Volume 3, Issue 8, August 2013 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

A Novel Approach to Privacy Preserving Data Publishing

[1] B.Manoj Kumar, [2] S.Kusuma

DEPT of CSE, JNTUA

India

Abstract- the anonymization techniques, such as generalization and bucketization are used for privacy preserving micro data publishing. Here the Generalization loses considerable amount of information, especially for multi dimensional data and the Bucketization method, 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. A novel technique called slicing has been proposed, which partitions the data both horizontally and vertically. Slicing preserves better data usage than generalization and also can be used for membership disclosure protection. The important advantage of slicing is that it can handle high-dimensional data. Hence slicing can be used for attribute disclosure protection and develop an efficient algorithm for computing the sliced data that obey the l-diversity requirement. Slicing preserves better data usage than generalization and is more effective than bucketization when dealing with the sensitive attribute. The basic idea of slicing is to break the linking the cross columns, and to preserve the linking within each column. It reduces the organizing of the data and preserves better utility than generalization and bucketization.

Key words: Slicing, Generalization, bucketization, Dendrogram Tree, CTREND Graph

I. Introduction

We live in a networked society which is experiencing substantial growth in the amount of data that contains person-specific information. Some organizations like census bureaus and renounced institutions are required to make personal information available, while other organizations, such as hospitals may want to publish their data voluntarily for research purposes. [1][2] For example, a hospital may provide patients' medical records to data analysers to facilitate the building of a classification model that uses patients' age, smoking history and obesity status to predict their life expectancy. On the other hand, data publishers are often prohibited by law from disseminating any person-specific information that compromises an individual's privacy. Therefore, a common precaution taken by data publishers is to remove all explicit identifiers such as name, address and SSN to make the resulting data look completely anonymous.

Unfortunately, the remaining information can still be used to disclose the individual's identity, and further sensitive information when linked to other data resources using the special characteristics (known as "identifying" attributes) found in the released data. For example, a recent study [6] estimates that 87% (216 million of 248 million) of the population of the United States can be uniquely identified using the seemingly innocuous attributes of gender, date of birth and 5-digit zip code. Clearly, the released data containing such information about individuals should not be considered anonymous, and more seriously, when such information is linked to a medical dataset which contains all above information together with the diagnosis and medication data, the sensitive information of individuals is leaked. Therefore, the data should be further processed before publication so that it is resistant to privacy leakage while still presenting a maximal amount of information to data analysers. To achieve this purpose, data publishers usually apply both the data generalization techniques and privacy schemes to process the dataset.[3] The data generalization technique works by substituting the original value of given attribute to a more generalized one based on the generalization hierarchy built on top of each attribute's domain. Specifically, the generalization hierarchy contains a mapping between the domain and its generalizations with its highest level representing the most generalized value, and its lowest level representing the most specific one that is the original value.[4] After applying this technique, the dataset will be partitioned into several groups so that every tuple is indistinguishable from any other in the same group based on the "identifying" attributes, which are a set of attributes that partially identify an individual. Due to its effectiveness to diminish the identity leakage, the approach has been widely utilized in the previous works that focused on privacy schemes and generalization algorithms.

II. PRIOR WORK

2.1 Constituents of the Micro-table

A micro-table is a table that consists of rows and columns. In our context, it contains person-specific information that could help data analysers explore hidden relationships between the data in the table. For example, a micro-table that consists of persons' obesity status, smoking history, and current age may help data analysers to find combinations of variables that are

useful to predict a patient's life expectancy. In a micro-table, rows are termed tuples which express a relationship among the set of values associated with a specific person. Columns are termed attributes which denote different aspects of semantic information related to a person. [7] The type of attribute value can be either categorical or numerical. We classify all attributes into three categories: identity attributes, quasi-identifier attributes, and sensitive attributes.

Identity Attribute: This kind of attribute can be used by itself to uniquely or nearly uniquely identify a certain individual. Name, social security number, and address are examples of identity attributes. Oftentimes, data analysers do not care about whose information a particular tuple corresponds to and hence, the identity attribute can be removed from the published micro-table without affecting the data analyser.

