{E \cap F}{E \cup F} \right|$$

- **Set-intersection indicator:** This is used when two objects sharing a single cluster label is sufficient to assert membership in the same cluster. This is defined as

$$I(E, F) = \begin{cases} 1 & if \ E \cap F \neq \phi \\ 0 & otherwise \end{cases}$$

Any of these similarity functions can be used for the algorithm and the initial non-overlapping clusters can be computed with *k*-medoid. The overlapping correlation clustering algorithm features a local-search algorithm [7] that finds the function b as shown in Algorithm 1. It should be noted that while this technique is ideal for high dimensional data, a dataset with few dimensions can instead use a modified *k*-member clustering algorithm by allowing each cluster to find data points irrespective of their inclusion in another cluster.

It should be noted that overlapped clustering produces fewer fake tuples than non-overlapped clustering. This is because the overlapped attributes tend to negate some of the potential fake tuples. For example, to reconstruct a tuple from Table 3, we know that the *(M, Army)* value can contain any of the values from the second column but has to contain the value *(32, Bronchitis)* from the third column as the *(Bronchitis, Army)* relation is implied in the fourth column. This produces a total of four possible tuples, each containing one of the values from the second column with the other column values remaining the same.

```
Initialize b to a valid labelling;
while C_OCC decreases do
    for each v ∈ V do
        find the label B that minimises cost;
        update b so that b(v) = B;
        compute totG, the total number of groups;
        if totG < beta then
            revert b(v) back to original labels;
        else
            continue;
        end
    end
end
return b;
```

ALGORITHM 1: Overlapped Clustering.

Figure 1 shows a representation of the attribute clusters formed in overlapped slicing. Here, the groups *G1* and *G2* each contain clusters that share one or more attribute amongst themselves. It should be noted that a lower number of groups result in a lower number of fake tuples. Hence, a metric *beta* has been introduced to control the minimum number of groups formed. Ideally, a low *beta* will result in high data utility due to the higher number of attribute correlations being released but lower privacy protection due to the lower number of fake tuples being generated whereas a high *beta* can provide higher privacy but hamper the data utility. This *beta* is used in the clustering phase to determine if an attribute should be assigned to a cluster or not as shown in Algorithm 1.

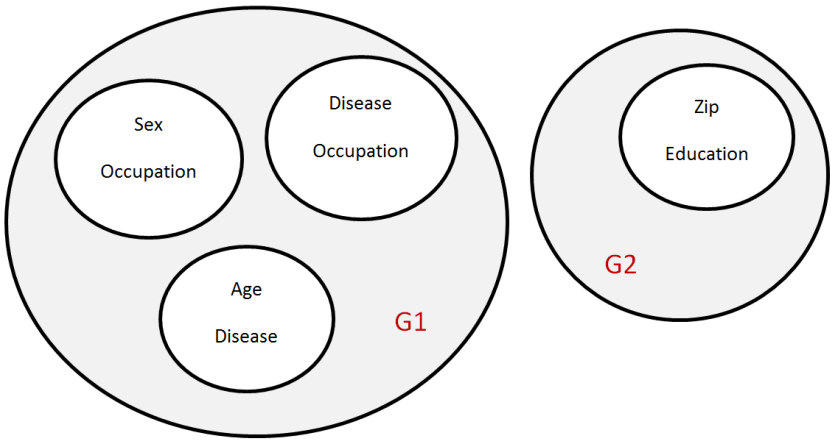

FIGURE 1: Overlapped Columns Clustered into Groups.

2.2 Tuple Partitioning

In this phase, the tuples are partitioned into buckets and checked for *l*-diversity. Improved slicing does not use the Mondrian algorithm [6], [9] as it incurs a high computational overhead yet fails to provide a better result. Instead, the dataset is partitioned horizontally as shown in Algorithm 2.

$Q = \{T\}$;
$SB = \emptyset$;
while Q *is not empty* **do**
  Remove the topmost bucket B from Q;
  Split B into m buckets;
  for *each of the* m *buckets* **do**
    Randomly allot half the tuples to B_1;
    Allot rest of the tuples in bucket to B_2;
  end
  if ***diversityCheck***$(T, Q \cup \{B_1, B_2\} \cup SB, l)$ **then**
    $Q = Q \cup \{B_1, B_2\}$;
  else
    $SB = SB \cup \{B\}$;
  end
end
return SB;

ALGORITHM 2: Tuple Partitioning.

