Privacy Preservation by Disassociation

TR-IMIS-2011-1

Manolis Terrovitis, John Liagouris, Nikos Mamoulis,
Spiros Skiadopoulos

Institute for the Management of Information Systems,
"Athena" RC, Greece

February, 2011

Abstract

In this work, we focus on the preservation of user privacy in the publication of sparse multidimensional data. Existing works protect the users' sensitive information by generalizing or suppressing quasi identifiers in the original data. In many real world cases, neither generalization nor the distinction between sensitive and non-sensitive items is appropriate. For example, web search query logs contain millions of terms that are very hard to categorize as sensitive or non sensitive. At the same time, a generalization-based anonymization would remove the most valuable information in the dataset; the original terms. Motivated by this problem, we propose an anonymization technique termed *disassociation* that preserves the original terms but hides the fact that two or more different terms appear in the same record. Up to now, such techniques were used to sever the link between quasi-identifiers and sensitive values in settings with a clear distinction between these types of values. Our proposal generalizes these techniques for sparse multidimensional data, where no such distinction holds. We protect the users' privacy by disassociating combinations of terms that can act as quasi-identifiers from the rest of the record or by disassociating the constituent terms, so that the identifying combination cannot be accurately recognized. To this end, we present an algorithm that anonymizes the data by first clustering them and then locally disassociating identifying combinations of terms. We analyze the attack model and extend the k^m-anonymity guaranty to the aforementioned setting. We empirically evaluate our method on real and synthetic datasets.

1 Introduction

In this paper, we focus on the preservation of privacy in the publication of sparse multidimensional data. Common sources of such data are transactional logs like supermarket sales logs, credit card logs, web search query logs etc. Consider for example a dataset D which contains records that trace web search query logs. Even without any direct identifier in the data (user's name or ID) the publication of D might lead to privacy breaches, if an attacker has background knowledge that associates some queries to a known user. For example, assume that the attacker John knows that the user Jane was interested on air tickets to Santorini, so he has a background knowledge consisting of terms Santorini and air tickets. If D is published without any modification then the attacker can trace all records that contain both terms Santorini and air tickets. If only one such record exists, then he can easily infer that this is the record of Jane.

To counter such privacy leaks, we propose an anonymization method that transforms the original dataset D to a dataset D', where the association of the partial background knowledge to a specific record is infeasible. Most existing anonymization techniques for sparse multidimensional data [1, 2, 3, 4], focus on hiding these combinations by either suppressing some terms or generalizing them to a more abstract form. For example, the term Santorini can be suppressed so the infrequent combination will not appear at all in the published data. Alternatively, other methods would generalize Santorini to Aegean Island, so that the infrequent combination would be replaced by the frequent {Aegean Island, air tickets}. Both methods result in publications which miss many original terms. In this paper, we propose a method that preserves the original terms but hides their combinations. As shown in the previous example, unique combinations of terms, e.g., Santorini and air tickets, lead to the re-identification of individuals related to specific records in the dataset. The proposed anonymization method hides identifying combinations, not by hiding their constituent terms, but instead by suppressing the fact that two or more terms appear together in the same record. We call this transformation *disassociation*.

The proposed anonymization technique works for many types of sparse multidimensional data but our work is significantly motivated by the need to publish web search query logs ($\mathcal{WQL}$) due to the huge amount of information they contain. Having a log with all query terms posed by a user over a period of time can be of great value to market experts, web portal owners but also to scientists from various fields ranging form social sciences to database researchers. The only, to the best of our knowledge, completely open publication of such log, the AOL data [5], led to serious privacy breaches [6]. Anonymizing such datasets, where the initial terms are completely arbitrary, by employing generalization is not only impractical (creating a generalization hierarchy is far for trivial) but it would also hide the original queries which are of major value. Our proposal overcomes this problem and offers an alternative anonymization method that introduces an information loss of different nature.

A consequence of having data terms that are arbitrary or hard to characterize and categorize is that it is hard to partition them into quasi identifiers and sensitive values. Because of this problem, we opted for a privacy guaranty that protects against identity disclosure and we have adopted the k^m-anonymity paradigm [1]. This means that an attacker, who knows up to m items from any record, will not be able to match his knowledge to less than k records in the published data. The lack of partitioning the data to sensitive values and quasi identifiers distinguishes our work from [7], which is the only work, to the best of our knowledge, that does not use suppression or generalization in anonymizing sparse multidimensional data.

The proposed anonymization method transforms the original data to a k^m-anonymous dataset by using two forms of partitioning. First, data are segmented *horizontally* creating clusters of similar records. Next, the records inside these groups are partitioned *vertically* to *chunks* of similar sub-records. The aim of vertical partitioning is twofold; on one hand, it aims at preserving combinations that are already frequent enough as not to be identifying, i.e., they appear more than k times. On the other hand, it aims at breaking up infrequent combinations so that the records that originally contained them cannot be identified. We complement our anonymization method by providing an algorithm that refines the result of the previous partitioning so that combinations of terms that are not locally frequent, i.e. inside a cluster, but they are frequent in a neighborhood of similar clusters, are preserved.

Finally, we define a weaker form of k^m-anonymity which protects against identity disclosure from adversaries that have only abstract background knowledge. For example, an attacker might not know any actual query posed by a user, but instead he might know the abstract subjects of the queries. For example, an attacker might not know that Jane issued the queries Santorini and air tickets, but instead he might know that she searched for an island of the Aegean and for a transportation to it. This background knowledge is weaker, but it is a realistic setting in several application areas. In brief the contributions of this work are:

1. We propose a new data transformation for sparse multidimensional data that preserves the original terms of the dataset.
2. We show how this transformation can be used to anonymize the data and prove that the resulting data adhere to our privacy guaranty.
3. We propose an algorithm that anonymizes the data and we evaluate it experimentally on real and synthetic datasets.
4. We propose a weaker form of k^m-anonymity, which protects against adversaries who have only abstract background knowledge.

