SCISPACE
formerly Typeset

Open access • Proceedings Article • DOI:10.1109/ICDE.2005.143

Top-down specialization for information and privacy preservation — Source link

Benjamin C. M. Fung, Ke Wang, Philip S. Yu

Institutions: Simon Fraser University

Published on: 05 Apr 2005 - International Conference on Data Engineering

Topics: Information privacy, Information sensitivity, Information sharing, Data publishing and k-anonymity

Related papers:

- k -anonymity: a model for protecting privacy
- Incognito: efficient full-domain K-anonymity
- Data privacy through optimal k-anonymization
- Mondrian Multidimensional K-Anonymity
- Transforming data to satisfy privacy constraints

Share this paper:

View more about this paper here: https://typeset.io/papers/top-down-specialization-for-information-and-privacy-1t22xcaf2d

Top-Down Specialization for Information and Privacy Preservation

Benjamin C. M. Fung
School of Computing Science
Simon Fraser University
BC, Canada V5A 1S6
bfung@cs.sfu.ca

Ke Wang
School of Computing Science
Simon Fraser University
BC, Canada V5A 1S6
wangk@cs.sfu.ca

Philip S. Yu
IBM T. J. Watson
Research Center
Hawthorne, NY 10532
psyu@us.ibm.com

Abstract

Releasing person-specific data in its most specific state poses a threat to individual privacy. This paper presents a practical and efficient algorithm for determining a generalized version of data that masks sensitive information and remains useful for modelling classification. The generalization of data is implemented by specializing or detailing the level of information in a top-down manner until a minimum privacy requirement is violated. This top-down specialization is natural and efficient for handling both categorical and continuous attributes. Our approach exploits the fact that data usually contains redundant structures for classification. While generalization may eliminate some structures, other structures emerge to help. Our results show that quality of classification can be preserved even for highly restrictive privacy requirements. This work has great applicability to both public and private sectors that share information for mutual benefits and productivity.

1. Introduction

Data sharing in today globally networked systems poses a threat to individual privacy and organizational confidentiality. An example in Samarati [9] shows that linking medication records with a voter list can uniquely identify a person's name and her medical information. In response, new privacy acts and legislations are recently enforced in many countries. In 2001, Canada launched the *Personal Information Protection and Electronic Document Act* [11] to protect a wide spectrum of information, e.g., age, race, income, evaluations, and even intentions to acquire goods or services. This information spans a considerable portion of many databases. Government agencies and companies have to revise their systems and practices to fully comply with this act in three years.

Consider a table T about patient's information on Diagnosis, Zip code, Birthdate, and Sex. If a description on (Zip code, Birthdate, Sex) is so specific that not many people match it, releasing the table will lead to linking a unique or a small number of individuals with the information on Diagnosis. Even if sensitive attributes like Diagnosis are not contained in the table to be released, but contained in an external source, they can be linked to identifying attributes (Zip code, Birthdate, Sex) [9].

Suppose that Zip code, Birthdate, Sex, and Diagnosis must be released, say, to some health research institute for research purpose. We can still protect privacy by replacing specific descriptions with less specific but semantically consistent descriptions. This operation is called *generalization*. For example, by replacing Birthdate with Birthyear (or replacing a specific age value with an interval), more records will match a description on (Zip code, Birthyear, and Sex), making the description less identifying. The premise is that a table T contains only a subset of all individuals. If a description is less specific, more individuals, both within and outside T, will match it, making it difficult to tell whether a matched individual actually has the diagnosis as recorded in T.

Protecting privacy is one side of the story. Another side of the story is making the released data useful to data analysis. Can the two goals be achieved at the same time? In this paper, we like to answer this question for using the data for classification. Our insight is that these two goals are really dealing with two types of information: The privacy goal requires to mask sensitive information, usually *specific* descriptions that identify individuals, whereas the classification goal requires to extract *general* structures that capture trends and patterns. If generalization is performed "carefully", identifying information can be masked while still preserving the trends and patterns for classification. Our experimental results support this insight.

Example 1 *[The running example]* Consider the data in Table 1 and taxonomy trees in Figure 1. Originally, the table has 34 records. We have removed irrelevant attributes and now each row represents one or more original records. The

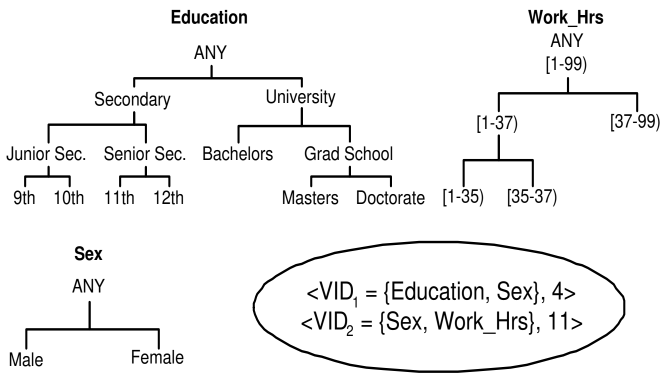


Figure 1. Taxonomy trees and VIDs

Education	Sex	Work_Hrs	Class	# of Recs.
9th	M	30	0Y3N	3
10th	M	32	0Y4N	4
11th	M	35	2Y3N	5
12th	F	37	3Y1N	4
Bachelors	F	42	4Y2N	6
Bachelors	F	44	4Y0N	4
Masters	M	44	4Y0N	4
Masters	F	44	3Y0N	3
Doctorate	F	44	1Y0N	1
			Total:	34

Table 1. (Compressed) table

$Class$ column contains the class frequency of the records represented, Y for income >50K and N for otherwise. To make "female doctor" less unique, we can generalize *Masters* and *Doctorate* to *Grad School*. As a result, "she" becomes less distinguishable by being one of the four inside and many more outside the table who are female and have graduate school degrees. As far as classification is concerned, no information is lost in this generalization because $Class$ does not depend on the distinction of *Masters* and *Doctorate*. ■

In this paper, the privacy goal is specified by the *anonymity* on a combination of attributes called a *virtual identifier*, where each description on a virtual identifier is required to be shared by some minimum number of records in the table. A generalization taxonomy tree is specified for each categorical attribute in a virtual identifier. We present a Top-Down Specialization (TDS) approach to generalize a table to satisfy the anonymity requirement while preserving its usefulness to classification. TDS generalizes the table by *specializing* it iteratively starting from the most general state. At each step, a general (i.e. parent) value is specialized into a specific (i.e. child) value for a categorical attribute, or an interval is split into two sub-intervals for a continuous attribute. This process is repeated until further specialization leads to a violation of the anonymity requirement.

A number of features make TDS practical:

- *Information and privacy-guided specialization.* Both information and privacy are considered in selecting the specialization at each step. Experiments show that classification based on the generalized data yields an accuracy comparable to classification based on the unmodified data, even for a very restrictive anonymity requirement.

- *Handling both categorical and continuous attributes.* This method does not require a priori discretized taxonomy tree for a continuous attribute, but dynamically obtains one in the top-down specialization process. TDS is especially efficient for handling continuous attributes in that a small number of intervals is obtained via a small number of specializations.

- *Handling multiple virtual identifiers.* Often, there are more than one virtual identifier to identify individuals. Treating all virtual identifiers as a single united virtual identifier leads to unnecessary generalization and loss of data quality. Our method allows specialization on multiple virtual identifiers to avoid unnecessary generalization.

