Data Min Knowl Disc (2008) 16:251–275
DOI 10.1007/s10618-008-0088-z

A framework for condensation-based anonymization of string data

Charu C. Aggarwal · Philip S. Yu

Received: 8 February 2007 / Accepted: 10 January 2008 / Published online: 6 February 2008

Abstract In recent years, privacy preserving data mining has become an important problem because of the large amount of personal data which is tracked by many business applications. An important method for privacy preserving data mining is the method of condensation. This method is often used in the case of multi-dimensional data in which pseudo-data is generated to mask the true values of the records. However, these methods are not easily applicable to the case of string data, since they require the use of multi-dimensional statistics in order to generate the pseudo-data. String data are especially important in the privacy preserving data-mining domain because most DNA and biological data are coded as strings. In this article, we will discuss a new method for privacy preserving mining of string data with the use of simple template-based models. The template-based model turns out to be effective in practice, and preserves important statistical characteristics of the strings such as intra-record distances. We will explore the behavior in the context of a classification application, and show that the accuracy of the application is not affected significantly by the anonymization process.

Keywords Privacy · Strings · Condensation

This article is an extended version of the conference version of the paper presented at the SIAM Conference on Data Mining, 2007 Aggarwal and Yu (2007). Available at http://www.charuaggarwal.net/str.pdf.

Responsible editor: Eamonn Keogh.

C. C. Aggarwal (✉)
IBM T. J. Watson Research Center, Hawthorne, NY, USA
e-mail: charu@us.ibm.com

P. S. Yu
University of Illinois at Chicago, Chicago, IL, USA
e-mail: psyu@cs.uic.edu

1 Introduction

Privacy-preserving data mining has become an important problem in recent years because of the large amount of personal data available with corporations and individuals. A number of interesting approaches for privacy-preserving data mining have been proposed in recent years. Two important methods for privacy-preserving data mining are those of randomization (Agrawal and Srikant 2000), and k-anonymity (Samarati and Sweeney 1998). A recent alternative of k-anonymity known as condensation is designed to construct pseudo-data which preserves statistical characteristics of the data, but not the actual records. In many cases, this can prove to be quite effective, since data mining algorithms can be directly adapted to this kind of pseudo-data. Other interesting methods for privacy-preserving data mining may be found in Aggarwal and Yu (2004), Agrawal and Srikant (2000), Agrawal and Aggarwal (2002), Aggarwal (2004), Bayardo and Agrawal (2005), Iyengar (2000), Meyerson and Williams (2004) and Samarati and Sweeney (1998).

The k-anonymity approach (Bayardo and Agrawal 2005; Kifer and Gehrke 2006; LeFevre et al. 2006; Samarati and Sweeney 1998) has been extensively explored in recent years because of its intuitive significance defining the level of privacy. The motivation behind the k-anonymity approach is that public databases can often be used by individuals to identify personal information about users. For example, a person's age and zip code can be used to identify them to a very high degree of accuracy. Therefore, the k-anonymity method attempts to reduce the granularity of representation of the data in order to minimize the risk of disclosure. To achieve this goal, the methods of generalization and suppression are used. In the method of generalization, the multi-dimensional values are generalized to a range. In addition some attributes or records may need to be suppressed in order to maintain k-anonymity. At the end of the process, the data are transformed in such a way that a given record cannot be distinguished from at least $(k-1)$ other records in the data. In such cases, the data are said to be k-anonymous, since it is not possible to map a given record to less than k-individuals in the public database. The concept of k-anonymity is intuitively appealing because of its natural interpretability in terms of the degree of privacy. In the case of multi-dimensional data, pseudo-identifiers are often distinguished from sensitive attributes. Many related approaches such as confidence bounding (Wang et al. 2006) and l-diversity (Machanavajjhala et al. 2006) are specifically designed in order to perform the anonymization in such a way so as to protect the values of the sensitive attributes. The l-diversity method of Machanavajjhala et al. (2006) constructs the groups in such a way that the diversity of the sensitive values of the different groups is preserved. In the case of string data, we note that crisp attributes cannot be defined especially if the strings are of variable length.

The *condensation-based* technique (Aggarwal and Yu 2004) has been proposed as an alternative to k-anonymity methods. The key difference between condensation and k-anonymity methods is that the former works with pseudo-data rather than the original records. Because of the use of pseudo-data, the identities of the records are even more secure from inference attacks. The idea is to utilize statistical summarization which is then leveraged in order to create pseudo-data. The condensation approach uses the following steps:

- We construct condensed groups of records. The number of records in each group is (at least) equal to the anonymity level k.
- The statistical information in the condensed groups can be utilized to synthetically generate pseudo-data which reflects the overall behavior of the original data.
- The condensed pseudo-groups can be utilized directly with minor modifications of existing data mining algorithms. Typically, such pseudo-data is useful in *aggregation-based* data mining algorithms which utilize the aggregate trends and patterns in the data rather than individual records.

The condensation approach is very similar to the k-anonymity model since it guarantees that at least k records in the data cannot be distinguished from one another. At the same time, since a one-to-one matching does not exist between the original and condensed data, it is more resistant to inference attacks. We note that the new data set need not even contain the same number of records as the original data set, as long as the records in different condensed groups are proportionately represented in the pseudo-data.

We note that most k-anonymity methods have been developed for the case of multi-dimensional data. Many of the typical methods such as attribute-specific generalization work well on multi-dimensional domains, but cannot be easily extended to other domains such as strings. The string domain is particularly important because of its applicability to a number of crucial problems for privacy preserving data mining in the biological domain. Recent research has shown that the information about diseases in medical data can be used in order to make inferences about the identity of DNA fragments (Malin and Sweeney 2001; Malin 2004). Many diseases of a genetic nature show up as specific patterns in the DNA of the individual. A possible solution is to anonymize the medical records (Sweeney 1996), but this can at best provide a partial solution. This is because the information about DNA segments can be obtained from a variety of sources other than medical records. For example, identifying information can be obtained from a number of defining characteristics which are public information about the individual. Similarly, if DNA string fragments from a relative of a target are available, it can be used to identify the target. In general, it can be assumed that partial or complete information about the individual fragments of the strings is available. It may also be possible to have strings which are structurally related to the base strings in a specific way. Therefore, it is important to anonymize the strings in such a way that it is no longer possible to use these individual fragments in order to make inferences about the identities of the original strings.

We note that most work on privacy-preservation in the health-care domain has concentrated on the multi-dimensional aspects of medical data (Malin and Sweeney 2001; Malin 2004; Sweeney 1996). These aspects are mostly concerned with characteristics such as identification or geographical location of patients and sources of records. However, in recent years, great progress has been made in tying the specific behavior of DNA strings to different diseases and disorders. For example, alleles at specific positions can be tied to distinguishing characteristics such as the skin or eye color, or other kinds of genetic diseases. Such characteristics increase the risk of identification, especially if the data are drawn from specific pools of individuals.

In this article, we will discuss the condensation model for anonymization of string data. We note that unlike the randomization approach for market basket data sets (Evfimievski et al. 2002), our condensation approach provides guarantees of k-anonymity, and is applicable to general strings, rather than simply sets of items. In order to perform the anonymization, we create summary statistics of groups of strings, and use these summary statistics to generate pseudo-strings. The summary statistics contains first- and second-order information about the distribution of the symbols in the strings. The distribution contains sufficient probabilistic parameters in order to generate pseudo-strings which are similar to the original strings. We will see that the *aggregate behavior* of the new set of strings maintains key characteristics such as composition, the order of the intra-string distances, and the accuracy of data mining algorithms such as classification. We note that the preservation of intra-string distances is a key goal in many string and biological applications which are deeply dependent upon the computation of such distances. In addition, we will show that the accuracy of applications such as classification are not affected by the anonymization process.

This article is organized as follows. In the next section, we will discuss the process of creating the segmentations from the sets of k-strings. In Sect. 3, we will show how these segmentations can be used to create pseudo-strings for mining purposes. In Sect. 4, we will present experimental results which illustrate the effectiveness of using the pseudo-strings in place of true strings. We will examine the compositional behavior of the pseudo-string data set with respect to the original set of strings. We will also show how intra string distances are affected by this segmentation model. Finally, we will test the behavior of a classification application with the use of pseudo-strings instead of the original strings. In Sect. 5, we discuss the conclusions and summary.