Quasi-Identifier Attribute: Quasi-identifier(QI) attributes (known as "identifying" attributes) are a set of non-sensitive attributes which can be combined together to almost uniquely identify individuals in the general population. One example of QI attributes is Gender, Date of Birth, and Zipcode.

Sensitive Attribute: Sensitive attributes such as disease, salary and credit ranking contain the most sensitive information about individuals. Sensitive attributes are valuable to the data analysers who can explore the relationship between the sensitive and QI attributes. Therefore, it is necessary to publish these sensitive attributes.

2.2 Basic Techniques

If an individual's QI and sensitive attributes are published in the same micro-table, an adversary may be able to deduce his/her identity from the QI attributes and thus, the sensitive information about an individual would be revealed. To mitigate these identification threats, some basic techniques have been introduced to process the micro-tables before publication, including: [6][8]

Data Generalization: This technique generalizes the values of certain attributes to make different values have the same semantic form. For example, a group of ages {23;25;29} can be generalized into a single group[20-29].

Data Suppression: In data suppression, a tuple (or parts of the tuple) is omitted from the published table. One widely studied approach is to partition the tuples into QI groups, where each QI group has the tuples that share the same QI attribute values. This grouping is achieved by data generalization and suppression techniques. When producing such a table, a data publisher will count for the following three aspects:

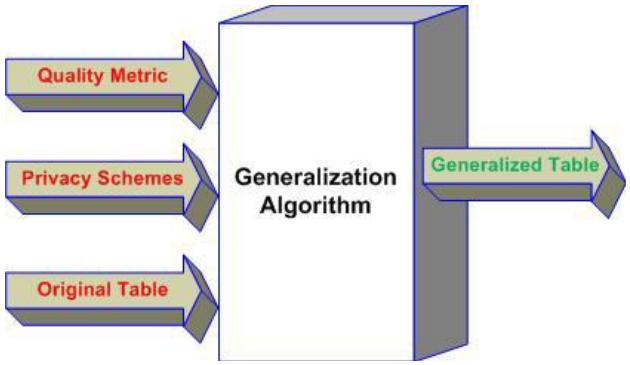


Figure 1: Relation between different aspects of data publishing

III. PROPOSED WORK

3.1 Preprocessing Phase

3.1.1 Partitioning According To Time Period

Before partitioning the dataset is to be imported in to the database. After importing the data set, it is partitioned based on the time periods in preprocessing phase, and after partitioning the each partition is clustered using a familiar traditional clustering techniques called hierarchical clustering . The results of this hierarchical clustering for each partition generates two data structure such as the node list and the edge list. After generating both these lists in the preprocessing phase which more effective (real-time) visualization updates of the evaluated C-TREND graphs. By considering these data structures, graph entities (nodes and edges) are generated and shown as a temporal cluster graph in the system output window.

3.1.2 Dendrogram Tree Construction And Extracting Cluster Solutions

In this module hierarchical clustering techniques are used for clustering the datasets.

Procedure for Hierarchical Clustering: Given a set of N items to be clustered, and an NxN distance (or similarity) matrix, the general working process of this hierarchical clustering is as follows:

1. Start by assigning each item to its own cluster, so that if we have N items, then it results N clusters, each containing just one item. Let the distances (similarities) between the clusters equal the distances (similarities) between the items they contain.

2. Find the closest (most similar) pair of clusters and combine them into a single cluster, so that this merging will decrements one cluster counts.

3. Calculate the distances (similarities) between the newly formed cluster and each of the previous clusters. Repeat steps 2 and 3 until all items are clustered into a single cluster of size N.