- *Scalable computation.* At each iteration, a key operation is updating some goodness metric of affected candidate specializations. In general, this requires accessing data records. Our method incrementally maintains some "count statistics" to eliminate the expensive data access. Simple but effective data structures are proposed for this purpose. Our method deals with the compressed table, which is usually smaller than the original table, and is amendable to disk-resident data.

- *Anytime solution.* The top-down approach provides a natural tree-growing procedure that allows the user to step through each specialization to determine a desired trade-off between privacy and accuracy. The user may stop at *any time* and have a table satisfying the anonymity requirement. This property is not shared by the bottom-up generalization that starts with the most precise state.

We review related work in Section 2, define the problem in Section 3, define the goodness criterion of specialization in Section 4, present the top-down specialization approach in Section 5, evaluate this approach in Section 6. Finally, we conclude the paper.

2. Related Work

Many techniques have been proposed previously to preserve privacy, but most are not intended for classification. For example, Kim et al. [7] and Fuller [3] show that some simple statistical information, like means and correlations, can be preserved by adding noise and swapping values. Recently, this technique was studied in data mining for classification [1]. In these works, privacy was measured by how closely the original values of a masked attribute can be estimated, which is very different from the notion of anonymity that quantifies how uniquely an individual can be linked with sensitive information.

The concept of anonymity was proposed in [2]. Bottom-up generalization was used to achieve anonymity in Datafly system [10] and μ-Argus system [5]. These works assumed a single virtual identifier that includes all attributes that could potentially be used to link the data to an external source. In our work, the linking can be either to an external source or among the attributes in the table itself. In the latter case all virtual identifiers may be known at the time of problem specification. They did not consider classification or a specific use of data, and simple heuristics were used, such as selecting the attribute having most number of distinct values [10] or values not having k occurrences [5] to generalize. Such heuristics did not address the quality for classification.

Iyengar [6] presented the anonymity problem for classification, and proposed a genetic algorithm solution. The idea is encoding each state of generalization as a "chromosome" and encoding data distortion into the fitness function, and employing the genetic evolution to converge to the fittest chromosome. Wang et al. [12] recently presented an effective bottom up approach, but it can only generalize categorical attributes. Unlike the *random* genetic evolution and the bottom-up generalization, our approach produces a *progressive* generalization process which the user can step through to determine a desired trade-off of privacy and accuracy. We handle both categorical and continuous attributes, and multiple virtual identifiers. We focus on scalability, in addition to data quality. Iyengar reported 18 hours to transform 30K records. Our algorithm took only 7 seconds to produce a comparable accuracy on the same data. For large databases, Iyengar suggested running his algorithm on a sample. However, a small sampling error could mean failed protection on the entire data.

3. Problem Definition

A data provider wants to release a person-specific table $T(D_1, \ldots, D_m, Class)$ to the public for modelling the class label $Class$. Each D_i is either a categorical or a continuous attribute. A record has the form $<v_1, \ldots, v_m, cls>$, where v_i is a domain value for D_i and cls is a class for $Class$. $att(v)$ denotes the attribute of a value v. The data provider also wants to protect against linking an individual to sensitive information either within or outside T through some identifying attributes, called a *virtual identifier* or simply VID. A sensitive linking occurs if some value of the virtual identifier is shared by only a "small" number of records in T. This requirement is formally defined below.

Definition 1 (Anonymity requirement) Consider p virtual identifiers $VID_1, \ldots, VID_p$ on T. $a(vid_i)$ denotes the number of data records in T that share the value vid_i on VID_i. The *anonymity* of VID_i, denoted $A(VID_i)$, is the smallest $a(vid_i)$ for any value vid_i on VID_i. A table T satisfies the *anonymity requirement* $\{<VID_1, k_1>, \ldots, <VID_p, k_p>\}$ if $A(VID_i) \geq k_i$ for $1 \leq i \leq p$, where k_i is the *anonymity threshold* on VID_i specified by the data provider. ■

We assume that VID_j is not a subset of VID_i, $j \neq i$, otherwise, VID_j can be removed from the requirement.

Example 2 Consider Table 1. To protect against sensitive linking through {*Education,Sex*}, we specify $<VID_1 = \{Education, Sex\}, 4>$, that is, every existing vid in T must be shared by at least 4 records in T. The following vids violate this requirement: <*9th*, *M*>, <*Masters*, *F*>, <*Doctorate*, *F*>. To protect against linking through {*Sex*, *Work_Hrs*} as well, the anonymity requirement could include the two VIDs such as in Figure 1. ■

To generalize T to satisfy the anonymity requirement, a *taxonomy tree* is specified for each categorical attribute in $\cup VID_i$, by either the data provider or the data recipient. A leaf node represents a domain value and a parent node represents a less specific value. For a continuous attribute in $\cup VID_i$, our algorithm dynamically grows a taxonomy tree at runtime, where each node represents an interval, and each non-leaf node has two child nodes representing some "optimal" binary split of the parent interval. More details on this will be discussed in Section 4. Figure 1 shows a dynamically grown taxonomy tree for *Work_Hrs*.

Let $child(v)$ be the set of child values of v in the taxonomy tree. A table T can be generalized by a sequence of generalizations starting from data records. A *generalization*, written $child(v) \rightarrow v$, replaces child values in $child(v)$ with their parent value v. To ensure that the result is easy to understand, prior to the generalization, all values below the values in $child(v)$ should be generalized to those in $child(v)$ first, so that a descendant value and an ancestor value do not coexist in the generalized data.

Alternatively, T can be generalized by a sequence of specializations starting from the most general state in which each attribute has the top most value of its taxonomy tree.

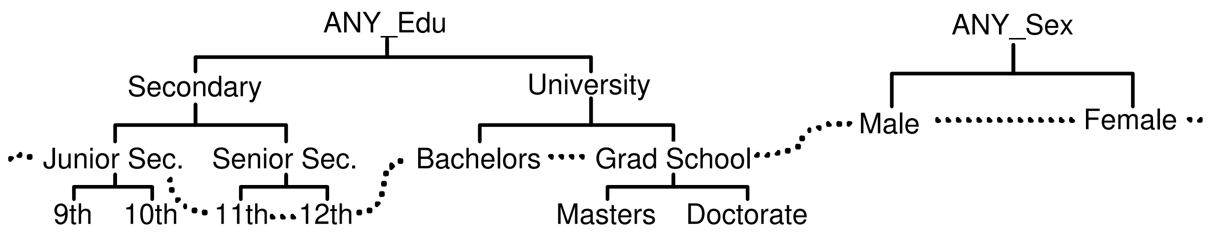


Figure 2. Example 3: A solution cut for $VID_1 = \{Education, Sex\}$

A *specialization*, written $v \rightarrow child(v)$, replaces the parent value v with a child value in $child(v)$ that generalizes the domain value in a record. A specialization is *valid* (with respect to T) if T satisfies the anonymity requirement after the specialization. A specialization is *beneficial* (with respect to T) if more than one class is involved in the records containing v. A specialization is performed only if it is both valid and beneficial.