2 The condensation model

In this section, we will discuss the condensation model for string data. Let us assume that we have a database $\mathcal{D}$ containing N strings. We would like to create a new anonymized database which satisfies the conditions of k-indistinguishability. The N strings are denoted by $S_1 \ldots S_N$. The condensation needs to be performed in such a way that it is no longer possible to use information about portions or fragments of the strings in order to identify the entire string.

In order to perform the privacy-preserving transformation of the strings, we would like to have a database in which the lengths of the strings are not too different from one another. In cases in which the database does contain strings of widely varying lengths, it is desirable to create a situation in which the lengths of strings are tightly distributed within certain ranges. In order to formalize this definition, we need to define some tightness parameters. Specifically, we define the (ϵ, k)-similarity assumption for a database in terms of user defined parameter $\epsilon > 0$ and anonymity level k. We formalize this definition below:

Definition 2.1 A set of strings $\mathcal{D} = \{S_1 \ldots S_N\}$ is said to satisfy the (ϵ, k)-similarity assumption, if a set of ranges $[l_1, u_1] \ldots [l_r, u_r]$ can be found such that the following properties are satisfied:

- $u_i \leq (1+\epsilon) \cdot l_i$
- For each $i \in \{1 \ldots r\}$ the range $[l_i, u_i]$ contains at least k strings from the database $\mathcal{D}$.
- All strings from $\mathcal{D}$ belong to at least one of the ranges $[l_i, u_i]$ for some $i \in \{1 \ldots r\}$.

We note that a given database may not necessarily satisfy the (ϵ, k)-similarity assumption. In the event that the database does not satisfy this assumption, some of the strings may need to be suppressed in order to preserve k-anonymity. We note that a large enough value of ϵ can always be found for which the strings in the database can be made to satisfy the (ϵ, k)-similarity assumption. However, larger choices of ϵ are not desirable since this allows the lengths of strings in the database to vary. We will see that this complicates the process of generating pseudo-strings from the unevenly distributed strings.

While we will propose methods to deal with non-homogeneous string lengths, this may not be a severe issue in many applications. This is because the strings may correspond to the same entity in a given application. In such cases, the lengths of the different strings may be exactly the same. Therefore, the entire database may trivially satisfies the $(\epsilon = 0, k = N)$-assumption in these cases. In such cases, the process of condensation and privacy preservation of the strings is further simplified, since one does not have to worry about creating the ranges with different sets of strings.

In the case that the database does not satisfy the (ϵ, k)-similarity assumption, we perform a preprocessing step in order to segment the database into different groups of strings. The length of these groups is homogeneous to a level chosen by the user-defined parameter ϵ. In addition, we remove those strings whose lengths are significantly different from the rest of the data. This is because some strings cannot easily be fit in any segment without violating the k-anonymity assumption. Once the database is segmented, we can apply the condensation procedure separately to each of these segments.

The preprocessing step works using a simple iterative approach in which we start from the string having the smallest length l_s, and try to find all strings which lie in the range $[l_s, (1+\epsilon) \cdot l_s]$. If at least k strings can be found within this range, we create a new homogenized segment containing all strings whose lengths lie in the range $[l_s, (1+\epsilon) \cdot l_s]$. We remove this set of strings from the database and proceed further. On the other hand, when the range contains fewer than k strings, then we exclude (i.e., suppress) the smallest string from the database and proceed further with the next smallest string. Thus, in each iteration, either a string is discarded from the database or a set of at least k strings is grouped together in one range and removed from the database. The procedure will terminate in at most N iterations, though the number of iterations is closer to N/k in practice. This is because a suppression operation should occur only in a small number of iterations, when a judicious choice of parameters is used. We also note that we do not need to repeatedly access the underlying string database for this purpose. We can perform an initial scan of the database in which the lengths of each of the strings are determined and stored in a vector of lengths. For modestly large databases containing millions of records, it is possible to maintain this vector of lengths in main memory. The iterative algorithm is applied to this vector of lengths to determine the segmentation. Once the segmentation of lengths

Fig. 1 Length homogenization algorithm

Algorithm *HomogenizeLength*(Database: $\mathcal{D}$, AnonymityValue: k, Similarity: ϵ)
begin
  Determine the lengths of the strings in the database $\mathcal{D}$ and denote by $P = \{L_1 \ldots L_N\}$;
  $i = 1$;
  while P is non-empty **do**
  begin
    Determine smallest length L_{min} in P;
    Q = Set of strings in $[L_{min}, (1+\epsilon) \cdot L_{min}]$;
    if $|Q| < k$ remove L_{min} from P as outlier **else**
    begin
      $\mathcal{P}_i = Q$; $i = i + 1$;
      Remove Q from P;
    end
  end
  $r = i$;
  Recreate database segments $\mathcal{D}_1 \ldots \mathcal{D}_r$ from corresponding length ranges;
end

has been determined, we can process the original database to create the segmentation on the actual strings and discard the outliers. This is done using the intervals determined by the algorithm. The overall algorithm for preprocessing is illustrated in Fig. 1. Since the algorithm partitions the string database based on length, we refer to it as *HomogenizeLength*.

At the end of the process, we have a new set of database segments denoted by $\mathcal{D}_1 \ldots \mathcal{D}_r$ in each of which the lengths are approximately equal. By approximately equal, we are referring to the fact that the lengths lie within a factor of $(1 + \epsilon)$ of one another. In further discussions, we will abstract out this preprocessing portion of the algorithm. In other words, we will assume without loss of generality that the database $\mathcal{D}$ contains only strings in which the length ratios lie within a factor of $(1 + \epsilon)$. This can be done without loss of generality because we can assume that the subsequent steps are applied to each homogenized segment in the data. A homogenized segment of the database is converted into a set of *templates*. We will discuss the process of template construction in the next section.

3 Template construction for condensation

In this section, we will discuss the process of template construction for string condensation and anonymization. Let us assume that the strings are drawn from the alphabet $\Sigma = \{\sigma_1 \ldots \sigma_l\}$ of size l. The process of string condensation requires the generation of pseudo-strings from groups of similar strings. In order to achieve this goal, we

first need to create groups of k similar strings from which the condensed templates are formed. The statistics from each group of k similar strings is used to generate pseudo-strings. As discussed earlier, we will assume that the process of statistical condensation is applied to each homogeneous segment.

As discussed earlier, the preprocessing phase ensures that the lengths of all the different strings lie within a factor of at most ϵ. Let us assume that the N strings in the database are denoted by $S_1 \ldots S_N$, with corresponding lengths $L_1 \ldots L_N$. Then, the length of the template representation of this set of strings is equal to $L = \lceil \sum_{j=1}^{N} L_j/N \rceil$.