The rest of the paper is organized as follows: In Section 2 we describe the problem and the basic notions of our proposal. In Section 3, we discuss issues in

disassociation-based anonymization and prove that the result of the disassociation transformation meets our privacy guarantee. Our anonymization algorithm is presented in Section 4. Section 5 contains the experimental evaluation of our method. In Section 6, we position our contribution with respect to the related work in the field and we conclude in Section 7.

2 Problem definition

In this section, we describe the problem setting and the basic idea behind our anonymization method.

Dataset. We assume a sparse multidimensional dataset D which is formed by *records* that encompass a variety of *terms* from a huge *domain* T. Protecting user privacy in such setting is a challenging task, since each record may contain a unique term combination that can identify the individual that is associated with the record. As a motivating example, we consider web search query logs ($\mathcal{WQL}$) that trace the queries posed by a user over a period of time. To simplify the discussion, we define $\mathcal{WQL}$ data records and terms as follows: *(a)* a *term* is a single query posed by a user and *(b) records* are formed by the terms (i.e., queries) of a specific user. In our setting, we do not make a distinction between terms containing sensitive values and quasi identifiers since any term might reveal information that is considered sensitive for a user and any term can be used to identify a user. Having a clear distinction between sensitive terms and quasi identifiers, simplifies the problem and, thus, our methods can easily be adjusted to such a case.

Anonymity guaranty. The anonymization process for multidimensional data should take into account not only the terms of each record but also *their relations*. Although a single term may be very common, its co-existence with other terms could be rare enough to uniquely identify a record. Thus, an attacker who knows a rare combination of (even frequent) terms can use this knowledge to retrieve the corresponding record and associate it to a specific person.

In this work, we focus on the privacy protection *against identity disclosure*. The most popular guaranty for the protection of user identity is k*-anonymity* [8], which makes each published record indistinguishable from other $k-1$ published records. In the multidimensional data setting where all terms can potentially act as quasi identifiers, k-anonymity is achieved by creating equivalence classes of identical records. In each class, every record should not be distinguished even if the attacker knows all of its terms (recall that all terms may act as quasi-identifiers). There two fundamental problems with this approach: *(a)* since records can be drastically different from each other (both in the values and the arity of their terms), their transformation into equivalent classes can result in huge information loss, and *(b)* it does not make much sense to hide a record from an attacker that knows all of its terms.

For the above reasons, we consider k^m-anonymity [1]. k^m-anonymity is a

conditional form of k-anonymity which guarantees that an attacker who knows up to m terms will not be able to distinguish any record from other $k-1$ records. By defining different values for m, we can guarantee different levels of protection at the expense of data quality. In our setting, we will use the following definition of k^m-anonymity.

Guaranty 1. *An attacker who knows up to m terms that identify an individual U, will not be able to infer that there are less than k candidates* for any part of the record *associated with U*.

Attack model. The identification of a user in a $\mathcal{WQL}$ is made possible by tracing records that contain unique combinations of terms. For example, if an attacker knows that a user U has searched for terms a and b and there is only one record that contains a and b in the anonymized dataset, the attacker is certain that this record is associated to the user U. We assume that the attacker:

- Knows that information about a user appears in the published dataset.
- Has background knowledge of up to m terms that correspond to queries posed by any user.
- Does not have negative knowledge, i.e., the attacker does not know if a user *did not* pose any specific query.
- Does not have background knowledge sufficient to infer negative knowledge about a user. For example, if term a appears 100 times in the data, we assume that the attacker does not have enough background knowledge that links a to 100 difference persons, so she can rule out the possibility that a is associated to any other person.
- Is interested in identifying a record or a part of a record that is associated with a specific user.

3 Disassociation-based Anonymization

Anonymization methods are characterized by their (a) data transformation, (b) privacy protection, and (c) information loss. In this section we explain these points in detail and motivate our framework which applies disassociation transformation to the original data. Furthermore, we define a weaker form of k^m-anonymity for attackers who only have abstract knowledge about the terms.

3.1 Data transformation

Our approach avoids generalizing the terms in the published data, but instead it reduces the structural information contained in the data. The key idea is to hide the

ID	Records
r_1	{itunes, swine flu, madonna, ikea, ruby}
r_2	{madonna, swine flu, viagra, ruby, audi a4, sony tv}
r_3	{itunes, madonna, audi a4, ikea, sony tv}
r_4	{itunes, swine flu, viagra}
r_5	{itunes, swine flu, madonna, audi a4}
r_6	{madonna, digital camera, panic disorder, sony tv, playboy}
r_7	{iphone sdk, madonna, ikea, ruby}
r_8	{iphone sdk, digital camera, madonna, playboy}
r_9	{iphone sdk, digital camera, panic disorder}
r_{10}	{iphone sdk, digital camera, madonna, ikea, ruby}

(a) Original dataset

1^{st} *cluster* - **Record chunks**		**Term chunk**
C_1	C_2	C_T
{itunes, swine flu, madonna} {madonna, swine flu} {itunes, madonna} {itunes, swine flu} {itunes, swine flu, madonna}	{audi a4, sony tv} {audi a4, sony tv} {sony tv} {audi a4}	ikea, viagra, ruby

2^{nd} *cluster* -**Record chunk**	**Term chunk**
C_1	C_T
{madonna, digital camera} {iphone sdk, madonna} {iphone sdk, digital camera, madonna} {iphone sdk, digital camera} {iphone sdk, digital camera, madonna}	panic disorder, playboy, ikea, ruby

(b) Anonymized dataset

Figure 1: Disassociation example

fact that certain terms appear together in the same record. In this sense, we aim at *disassociating* those terms whose combination can work as a quasi identifier.