Definition 2 (Anonymity for Classification) Given a table T, an anonymity requirement $\{<VID_1, k_1>, \ldots, <VID_p, k_p>\}$, and a taxonomy tree for each categorical attribute contained in $\cup VID_i$, generalize T on the attributes $\cup VID_i$ to satisfy the anonymity requirement while preserving as much information as possible for classification. ■

A generalized T can be viewed as a "cut" through the taxonomy tree of each attribute in $\cup VID_i$. A *cut* of a tree is a subset of values in the tree that contains exactly one value on each root-to-leaf path. A *solution cut* is $\cup Cut_j$, where Cut_j is a cut of the taxonomy tree of an attribute in $\cup VID_i$, such that the generalized T represented by $\cup Cut_j$ satisfies the anonymity requirement. We are interested in a solution cut that maximally preserves the information for classification.

Example 3 Continue with Example 2. Figure 2 shows a solution cut, indicated by the dashed curve. This solution cut is the lowest in the sense that any specialization on *Junior Sec.* or *Grad School* would violate the anonymity requirement, i.e., is invalid. Also, specialization on *Junior Sec.* or *Grad School* is non-beneficial since none of them specializes data records in different classes. ■

4. Search Criteria

Our top-down specialization starts from the top most solution cut and pushes down the solution cut iteratively by specializing some value in the current solution cut until violating the anonymity requirement. Each specialization tends to increase information and decrease anonymity because records are more distinguishable by specific values. The key is selecting a specialization at each step with both impacts considered. In this section, we consider a goodness metric for a single specialization.

Consider a specialization $v \rightarrow child(v)$. Let R_v denote the set of records generalized to the value v, and let R_c denote the set of records generalized to a child value c in $child(v)$ after specializing v. Let $|x|$ be the number elements in a set x. $|R_v| = \sum_c |R_c|$, where $c \in child(v)$. The effect of specializing v is summarized by the "information gain", denoted $InfoGain(v)$, and the "anonymity loss", denoted $AnonyLoss(v)$. To heuristically maximize the information of the generalized data for achieving a given anonymity, we favor the specialization on v that has the maximum information gain for each unit of anonymity loss:

$$Score(v) = \begin{cases} \frac{InfoGain(v)}{AnonyLoss(v)} & if\ AnonyLoss(v) \neq 0 \\ InfoGain(v) & otherwise \end{cases}$$

InfoGain(v): defined as

$$InfoGain(v) = I(R_v) - \sum_c \frac{|R_c|}{|R_v|} I(R_c)$$

where $I(R_x)$ is the *entropy* of R_x [8]:

$$I(R_x) = -\sum_{cls} \frac{freq(R_x, cls)}{|R_x|} \times log_2 \frac{freq(R_x, cls)}{|R_x|}$$

$freq(R_x, cls)$ is the number of data records in R_x having the class cls. Intuitively, $I(R_x)$ measures the "mix" of classes for the data records in R_x. The two terms in $InfoGain(v)$ are the mix of classes before and after specializing v. A good specialization will reduce the mix of classes, i.e., have a large $InfoGain(v)$. Note that $InfoGain(v)$ is non-negative [8].

AnonyLoss(v): defined as

$$AnonyLoss(v) = avg\{A(VID_j) - A_v(VID_j)\},$$

where $A(VID_j)$ and $A_v(VID_j)$ represents the anonymity before and after specializing v. $A(VID_j) - A_v(VID_j)$ is the loss of anonymity of VID_j, and $avg\{A(VID_j) - A_v(VID_j)\}$ is the average loss of all VID_j that contain the attribute of v.

Example 4 The specialization on *ANY_Edu* refines the 34 records into 16 records for *Secondary* and 18 records for *University*. The calculation of $Score(ANY\_Edu)$ is shown below.

$I(R_{ANY\_Edu}) = -\frac{21}{34} \times log_2 \frac{21}{34} - \frac{13}{34} \times log_2 \frac{13}{34} = 0.9597$
$I(R_{Secondary}) = -\frac{5}{16} \times log_2 \frac{5}{16} - \frac{11}{16} \times log_2 \frac{11}{16} = 0.8960$
$I(R_{University}) = -\frac{16}{18} \times log_2 \frac{16}{18} - \frac{2}{18} \times log_2 \frac{2}{18} = 0.5033$
$InfoGain(ANY\_Edu) = I(R_{ANY\_Edu}) - (\frac{16}{34} \times I(R_{Secondary}) + \frac{18}{34} \times I(R_{University})) = 0.2716$
$AnonyLoss(ANY\_Edu) = avg\{A(VID_1) - A_{ANY\_Edu}(VID_1)\} = (34 - 16)/1 = 18$
$Score(ANY\_Edu) = \frac{0.2716}{18} = 0.0151$. ■

For a continuous attribute, the specialization of an interval refers to the optimal binary split of the interval that maximizes the information gain. We use information gain, instead of $Score$, to determine the split of an interval because anonymity is irrelevant to finding a split good for classification. This is similar to the situation that the taxonomy tree of a categorical attribute is specified independently of the anonymity issue.

Example 5 For the continuous attribute *Work_Hrs*, the top most value is the full range interval of domain values, *[1-99)*. To determine the split point of *[1-99)*, we evaluate the information gain for the five possible split points for the values 30, 32, 35, 37, 42, and 44. The following is the calculation for the split point at 37:
$InfoGain(37) = I(R_{[1-99)}) - (\frac{12}{34} \times I(R_{[1-37)}) + \frac{22}{34} \times I(R_{[37-99)}))$
$= 0.9597 - (\frac{12}{34} \times 0.6500 + \frac{22}{34} \times 0.5746) = 0.3584.$
As $InfoGain(37)$ is highest, we grow the taxonomy tree for *Work_Hrs* by adding two child intervals, *[1-37)* and *[37-99)*, under the interval *[1-99)*. ■

The next example shows that $InfoGain$ alone may lead to a quick violation of the anonymity requirement, thereby, prohibiting specializing data to a lower granularity.

Example 6 Consider Table 2, an anonymity requirement $<VID = \{Education, Sex, Work\_Hrs\}, 4>$, and specializations:
ANY_Edu → {*8th, 9th, 10th*},
ANY_Sex → {*M, F*}, and
[1-99) → {*[1-40), [40-99)*}.
The class frequency for the specialized values is:
Education: 0Y4N (8th), 0Y12N (9th), 20Y4N (10th)
Sex: 20Y6N (M), 0Y14N (F)
Work_Hrs: 0Y12N ([1-40)), 20Y8N ([40-99))
Specializing *Education* best separates the classes, so is chosen by $InfoGain$. After that, the other specializations become invalid. Now, the two classes of the top 24 records become indistinguishable because they are all generalized into <*10th, ANY_Sex, [1-99)*>.

In contrast, the $Score$ criterion will first specialize *Sex* because of the highest $Score$ due to a small $AnonyLoss$. Subsequently, specializing *Education* becomes invalid, and the next specialization is on *Work_Hrs*. The final generalized table is shown in Table 3 where the information for distinguishing the two classes is preserved. ■

Education	Sex	Work_Hrs	Class	# of Recs.
10th	M	40	20Y0N	20
10th	M	30	0Y4N	4
9th	M	30	0Y2N	2
9th	F	30	0Y4N	4
9th	F	40	0Y6N	6
8th	F	30	0Y2N	2
8th	F	40	0Y2N	2
			Total:	40

Table 2. (Compressed) table for Example 6

Education	Sex	Work_Hrs	Class	# of Recs.
ANY_Edu	M	[40-99)	20Y0N	20
ANY_Edu	M	[1-40)	0Y6N	6
ANY_Edu	F	[40-99)	0Y8N	8
ANY_Edu	F	[1-40)	0Y6N	6

Table 3. Generalized table by $Score$ for Example 6

5. Top-Down Specialization

5.1. The Algorithm

We present our algorithm, *Top-Down Specialization (TDS)*. In a preprocessing step, we compress the given table by removing all attributes not in $\cup VID_j$ and collapsing duplicates into a single row with the $Class$ column storing the class frequency as in Table 1. The compressed table is typically much smaller than the original table. Below, the term "data records" refers to data records in this compressed form. To focus on main ideas, we assume that the compressed table fits in the memory. In Section 5.5, we will discuss the modification needed if the compressed table does not fit in the memory.

