On Anonymization of String Data

Charu C. Aggarwal* Philip S. Yu†

Abstract

String data is especially important in the privacy preserving data mining domain because most DNA and biological data is coded as strings. In this paper, we will discuss a new method for privacy preserving mining of string data with the use of simple template based condensation models. The template based model turns out to be effective in practice, and preserves important statistical characteristics of the strings.

Keywords: Privacy, strings, condensation

1 Introduction

The k-anonymity approach [4, 7] 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.

The *condensation-based* technique [1] 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. In this approach we construct condensed groups of records with the appropriate anonymity level, and generate pseudo-data which uses the aggregate statistics of each condensed group.

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 [5]. Many diseases of a genetic nature show up as specific patterns in the DNA of the individual. In general, it can be assumed that partial or complete information about the individual fragments of the strings is available. 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.

In this paper, we will discuss the condensation model for anonymization of string data. 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.

This paper is organized as follows. In the next section, we will discuss the process of creating the pseudo-strings for mining purposes. In section 3, we will present experimental results which illustrate the effectiveness of using the pseudo-strings in place of true strings. In section 4, 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*

*IBM T. J. Watson Research Center, charu@us.ibm.com
†IBM T. J. Watson Research Center, psyu@us.ibm.com

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* $i \in \{1 \ldots r\}$.

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 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 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. 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*.

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 $L_1 \ldots L_k$. Then, the length of the template representation of this set of strings is equal to $L = [\sum_{j=1}^{k} L_j / N]$.

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.

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-order 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:

$$Sc_{rpq}(\mathcal{G}) = \sum_{j=1}^{k} f_{pr}(T_j) \cdot f_{q(r+1)}(T_j) \tag{2.1}$$

  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 2 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 $\mathcal{F}f(\mathcal{G})$ 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:

$$Fs_{rp}(\mathcal{G}) = \sum_{j=1}^{k} f_{pr}(T_j) \tag{2.2}$$

  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 $\mathcal{D}$.

In this section we will discuss the partitioning approach for condensation and subsequent generation of the pseudo-strings. Since the anonymity level is assumed to be k, each partition should contain at least k strings. We also assume that the set of strings have already been homogenized and templates have been constructed from each segment. Therefore, the algorithm described in this section really applies to *each segment* of the homogenized database. Therefore, we can assume that all strings lengths are within a factor of $(1+\epsilon)$ from one another. In the event that the database contains strings with widely varying lengths, we need to perform the homogenization step, and then apply the partitioning process on each homogenized segment. 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 to each segment of this partitioning.
- In the next level of partitioning, we create groups of k strings 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 strings, we use an iterative algorithm in which the groups of records 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 strings. We start off by picking a random string and assign the $(k-1)$-closest strings to it. We will discuss the distance calculation process in more detail slightly later. We repeat the process of picking random strings and assigning the $(k-1)$ closest strings in the data to the currently selected string until all strings have been exhausted. As soon as each set of $(k-1)$ strings have been assigned, they are marked as assigned. We note that at the end of the process, fewer than k unmarked strings may remain because of successive assignments. Each string 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 strings assigned to them, but a few may have more than k strings 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 strings. The centroid of each group is defined as the average of all the strings in the group. This is an easy computation because of the homogenized length of each set of templates. 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 strings and create a new group from these strings. After forming this group, we mark this set of k strings from the database and continue the process until all strings 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 string database multiple times in order to improve the quality of the assignment. 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 less than 1%.

2.1 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 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:

$$dist(T_p, T_q) = \sum_{i=1}^{|T_p|} \sum_{r=1}^{l} |f_{ir}(T_p) - f_{ir}(T_q)| \tag{2.3}$$

2.2 Optimizations In some cases, it may be difficult to create a coherent group of strings. This is especially true towards the end of each iterative pass over the string database, when only a small number of strings remain. In such cases, it may be desirable to assign these strings 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 reduces. 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.

Data Set Name	Keywords
YST1	Yeast
YST2	Yeast
MS	Mouse
HS	Homo Sapiens

Table 1: Keywords used in SWISS-PROT database

2.3 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 $l(\mathcal{G})/n(\mathcal{G})$. 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 leftmost 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 statistics Fs_{1p} for the different symbols. Specifically the pth symbol is generated for the first position with probability $Fs_{1p}(\mathcal{G})/n(\mathcal{G})$.
- 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.