To achieve anonymization and provide Guaranty 1, our method performs a *horizontal* followed by a *vertical* partitioning of the input dataset. The horizontal partitioning brings similar records together, but the heart of the anonymization procedure lies in the vertical partitioning, which breaks up infrequent combinations of terms.

Horizontal partitioning. This partitioning results in *clusters* that group together records having similar terms. Thus, this step only regroups the original dataset without anonymizing it. Horizontal partitioning reduces the anonymization of the initial dataset to the anonymization of a number of independent clusters with severely reduced cardinality. The benefits are twofold. First, the anonymization process can be done more efficiently and even in parallel. Second, it limits the scope of the term disassociation to the records that are contained in the cluster, thus it limits the effect of disassociation to the information quality of the dataset. Bringing similar records together reduces the need for disassociating terms, thus it further reduces information loss.

Vertical partitioning. This partitioning is the core of the anonymization process. Intuitively, vertical partitioning leaves term combinations that appear many times intact and disassociates terms that create infrequent and, thus, identifying combina-

tions. The disassociation is achieved by suppressing the fact that these terms appear together in a single record. Specifically, vertical partitioning further divides clusters into *chunks*. There are two types of chunks: record and term chunks. *Record chunks* contain subrecords of the original dataset formed by terms appearing more than k times, and they are k^m-anonymous. That is, every m-sized combination of terms that appears in a chunk, appears at least k times. *Term chunks* contain the terms which appear less than k times. Each cluster may contain an arbitrary number of record chunks (≥ 0) and exactly one term chunk (which might be empty).

Vertical partitioning is applied to each cluster (resulted from the horizontal partitioning) independently. Let us consider a cluster C and let T^C be the set of terms that appear in C. To partition C into v record chunks $C_1, \ldots, C_v$ and a term chunk C_T, we divide T^C into $v+1$ subsets $T_1, \ldots, T_v, T_T$ that are pairwise disjoint (i.e., $T_i \cup T_j = \emptyset$, $i \neq j$) and jointly exhaustive (i.e., $\bigcup T_i = T^C$). Subsets $T_1, \ldots, T_v$ are used to define record chunks $C_1, \ldots, C_v$ while subset T_T, is used to define term chunk C_T. Specifically, record chunks C_i, $1 \leq i \leq v$ are defined as:

$$C_i = \{\!\{ T_i \cap r \mid \text{for every record } r \in C \}\!\} \tag{1}$$

where $\{\!\{\cdot\}\!\}$ denotes a collection with bag semantics (i.e., duplicate records are allowed in C_i). The partitioning of T^C to $T_1, \ldots, T_v, T_T$ is done in that each record chunk $C_1, \ldots, C_v$ is k^m-anonymous.

Finally, the term chunk is defined as:

$$C_T = T_T. \tag{2}$$

Note that chunks $C_1, \ldots, C_v$ are collections of records while chunk C_T is a set of terms.

Example 1. *An example of the anonymization process is illustrated in Figure 1. The original $\mathcal{WQL}$ consisting of 10 records (each being the web search history of a different user) is presented in Figure 1a. An attacker that knows few query terms may easily identify a specific user. For instance, knowing that a specific user requested information about* swine flu *and* sony tv*, an attacker can easily identify record r_2 as the web history of the related user in the published data.*

The proposed anonymization method initially partitions horizontally the input data into two clusters; the first cluster corresponds to records r_1 to r_5 while the second cluster corresponds to records r_6 to r_{10} (see also Figure 1a).

Following, each cluster is vertically partitioned into chunks. The first cluster contains 3 chunks (2 record chunk and 1 term chunk) while the second contains 2 chunks (1 record chunk and 1 term chunk). For the first cluster we have $v = 3$, $T_1 =$ {itunes, swine flue, madonna}, $T_2 =$ {audi a4, sony tv} and $T_T =$ {ikea, viagra, ruby} and for the second cluster we have $v = 2$, $T_1 =$ {madonna, digital camera, iphone sdk} and $T_T =$ {panic disorder, playboy, ikea, ruby}. The record and term chunks are populated according to Equations 1 and 2 respectively.

The published data of Figure 1b are 3^2-anonymous. Note that the attacker that could identify record r_2 in the original dataset using terms swine flu *and* sony tv

cannot do so in the anonymized dataset since they do not appear in any subrecord together. Additionally, the attacker is not able to identify even a subrecord of the original record r_2 *since* swine flu *occurs in 4 subrecords of the first chunk and* sony tv *occurs in 3 subrecords of the second chunk.*

In the general case, the records that originate a cluster cannot be deduced from chunks. In fact, given an anonymized dataset, we know that the original cluster contains at least as many records as the respective number of the larger chunk but we do not know the exact number. This is not desirable, because it introduces significant information loss. For instance, if the number of records that originate each cluster is not available then it is not feasible to make reasonable estimations about the co-existence of terms in different clunks. To handle this situation, we *publish* the original size of each cluster in the anonymized dataset.

To summarize the anonymization process, every record of the original dataset is assigned to a cluster, split into subrecords (according to Equations 1 and 2), and stored into some chunks (of the respective cluster). For instance in Example 1 and Figure 1, r_1 belongs to the first cluster and is split into *(a)* the 1st record of its first chunk (C_1) and *(b)* terms ikea and ruby of its term chunk (C_T). Note that r_1 does not have terms in the second chunk (C_2).

Semantics. The anonymized dataset models a set of possible datasets. This is consistent with previous anonymization techniques that do the same via generalization. However, in our case the possible databases are not described in a closed form captured by the generalization ranges, but by the possible combinations of the chunks contents. This is a novel approach that has not been investigated in the past.