Table 4 summarizes the conceptual algorithm. Initially, Cut_i contains only the top most value for its attribute. The valid, beneficial specializations in $\cup Cut_i$ form the set of *candidates* to be performed next. At each iteration, we find the candidate of the highest $Score$, denoted $Best$ (Line 5), apply $Best$ to T and update $\cup Cut_i$ (Line 6), and update $Score$ and validity of the candidates in $\cup Cut_i$ (Line 7). The algorithm terminates when there is no more candidate in $\cup Cut_i$, in which case it returns the generalized table together with $\cup Cut_i$.

Example 7 Consider the anonymity requirement:
$\{<VID_1 = \{Education, Sex\}, 4>,$
$<VID_2 = \{Sex, Work\_Hrs\}, 11>\}.$
Initially, all data records in Table 1 are generalized to
<*ANY_Edu, ANY_Sex, [1-99)*>,
and

$\cup Cut_i = \{$*ANY_Edu, ANY_Sex, [1-99)*$\}$.
All specializations in $\cup Cut_i$ are candidates. To find the best specialization, we need to compute $Score(ANY\_Edu)$, $Score(ANY\_Sex)$, and $Score($*[1-99)*$)$. ■

```
1  Algorithm TDS
2    Initialize every value in $T$ to the top most value.
3    Initialize $Cut_i$ to include the top most value.
4    **while** some $x \in \cup Cut_i$ is valid and beneficial **do**
5      Find the $Best$ specialization from $\cup Cut_i$.
6      Perform $Best$ on $T$ and update $\cup Cut_i$.
7      Update $Score(x)$ and validity for $x \in \cup Cut_i$.
8    **end while**
9    **return** Generalized $T$ and $\cup Cut_i$.
```

Table 4. Top-Down Specialization (TDS)

The algorithm in Table 4 makes no claim on efficiency. In a straightforward method, Line 6 and 7 require scanning all data records and recomputing $Score$ for all candidates in $\cup Cut_i$. Obviously, this is not scalable. The key to the efficiency of our algorithm is *directly* accessing the data records to be specialized, and updating $Score$ based on some statistics maintained for candidates in $\cup Cut_i$, instead of accessing data records. In the rest of this section, we explain our scalable implementation and data structures in details.

5.2. Find the Best Specialization

This step makes use of computed $InfoGain(x)$ and $A_x(VID_j)$ for all candidates x in $\cup Cut_i$ and computed $A(VID_j)$ for each VID_j. Before the first iteration, such information is computed in an initialization step for every top most value. For each subsequent iteration, such information comes from the update in the previous iteration (Line 7). Finding the best specialization $Best$ involves at most $| \cup Cut_i|$ computations of $Score$ without accessing data records. Updating $InfoGain(x)$ and $A_x(VID_j)$ will be considered in Section 5.4.

5.3. Perform the Best Specialization

First, we replace $Best$ with $child(Best)$ in $\cup Cut_i$. Then, we need to retrieve R_{Best}, the set of data records generalized to $Best$, to tell the child value in $child(Best)$ for individual data records. We first present a data structure, *Taxonomy Indexed PartitionS (TIPS)*, to facilitate this operation. This data structure is also crucial for updating $InfoGain(x)$ and $A_x(VID_j)$ for candidates x. The general idea is to group data records according to their generalized records on $\cup VID_j$.

Definition 3 (TIPS) TIPS is a tree structure with each node representing a generalized record over $\cup VID_j$, and each child node representing a specialization of the parent node on exactly one attribute. Stored with each leaf node is the set of (compressed) data records having the same generalized record, called a *leaf partition*. For each x in $\cup Cut_i$, P_x denotes a leaf partition whose generalized record contains x, and $Link_x$ denotes the link of all P_x, with the head of $Link_x$ stored with x. ■

At any time, the generalized data is represented by the leaf partitions of TIPS, but the original data records remain unchanged. $Link_x$ provides a direct access to R_x, the set of data records generalized to the value x. Initially, TIPS has only one leaf partition containing all data records, generalized to the top most value on every attribute in $\cup VID_j$. In each iteration, we perform the best specialization $Best$ by refining the leaf partitions on $Link_{Best}$.

Update TIPS. We refine each leaf partition P_{Best} found on $Link_{Best}$ as follows. For each value c in $child(Best)$, a child partition P_c is created under P_{Best}, and data records in P_{Best} are split among the child partitions: P_c contains a data record in P_{Best} if c generalizes the corresponding domain value in the record. An empty P_c is removed. $Link_c$ is created to link up all P_c's for the same c. Also, link P_c to every $Link_x$ to which P_{Best} was previously linked, except for $Link_{Best}$. Finally, mark c as "beneficial" if R_c has more than one class. Recall that R_c is the set of data records generalized to c.

We emphasize that this is the only operation in the whole algorithm that requires accessing data records. The overhead of maintaining $Link_x$ is small. For each attribute in $\cup VID_j$ and each leaf partition on $Link_{Best}$, there are at most $|child(Best)|$ "relinkings", or at most $| \cup VID_j| \times |Link_{Best}| \times |child(Best)|$ "relinkings" in total for applying $Best$.

Example 8 Continue with Example 7. Initially, TIPS has only one leaf partition containing all data records and representing the generalized record <*ANY_Edu, ANY_Sex, [1-99)*>. Let the best specialization be *[1-99)* $\rightarrow$ $\{$*[1-37), [37-99)*$\}$ on *Work_Hrs*. We create two child partitions under the root partition as in Figure 3, and split data records between them. Both child partitions are on $Link_{ANY\_Edu}$ and $Link_{ANY\_Sex}$. $\cup Cut_i$ is updated into $\{$*ANY_Edu, ANY_Sex, [1-37), [37-99)*$\}$. Suppose that the next best specialization is *ANY_Edu* $\rightarrow$ $\{$*Secondary, University*$\}$, which specializes the two leaf partitions on $Link_{ANY\_Edu}$, resulting in the TIPS in Figure 3. ■

A scalable feature of our algorithm is maintaining some statistical information for each candidate x in $\cup Cut_i$ for updating $Score(x)$ without accessing data records. For each new value c in $child(Best)$ added to $\cup Cut_i$ in the current iteration, we collect the following *count statistics* of c while scanning data records in P_{Best} for updating TIPS:

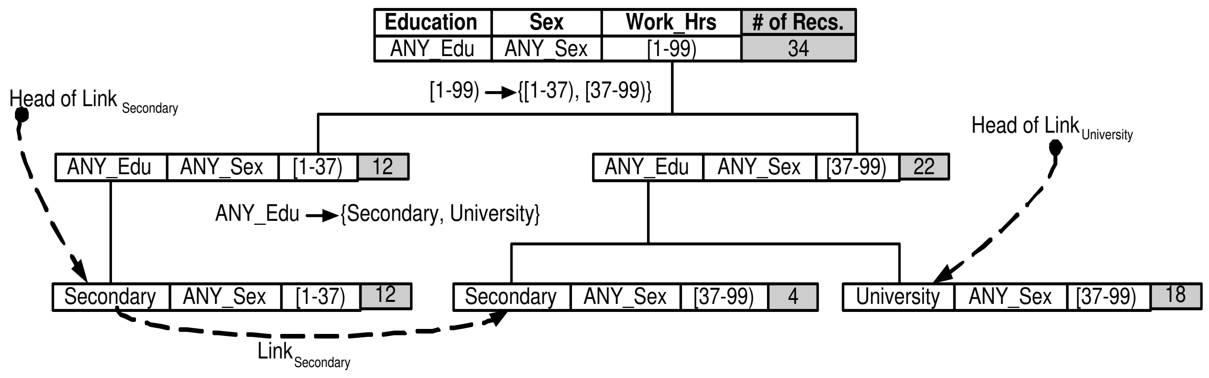


Figure 3. The TIPS data structure

(1) $|R_c|$, $|R_d|$, $freq(R_c, cls)$, and $freq(R_d, cls)$ for computing $InfoGain(c)$, where $d \in child(c)$ and cls is a class label. Refer to Section 4 for these notations. (2) $|P_d|$, where P_d is a child partition under P_c *as if* c is specialized, kept together with the leaf node for P_c. These information will be used in Section 5.4.

TIPS has several useful properties. (1) All data records in the same leaf partition have the same generalized record although they may have different lower level values. (2) Every data record appears in exactly one leaf partition. (3) Each leaf partition P_x has exactly one generalized vid_j on VID_j and contributes the count $|P_x|$ towards $a(vid_j)$. Later, we use the last property to extract $a(vid_j)$ from TIPS.

5.4. Update the Score

This step updates $Score(x)$ for candidates x in $\cup Cut_i$ to reflect the impact of $Best \to child(Best)$. The key to the efficiency of our algorithm is computing $Score(x)$ from the count statistics maintained in Section 5.3 without accessing data records. We update $InfoGain(x)$ and $A_x(VID_j)$ separately. Note that the updated $A(VID_j)$ is obtained from $A_{Best}(VID_j)$.

5.4.1 Update $InfoGain(x)$

A quick observation is that $InfoGain(x)$ is not affected by applying $Best \to child(Best)$ except that we need to compute $InfoGain(c)$ for each value c in $child(Best)$. $InfoGain(c)$ can be computed while collecting the count statistics for c in Section 5.3.

5.4.2 Update $AnonyLoss(x)$

Unlike information gain, it is not enough to compute $A_c(VID_j)$ only for each c in $child(Best)$. Recall that $A_x(VID_j)$ is the minimum $a(vid_j)$ after specializing x. If $att(x)$ and $att(Best)$ are contained in VID_j, the specialization on $Best$ may affect this minimum, hence, $A_x(VID_j)$. Below, we present a data structure, *Virtual Identifier TreeS* ($VITS$), to extract $a(vid_j)$ efficiently from TIPS for updating $A_x(VID_j)$.

Definition 4 (VITS) VIT_j for $VID_j = \{D_1, \ldots, D_w\}$ is a tree of w levels. The level $i > 0$ represents the generalized values for D_i. Each root-to-leaf path represents an existing vid_j on VID_j in the generalized data, with $a(vid_j)$ stored at the leaf node. A branch is trimmed if its $a(vid_j) = 0$. $A(VID_j)$ is the minimum $a(vid_j)$ in VIT_j. ■

In other words, VIT_j provides an index of $a(vid_j)$ by vid_j. Unlike TIPS, VITS does not maintain data records. On applying $Best \to child(Best)$, we update every VIT_j such that VID_j contains $att(Best)$.

Update VIT_j. For each occurrence of $Best$ in VIT_j, create a separate branch for each c in $child(Best)$. The procedure in Table 5 computes $a(vid_j)$ for the newly created vid_j's on such branches. The general idea is to loop through each P_c on $Link_c$ in TIPS, increment $a(vid_j)$ by $|P_c|$. This step does not access data records because $|P_c|$ was part of the count statistics of $Best$. Let r be the number of VID_j containing $att(Best)$. The number of $a(vid_j)$ to be computed is at most $r \times |Link_{Best}| \times |child(Best)|$.

```
1  Procedure UpdateCounts
2  for each $P_c \in Link_c$ do
3    for each $VID_j$ containing $att(Best)$ do
4      $a(vid_j) = a(vid_j) + |P_c|$, where
         $vid_j$ is the generalized value on $VID_j$ for $P_c$
5    end for
6  end for
```

Table 5. Computing $a(vid_j)$ for new vid_j

Example 9 In Figure 4, the initial VIT_1 and VIT_2 (i.e., left most) have a single path. After applying *[1-99)* $\rightarrow$ $\{$*[1-37)*, *[37-99)*$\}$, the vid <*ANY_Sex*, *[1-99)*> in VIT_2 is replaced with two new vids <*ANY_Sex*, *[1-37)*> and <*ANY_Sex*, *[37-99)*>, and $A(VID_2) = 12$. Since VID_1 does not include *Work_Hrs*, VIT_1 remains unchanged and $A(VID_1) = 34$.

After applying the second specialization *ANY_Edu* $\rightarrow$ $\{$*Secondary*, *University*$\}$, VIT_2 remains unchanged, and $A(VID_2) = 12$. The vid <*ANY_Edu,ANY_Sex*> in VIT_1 is replaced with two new vids <*Secondary,ANY_Sex*> and <*University, ANY_Sex*>. To compute $a(vid)$ for these new vids, we add $|P_{Secondary}|$ on $Link_{Secondary}$ and $|P_{University}|$ on $Link_{University}$ (see Figure 3): $a($<*Secondary*, *ANY_Sex*>$)$ = 0 + 12 + 4 = 16, and $a($<*University*, *ANY_Sex*>$)$ = 0 + 18 = 18. So $A_{ANY\_Edu}(VID_1) = 16$. ■

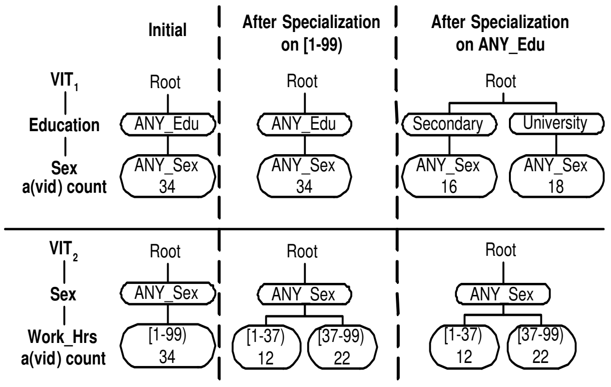


Figure 4. The VITS data structure

Now, we update $A_x(VID_j)$ for candidates x in $\cup Cut_i$ in the impact of $Best \rightarrow child(Best)$. Doing this by specializing x requires accessing data records, hence, is not scalable. We like to compute $A_x(VID_j)$ using the count statistics maintained for x in $\cup Cut_i$ without accessing data records.