Our first step is to convert each string into a probabilistic template representation of length L. This is done in order to facilitate further probabilistic analysis of different positions on the strings. Unlike the original string, each position in the template may correspond to one or more symbols. Let us assume that the template representation of string S_j is denoted by T_j. In order to define the symbols corresponding to the ith position of string T_j, we determine a corresponding start and end position within string S_j. The start position within the string S_j for the ith position of string T_j is defined by $(i-1) \cdot L_j/L$. We note that this value may correspond to a floating point number. Let us assume that the floating point value of the beginning position is defined by p. Similarly, the ending position within the string S_j for T_j is defined by $i \cdot L_j/L$. Let us assume that this value is equal to q. As in the previous case, the value of q may also be a floating point value. In many cases, when ϵ is small, the condition $q \approx p + 1$ holds. Then, we compute the frequency of the presence of the different symbols from positions p to q (start and end points inclusive) in string S_j. Let us denote the frequency of the symbol σ_i by $n(\sigma_i)$. Since p and q are floating point numbers, we need to include the contribution of the floating point portions between p and q. For example, we have illustrated two cases in Fig. 2, in which we used $p = 1.3$ and $q = 5.4$. In such a case, we need to include a fraction of the second position and the sixth positions, respectively. Since the second position covers the range [1, 2], the fraction attributed to the second position is $2 - 1.3 = 0.7$. Since the sixth position covers the range [5, 6], the fraction attributed to the sixth position is $5.4 - 5 = 0.4$. For the case of Fig. 2(a) on the left, the second and sixth positions are both populated with the alphabet D. Correspondingly, the frequency of the alphabet D is given by $0.7 + 1 + 0.4 = 2.1$. This corresponds to the beginning, end and middle positions for D. The frequencies for alphabets R and E between these positions are each given by 1. Next, we convert these frequencies into fractional form by normalizing (dividing) each frequency with the total frequency computed for each symbol. Therefore, the fractional frequency of a given symbol σ_i is given by $n(\sigma_i)/\sum_{r=1}^{l} n(\sigma_r)$. Using this computation, the normalized frequencies of the symbols D, R, and E are 2.1/4.1, 1/4.1 and 1/4.1 respectively. For the case of Fig. 2(b) on the right, the normalized frequencies can be computed similarly using the fact that the start and end positions are populated by E and L, respectively. The corresponding normalized frequency calculations are illustrated in Fig. 2(b). In most cases, the value of p and q will differ by about one, and therefore only one or two symbols may have non-zero frequency. This is true when the value of ϵ chosen is relatively small. In some cases, p and q may correspond to the same position with different floating point values. In such cases, the computation will yield a normalized frequency of 1 for the symbol at that position.

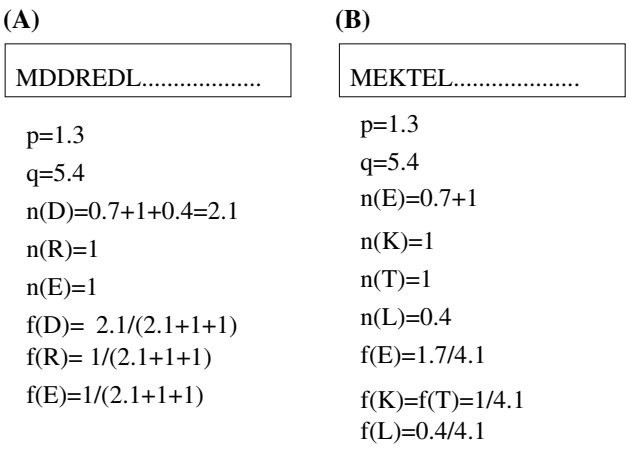

Fig. 2 Symbol frequency computation for template construction

Let us assume that the (normalized) frequencies of the l alphabets $\sigma_1 \ldots \sigma_l$ for position i in string T_j are denoted by $f_{i1}(T_j) \ldots f_{il}(T_j)$. Because of the normalization process, we have $\sum_{m=1}^{l} f_{im}(T_j) = 1$ for each position i, and string j. We note that most of the values of $f_{im}(T_j)$ are zero. As discussed later, we can use this fact to improve the efficiency of the summary statistic representation. We also note that the only goal achieved by the process is to convert the string to a length of L. In order to represent this summary process of conversion of the strings into a new representation with length L, we denote the procedure by $ConvertLength(S_i, L)$. The use of this notation is helpful in further discussion. We refer to the converted strings as *extended template strings* since they represent the probabilistic templates over an extended string length.

The normalized frequencies are computed for each of the N strings $\{S_1 \ldots S_N\}$, which are then converted into the N extended template strings denoted by $\mathcal{G} = \{T_1 \ldots T_N\}$. We note that unlike the set of strings $S_1 \ldots S_N$, the set of strings in $\mathcal{D}$ have the same length which is defined by L. The homogeneous length of the strings helps us define a set of first- and second-order statistics for the a group $\mathcal{G}$ of strings drawn from database $\mathcal{D}$.

We define the summary statistics for the group $\mathcal{G} = \{T_1 \ldots T_k\}$ as follows:

- For each group $\mathcal{G}$, we define the second order statistics $\mathcal{S}c(\mathcal{G})$ which are defined for each position $r \in \{1 \ldots L-1\}$ and pair of symbols $p, q \in \{\sigma_1 \ldots \sigma_l\}$ by Sc_{rpq}. This value is defined as follows:

$$Sc_{rpq}(\mathcal{G}) = \sum_{j=1}^{k} f_{rp}(T_j) \cdot f_{(r+1)q}(T_j) \tag{1}$$

  Note that we are computing the correlation of symbol presence between the rth and $(r+1)$th position. We also note that there are at most $(L-1) \cdot l^2$ such values, but most of these values are zero since most of the symbols have zero frequency

at a given position. Therefore, we can choose a sparse representation in which we maintain only the non-zero values. In practice, since each position will typically contain only about two symbols with non-zero frequency in T_i, we only need to maintain a total of about $4 \cdot L$ such values. The corresponding savings can be significant when the value of l is relatively large. For example, in domains such as protein analysis, the value of l corresponds to the number of amino-acids which is 20. We note that the second-order statistics measure the correlation between different symbols at adjacent values. This is useful for effective re-generation of successive positions of pseudo-strings from group statistics.

- For each group $\mathcal{G}$, we define the first order statistics $Fs(\mathcal{G})$ which is defined for each position $r \in \{1 \ldots L\}$ and symbol $\sigma_p \in \{\sigma_1 \ldots \sigma_l\}$ by $Fs_{rp}(\mathcal{G})$. This value is defined as follows:

$$Fs_{rp}(\mathcal{G}) = \sum_{j=1}^{k} f_{rp}(T_j) \tag{2}$$

  There are at most $L \cdot l$ such non-zero values. As in the previous case, we can maintain a sparse representation for efficiency.
- For each group $\mathcal{G}$, we maintain the corresponding string length L.
- For each group, we maintain the number of strings $n(\mathcal{G}) = k$.

We note that the summary statistics turn out to be useful in generating the pseudo-data for the different groups. It remains to explain how the different groups are constructed. In order to construct the groups, we need to partition the strings into groups of k records. In this section, we will discuss the partitioning approach for constructing the groups from the homogenized database $\mathcal{D}$.

4 The partitioning method for condensation

In this section, we will discuss the partitioning approach for condensation and subsequent generation of the pseudo-strings. We also assume that the set of strings have already been homogenized. The partitioning process generates probabilistic templates from each segment. Therefore, the algorithm described in this section really applies to *each segment* of the homogenized database. The uniformity in the length of different templates facilitates the application of different kinds of operations on the probabilistic templates. Thus, there are two levels of partitioning:

- In the first level of partitioning, we perform the previously discussed homogenization process to create segments containing strings with lengths within a factor of $(1 + \epsilon)$ of one another. The algorithm of this section is applied (independently) to each segment of homogenized strings. Sine the groups are already homogenized, it is easy to construct probabilistic templates from them.
- In the next level of partitioning, the first step is the construction of probabilistic templates from each homogenized segment. Then, we create groups of k probabilistic templates from each database segment. The statistics of each segment are

then computed. These statistics are used to generate the pseudo-strings. We will discuss this partitioning step in this section.

In order to partition the data into sets of k templates, we use an iterative algorithm in which the groups of templates are constructed around different sets of seeds. In each iteration, we make a pass through the database in random order. In the first iteration, the *ConvertLength* procedure is applied to each string in the database in order to express it as a template of length L.

Next, we begin an iterative process of constructing groups by building them around randomly chosen template. We start off by picking a random template and assign the $(k-1)$-closest templates to it. We will discuss the distance calculation process in more detail slightly later. We repeat the process of picking random templates and assigning the $(k-1)$ closest templates in the data to the currently selected template until all templates have been exhausted. As soon as each set of $(k-1)$ templates have been assigned, they are marked as assigned. We note that at the end of the process, fewer than k unmarked templates may remain because of successive assignments. Each template in this set is assigned to the closest existing centroid among all groups formed so far. At the end of the process, most centroids will have k templates assigned to them, but a few may have more than k templates assigned.