In other words, the exact initial dataset is hidden amongst multiple possible assignments of terms and subrecords to records. Additionally, these assignment are similar enough to allow a meaningful analysis. The potential analyst could exploit the anonymized dataset as it is or by performing a random assignment of the chunks' elements to records to obtain a *reconstructed* dataset. In Section 3.4 we elaborate on reconstructed datasets and their quality.

3.2 Anonymity guaranty

Since our anonymization method works independently in each cluster, to enforce Guaranty 1, we consider each cluster separately. Let C be an arbitrary cluster of the anonymized dataset which is vertically partitioned into record chunks $C_1, \ldots, C_v$ and term chunk C_T.

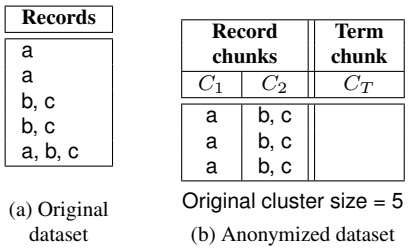
Let us now assume that the attacker knows m terms of a user's record and wants to identify this record. For this task, the attacker cannot employ the terms that reside in the term chunk C_T since this chunk does not contain records. Thus, in the following we will assume the worst case, that all terms that the attacker possesses reside in the record chunks $C_1, \ldots, C_v$. We consider the following two cases.

Case 1: Terms belong to one chunk. It is not hard to see that if all record chunks $C_1, \ldots, C_v$ are k^m-anonymous then Guaranty 1 holds.

Case 2: Terms belong to more than one chunk. Since each chunk is k^m-anonymous, the attacker cannot use the terms to identify less than k subrecords that contain them in any chunk. To enforce Guaranty 1 we may employ the following lemma.

Lemma 1. *Let $C_1, \ldots, C_v$ be the record chunks and C_T be the term chunk that correspond to the anonymization of a cluster C. If* (a) *chunks $C_1, \ldots, C_v$ are k^m-anonymous and* (b) *$C_T \neq \emptyset$ then Guaranty 1 holds.*

In the (rare) case where $C_T = \emptyset$, the k^m-anonymity of all record chunks $C_1, \ldots, C_v$ does not ensure Guaranty 1, as the following example illustrates.



(a) Original dataset

(b) Anonymized dataset

Figure 2: Illustration of Example 2

Example 2. *Let us consider the dataset of Figure 2a. To anonymize this dataset, we set $T_1 = \{\mathsf{a}\}$, $T_2 = \{\mathsf{b},\mathsf{c}\}$ and $T_T = \emptyset$ and use the method presented in Section 3. The anonymized dataset (illustrated in Figure 2b) contains 2 record chunks (C_1 and C_2) having 3 subrecords each and an empty term chunk (C_T). It is not hard to verify that all chunks are 3^2-anonymous.*

Let us now consider an attacker X that knows: (a) *the anonymized dataset of Figure 2,* (b) *that the size of the original cluster is 5 (see also the discussion in Section 3.1) and* (c) *that a user had used terms* a *and* b*, i.e.,* (a, b) *is a subrecord of the original dataset.*

Attacker X also knows that the original dataset is composed by a combination of the records stored in chunks C_1 and C_2. Out of all these combinations, X may only keep those that result in datasets with 5 records. Using this procedure X is certain that the original dataset is the one presented in Figure 2a since there does not exist any other combination resulting in a dataset with 5 records.

To handle the situation demonstrated by Example 2 and enforce Guaranty 1, we must establish that for any attacker who knows up to any m terms of a record, there will be a possible initial dataset that has at least k records that contain the m terms. Fortunately, we do no need to reconstruct all possible initial datasets to see if this condition is satisfied. It suffices to enforce the condition of the following lemma.

Lemma 2. *Let $C_1, \ldots, C_v$ be the record chunks that correspond to the anonymization of a cluster with size s. If* (a) *chunks $C_1, \ldots, C_v$ are k^m-anonymous and* (b) *the total number of subrecords in all chunks $\sum(|C_i|)$ is greater or equal to $s + m \cdot (k-1)$ then Guaranty 1 holds.*

Proof. To prove this lemma, it suffices to construct a cluster C from record chunks $C_1, \ldots, C_v$ for which $|C| = s$ and Guaranty 1 holds. To this end, we initially consider m terms $t_1, \ldots, t_m$ that are scattered in m distinct chunks C_{t_i}. Every term t_i (since all chunks are k^m-anonymous) appears in least k times in C_{t_i}. Let $r_1^i, \ldots, r_k^i$ be the k records that term t_i appears ($1 \leq i \leq m$) in C_{t_i}. We also have $\sum(|C_{t_i}|) \geq k \cdot m$. It follows from the second condition of the lemma that, apart from records r_j^i ($1 \leq i \leq m$, $1 \leq j \leq k$), chunks $C_1, \ldots, C_v$ contain at least $s + m \cdot (k-1) - k \cdot m = s - m$ records. Let us denote these records by $e_1, \ldots, e_{s-m}$.

We can now construct cluster C as follows:

$$C = \{r_1^1 \cup \cdots \cup r_1^m, \ldots, r_1^1 \cup \cdots \cup r_1^m, e_1, \ldots, e_{s-m}\}.$$

It is easy to verify that C is k^m-anonymous (due to the first k records) and its size is s. The proof for the case where terms $t_1, \ldots, t_m$ are scattered in less than m distinct chunks is analogous. □

3.3 Extensions

In the presented anonymization method when a term t appears less than k times in a cluster it is moved to the term cluster without considering its support in any other cluster. Doing so, we might lose important correlations, even if t is frequent in other clusters.