Update $A_x(VID_j)$. For a candidate x in $\cup Cut_i$, computing $A_x(VID_j)$ is necessary in two cases. First, x is in $child(Best)$ because $A_x(VID_j)$ has not been computed for newly added candidates x. Second, $A_x(VID_j)$ might be affected by the specialization on $Best$, in which case $att(x)$ and $att(Best)$ must be contained in VID_j. In both cases, we first compute $a(vid_j^x)$ for the new vid_j^x's created *as if* x is specialized. The procedure is the same as in Table 5 for specializing $Best$, except that $Best$ is replaced with x and no actual update is performed on VIT_j and TIPS. Note that the count $|P_c|$, where c is in $child(x)$, in the procedure is part of the count statistics maintained for x.

Next, we compare $a(vid_j^x)$ with $A(VID_j)$ to determine the minimum, i.e., $A_x(VID_j)$. There are two cases:

Case 1: If no contributing vid of $A(VID_j)$ (i.e., those vids such that $a(vid) = A(VID_j)$) contains the value x, then such vids remain existing if x is specialized, so $A_x(VID_j)$ is the minimum of $A(VID_j)$ and $a(vid_j^x)$.

Case 2: If some contributing vid of $A(VID_j)$ contains the value x, such vid's become new vid_j^x if x is specialized, so $A_x(VID_j)$ is the minimum of $a(vid_j^x)$.

Finally, if the new $A_x(VID_j) \geq k_j$, we keep it with x and mark x as "valid" in the cut.

5.5. Discussion

Each iteration involves two types of work. The first type accesses data records in R_{Best} for updating TIPS and count statistics in Section 5.3. The second type computes $Score(x)$ (i.e., $InfoGain(x)$ and $A_x(VID_j)$) for the candidates x in $\cup Cut_i$ without accessing data records in Section 5.4. For a table with m attributes and each taxonomy tree with at most p nodes, the number of such x is at most $m \times p$. This computation makes use of the maintained count statistics, rather than accessing data records. In other words, each iteration accesses only the records being specialized.

In the special case that there is only a single VID, each root-to-leaf path in TIPS has represented a vid, and we can store $a(vid)$ directly at the leaf partitions in TIPS without VITS. A single VID was considered in [10, 6] where the VID contains all potentially identifying attributes to be used for linking the table to an external source. Our algorithm is more efficient in this special case.

To focus on main ideas, our current implementation assumes that the compressed table fits in memory. Often, this assumption is valid because the compressed table can be much smaller than the original table. If the compressed table does not fit in the memory, we can store leaf partitions of TIPS on disk if necessary. Favorably, the memory is used to keep only leaf partitions that are smaller than the page size to avoid fragmentation of disk pages. A nice property of TDS is that leaf partitions that cannot be further specialized (i.e., on which there is no candidate specialization) can be discarded, and only some statistics for them needs to be kept. This likely applies to small partitions in memory, therefore, the memory demand is unlikely to build up.

Compared to iteratively generalizing the data bottom-up starting from domain values, the top-down specialization is more natural and efficient for handling continuous attributes. To produce a small number of intervals for a continuous attribute, the top-down approach needs only a small number of interval splitting, whereas the bottom-up approach needs many interval merging starting from many domain values. In addition, the top-down approach can discard data records that cannot be further specialized, whereas the bottom-up approach has to keep all data records until the end of computation.

Attribute	Type	Numerical Range	
		# of Leaves	# of Levels
Age (A)	continuous	17 - 90	
Capital-gain (Cg)	continuous	0 - 99999	
Capital-loss (Cl)	continuous	0 - 4356	
Education-num (En)	continuous	1 - 16	
Final-weight (Fw)	continuous	13492 - 1490400	
Hours-per-week (H)	continuous	1 - 99	
Education (E)	categorical	16	5
Martial-status (M)	categorical	7	4
Native-country (N)	categorical	40	5
Occupation (O)	categorical	14	3
Race (Ra)	categorical	5	3
Relationship (Re)	categorical	6	3
Sex (S)	categorical	2	2
Work-class (W)	categorical	8	5

Table 6. Attributes for the *Adult* data set

6. Experimental Evaluation

We have three objectives in this section. (1) Verify if the proposed method, TDS, can generalize a given data set to satisfy a broad range of anonymity requirements without sacrificing significantly the usefulness to classification. (2) Compare TDS with the genetic algorithm presented in Iyengar [6] in terms of both accuracy and efficiency. (3) Evaluate the scalability of TDS on larger data sets. The result of Iyengar [6] was quoted from that paper. All experiments based on TDS were conducted on an Intel Pentium IV 2.6GHz PC with 1GB RAM.

We adopted the publicly available benchmark *Adult* data set from [4]. It has 6 continuous attributes, 8 categorical attributes, and a binary $Class$ column representing two income levels, $\leq$50K or $>$50K. Table 6 describes each attribute. After removing records with missing values from the pre-split training and testing sets, we have 30,162 records and 15,060 records for training and testing respectively. This is exactly the same data set as used in [6].

Although the author of [6] has specified taxonomy trees for categorical attributes, we do not agree with the author's groupings. For example, the author grouped *Native-country* according to continents, except Americas. We followed the grouping according to the World Factbook published by CIA[1]. Our taxonomy trees and an executable program of TDS can be obtained from our website[2]. Nevertheless, we did use Iyengar's taxonomy trees in Section 6.2 for comparison purpose.

[1]http://www.cia.gov/cia/publications/factbook/

[2]http://www.cs.sfu.ca/~ddm/

6.1. Data Quality

Our first objective is to evaluate if the proposed TDS preserves the quality for classification while generalizing the data to satisfy various anonymity requirements. We used the C4.5 classifier [8] and Naive Bayesian classifier[3] as classification models. Unless stated otherwise, all 14 attributes were used for building classifiers.

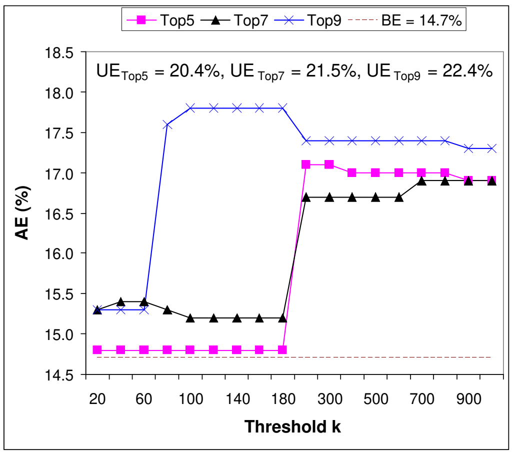
In a typical real life situation, the data provider releases all data records in a single file, leaving the split of training and testing sets to the data miner. Following this practice, we combined the training set and testing set into one set for generalization, and built a classifier using the generalized training set and collected the error using the generalized testing set. This error, called the *anonymity error* and denoted AE, was compared with the *baseline error*, denoted BE, for the unmodified training and testing sets. Note that AE depends on the anonymity requirement. BE is 14.7% for the C4.5 classifier and 18.07% for the Naive Bayesian classifier. $AE - BE$ measures the quality loss due to data generalization.