Step 3 can be evaluated in many ways. It clearly differentiates the single-link from complete-link and average-link clustering. In single-link clustering (also called the connectedness or minimum method), the distance between one cluster and another cluster to be equal to the shortest distance from any member of one cluster to any member of the other cluster. If the data consist of similarities then the similarity between one cluster and another cluster to be equal to the greatest similarity from any member of one cluster to any member of the other cluster. In complete-link clustering (also called the diameter or maximum method), the distance between one cluster and another cluster to be equal to the longest distance from any member of one cluster to any member of the other cluster. In average-link clustering, the distance between one cluster and another cluster to be equal to the average distance from any member of one cluster to any member of the other cluster. This kind of hierarchical clustering is called agglomerative because it merges clusters iteratively.

Single Linkage Clustering: It is an agglomerative scheme in which it erases the rows and columns in the proximity matrix when the old clusters are merged into new clusters. The N*N proximity matrix is D = [d(i,j)]. The clusterings are assigned with the sequence numbers starting from 0, 1... (n-1) and L (k) is the level of the kth clustering. A cluster with sequence number m is denoted (m) and the proximity between the clusters (r) and (s) and is denoted as d [(r),(s)].

This algorithm works according to the following steps:

1. Start with the disjoint clustering having the level L (0) =0 and the sequence number as m=0.

2. Find the least dissimilar pair of clusters in the current clustering, say pair (r), (s), according to d [(r), (s)] = min d [(i), (j)] where the minimum is over all pairs of clusters in the current clustering.

3. Increment the sequence number: m = m +1. Merge clusters (r) and (s) into a single cluster to form the next clustering m. Set the level of this clustering to L (m) = d [(r), (s)]

4. Update the proximity matrix, D, by deleting the rows and columns corresponding to clusters (r) and (s) and adding a row and column corresponding to the newly formed cluster. The proximity between the new cluster, denoted (r,s) and old cluster (k) is defined in this way: d[(k), (r,s)] = min d[(k),(r)], d[(k),(s)].

5. If all objects are found in one cluster, abort the process otherwise proceed with step 2.

The following pages trace a hierarchical clustering of distances in miles between U.S. cities. The method of clustering is single-link.

Input distance (dissimilarity) matrix

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**
A	0	206	429	1504	963	2976	3095	2979	1949
B	206	0	233	1308	802	2815	2934	2786	1771
C	429	233	0	1075	671	2684	2799	2631	1616
D	1504	1308	1075	0	1329	3273	3053	2687	2037
E	963	802	671	1329	0	2013	2142	2054	996
F	2976	2815	2684	3273	2013	0	808	1131	1307
G	3095	2934	2799	3053	2142	808	0	379	1235
H	2979	2786	2631	2687	2054	1131	379	0	1059
I	1949	1771	1616	2037	996	1307	1235	1059	0

The nearest pair of cities is A and B, at distance 206. These are merged into a single cluster called "A/B" in the first step. Then we compute the distance from this new compound object to all other objects. In single link clustering the rule is that the distance from the compound object to another object is equal to the shortest distance from any member of the cluster to the outside object. So the distance from "A/B" to C is chosen to be 233, which is the distance from B to C. Similarly, the distance from "A/B" to I am chosen to be 1771.

After merging a with B

	A/B	**C**	**D**	**E**	**F**	**G**	**H**	**I**
A/B	0	223	1308	802	2815	2934	2786	1771
C	223	0	1075	671	2684	2799	2631	1616
D	1308	1075	0	1329	3273	3053	2687	2037
E	802	671	1329	0	2013	2142	2054	996
F	2815	2684	3273	2013	0	808	1131	1307
G	2934	2799	3053	2142	808	0	379	1235
H	2786	2631	2687	2054	1131	379	0	1059
I	1771	1616	2037	996	1307	1235	1059	0

The nearest pair of objects is A/B and C, at distance 223. These are merged into a single cluster called "A/B/C". Then we compute the distance from this new cluster to all other clusters, to get a new distance matrix:

After merging C with A-B

	A/B/C	**D**	**E**	**F**	**G**	**H**	**I**
A/B/C	0	1075	671	2684	2799	2631	1616
D	1075	0	1329	3273	3053	2687	2037
E	671	1329	0	2013	2142	2054	996
F	2684	3273	2013	0	808	1131	1307
G	2799	3053	2142	808	0	379	1235
H	2631	2687	2054	1131	379	0	1059
I	1616	2037	996	1307	1235	1059	0