After the first assignment pass, we repeat the process with a modification. Specifically, we use the segmentation of the templates from the previous iteration in order to improve the quality of the groups. This is done by using the centroid of each template set in order to decide the assignment of templates. The centroid of each group is defined as the average of all the templates in the group. This is an easy computation, all the templates in a given database segment have the same length. Therefore, the averaging can be done in a position-specific way. As in the previous case, we use an iterative process to segment the data. The difference is that we use centroids rather than randomly picked strings from the database in order to create groups. For each of the centroids, we find the closest k templates and create a new group from these templates. After forming this group, we mark this set of k templates from the database and continue the process until all templates have been exhausted. Finally, we re-compute the summary statistics of each of the groups formed, and use them to generate the pseudo-data. We repeat this iterative process over the database multiple times in order to improve the quality of the assignment. The quality of the assignment is defined in terms of an objective function which measures the average distance of the different templates from their corresponding centroids. This distance will be defined slightly later by Eq. 3. We use a termination criterion which computes the objective function of each assignment pass. The algorithm terminates when the difference in objective function between two successive assignments improves by $<1\%$. The overall approach is illustrated in Fig. 3.

4.1 Distance function computation

It remains to explain how the distances are computed between pairs of template strings. This is required for assignment computations. Let us consider two template strings

Fig. 3 Creation of groups for condensation

Algorithm *CreateGroups*((Homogen.) Dbs. (Seg.): $\mathcal{D}$, Anonymity Level: k);
begin
$L =$ Average string length of homogenized database $\mathcal{D}$;
{ The next step constructs templates from each string in $\mathcal{D}$ };
for each string $S \in \mathcal{D}$
$S = ConvertLength(S, L)$;
$\mathcal{D}_c = \mathcal{D}$; $C = \{\}$
while $\mathcal{D}_c$ is non-empty
begin
Pick a template T from $\mathcal{D}_c$ and find the $(k-1)$ closest templates to it, thus creating $\mathcal{G}$;
Remove templates in $\mathcal{G}$ from $\mathcal{D}_c$;
Add centroid of $\mathcal{G}$ to C;
end
$C' = \{\}$; $\mathcal{D}_c = \mathcal{D}$;
while not(termination_criterion)
begin
while $\mathcal{D}_c$ is not empty
begin
Pick the next template T from C;
Assign the k closest templates from $\mathcal{D}_c$ to T to create group $\mathcal{G}$ with k templates (not including chosen template);
Add centroid of $\mathcal{G}$ to C';
end
Reset $\mathcal{D}_c = \mathcal{D}$; $C = C'$; $C' = \{\}$;
end
end

T_p and T_q. Then, the distance $dist(T_p, T_q)$ between two strings T_p and T_q of equal lengths $|T_p| = |T_q|$ is defined as follows:

$$dist(T_p, T_q) = \sum_{i=1}^{|T_p|} \sum_{r=1}^{l} |f_{ir}(T_p) - f_{ir}(T_q)| \tag{3}$$

This distance function is also used in order to compute the objective function for clustering quality in each iteration.

4.2 Optimizations

In some cases, it may be difficult to create a coherent group of templates. This is especially true towards the end of each iterative pass over the database, when only a small number of templates remain. In such cases, it may be desirable to assign these templates to different groups and create corresponding groups with size larger than k.

Therefore, at the end of the process, each group is examined to check if the strings in it can be re-assigned to other groups in order to reduce the objective function value. If this is the case, then the assignment is performed and the number of groups further reduce. At the end of the process, we pick the last set of groups that was created, and perform the re-assignment test on this last set. The process of generating the pseudo-data is discussed in the next section.

4.3 Generation of pseudo-strings

The condensed data from each group $\mathcal{G}$ can be be used to generate the pseudo-strings. Each pseudo string is generated with the same string length which is denoted by L. For each group $\mathcal{G}$, we generate $n(\mathcal{G})$ strings. We use the auto-correlation behavior of the second-order statistics in order to generate the strings. We generate the left-most position using the first-order statistics. Subsequent positions are generated using second-order correlations. We discuss the steps systematically below.

- The first position is generated using the statistic $Fs_{1p}(\mathcal{G})$ for each symbol σ_p. Specifically the pth symbol is generated for the first position with probability $Fs_{1p}(\mathcal{G})/n(\mathcal{G})$. We note that the sum of the different values of $Fs_{1p}(\mathcal{G})/n(\mathcal{G})$ is equal to $n(\mathcal{G})$, since the sum of the probabilities of different symbols at a given position in a template is one unit. Since the first-order statistics are defined in terms of the sum of $n(\mathcal{G})$ templates, it follows that the the sum of the different values of $Fs_{1p}(\mathcal{G})/n(\mathcal{G})$ is equal to 1.
- Once the ith position has been generated, we use the second-order correlations in order to generate the $(i+1)$th position. The conditional probability of $(i+1)$th position taking on a particular symbol value can be calculated using the first- and second-order statistics. Let us assume that the symbol at the ith position is σ_p. Then, the conditional probability of the $(i+1)$th position taking on the symbol σ_q is defined by the expression $Sc_{ipq}(\mathcal{G})/Fs_{ip}(\mathcal{G})$. We use this conditional probability in order to generate the symbol at the $(i+1)$th position from the symbol at the ith position. This is done by flipping a biased die for that position, using the conditional probabilities to decide the weights on different sides. This step is iteratively repeated over the entire length of the pseudo-string.

We note that the above process preserves the correlations between adjacent data points, but not the correlations between non-adjacent points. In the next section, we will show that the pseudo-data generated by the process retains similar aggregate characteristics to the original data.

4.4 Computational efficiency

The technique requires two main steps for the creation of pseudo-strings: (1) the homogenization process of the underlying strings, (2) the process of template construction, and (3) the creation of groups from the homogenized strings. We note that the process of homogenization simply requires a linear scan through the strings, and is therefore not the main time-consuming operation. Similarly, the process of template

construction is a linearly scalable operation, one the homogenized length is known. Therefore, the main bottleneck if the construction of groups from the homogenized partitions. In the worst-case, the entire database may be represented in one partition. Let us assume that the database contains N records, and a total of g groups are created. Let us also assume that the homogenized string length of the partition is L. We note that in order to create each group, we may need to perform $O(N)$ distance function computations. Furthermore, each distance function computation may required $O(L \cdot l)$ steps, where l is the number of symbols. This means that the number of steps for each group is given by $O(L \cdot l \cdot N)$. Since this is repeated for the g groups, the overall running time is given by $O(N \cdot g \cdot L \cdot l)$.

4.5 Alternative techniques

The aim of this paper is to provide a framework for condensation-based anonymization of strings. We have also provided a broad class of algorithms which are specifically designed for this frame-work. We note that a number of alternatives are possible in order to optimize the underlying algorithms both theoretically and experimentally. We mention some of these possibilities, since they can serve as the basis for future research on the topic. The two main procedures are the creation of homogenized string templates, and the construction of groups from the underlying data. The possible techniques which can be used in order to improve the effectiveness are as follows:

- *String homogenization:* We have used a linear time algorithm for string homogenization because of its efficiency. However, the strings of different lengths can be partitioned optimally with the use of a dynamic programming algorithm. Such an algorithm should minimize the number of strings which are excluded during the homogenization process. We note that only the lengths of the strings need to be used rather than the strings themselves.
- *Group generation:* We note that the process of group generation is a constrained clustering problem. While we have proposed one possible technique for creating the anonymized groups, it is possible to draw on a number of related techniques for balanced clustering in order to determine how to create the constrained clusters (Banerjee and Ghosh 2006). We note that the techniques in Banerjee and Ghosh (2006) have been designed for the case of multi-dimensional data. However, some of the broad techniques for managing the constraints on the cardinality of different clusters are also applicable to the case of other kinds of data. The challenge in future research is to extend these techniques to the case of string data.