A way to deal with such cases is to relax the partitioning scheme by allowing records from different clusters to share a chunk. In other words, instead of considering the horizontal and vertical partitioning in two independent steps, we may introduce processing rounds of refinement. In each round, terms that were considered infrequent and were placed at the term chunk of a cluster (in a previous round), are used to create chunks that are shared between different clusters.

Consider the example of Figure 1. The correlation between ikea and ruby is not preserved since the anonymity method takes into account the appearances of these terms in each cluster independently. Note that, if we consider both clusters as one, the number of appearances of ikea and ruby is not small. We address this issue by creating a shared chunk storing the subrecords that contain ikea and ruby as depicted in Figure 3. This procedure can be recursively repeated; we can first create shared chunks for two clusters, then for four clusters and so on.

The most important question associated with refining the clusters and creating shared chunks, is whether there is information gain in doing so or not. The basic question is whether it is better to state that a term appears in few records that comprise a single cluster (by publishing it at the term chunk) or to create a shared

Record chunks		Term chunk	Shared chunk
1^{st} *cluster*			
{itunes, swine flu, madonna} {madonna, swine flu} {itunes, madonna} {itunes, swine flu} {itunes, swine flu, madonna}	{audi a4, sony tv} {audi a4, sony tv} {sony tv} {audi a4}	viagra	{ikea,ruby} {ruby} {ikea} {ikea,ruby} {ikea,ruby}
2^{nd} *cluster*			
{madonna, digital camera} {iphone sdk, madonna} {iphone sdk, digital camera, madonna} {iphone sdk, digital camera} {iphone sdk, digital camera, madonna}		panic disorder, playboy	

Figure 3: Disassociation with a shared chunk. Clusters 1 and 2 have formed a greater cluster.

chunk that contains the projections of all the records that contain it, but associate the chunk with numerous records. In Section 4.3 we present an algorithm that answers these questions and refines the initial anonymization results.

A final note about shared chunks is that they can provide the attacker with an opportunity to breach the privacy of the dataset by performing a reverse engineering attack; the attacker can exploit conflicts between assignments of records from shared chunks, relying on the fact that terms cannot appear twice in a record. If a cluster C has a record chunk built over terms a, b and then participates in a shared chunk sc built over terms a, b, c, then the attacker can infer that subrecords from sc than contain either a or b cannot be originating from any of the records of the cluster C. This attack is countered by requiring that all projections of a shared chunk sc to terms that have already appeared in record or shared chunks in lower levels (i.e., smaller chunks whose records participate to sc) appear at least k times in *distinct subrecords* in the shared chunk. In the interest of space we refer the interest reader to the long version of the paper for details [9]. The anonymization algorithm implemented for this paper, counters these attacks and all the reported experiments have taken into account the protection from such attacks.

3.4 Utility metric

The anonymization method aims at preserving as much information as possible from the initial data. We do not tailor our algorithms to specific queries, since we do not make any strict assumptions about how the published data are going to be used. Considering that our method hides the exact details about the existence of infrequent combinations, we use utility metrics that capture the deviation of the supports of term combinations from the original to the anonymized dataset. We use a series of similar metrics, as to avoid overspecializing to one of them:

tKd **(top-K deviation)**. The idea behind this metric is simple; we want to measure how the top-K frequent itemsets from the initial dataset change in the published

data. Let FI_o be the top-K frequent itemsets in D and FI_p be the top-K in the published data. To trace the FI_p we count only combinations that appear in a single chunk, and we do not take into account combinations that might appear when joining the subrecords of different chunks. For example if a and b are two terms that appear together as a subrecord $\{a, b\}$ in one chunk, we are going to count them towards calculating the support of $\{a, b\}$ in the published data. On the other hand, even if a appears 4 times in a chunk C_1 and b appears again 4 times in a chunk C_2 of the same cluster C with size 5, they will not affect the calculation of the support of $\{a, b\}$ since they appear on separate chunks. The measure for tKd is defined as follows:

$$tKd = 1 - \frac{|FI_o \bigcap Fi_p|}{|FI_o|} \tag{3}$$

tKd expresses the ratio of the top-K frequent itemsets of the original dataset that appear in the top-K frequent itemsets of the anonymized data.

bD **(Belief deviation)**. Since we are not interested in publishing only the top-K most frequent itemsets, but we are also interested in less frequent combinations we have to find a way to measure the effect of anonymization on them. A side-effect of the disassociation transformation is that is also creates the belief that some terms might appear together at the initial data, whereas they do not. We would like to measure the change on our belief for the support of any combination of terms from T due to the anonymization. For practical purposes we limit ourselves to measuring the deviation between combinations of two terms. If $s_o(a, b)$ is a function that gives us the support of combination $\{a, b\}$ in the original data and $es_p(a, b)$ is the function that gives us the *estimated* support of the same combination in the published data, then we define bD as:

$$bD = \frac{|es_p(a, b) - s_o(a, b)|}{AVG(s_o(a, b), es_p(a, b))} \tag{4}$$

Using the average of the two supports as denominator, instead of using the original support $s_o(a, b)$, has a smoothing effect on the metric, since it normalizes it to 0-2, and it avoids division by 0 in the case of combinations that do not appear in the original data. Note that the es_p does not estimate the support of a term combination in D' as conservatively as we calculated the support of the top-K frequent terms for tKd. Unlike the case of tKd we do not count only the times that two terms a and b appear in the same subrecord of a record chunk, but instead we calculate the probability of appearance for $\{a, b\}$ even if a and b appear in different chunks of the same cluster. Consider the example we presented for tKd where a and b appear 4 times in two different chunks of cluster C, which contains 5 records. Then the probability of appearance of a in any record is 0.8 and the same holds for b. The probability that both of them appear in the same record is the product of these probabilities, thus $0.8 \times 0.8 = 0.64$. Because we have 5 records in the cluster

we estimate the support of $\{a, b\}$ in C as $es_p(a, b) = 0.64 \times 5 = 3.2$. Thus we estimate that $\{a, b\}$ appear 3.2 times in this cluster.