For the same anonymity threshold k, a single VID is always more restrictive than breaking it into multiple VIDs. For this reason, we first consider the case of single VID. To ensure that generalization is working on attributes that have impacts on classification, the VID contains the top N attributes ranked by the C4.5 classifier. The top rank attribute is the attribute at the top of the C4.5 decision tree. Then we remove this attribute and repeat this process to determine the rank of other attributes. The top 9 attributes are $Cg, A, M, En, Re, H, S, E, O$ in that order. We specified three anonymity requirements denoted Top5, Top7, and Top9, where the VID contains the top 5, 7, and 9 attributes respectively. The *upper error*, denoted UE, refers to the error on the data with all the attributes in the VID removed (equivalent to generalizing them to the top most ANY). $UE - BE$ measures the impact of the VID on classification.

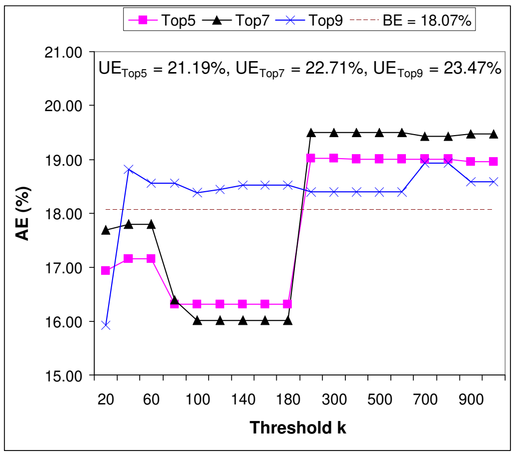
Figure 5a displays AE for the C4.5 classifier with the anonymity threshold $20 \leq k \leq 1000$. Note that k is not spaced linearly. We summarize the analysis for Top7 as follows. First, $AE - BE$, where $BE = 14.7\%$, is less than 2% over the range of anonymity threshold $20 \leq k \leq 600$, and AE is much lower than $UE = 21.5\%$. This supports that accurate classification and privacy protection can coexist. Second, AE *generally* increases as the anonymity threshold k increases, but not monotonically. For example, the error slightly drops when k increases from 60 to 100. This is due to the variation between the training set and testing set and the fact that a better structure may appear in a more general state.

A closer look at the generalized data for Top7 with

[3]http://magix.fri.uni-lj.si/orange/



(a) C4.5



(b) Naive Bayesian

Figure 5. Anonymity error of TDS on TopN

$k = 500$ reveals that among the seven top ranked attributes, three are generalized to a different degree of granularity, and four, namely *A* (ranked 2nd), *Re* (ranked 5th), *S* (ranked 7th), and *Cg* (ranked 1st), are generalized to the top most value *ANY*. Even for this drastic generalization, AE has only increased by 2% from $BE = 14.7\%$, while the worst case can be $UE = 21.5\%$. With the generalization, classification now is performed by the remaining three attributes in the VID and the unmodified but lower ranked attributes. Clearly, this is a different classification structure from what would be found from the unmodified data. As a result, though generalization may eliminate some structures, new structures emerge to help.

Figure 5b displays AE for the Naive Bayesian classifier. Compared to the C4.5 classifier, though BE and UE are higher (which has to do with the classification method, not the generalization), the quality loss due to generalization, $AE - BE$ (note $BE = 18.07\%$), is smaller, no more than 1.5% for the range of anonymity threshold $20 \leq k \leq 1000$. This suggests that the information based generalization is also useful to other classification methods such as the Naive Bayesian that do to use the information bias. Another observation is that AE is even lower than BE for the anonymity threshold $k \leq 180$ for Top5 and Top7. This confirms again that the best structure may not appear in the most specialized data. Our approach uses this room to mask sensitive information while preserving classification structures.

Figure 6 shows the generated taxonomy trees for continuous attributes *Hours-per-week* and *Education-num* with Top7 and $k = 60$. The splits are very reasonable. For example, in the taxonomy tree of *Education-num*, the split point at 13 distinguishes whether the person has post-secondary education. If the user does not like these trees, she may modify them or specify her own and subsequently treat

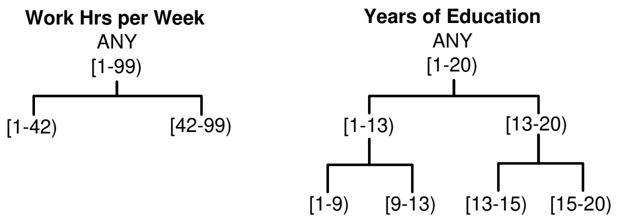


Figure 6. Generated taxonomy trees of *Hours-per-week* **and** *Education-num*

continuous attributes as categorical attributes with specified taxonomy trees.

Our method took at most 10 seconds for all previous experiments. Out of the 10 seconds, approximately 8 seconds were spent on reading data records from disk and writing the generalized data to disk. The actual processing time for generalizing the data is relatively short.

In an effort to study the effectiveness of multiple VIDs, we compared AE between a multiple VIDs requirement and the *corresponding* single united VID requirement. We randomly generated 30 multiple VID requirements as follows. For each requirement, we first determined the number of VIDs using the uniform distribution $U[3, 7]$ (i.e., randomly drawn a number between 3 and 7) and the length of VIDs using $U[2, 9]$. For simplicity, all VIDs in the same requirement have the same length and same threshold $k = 100$. For each VID, we randomly selected some attributes according to the VID length from the 14 attributes. A repeating VID was discarded. For example, a requirement of 3 VIDs and length 2 is

$\{< \{A, En\}, k >, < \{A, R\}, k >, < \{S, H\}, k >\}$,

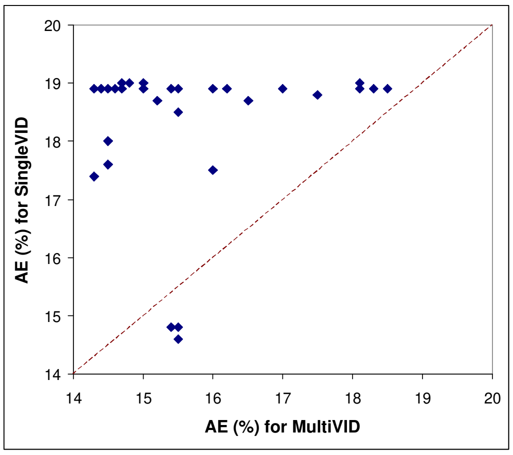


Figure 7. SingleVID **vs** MultiVID **(k=100)**

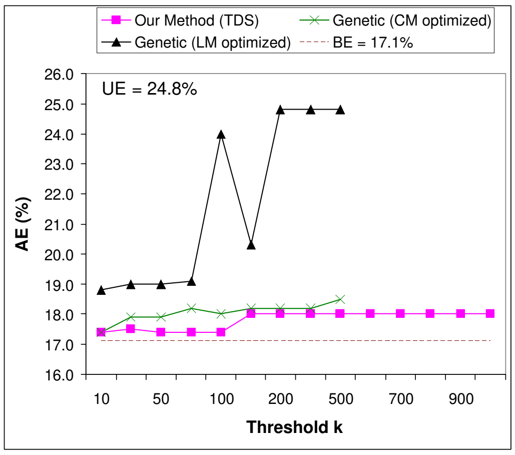


Figure 8. Comparing with genetic algorithm

and the corresponding single VID requirement is

$\{< \{A, En, R, S, H\}, k >\}$.