4.6 Use of higher order statistics

This article is primarily focussed on the use of second-order statistics for pseudo-data generation. The technique can be easily extended to the case of higher-order statistics. Consider the group of strings denoted by $T_1 \ldots T_k$. For each group $\mathcal{G}$ we define statistics of order $s \geq 2$ by $\mathcal{O}_s$. For each position $r \in \{1 \ldots L - s + 1\}$ and symbol set $q_1 \ldots q_s$, we define the corresponding statistics $\mathcal{O}_s(q_1 \ldots q_s, r, \mathcal{G})$ as follows:

$$\mathcal{O}_s(q_1 \ldots q_s, r, \mathcal{G}) = \sum_{j=1}^{k} f_{rq_1} \cdot f_{(r+1)q_2} \cdots f_{(r+s-1)q_s} \quad (4)$$

Note that there are at most $(L - s + 1) \cdot l^s$ possible symbol sets for which the statistics may need to be maintained. As in the previous case, most of these values are zero, and therefore, the actual storage requirement may be much lower.

The process of generation of pseudo-data is also similar to the more simple case of second-order statistics. In this case, in order to generate the $(i + 1)$th position, we use the correlations of order s and correlations of order $(s-1)$. The conditional probability of the $(i + 1)$th position taking on the symbol q_s, given that the last $(s - 1)$ positions are $q_1 \ldots q_{s-1}$ is given by $\mathcal{O}_s(q_1 \ldots q_s, i - s + 1, \mathcal{G})/\mathcal{O}_s(q_1 \ldots q_{s-1}, i - s + 1, \mathcal{G})$. This conditional probability can be used to weight the sides of a die, which is then flipped in order to generate the symbol at the $(i+1)$th position. In order to generate the first s symbols, we need to use the lower-order statistics, in the same way to generate the corresponding conditional probabilities.

We note that the process of generation of higher-order pseudo-data is almost as efficient as that of the generation of second-order pseudo-data from a time perspective. The only difference is that the conditional probabilities are computed as a function of s values rather than two values. In terms of space, the storage requirement is quite a bit higher. In the worst case, we may need to maintain the storage statistics for $(L - s + 1)) \cdot l^s$ possible symbol sets. However, since most of these values are typically zero, the actual storage requirement is much lower.

5 Experimental results

In this section, we will discuss the experimental results of the condensation approach to privacy based mining of the strings. The data were obtained from the SWISS-PROT database which contained DNA sequences. Each of these sequences was expressed in terms of the 20 different amino acids. Correspondingly the alphabet size $|\Sigma|$ was 20. As illustrated, we used protein strings from different species in order to generate the strings. In Table 1, we have illustrated the keywords which were used to query the database along with the statistics of the strings. The SWISS-PROT database provides a query interface for keywords, and this query interface was used in conjunction with the corresponding keyword in order to derive the data. The query interface allows for download of a pre-defined number of records which are the best matches for a given query. In each case, the first 1,000 matches were downloaded from the query interface to create the data set. We note that smaller data sets are more difficult for privacy preservation purposes, since a given data locality may contain only a small number of similar strings. This makes it more difficult to generate pseudo-data from a similar group of strings in a robust way.

We note that the purpose of the condensation approach is to create pseudo-data which is similar in distribution to the original data. This ensures that aggregation-based data mining algorithms can be used on the data without affecting the overall results. Therefore, we tested the technique using the following methods:

Table 1 Statistics of SWISS-PROT data

Data set name	Keywords	Avg	Std	Min	Max
YST1	Yeast	513.8	345.7	7	3,122
YST2	Yeast	527.12	394.74	21	3,856
MS	Mouse	541.64	337.55	21	2,434
HS	Homo Sapiens	549.22	399.84	51	3,911

- We tested how the distribution of the different amino-acids varied from the original data set to the synthetic data set. For this purpose, we generated the histogram of the distribution of the different amino-acids for the different data sets. We computed the level of variation among the different histograms for the different data sets over different group sizes.
- Many string mining algorithms are critically dependent upon the computation of alignment distance functions among strings. For example, many data mining algorithms such as clustering or classification can be converted into distance based variations. We will compute inter-group alignment functions and show that the relative distances do not vary too much as long as the computations between strings of different groups. The latter assumption follows from the fact that the anonymization process requires indistinguishability of the strings within a group. At the same time, we retain the ability to distinguish the distances of the groups from one another for mining purposes.
- We tested the effect of the anonymization process on a classification application. Specifically, we tested how the accuracy of a classification application was affected with increasing group size. With increasing level of anonymization, it is expected that a training model constructed with the pseudo-strings should reduce in accuracy. While this turns out to be correct, we find that the level of degradation is relatively small. This is because of the fact that the newly generated strings reflect the behavior of the original data on an aggregate basis.

The value of ϵ was chosen so as to have sufficient number of strings in each segment. Therefore, we chose a value of $\epsilon = 1.5$. The algorithm was tested over different anonymity levels for robustness, since this parameter is controlled by the privacy-provider, and should not be tuned to the advantage of the underlying algorithm. In Figs. 4–7, we have illustrated the amino-acid composition of the original data and pseudo-data using the condensation based approach. In each case, the generated string length was set to the average string length over the entire database. The group size used in each case was 20. The histogram illustrates the compositional behavior for each if the 20 different amino-acids. The label on the X-axis indicates the index of the amino-acid alphabet, which can vary from 1 to 26 depending upon the index of the alphabet. Note that since some of the alphabets (such as B or Z) do not correspond to amino-acids, the frequency of the corresponding alphabet index is always zero in both the original data and generated pseudo-data. For the other amino-acids, the frequencies of each amino-acid in both the original and generated pseudo-data are approximately the same. This is true of all the four data sets illustrated in Figs. 4–7, respectively. In each case, we also computed the average error over the entire set of amino-acids. This is useful to

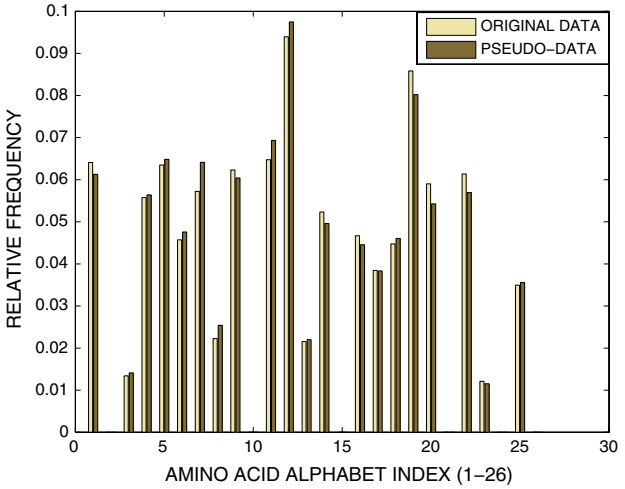

Fig. 4 Comparison of original and pseudo-data in amino-acid composition (YST1)

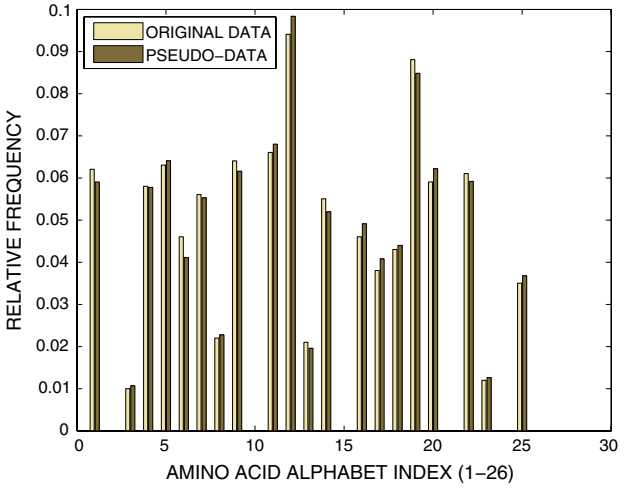

Fig. 5 Comparison of original and pseudo-data in amino-acid composition (YST2)

get an idea of the overall error in composition by using the condensation approach. Let us assume that the fractional frequency of the alphabet σ_i in the original database is denoted by f_i, and the fractional frequency of the alphabet σ_i in the pseudo-data is denoted by f_i'. Then, the overall compositional difference $CD(\overline{f}, \overline{f'})$ is defined as follows:

$$CD(\overline{f}, \overline{f'}) = \sum_{\sigma_i \in \Sigma} |f_i - f_i'| \tag{5}$$