3 Experimental Results

In this section, we will discuss the experimental results of the condensation approach to privacy based mining of the strings. The data was obtained from the SWISS-PROT database which contained DNA sequences. Each

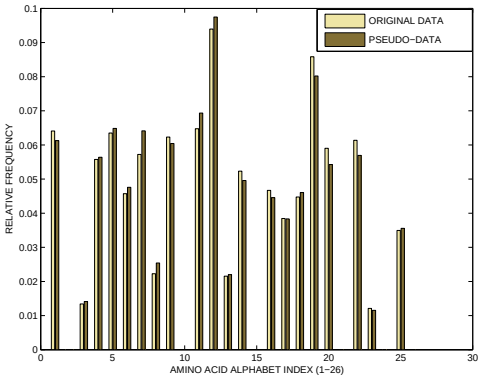

Figure 1: Comparison of original and pseudo-data in amino-acid composition (YST1)

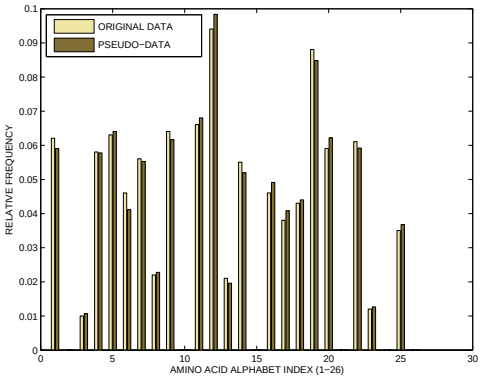

Figure 2: Comparison of original and pseudo-data in amino-acid composition (YST2)

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 names of the different databases and the corresponding keywords which were used to query the database for this purpose. 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. In each case 1000 strings from the database were used 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.

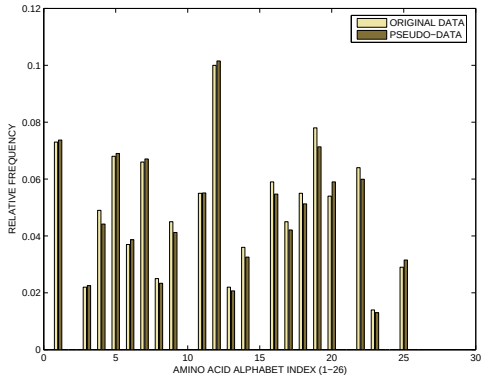

Figure 3: Comparison of original and pseudo-data in amino-acid composition (HS)

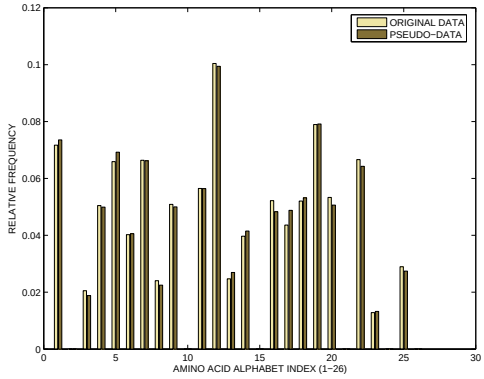

Figure 4: Comparison of original and pseudo-data in amino-acid composition (MS)

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. 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. In Figures 1, 2, 3, and 4, we have illustrated the aminoacid 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 Figures 1, 2, 3, and 4 respectively. Other aggregate tests on classification and distance function computation are available in an extended version of this paper [2].

4 Conclusions and Summary

In this paper, we proposed a methods for condensation based privacy preserving data mining of strings. We discussed a method to segment the string data into groups. The segmented string data is 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 both 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 aggregate composition, and showed that the approach retained similarity in composition while preserving privacy.

References

[1] C. C. Aggarwal, and P. S. Yu. *A Condensation Based Approach to Privacy Preserving Data Mining.* EDBT Conference, 2004.

[2] C. C. Aggarwal, and P. S. Yu. *On Anonymization of Strings.* IBM Research Report, 2007.

[3] C. C. Aggarwal. *On k-anonymity and the curse of dimensionality.* VLDB Conference, 2004.

[4] R. Bayardo, and R. Agrawal. *Data Privacy through optimal k-anonymization.* ICDE Conference, 2005.

[5] B. Malin, and L. Sweeney. *Re-identification of DNA through an automated linkage process.* American Medical Informatics Association, 423–427, 2001.

[6] A. Meyerson, and R. Williams. *On the complexity of optimal k-anonymity.* ACM PODS Conference, 2004.

[7] P. Samarati. *Protecting Respondents' Identities in Microdata Release.* IEEE Trans. Knowl. Data Eng. 13(6): 1010-1027, 2001.