The queue *Q* initially has only one bucket containing all the tuples and the queue *SB* is empty. In each iteration, the algorithm removes a bucket *B* from *Q* and splits the bucket along the SAs into *m* buckets where *m* is the total number of different sensitive attribute values in *B*. Half the tuples in each of the *m* buckets are then randomly chosen and allotted to bucket B_1 and the rest to bucket B_2. If the sliced table after the split satisfies *l*-diversity, then the algorithm puts the two buckets B_1 and B_2 at the end of the queue *Q* for further splits. Otherwise, we cannot split the bucket anymore and the algorithm puts the bucket *B* into *SB*. When *Q* becomes empty, we have computed the sliced table and the set of sliced buckets is in *SB*.

The time complexity of Mondrian is $O(n \log n)$ [9] whereas the alternate tuple partitioning algorithm presented here takes only $O(n)$ time. The *diversityCheck* algorithm is the same as in slicing except that the computation of $p(t, B)$ and $D(t, B)$ requires us to calculate the total number of possible tuples generated in each bucket.

3. EXPERIMENTAL ANALYSIS

In this section, the effectiveness in preserving data utility of the proposed improved slicing technique is evaluated against the existing slicing method. All algorithms are implemented in Java and the experiments were run on an Intel Core i3 2.27GHz machine with 2GB of RAM and Windows 7 OS.

Attribute	Type	Values
Age	Continuous	74
Workclass	Categorical	8
Final-Weight	Continuous	NA
Education	Categorical	16
Education-Num	Continuous	16
Marital-Status	Categorical	7
Occupation	Categorical	14
Relationship	Categorical	6
Race	Categorical	5
Sex	Categorical	2
Capital-Gain	Continuous	NA
Capital-Loss	Continuous	NA
Hours-Per-Week	Continuous	NA
Country	Categorical	41
Salary	Categorical	2

TABLE 4: Description of the Adult Dataset.

The experiments made use of the Adult data set from the UC Irvine machine learning repository [10], as described in Table 4. Tuples with missing values were eliminated and the experiments were performed on the remaining 30,162 valid tuples considering *Occupation* to be the sensitive attribute. Attributes with continuous type values were discretized into equal sized bins and then treated as a discrete domain.

Improved slicing aims to provide a higher privacy standard than conventional techniques like generalization and bucketization by utilizing the inherent privacy preserving properties of slicing [4] such as attribute and membership disclosure protection. The following analysis aims to show that improved slicing not only maintains the privacy protection offered by slicing but also offers a higher utility in the anonymized dataset.

The main aim of this paper is to present a technique to increase the utility of a sliced dataset and because the data utility of a sliced table depends on the number of attribute correlations released, the improved slicing algorithm is evaluated on the basis of the average correlation coefficients between the attributes in each column against the number of columns released. Due to the random nature of the clustering algorithms used, each run may produce different column attributes. Hence, for a given number of columns, each technique was implemented 50 times and the average results were reported as shown in Figure 2. During the whole experiment, the minimum number of attributes in the column containing the SA was limited to 3 and the *beta* for improved slicing was set to 2.