Now, the nearest pair of objects is G and H, at distance 379. These are merged into a single cluster called "G/H". Then we compute the distance from this new cluster to all other objects, to get a new distance matrix:

After merging G with H

	A/B/C	**D**	**E**	**F**	**G/H**	**I**
A/B/C	0	1075	671	2684	2631	1616
D	1075	0	1329	3273	2687	2037
E	671	1329	0	2013	2054	996
F	2684	3273	2013	0	808	1307
G/H	2631	2687	2054	808	0	1059
I	1616	2037	996	1307	1059	0

Now, the nearest pair of objects is E and A/B/C, at distance 671. These are merged into a single cluster called "A/B/C/E". Then we compute the distance from this new cluster to all other clusters, to get a new distance matrix:

After merging E with A/B/C

	A/B/C/E	**D**	**F**	**G/H**	**I**
A/B/C/E	0	1075	2013	2054	996
D	1075	0	3273	2687	2037
F	2013	3273	0	808	1307
G/H	2054	2687	808	0	1059
I	996	2037	1307	1059	0

Now, the nearest pair of objects is F and G/H, at distance 808. These are merged into a single cluster called "G/H/F". Then we compute the distance from this new cluster to all other clusters, to get a new distance matrix:

After merging F with G/H: Now, the nearest pair of objects is I and A/B/C/E, at distance 996. These are merged into a single cluster called "A/B/C/E/I". Then we compute the distance from this new cluster to all other clusters, to get a new distance matrix:

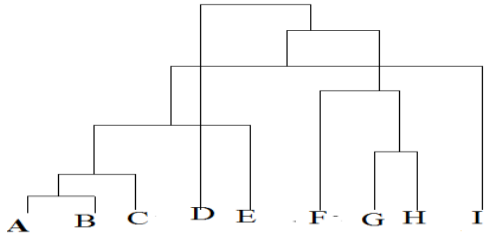
	A/B/C/E	D	G/H/F	I
A/B/C/E	0	1075	2013	996
D	1075	0	2687	2037
G/H/F	2054	2687	0	1059
I	996	2037	1059	0

After merging me with A/B/C/E: Now, the nearest pair of objects is A/B/C/E/I and G/H/F, at distance 1059. These are merged into a single cluster called "A/B/C/E/I/G/H/F".

	A/B/C/E/I	**D**	**G/H/F**
A/B/C/E/I	0	1075	1059
D	1075	0	2687
G/H/F	1059	2687	0

After merging G/H/F with A/B/C/E/I: Finally, we merge the last two clusters at level 1075.

	A/B/C/E/I/G/H/F	**D**
A/B/C/E/I/G/H/F	0	1075
D	1075	0

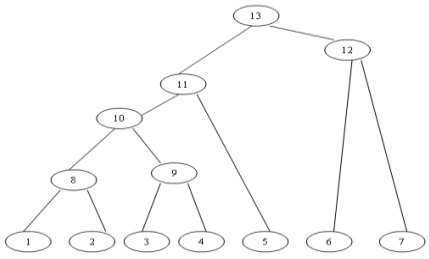


DENDOGRAM TREE

Algorithm: DENDRO_EXTRACT

INPUT: ki =desired number of clusters
i=data partition indicator
1 begin
2 if ki <=N then
3 CurrCl = {DendrogramRooti}
4 while |CurrCl| < ki
5 MaxCl = DendrogramRooti - |CurrCl| + 1
6 CurrCl = (CurrCl \ MaxCl) U (MaxCl.Left) U (MaxCl.Right)
7 return CurrCl
8 else request new ki
9 end