The aim of tKd is to give us a conservative estimation of the information loss due to the anonymization procedure by calculating how the most frequent patterns are preserved in the published data. To this end, it considers only the combinations of terms that appear in the same subrecord of each chunk (shared or not). The bD metric aims at estimating the information loss in cases of less frequent patterns where the data analyst must make some estimations about their support. To this end, we take into account the probability that terms from different chunks of the same cluster will appear together and we also take into account combinations of terms that do not appear at all in the original data.

Reconstructed datasets. A way to avoid adjusting existing analysis methods to the type of dataset produced by the disassociation anonymization method is to reconstruct a possible initial dataset and use it for analysis. This is achieved by randomly assigning the subrecords of a cluster's record chunks to the records that participate in the cluster. We implemented such an algorithm using the following assumptions: a) it considers all records of a chunk equally probable when assigning a subrecord, b) it does not place subrecords to a record that already contains some items of a subrecord, e.g., if terms a, b, c have already be assigned to record r, then subrecord c, d cannot be assigned to it and c) it makes the most conservative assumption about the term chunk, i.e., it assumes that each term appears only once and assigns it randomly to a record of the cluster. The algorithm is fairly straightforward so we omit its details due to space constraints. Clusters and record chunks are created in a way that they group together similar records and subrecords, thus reconstructing a random possible initial dataset results to a dataset that is very similar to the original one. To measure the quality of the reconstructed dataset, i.e., how similar it is to the original one, we may again use the tkd and bD metrics. In this case we measure tkd by mining the reconstructed dataset and we measure bD not by estimating the support of a combination as we did for the anonymized data, but we can explicitly count it as we do in the original data. A way to further increase the accuracy of the analysis of the reconstructed data is to create more than one reconstructed datasets and take into account the average support of each combination and itemset in every reconstructed dataset. In Section 5 we investigate experimentally how tkd and bD are affected when we take into account more that one reconstructed datasets.

3.5 Attacks with abstract knowledge

In $\mathcal{WQL}$, as in other application areas, the domain of terms is arbitrary since each user freely introduces queries. Having a huge diversity in queries might be a problem for anonymization methods, since users pose unique queries that can reveal their identify. For example in the AOL data, 90% of the queries are unique and 97% are posed 3 times or less [10]. On the other hand, an arbitrary vocabulary renders very unlikely the fact that the attacker knows the exact query posed by a

user. For example, an attacker might know that a user U searched for information cornering sport cars but since U can express the exact query anyway he wants, the attacker cannot know that U used terms diesel sport cars and new sport cars.

Based on this observation it is possible for the publisher to define a weaker form of privacy for $\mathcal{WQL}$ and other sparse multidimensional data, which provides guaranties against attackers which do not know the exact terms that are related to a user, but only an abstraction of them.

Let us consider a dataset D. Let T be the set of terms that appear in D and assume a partition of T to subsets $L_1, \ldots, L_u$ that are pairwise disjoint (i.e., $L_i \cup L_j = \emptyset$, $i \neq j$) and jointly exhaustive ($\bigcup L_i = T$). For every set L_i, we introduce a term l_i that abstractly represents all members of L_i. Thus, l_i represents a class of terms. We define *abstract* k^m-anonymity as:

Guaranty 2. *An attacker who knows up to m classes of terms (among $l_1, \ldots, l_u$) that identify an individual U, will not be able to infer that there are less than k candidates for any part of the record associated with U.*

Guaranty 2 is weaker than Guaranty 1; it protects against attackers, which know only that a combination of up to m classes of terms (among $l_1, \ldots, l_u$) appears in the record of a user. Unique combinations of terms might still appear in a dataset that adheres to abstract k^m-anonymity. In order for the publisher to be able to provide such a guaranty he must be able to create a classification $L_1, \ldots, L_u$ that captures as accurately as possible the knowledge of the attacker. Still, the publisher does not necessary have to group all terms appearing in the original data. If the publisher cannot decide on the similarity of some terms, he can always choose to create a class containing only 1 term. The size of the classes affects the power of the privacy guaranty and the quality of the published data; the more abstract the classes are, the weaker the privacy guaranty, but the better the data quality will be.

4 The proposed algorithm

In this section, we present the details of our algorithm, which is based on two simple heuristics for the horizontal and vertical partitioning phases. In addition, we discuss how its results can be refined by creating shared chunks. Finally, we explain how our method can be adapted for anonymizing data in different abstraction levels.

4.1 Horizontal partitioning

This partitioning can be theoretically performed by adopting any of the related clustering algorithms in the literature for set-valued data [11]. Unfortunately, the adoption of a clustering algorithm to our setting is hard mainly because such methods are not efficient on large datasets like $\mathcal{WQL}$ and because they do not give us explicit control over the size of the clusters. Therefore, we have opted to design a

lightweight algorithm able to efficiently cluster huge datasets. To this end, we propose the *divide and conquer* strategy of Algorithm HORPART, illustrated in Figure 4. The key idea is to split the dataset into two parts: one with the records that contain the most frequent term a in the dataset and one with the rest of the records. This procedure is recursively applied to the new datasets until the final datasets are small enough to become clusters. Note that terms that have been previously used for partitioning are stored in set $ignore$ and are not used for subsequent partitions (see also Line 3).

As a final step, we unify small adjacent clusters in order to create clusters that have at least a minimum size of $minClusterSize$. To simplify the presentation, we have omitted this step in Algorithm HORPART.