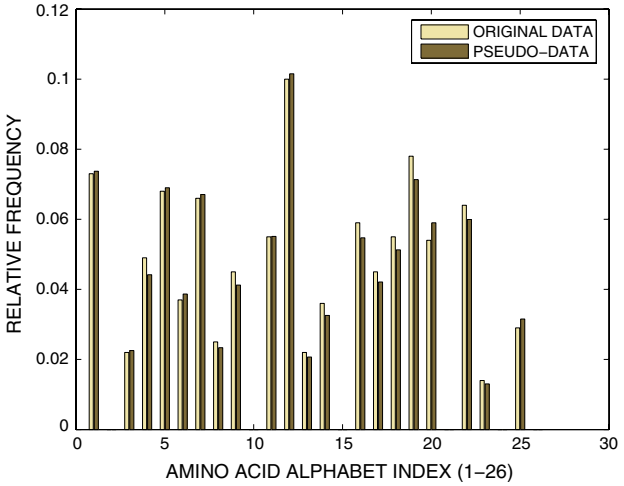

Fig. 6 Comparison of original and pseudo-data in amino-acid composition (HS)

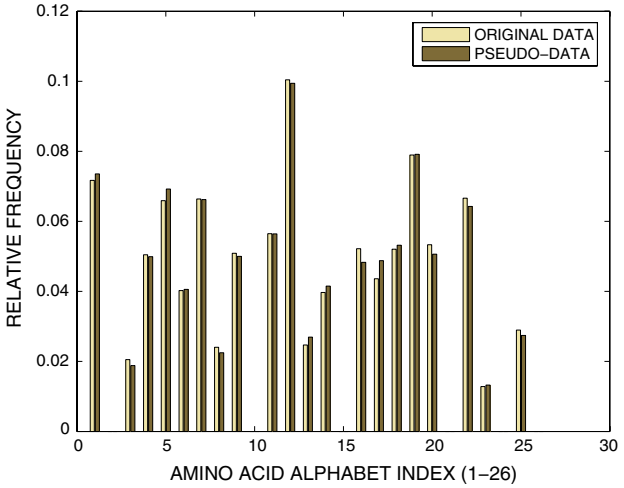

Fig. 7 Comparison of original and pseudo-data in amino-acid composition (MS)

Note that the value of $CD(\overline{f}, \overline{f'})$ is a fraction which always lies between 0 and 1. A value of zero corresponds to histograms which are exactly identical, whereas a value of 1 indicates histograms drawn from complementary alphabets. In each case, the value of $CD(\overline{f}, \overline{f'})$ ranged from 3 to 5% for a group size of 20. This indicates that there is only a small difference in the composition between the original data and the generated pseudo-data.

Since the value of $CD(\overline{f}, \overline{f'})$ is sensitive to the actual group size, it is interesting to plot this value for synthetic generations with different group sizes. In Fig. 8, we have illustrated the compositional difference between the original data and the pseudo-data for different group sizes. In each case, the compositional difference varied between 1

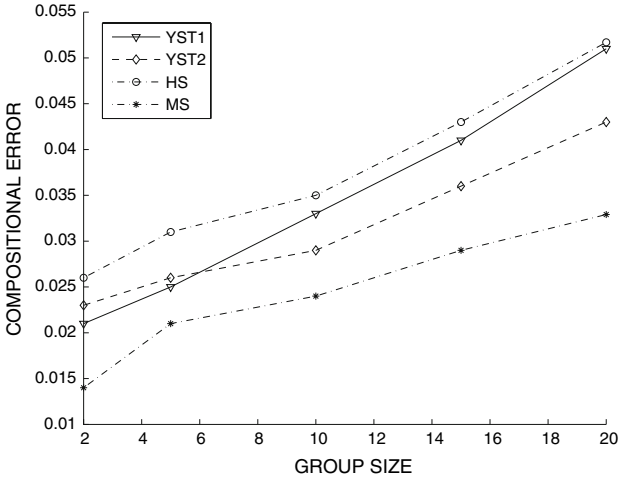

Fig. 8 Compositional match index with increasing group size

and 5%, and typically increased with increasing group size. This is a natural trade-off between accuracy and privacy. We further note that the behavior of the compositional accuracy for each of the data sets was quite similar in terms of the trend with increasing group size.

Many string mining algorithms such as clustering and classification are dependent upon the computation of intra-string distances. However, we note that the condensation-based approach does not distinguish between a group of strings. Therefore, it is instructive to examine how inter-group distances are affected by the condensation based transformation. Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be two groups of strings which from each of which group-specific pseudo-data is generated. Let the strings in the pseudogroups corresponding to $\mathcal{G}_1$ and $\mathcal{G}_2$ be denoted by $\mathcal{G}_1'$ and $\mathcal{G}_2'$. We note that since the strings within an anonymized group cannot be distinguished, we can only compare the inter-group distances between the original and the perturbed data. For this purpose, we use the Needleman–Wunsch distance function (Needleman and Wunsch 1970) between two groups of strings. Let the corresponding distances between strings S_i and S_j be denoted by $NW(S_i, S_j)$. Then, we define the group-specific distance $Gdist(\mathcal{G}_1, \mathcal{G}_2)$ between groups $\mathcal{G}_1$ and $\mathcal{G}_2$ as follows:

$$Gdist(\mathcal{G}_1, \mathcal{G}_2) = \sum_{i,j:S_i \in \mathcal{G}_1, S_j \in \mathcal{G}_2} NW(S_i, S_j) \tag{6}$$

In order to compare the behavior of the distances, we sample m pairs of groups. For each pair we compute the value of $Gdist(\mathcal{G}_1, \mathcal{G}_2)$. Let these values be denoted by $v_1 \ldots v_m$. We also compute the same values $Gdist(\mathcal{G}_1', \mathcal{G}_2')$ with respect to the pseudo-groups $\mathcal{G}_1'$ and $\mathcal{G}_2'$. These values are denoted by $v_1' \ldots v_m'$. For pair of values v_i and v_j, the ordering of distances is preserved if the relative order of v_i and v_j is the same as v_i' and v_j'. For each of the $m \cdot (m-1)/2$ pairs, we calculate the fraction of pairs for which the relative order is preserved. The closer this value is to 1, the greater the

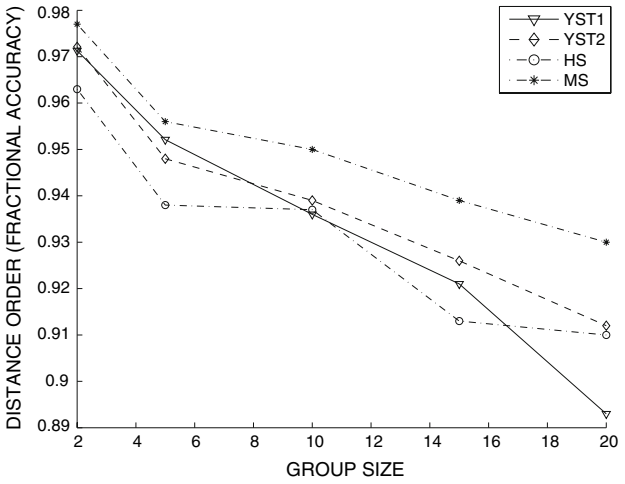

Fig. 9 Distance function computation accuracy with increasing group size

fraction of distance orderings which are preserved from the original data. In Fig. 9, we have illustrated the fraction of times that this relative ordering is preserved for different group sizes in each of the data sets. It is clear that this fraction is quite high in most cases, but reduces with increasing group size. However the overall accuracy of distance function ordering continues to be >90% in most cases. This means that applications which are dependent upon distance function ordering can be used effectively with the condensation-based method. In the next section, we will discuss the specific case of a classification application, and show that the effectiveness is retained with the condensation based approach.