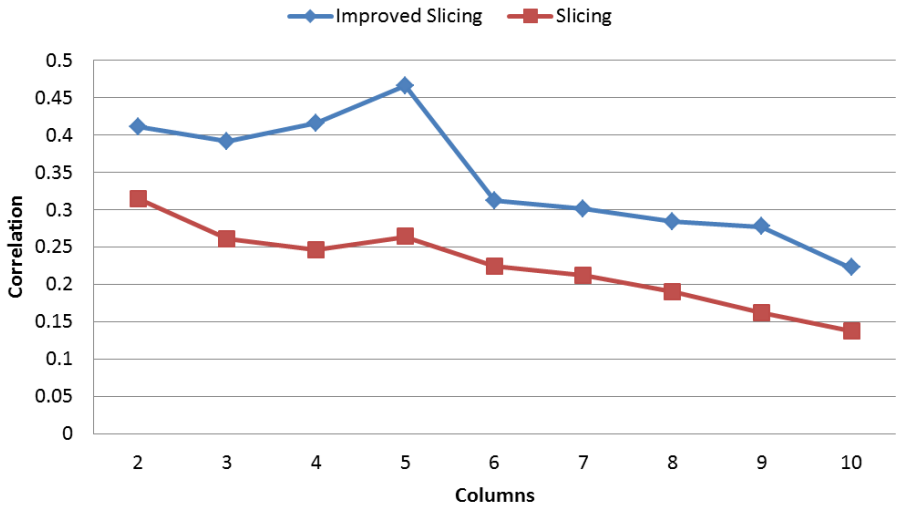

FIGURE 2: Average Correlation vs Number of Columns.

It can be seen from the graph that for slicing, the average correlation between the attributes in the columns released tends to fall as the number of columns increase. This is to be expected as the number of attributes in each column would decrease with the rise in number of columns resulting in a lower average of the correlation coefficient and formation of lone columns. Improved slicing, on the other hand, tends to increase its average correlation coefficient up to a maxima and then decline like slicing. This could suggest the existence of an optimal number of columns that provides the highest utility for a sliced dataset. Improved slicing can provide the same level of privacy as non-overlapped slicing by satisfying the privacy measure used. Keeping the minimum size of each bucket limited to 250, both slicing and improved slicing were able to satisfy *l*-diversity for *l* = 2, 3, 4, 5 and 6.

4. CONCLUSION AND FUTURE WORK

This paper presents a novel technique for increasing the utility of anonymized datasets by improving upon some of the shortcomings of slicing. Improved slicing can duplicate an attribute in

more than one column and this leads to greater data utility because of an increased release of attribute correlations. Improved slicing satisfies all the privacy safeguards of traditional slicing such as prevention of attribute disclosure and membership disclosure. This work also presents an alternate tuple partitioning algorithm that runs faster and is more efficient. The experimental results demonstrate the greater data utility provided by improved slicing while satisfying *l*-diversity.

Future research work in this area can include the extension of the notion of improved slicing to datasets satisfying more severe anonymity parameters such as *t*-closeness and *m*-invariance. Further analysis on the effect of the number of released columns on data privacy and utility should also be considered. Improved slicing for datasets containing more than one sensitive attribute is also a possible future research direction.

5. REFERENCES

[1] P. Samarati, "Protecting respondents' identities in microdata release," *IEEE Trans. on Knowl. and Data Eng.*, vol. 13, no. 6, pp. 1010–1027, 2001.

[2] L. Sweeney, "Achieving k-anonymity privacy protection using generalization and suppression," *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, vol. 10, no. 6, pp. 571–588, 2002.

[3] X. Xiao and Y. Tao, "Anatomy: Simple and effective privacy preservation," *in Proceedings of the 32Nd International Conference on Very Large Data Bases*, ser. VLDB '06, 2006, pp. 139–150.

[4] T. Li, N. Li, J. Zhang, and I. Molloy, "Slicing: A new approach for privacy preserving data publishing," IEEE *Transactions on Knowledge and Data Engineering*, vol. 24, pp. 561 – 574, 2012.

[5] C. C. Aggarwal, "On k-anonymity and the curse of dimensionality," in *Proceedings of the 31st International Conference on Very Large Data Bases*, 2005, pp. 901–909.

[6] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan, "Mondrian multidimensional k-anonymity," *in Proceedings of the 22nd International Conference on Data Engineering*, 2006.

[7] F. Bonchi, A. Gionis, and A. Ukkonen, "Overlapping correlation clustering," in *Proceedings of the 2011 IEEE 11th International Conference on Data Mining*, 2011, pp. 51–60.

[8] Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "L-diversity: Privacy beyond k-anonymity," *ACM Trans. Knowl. Discov. Data*, vol. 1, no. 1, 2007.

[9] J. H. Friedman, J. L. Bentley, and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Trans. Math. Softw.*, vol. 3, no. 3, pp. 209–226, 1977.

[10] Internet: http://archive.ics.uci.edu/ml/datasets/Adult, [Mar. 14, 2014].