Algorithm: HORPART
Input : Dataset D, set of terms $ignore$ (initially empty)
Output : A HORizontal PARTitioning of D
Param. : The maximum cluster size $maxClusterSize$

1 **if** $|D| < maxClusterSize$ **then return** $\{\!\{D\}\!\}$;
2 Let T be the set of terms of D;
3 Find the most frequent term a in $T - ignore$;
4 $D_1 =$ all records of D having term a;
5 $D_2 = D - D_1$;
6 **return** HORPART$(D_1, ignore \cup \mathsf{a}) \cup$HORPART$(D_2, ignore)$

Figure 4: Algorithm HORPART

4.2 Vertical partitioning

To vertically partition the cluster, we follow the *greedy* strategy of Algorithm VERPART illustrated in Figure 5, which is executed independently for each cluster. Thus, VERPART takes as input a cluster C and integers k and m. The algorithm performs a vertical partition and outputs the k^m-anonymous record chunks $C_1, \ldots, C_v$ and the term chunk C_T of C.

Let T^C be the set of terms of C. Initially, the algorithm computes the number of appearances (support) $s(t)$ of every term t and sorts T^C with decreasing $s(t)$. All terms that appear less than k times are moved from T^C to T_T. Next, the algorithm computes sets $T_1, \ldots, T_v$ (while loop). To this end, the algorithm uses sets T_{remain} (that contains the terms that have not been assigned – similarly to T^C, T_{remain} is sorted with decreasing s) and T_{cur} (that contains the terms that will be assigned to the current set). To compute T_i ($1 \leq i \leq v$), Algorithm VERPART considers all terms of set T_{remain}. A term t is inserted into T_{cur} only if the chunk constructed from $T_{cur} \cup \{t\}$ remains k^m-anonymous (Line 12). If the insertion of a term t to T_{cur} renders $T_{cur} \cup \{t\}$ non k^m-anonymous, t is skipped and the algorithm continues with the next term. After having assigned to T_{cur} as many terms from T_{remain} as possible, the algorithm *(a)* assigns T_{cur} to T_i, *(b)* removes the terms of T_{cur} from T_{remain} and *(c)* continues to the next set T_{i+1}. Finally, Al-

gorithm VERPART constructs record chunks $C_1, \ldots, C_v$ using $T_1, \ldots, T_v$ and the term chunk C_T using T_T (using Equations 1 and 2 respectively)

Algorithm: VERPART
Input : A cluster C, integers k and m
Output : A k^m-anonymous VERtical PARTitioning of c

Let T^C be the set of terms of C;
for *every term* $t \in T$ **do**
    Compute the number of appearances $s(t)$;
Sort T^C with decreasing $s(t)$;
Move all terms with $s(t) < k$ into T_T;
$v = 0$;
$T_{cur} = \emptyset$;
$T_{remain} = T^C - T_{term}$; //$T_{remain}$ has the ordering of T^C
while $T_{remain} \neq \emptyset$ **do**
    for *every term* $t \in T_{remain}$ **do**
        Create a chunk C using $T_{cur} \cup \{t\}$ (Equation 1);
        if C *is* k^m*-anonymous* **then** $T_{cur} = T_{cur} \cup \{t\}$;
    v++;
    $T_v = T_{cur}$;
    $T_{remain} = T_{remain} - T_{cur}$;
Create record chunks $C_1, \ldots, C_v$ using $T_1, \ldots, T_v$;
Create term chunk C_T using T_T;
return $C_1, \ldots, C_v, C_T$

Figure 5: Algorithm chunks

The algorithm has always the trivial solution of putting everything to the term chunk, thus it always finds a solution. Moreover, in every iteration, at least one term that has support over k will be placed in a record chunk. Since record chunks are only created if they are k^m anonymous (a single item with support over k in a record chunk trivially creates a k^∞-anonymous chunk), the resulting cluster will be k^m-anonymous, according to Guaranty 1.

Complexity. Since the proposed algorithm is a heuristic it cannot provide a strong guaranty for the quality of the result. On the other hand, its computational complexity is low. Asymptotically, the most expensive part of the algorithm is the vertical partitioning. Creating the chunks requires counting the support of itemsets, thus the optimal solution is a computationally intensive problem (even simple mining for frequent itemsets has been shown to be $\#P$-complete [12]). Additionally, we have the problem of creating groups of terms according to the support of their combinations. The cost of vertically partitioning each cluster depends on the size of the created chunks, because the chunk size affects the number of combinations to be examined. In the worst case the size of the created chunks increases as the domain of the cluster increases. The size of the cluster domain is affected both by the average record length and the total domain of the dataset. The advantage of the

anonymization algorithm is that the vertical partitioning heuristic processes only the records of one cluster at a time. Thus, despite the significant asymptotic cost of the vertical partitioning, the algorithm has explicit control over the size of its input. Since the size of the cluster does not depend on the size of the dataset $|D|$, increasing $|D|$ will only produce linearly more clusters, thus the cost of vertical partitioning phase will only increase linearly following the increase of the clusters number. The horizontal partitioning phase on the other hand is $O(|D|^2)$, since in the worst case it requires $\frac{|D|}{|C_{min}|}$ partitionings (where $|C_{min}|$ is the minimum cluster size), when one cluster is created in each step. Each partitioning has cost $O(|D|)$ since all records might have to be examined to detect their most frequent item, thus the overall cost of horizontal partitioning has worst case complexity $O(D|^2)$.