5.1 Classification application

In this section, we will discuss a classification application in which we tested the condensation approach. The idea was to test how well the condensed pseudo-strings represented the information in the training model. The sequence data sets for classification were generated synthetically using the Apriori market basket data generator (Agrawal and Srikant 1994). We generated the sequences of items using exactly the same methodology as that discussed in Agrawal and Srikant (1994). In order to create a classification data set, we created two instances of the same data with different random seeds, and combined them into one larger data set. The two parts of the data set defined the two different classes. The class variable for each record was defined by the particular seed driven instance to which it belonged. We generated two data sets corresponding to instantiations from T20.I4.D50K and T20.I6.D50K, respectively. Thus, the first data set contained two sets of 50,000 records from different instantiations of T20.I4.D50K. This resulted in a total of 100,000 records. The second data set contained two sets of 50,000 records from different instantiations of T20.I6.D50K.

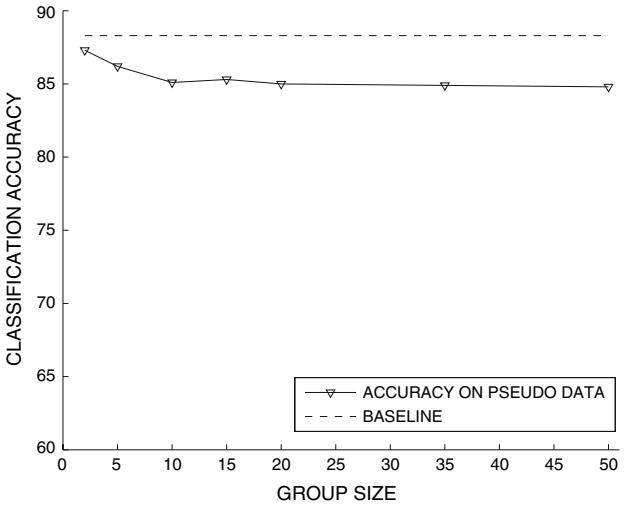

Fig. 10 Classification accuracy on data set V20.I6.D100K (nearest neighbor classifier)

We refer to these data sets as V20.I4.D100K and V20.I6.D100K, respectively. In each case, we used 99% of the data for training and the last set of 1,000 records for testing.

In order to re-construct the training data using the privacy approach, we constructed a separate set of pseudo-strings for each class in the data. The class of each pseudo-string was set to the class of the original data that was used to generate it. The training data set was then used in conjunction with a nearest neighbor classifier in order to perform the classification. Specifically, we constructed a k-nearest neighbor classifier on the training data. We constructed training data sets for different groups sizes and tested the classification accuracy over these different group sizes.

We have illustrated the classification accuracy of the data set V20.I6.D100K in Fig. 10. The group size is marked on the X-axis, and the classification accuracy is marked on the Y-axis. The classification accuracy generally reduces with group size because of the natural tradeoff between accuracy and privacy. It is also interesting to see that while the classification accuracy reduces slightly, it is generally quite competitive to the original method. In most cases, the overall classification accuracy on the pseudo-data is within about 4% of the classification accuracy of the original method. Another observation is that the classification accuracy initially reduces with increasing group size, and then flattens out rapidly. Most of the reduction in accuracy is for group sizes which are <10 or so. In Fig. 11, we have illustrated the classification accuracy of the method with increasing group size for the data set V20.I4.D100K. The baseline classification accuracy was slightly lower in this case because of the fact that the new data set had fewer inter-attribute correlations. However, the accuracy behavior with increasing group size is sightly better in this case. In this case, the classification accuracy is within 3% of the baseline accuracy. In order to illustrate robustness across different classifiers, we also tested the technique with the use of the *WavRule* classifier (Aggarwal 2002) over different anonymization levels. The results are illustrated

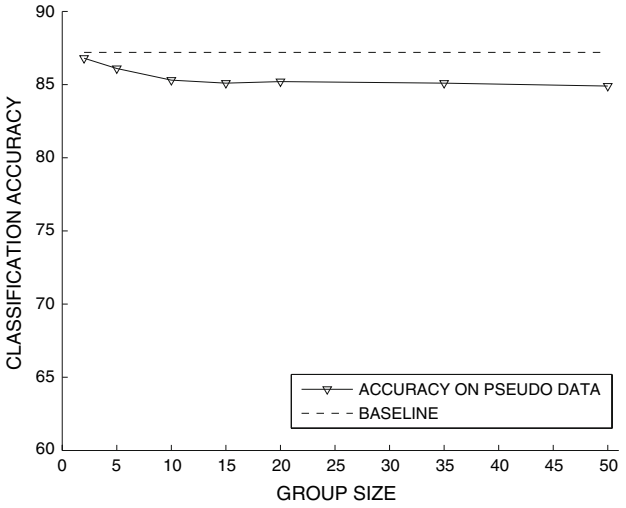

Fig. 11 Classification accuracy on data set V20.I4.D100K (nearest neighbor classifier)

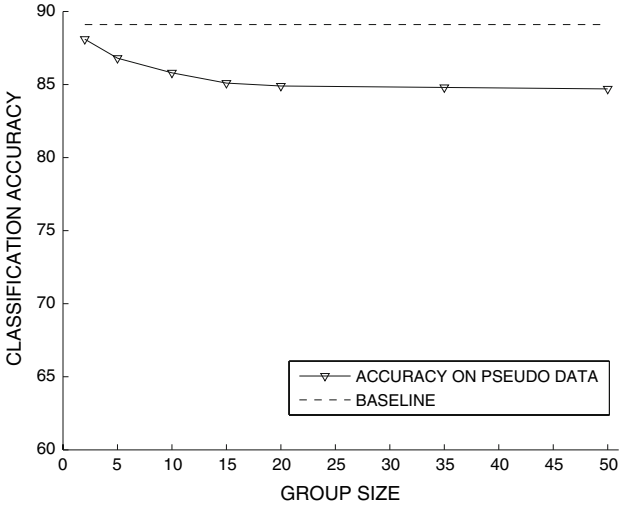

Fig. 12 Classification accuracy on data set V20.I6.D100K (WavRule Classifier)

in Figs. 12 and 13, respectively. It is evident that the results are quite similar to the case of the nearest neighbor classifier. It is clear that in each case, the accuracy of the technique reduced quite slowly with increasing anonymity level. Thus, the technique si robust across different kinds of classifiers.

Thus, the general observation from these results is that the classification accuracy was largely retained and reduced slowly with increasing group size. This shows that the process of anonymization retains a sufficient amount of statistical information

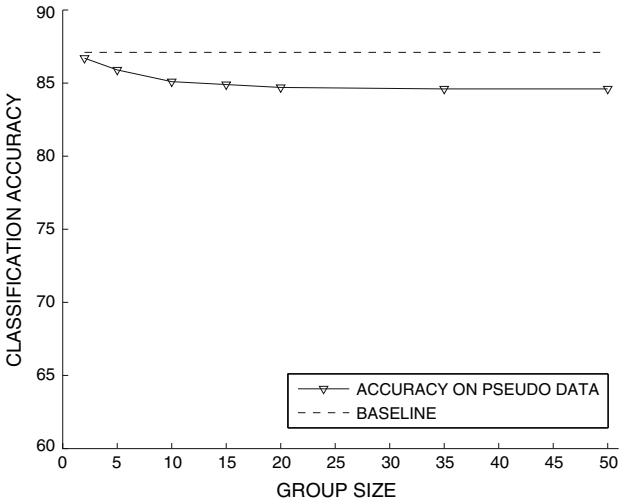

Fig. 13 Classification accuracy on data set V20.I4.D100K (WavRule Classifier)

Table 2 Effect of higher order transformations (Classification accuracy for anonymity level of 20)

Data set	NN (Ord. 2)	NN (Od. 3)	NN (Od. 4)	WavR. (Od. 2)	WavR. (Od. 3)	WavR. (Od. 4)
V20.I6.D100K	85.0	85.1	85.1	84.9	85.0	85.1
V20.I4.D100K	85.2	85.2	85.3	84.7	84.9	84.9

from the original data, so that it matches the latter in terms of composition, distance calculations and classification accuracy.

We also tested the effect of higher order transformations on the quality of the results. The results are illustrated in Table 2. In this table, we have illustrated the classification accuracy for each data set for transformations of order 2, 3, and 4. In each case, the anonymity level was set at 20. It is clear that in each case, the use of higher-order transformation improves the quality of the classification, but not by very much. This is because the additional information from higher-order correlations is often redundant over the use of information which can be constructed from different combinations of second-order correlations.