A dendrogram data structure allows for quick extraction of any specific clustering solution for each data partition when the user changes partition zoom level ki. To obtain a proper clustering solution from the data structure for the data partition Di, the CTREND uses an algorithm called DENDRO_EXTRACT algorithm (Algorithm 1), which takes the preferred number of clusters in the solution ki as an input and returns the set CurrCl containing the clusters corresponding to the ki-sized solution. Cluster attributes such as center and size are then accessible from the corresponding dendrogram data structure by referencing the clusters in CurrCl. DENDRO_EXTRACT starts at the root of the dendrogram and traverses the dendrogram by splitting the highest numbered node (where the nodes are numbered according to how close they are to the root, as in figure below) in the current set of clusters until k clusters are included in the set. MaxCl in this algorithm acts as the highest element in the current cluster set CurrCl. With this specific dendrogram structure it was clearly observed, it is always the case that MaxCl = DendrogramRooti -|CurrCl| + 1. The dendrogram data array maintains the successive levels of the hierarchical solution in order; therefore by replacing the MaxCl to both of its children: MaxCl.Left (left child) and MaxCl.Right (right child) after replacing it identifies the next solution level in the dendrogram. D The time complexity of DENDRO_EXTRACT is linear and is of O (ki) and hence it gives real-time extraction of cluster solutions.



Dendrogram tree (N=7)

Working of Algorithm

Let the desired number of clusters are 7 i.e. $\mathbf{K_i}$=7
$DendrogramRoot_i$=13
CurrCl=13

Iteration 1:

|CurrCl|<7 i.e. 1<7
$MaxCl = DendrogramRoot_i - |CurrCl| + 1$
CurrCl = (CurrCl \ MaxCl) U (MaxCl.Left) U (MaxCl.Right)
MaxCl=13-1+1=13
CurrCl= (13/13) U (13.left) U (13.right)
CurrCl= {11, 12}

Iteration 2:

|CurrCl|<7 i.e. 2<7
$MaxCl = DendrogramRoot_i - |CurrCl| + 1$

CurrCl = (CurrCl \ MaxCl) U (MaxCl.Left) U (MaxCl.Right)
MaxCl=13-2+1=12
CurrCl= (11, 12/12) U (12.left) U (12.right)
CurrCl= {11, 6, 7}

Iteration 3:

|CurrCl|<7 i.e. 3<7
MaxCl = $DendrogramRoot_i$ - |CurrCl| + 1
CurrCl = (CurrCl \ MaxCl) U (MaxCl.Left) U (MaxCl.Right)
MaxCl=13-3+1=11
CurrCl= (11, 6, 7/11) U (11.left) U (11.right)
CurrCl= {10, 5, 6, 7}

Iteration 4:

|CurrCl|<7 i.e. 4<7
MaxCl = $DendrogramRoot_i$ - |CurrCl| + 1
CurrCl = (CurrCl \ MaxCl) U (MaxCl.Left) U (MaxCl.Right)
MaxCl=13-4+1=10
CurrCl= (10, 5, 6, 7/10) U (10.left) U (10.right)
CurrCl= {8, 9, 5, 6, 7}

Iteration 5:

|CurrCl|<7 i.e. 5<7
MaxCl = $DendrogramRoot_i$ - |CurrCl| + 1
CurrCl = (CurrCl \ MaxCl) U (MaxCl.Left) U (MaxCl.Right)
MaxCl=13-5+1=9
CurrCl= (8, 9, 5, 6, 7/9) U (9.left) U (9.right)
CurrCl= {8, 5, 6, 7, 3, 4}

Iteration 6:

|CurrCl|<7 i.e. 6<7
MaxCl = $DendrogramRoot_i$ - |CurrCl| + 1
CurrCl = (CurrCl \ MaxCl) U (MaxCl.Left) U (MaxCl.Right)
MaxCl=13-6+1=8
CurrCl= (8, 5, 6, 7, 3, 4/8) U (8.left) U (8.right)
CurrCl= {1, 2, 5, 6, 7, 3, 4}

Iteration 7:

|CurrCl|<7 i.e. 7<7
Condition violated.