In Figure 7, each data point represents the AE of a multiple VID requirement, denoted MultiVID, and the AE of the corresponding single VID requirement, denoted SingleVID. The C4.5 classifier was used. Most data points appear at the upper left corner of the diagonal, suggesting that MultiVID generally yields lower AE than its corresponding SingleVID. This verifies the effectiveness of multiple VIDs to avoid unnecessary generalization and improve data quality.

6.2. Comparing with Genetic Algorithm

Iyengar [6] presented a genetic algorithm solution. See Section 2 for a brief description or [6] for the details. Experiments in this section were customized to conduct a fair comparison with his results. We used the same *Adult* data set, same attributes, and the same anonymity requirement as specified in [6]:

GA = $< \{A, W, E, M, O, Ra, S, N\}, k >$.

We obtained the taxonomy trees from the author, except for the continuous attribute A. Following Iyengar's procedure, all attributes not in GA were removed and were not used to produce BE, AE, and UE in this experiment, and all errors were based on the 10-fold cross validation and the C4.5 classifier. For each fold, we first generalized the training data and then applied the generalization to the testing data.

Figure 8 compares AE of TDS with the errors reported for two methods in [6], Loss Metric (LM) and Classification Metric (CM), for $10 \leq k \leq 500$. TDS outperformed LM, especially for $k \geq 100$, but performed only slightly better than CM. TDS continued to perform well from $k = 500$ to $k = 1000$, for which no result was reported for LM and CM in [6]. This analysis shows that our method is at least comparable to Iyengar's genetic algorithm in terms of accuracy. However, our method took only 7 seconds to generalize the data, including reading data records from disk and writing the generalized data to disk. Iyengar [6] reported that his method requires 18 hours to transform this data, which has about only 30K data records. Clearly, the genetic algorithm is not scalable.

6.3. Efficiency and Scalability

This experiment evaluates the scalability of TDS by blowing up the size of the *Adult* data set. First, we combined the training and testing sets, giving 45,222 records. For each original record r in the combined set, we created $\alpha - 1$ "variations" of r, where $\alpha > 1$ is the blowup scale. For each variation of r, we randomly selected q attributes from $\cup VID_j$, where q has the uniform distribution $U[1, |\cup VID_j|]$, i.e., randomly drawn between 1 and the number of attributes in VIDs, and replaced the values on the selected attributes with values randomly drawn from the domain of the attributes. Together with all original records, the enlarged data set has $\alpha \times 45,222$ records. In order to provide a more precise evaluation, the runtime reported in this section excludes the time for loading data records from disk and the time for writing the generalized data to disk.

Figure 9 depicts the runtime of TDS for 200K to 1M data records and the anonymity threshold $k = 50$ based on two types of anonymity requirements. AllAttVID refers to the single VID having all 14 attributes. This is one of the most time consuming settings because of the largest number of candidate specializations to consider at each iteration. For TDS, the small anonymity threshold of $k = 50$ requires more iterations to reach a solution, hence more runtime,

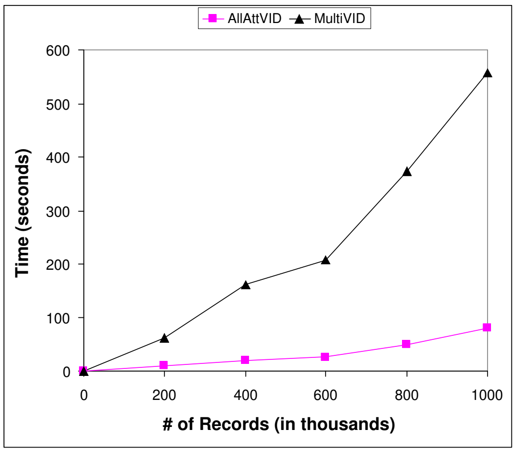


Figure 9. Scalability vs # of records (k=50)

than a larger threshold. TDS takes approximately 80 seconds to transform 1M records.

In Figure 9, MultiVID refers to the average runtime over the 30 random multiple VID requirements in Section 6.1 with $k = 50$. Compared to AllAttVID, TDS becomes less efficient for handling multiple VIDs for two reasons. First, an anonymity requirement on multiple VIDs is a less restrictive constraint than the single VID anonymity requirement containing all attributes; therefore, TDS has to perform more specializations before violating the anonymity requirement. Moreover, TDS needs to create one VIT for each VID and maintains $a(vid)$ in VITS. The increase is roughly by a factor proportional to the number of VIDs in an anonymity requirement.

6.4. Summary

Our experiments verified several claims about the proposed TDS method. First, TDS generalizes a given table to satisfy a broad range of anonymity requirements without sacrificing significantly the usefulness to classification. Second, while producing a comparable accuracy, TDS is much more efficient than previously reported approaches, particularly, the genetic algorithm in [6]. Third, TDS scales well with large data sets and complex anonymity requirements. These performances together with the features discussed in Section 1 make TDS a practical technique for privacy protection while sharing information.

7. Conclusions

We considered the problem of protecting individual privacy while releasing person-specific data for classification modelling. Our approach is based on two observations: sensitive information tends to be overly specific, thus of less utility, to classification; even if masking sensitive information eliminates some useful structures, alternative structures in the data emerge to help. We presented a top-down approach to iteratively specialize the data from a general state into a special state, guided by maximizing the information utility and minimizing the privacy specificity. This top-down approach serves a natural and efficient structure for handling categorical and continuous attributes and multiple anonymity requirements. Experiments showed that our approach effectively preserves both information utility and individual privacy and scales well for large data sets.

References

[1] R. Agrawal and S. Ramakrishnan. Privacy preserving data mining. In *Proc. of the ACM SIGMOD Conference on Management of Data*, pages 439–450, Dallas, Texas, May 2000. ACM Press.

[2] T. Dalenius. Finding a needle in a haystack - or identifying anonymous census record. *Journal of Official Statistics*, 2(3):329–336, 1986.

[3] W. A. Fuller. Masking procedures for microdata disclosure limitation. *Official Statistics*, 9(2):383–406, 1993.

[4] S. Hettich and S. D. Bay. The UCI KDD Archive, 1999. http://kdd.ics.uci.edu.

[5] A. Hundepool and L. Willenborg. μ- and τ-argus: Software for statistical disclosure control. In *Third International Seminar on Statistical Confidentiality*, Bled, 1996.

[6] V. S. Iyengar. Transforming data to satisfy privacy constraints. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 279–288, Edmonton, AB, Canada, July 2002.

[7] J. Kim and W. Winkler. Masking microdata files. In *ASA Proceedings of the Section on Survey Research Methods*, pages 114–119, 1995.

[8] R. J. Quinlan. *C4.5: Progams for Machine Learning*. Morgan Kaufmann, 1993.

[9] P. Samarati. Protecting respondents' identities in microdata release. In *IEEE Transactions on Knowledge Engineering*, volume 13, pages 1010–1027, 2001.

[10] L. Sweeney. Datafly: A system for providing anonymity in medical data. In *International Conference on Database Security*, pages 356–381, 1998.

[11] The House of Commons in Canada. The personal information protection and electronic documents act, April 2000. http://www.privcom.gc.ca/.

[12] K. Wang, P. Yu, and S. Chakraborty. Bottom-up generalization: a data mining solution to privacy protection. In *The Fourth IEEE International Conference on Data Mining 2004 (ICDM 2004)*, November 2004.