5.2 Scalability

We also tested the scalability of the approach with increasing data set size. For the purpose of scalability testing, the synthetic data sets were particularly useful, since the parameters of the data set could be easily varied. In Fig. 14, we have illustrated the scalability of the data sets T(x).I4.D10K, and T(x).I6.D10K, respectively. An anonymity level of 40 was used in each case. The average string size x is illustrated on the X-axis, and the running time (in seconds) is illustrated on the Y-axis. It is clear that in each case, the running time scaled linearly with increasing string size. Furthermore,

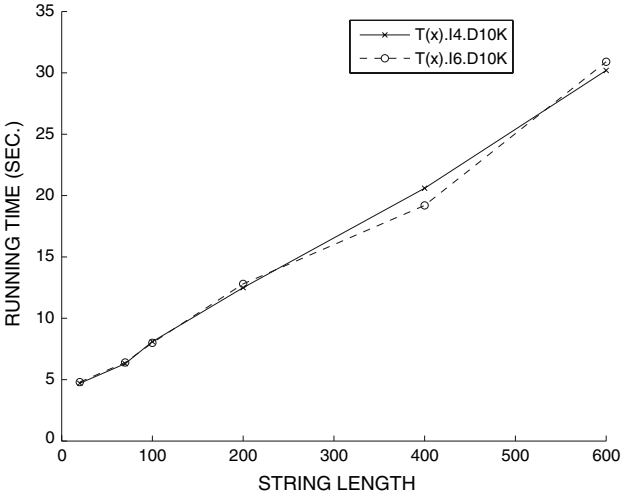

Fig. 14 Running time with increasing string length

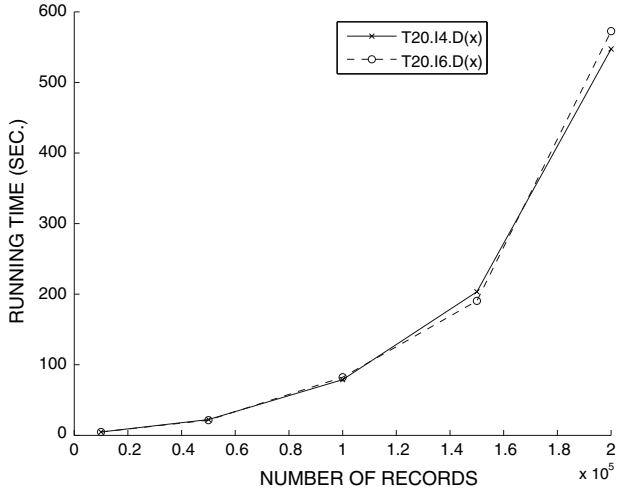

Fig. 15 Running time with increasing database size

the data set required a few seconds in each case in order to create the anonymized data set. Since the two curves for T(x).I4.D10K and T(x).I6.D10K are almost overlapping, this seems to suggest that the algorithm scalability is not too sensitive to the underlying characteristics of the data set. This is not surprising, since most of the operations of grouping the strings are not affected by the underlying behavior of the string. We also tested the scalability with increasing data set size. The results are illustrated in Fig. 15. We used the two data sets T20.I4.D(x), and T20.I6.D(x), respectively. The number of records is illustrated on the X-axis, and the running time in seconds is illustrated on the Y-axis. In each case, it is clear that the time complexity scaled superlinearly with

increasing number of strings. This is because the distance calculations needed to be performed multiple times for the different strings.

6 Conclusions and summary

In this article, we proposed a methods for condensation based privacy preserving data mining of strings. The string anonymization problem is important because of its natural application to the healthcare domain. Most current work in this domain (Malin and Sweeney 2001; Malin 2004) focusses on the problem of privacy- preservation of the strings only from the perspective of the underlying multi-dimensional information. This work has greater application to the health care domain, since it also focusses on the patterns in the strings themselves. In many cases, biological diseases or other physical characteristics occur as patterns in DNA string segments. For example, the most common (and easily observable) physical characteristics such as eye color, skin color, and hair quality can be derived from the underlying DNA string patterns. Such information can be used in order to narrow down the possibilities for a given record. When additional information is available in terms of relative DNA or possible diseases that a subject might have, the narrowing down process may be sufficient to uniquely identify the underlying record. Since our approach generates pseudo-data which retains the overall characteristics of the data, but does not maintain a one-to-one mapping, it retains the privacy of the underlying records.

The broad thrust of our approach is to utilize group-based segmentation in order to perform the anonymization. The segmented string data are then used in order to generate pseudo-data from the different strings. This generation is done by constructing a probabilistic model from each group. The probabilistic model stores first- and second-order information about the string templates in each group, and uses these summary statistics to generate strings which fit this model. We tested the resulting pseudo strings for a variety of aggregate statistics such as the composition, distance calculations, and the classification accuracy of the training model generated. We showed that the method turns out to be an effective approach for condensation based pseudo-data generation. This approach has the capability of preserving the statistical behavior of the original strings while retaining a high level of privacy. In future research, we will examine the possibility of improving the effectiveness of the string generation algorithms. We will also research techniques for extending this approach to the case of data streams.

References

Aggarwal CC (2002) On effective classification of strings with wavelets. In: ACM KDD conference

Aggarwal CC (2004) On *k*-anonymity and the curse of dimensionality. In: VLDB conference. Scalable clustering with balancing constraints

Agrawal D, Aggarwal CC (2002) On the design and quantification of privacy preserving data mining algorithms. In: ACM PODS conference

Agrawal R, Srikant R (1994) Fast algorithms for mining association rules in large databases. In: Proceedings of the VLDB conference

Agrawal R, Srikant R (2000) Privacy preserving data mining. In: Proceedings of the ACM SIGMOD conference

Aggarwal CC, Yu PS (2004) A condensation based approach to privacy preserving data mining. In: EDBT conference
Aggarwal CC, Yu PS (2005) On variable constraints in privacy preserving data mining. In: ACM SIAM data mining conference
Aggarwal CC, Yu PS (2007) On anonymization of strings. In: SIAM conference on data mining. http://www.charuaggarwal.net/str.pdf
Banerjee A, Ghosh J (2006) Scalable clustering with balancing constraints. Data Min Knowl Discov J 13: 365–395
Bayardo RJ, Agrawal R (2005) Data privacy through optimal *k*-anonymization. In: ICDE conference
Evfimievski A, Srikant R, Agrawal R, Gehrke J (2002) Privacy preserving mining of association rules. In: KDD conference
Iyengar V (2000) Transforming data to satisfy privacy constraints. In: ACM KDD conference
Kifer D, Gehrke J (2006) Injecting utility into anonymized data sets. In: ACM SIGMOD conference
LeFevre K, Dewitt DJ, Ramakrishnan R (2006) Mondrian multi-dimensional *k*-anonymity. In: ICDE conference
Machanavajjhala A, Gehrke J, Kifer D, Venkitasubramaniam M (2006) l-Diversity: privacy beyond *k*-anonymity. In: ICDE conference
Malin B (2004) Why methods for genomic data privacy fail and what we can do to fix it. In: AAAS Annual Meeting, Seattle, WA
Malin B, Sweeney L (2001) Re-identification of DNA through an automated linkage process. In: Proceedings, Journal of American Medical Informatics Associations. Hanley & Belfus, Inc, Washington, DC, pp 423–427
Meyerson A, Williams R (2004) On the complexity of optimal *k*-anonymity. In: ACM PODS conference
Needleman S, Wunsch C (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. J Mol Biol 48(3): 443–453
Rizvi S, Haritsa J (2002) Maintaining data privacy in association rule mining. In: VLDB conference
Samarati P, Sweeney L (1998) Protecting privacy when disclosing information: *k*-anonymity and its enforcement through generalization and suppression. In: Proceedings of the IEEE symposium on research in security and privacy
Sweeney L (1996) Replacing personally identifying information in medical records: the scrub system. In: Proceedings of the AMIA symposium
Wang K, Fung BCM, Yu PS (2006) Handicapping attacker's confidence: an alternative to *k*-anonymization. Knowledge Inform Sys Int J