3.2 Interactive Analysis Phase

3.2.1 Node Extraction and Edge Extraction

The final step in the preprocessing phase is to generate both node and edge lists. Here the node list contains all possible nodes and their sizes, and the edge list contains all possible edges and their weights, for the entire data set. Generating these lists in the preprocessing phase allows for more effective (real-time) visualization updates of the C-TREND graphs. Each data partition possesses an array-based Dendrogram data structure containing all its possible clustering solutions. The node list generated here is just an aggregate list of all dendrogram data structures indexed for optimal node lookup. Here these lists are dependent. Hence the edge list is generated based on the node list(since an edge is possible between any two nodes in adjacent data partitions) . Therefore, the edge list that was generated here basically a list of ordered pair and each pair represents the adjacent nodes that define an edge. Since each dendrogram contains 2N -1 nodes (i.e., N leaves and N - 1 internal nodes), there are (2N – 1)2 possible edges between two adjacent data partitions. If the given data set contains t data partitions, then there are (t-1) (2N – 1)2 possible edges for the entire data set. Note that we use t -1 because the first partition only possesses outgoing edges and the last partition only possesses incoming edges. By considering both of these partitions , the time complexity of the edge list generation would have an asymptotic upper bound of O(N 2tm) ,where m is the number of attributes in the data (needed to calculate edge weights). Hence through this preprocessing phase all the possible edges and their weights in the are calculated and the C-TREND graph was implemented that can achieve real-time functionality in the analysis phase, as the output graph parameters are being interactively adjusted.

3.2.2 Extracting Cluster Solutions According To K_i (Partition Zoom)

We refer to the ability to dynamically change the size of the clustering solution in a data partition as the zoom feature. Specifically, each data partition Di has a corresponding k_i value, where k_i refers to the number of clusters estimated In the clustering solution for that partition. For example, a value of k_i = 5 corresponds to the clustering solution for the i^{th} partition that contains exactly five clusters. The k_i value here is analogous with the k in the case of k-means clustering, where

the user typically specifies a k value up front, and the data are then partitioned into k clusters. Temporal cluster graphs provide the user with the control to adjust and visualize the clustering solution for each time partition in real time. For example, in a partition with 100 data points, changing $k_i = 3$ to $k_i = 2$ would recluster the data points from three clusters into two clusters. In all clustering methods, there is variability in the solution based on the user's understanding of the data and his or her interpretation of the output. Many clustering techniques assume that the number of clusters is known ahead of time (e.g., k-means clustering) and, therefore, a common problem in cluster analysis is deciding the optimal number of clusters that are present in a data set.

3.2.3 With In Period Time Strength

As discussed earlier, nodes of the trend graph are created by clustering each data partition. These data partitions identify the common naturally occurring patterns in the data. However, not all the clusters that are formed would not be large enough to be considered. For example, in a data set of 2,000 data points, a cluster of size s=2 (i.e., containing only two data points) would likely be spurious for many practical applications. Additionally, it is possible to have singleton (means only one) clusters appear in the final cluster solution for some clustering approaches such as agglomerative hierarchical clustering. To address these issues and provide more data visualization control to the user of temporal cluster graphs, an user-specified parameter was introduced and it can be used to determine if nodes generated by the clustering solution are "strong" enough to be included in the trend analysis. For every data partition Di,it gives ki clusters in clustering solution and some of these ki clusters can be filtered out for fake edges based on the "within-period trend strength parameter α". The Є [0, 1] parameter used as universal paramter for the provided entire data set. Here the each and every data partition utilizes the same value of α. Let Vi be the set of clusters in data partition Di, i.e., Vi contains the ki clusters identified in the clustering solution for Di. For every cluster j Є Vi, if the cluster size Sj >= α|Di|, then cluster j is included as a node in the output graph. Thus, α|Di| is the minimum node size threshold for data partition Di, where α|Di| is the number of data points in Di.

3.2.4 Cross Period Time Strength

Temporal cluster graphs, edges are used to represent relationships between nodes (clusters) in adjacent time partitions. In general an edge is possible between two nodes. In the adjacent time partitions the edges that are included in the graph are limited those are incident to very similar nodes, to represent a trend over time. Because the concept of what is "very similar" can be domain specific, we introduce a user-specified cross-period trend strength parameter β that is used to filter out fake edges based on their weight. An edge is included in the output graph if it meets two criteria: 1) the edge is incident to two nodes that are both included in the output graph (as determined by the clustering solution and within-period trend strength α), and 2) If the edge weight is less than or equal to a threshold η that depends on the user-specified cross-period trend strength parameter β. In practical this edge threshold η is calculated by taking the average of the weights of all the possible edges among the nodes in two adjacent data partitions (say, partitions i and i+1) and adjusting it by the user-specified β parameter

$$\eta_{i,\,i+1} = \beta \text{AvgDist}(V_i, V_{i+1}) = \beta \left(\sum_{p=}^{K_i} \sum^{K_{i+1}} (V_{i,\,p}, V_{i+1,\,p}) \right) / (k_i\, k_{i+1})$$

Here, $V_{i,p}$ is the p^{th} node in the i^{th} partition, and $d(v_{i,p}, v_{i,p+1})$ is the distance between nodes $V_{i,p}$ and $V_{i,p+1}$. The calculation employs the average edge weight between partitions i and i + 1.

3.2.5 Implementation Of Graph

One of the main goals of a data visualization technique is to reduce complexity and present information in a useful and perceptive manner, it is necessary to provide a means for displaying information at different levels of analysis (i.e. facilitate interactive zooming), as well as filtering the fake entities from a temporal cluster graph (i.e., facilitate interactive filtering). So we define a new analytical construct called the temporal cluster graph. We finally construct the graph using jsGraphics package. It gives the graphical form of the multi dimensional data and allows user to modify Ki, α, β values and analyze the trend graph.

IV. Conclusion

This paper presents a new approach called slicing to privacy-preserving microdata publishing. The drawbacks of generalization and bucketization was overcome by using this slicing and also preserves better utility while protecting against privacy threats. Slicing prevents attribute disclosure, membership disclosure and also provides better data utility .The tradeoffs between data utility and privacy was clearly observed by using Hierarchical clustering. Here the data sets are partitioned based on node and time strengths and are clearly visualized using CTREND graphs.

References

[1] Roberto J. Bayardo and Rakesh Agrawal. Data privacy through optimal k -anonymization. In ICDE '05: Proceedings of the 21st International Conference on Data Engineering, pages 217–228, Washington, DC, USA, 2005. IEEE Computer Society.

[2] B. Chen, R. Ramakrishnan, and K. LeFevre. Privacy skyline: Privacy with multidimensional adversarial knowledge. In VLDB '07: Proceedings of the 33rd international conference on Very large data bases, pages 770–781. VLDB Endowment, 2007.

[3] Sweeney, Latanya. Uniqueness of Simple Demographics in the U.S. Population. In LIDAP-WP4 Carnegie Mellon University, Laboratory for International Data Privacy, Pittsburgh, PA: 2000

[4] Kristen LeFevre, David J. DeWitt, and Raghu Ramakrishnan. Incognito: efficient full-domain k-anonymity. In SIGMOD ’05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data, pages 49–60, New York, NY, USA, 2005. ACM.

[5] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkatasubramaniam. L-diversity: Privacy beyond k-anonymity. ACM Trans. Knowl. Discov. Data, 1(1):3, 2006.

[6] D. Martin, D. Kifer, A. Machanavajjhala, J. Gehrke, and J. Halpern. Worst-case background knowledge. In ICDE ’07: Proceedings of the 21st International Conference on Data Engineering, pages 126–135. IEEE Computer Society, 2007.

[7] B. C. M. Fung, K. Wang, and P. S. Yu. Top-down specialization for information and privacy preservation. In ICDE, pages 205–216, 2005.

[8] G. Ghinita, Y. Tao, and P. Kalnis. On the anonymization of sparse high-dimensional data. In ICDE, pages 715–724, 2008.

[9] A. Inan, M. Kantarcioglu, and E. Bertino. Using anonymized data for classification. In ICDE, 2009.