4.3 Refining

Placing terms in the term chunk introduces a significant information loss to the dataset. As we discussed in Section 3, we can avoid placing a term a in the term chunk if we can create a shared chunk between existing clusters that contain a at least k times in total. An optimal solution to the creation of shared chunks problem would be to check how any subset of the exiting clusters can be combined to create a shared chunk. This is a very expensive solution and we have opted for some simplifications that permit the fast creation of shared chunks:

- When clusters share a chunk, we assume that they form a new *greater cluster* and they participate all together in further creation of shared chunks or they do not participate at all. For example, the clusters of Figure 3 have formed one single greater cluster.
- Shared chunks are created using only terms from the term chunk and the terms that have already been assigned to record chunks are not affected[1]. Greater clusters form new greater clusters based on the terms that appear in the term chunks of their constituent clusters. For example, the greater cluster of Figure 3 can be combined with other greater clusters based on viagra, panic disorder or playboy.

The aim of creating greater clusters with shared chunks is to improve the quality of the published dataset. Avoiding to place a term in the term chunk has obvious merits, but it is not always the best choice. Consider for example placing a term a in the term chunk of a small cluster with 10 records. We are then revealing that the term a appears in some of these 10 records, without disclosing any information about the number of appearances or about how it is combined with other terms. Alternatively, we might have the option to create a shared chunk containing a, but the shared chunk would be associated with a greater cluster of 1000 records. In

[1]Even if a term a appears in the term chunk of one cluster and in a record chunk of another cluster, these clusters cannot create a shared chunk containing a.

the latter case, we are revealing accurate information about its support, e.g., it appears 3 times in total, but we have 1000 instead of 10 candidate records that might contain it. Choosing the best option is not straightforward; the best way to answer this dilemma would be to calculate the information loss as defined in Section 3.4 and choose the solution with the best effect on data quality. Since, such an option is very expensive we use a simpler criterion. Our criterion for creating a shared chunk is based on whether the probability of attributing the terms of the shared chunk to a record increases or decreases after the creation of the chunk.

Assume for example that we want to create a shared chunk between clusters C and C' on terms $a_1, \ldots, a_n$. Moreover, let $s(a_1), \ldots, s(a_n)$ be the supports of $a_1, \ldots, a_n$ in C and $s'(a_1), \ldots, s'(a_n)$ the respective supports for C'. We create a greater cluster only if:

$$\frac{s(a_1) + \cdots + s(a_n) + s'(a_1) + \cdots + s'(a_n)}{|C| + |C'|} \geq \frac{2 \times n}{|C| + |C'|} \tag{5}$$

Prerequisite for creating a shared chunk on $a_1, \ldots, a_n$ is that the support of each term in the greater cluster is greater than k and that the shared chunk is k^m-anonymous.

The right part of the inequality expresses the probability of attributing one of $a_1, \ldots, a_n$ to the records of C and C' if no shared chunk is created. For example, from Figure 1 we infer that ruby and ikea appear at least 2 times in 8 records. If they are placed in a shared chunk, as in Figure 3 we can learn that they appear 4 times each in the 8 records, and in addition they appear thrice together.

Due to space limitations we describe only the basic idea behind our criterion for creating shared chunks. The case for creating a shared chunk between two greater clusters is more complicated, but follows the same line of reasoning. Finally, we have some additional weight in Inequality 5 that favors the creation of a shared chunk if it contains multiple terms, because of the information it reveals about the correlation of the terms. We sketch the basic steps of algorithm for creating the greater clusters and the shared chunks:

1. Sort all clusters according to their term chunk contents.
2. Merge adjacent pairs of clusters and greater clusters to new greater clusters (if there is information gain). The terms that are used in the creation of shared chunks are removed from the term chunks.
3. Sort the resulting clusters and greater clusters according to their term chunk. For greater clusters we do the sorting based on the union of the term chunks of their clusters.
4. If some change took place, go back to Step 2.

The algorithm takes as input the anonymized dataset produced by Algorithms 4 and 5 and produces a dataset which contains: a) all the initial chunks b) the new

shared chunks and c) the updated term chunks, which miss the terms that have been moved to the new shared chunks.

4.4 Handling different abstraction levels

Anonymizing the data in different abstraction levels can be performed by our technique for the simple k^m-anonymity. The only difference here is that the input to the algorithm is not the original data but the data *translated* to the abstraction terms $l_1, \ldots, l_n$. Respectively, the anonymized database has to be translated back to the original data before being published. The translation of the input substitutes every term $i \in L_j$ by l_j in all records and removes duplicate values, since we assume that an attacker has no information about the multiplicity of each term. The translation of the output data is a bit more complex. Each subrecord of the first $n-1$ buckets has to be re-linked to the original record, and all the abstract terms of the subrecord have to be replaced with all the original terms they represent from the original record. To facilitate this mapping we mark each subrecord with the original record-id, so that we can always know which record produced which subrecord (of course this id is not published). For the abstract terms of the term bucket things are simpler; each of them is replaced by the set of all original terms it represents and that appear in any of the records that comprise the cluster.

5 Experimental Evaluation

We have conducted an experimental evaluation by applying our algorithms on both real and synthetic datasets. In the following we detail the experimental setup, the datasets, the factors we used to evaluate the anonymization procedure and we comment on the results. All our implementations were done in C++. The experiments were carried out on a Intel Xeon E5504, running Linux, with g++ 4.3.2.

dataset	$\|D\|$	$\|T\|$	max $\|r\|$	avg $\|r\|$
BMS-POS	515,597	1,657	164	6.5
BMS-WebView-1	59,602	497	267	2.5
BMS-WebView-2	77,512	3,340	161	5.0
AOL	81364	324,386	40,38	483

Figure 6: The real datasets ($|r|$ stands for record size)

5.1 Experimental Setup

Evaluation metrics. We trace the performance of our algorithms with respect to the following: *(a)* the tKd and bD information loss metrics, *(b)* the percentage of terms $tlost$ that have support more than k in D and are placed in the term chunk and *(c)* the total execution time (since our implementations are in main memory

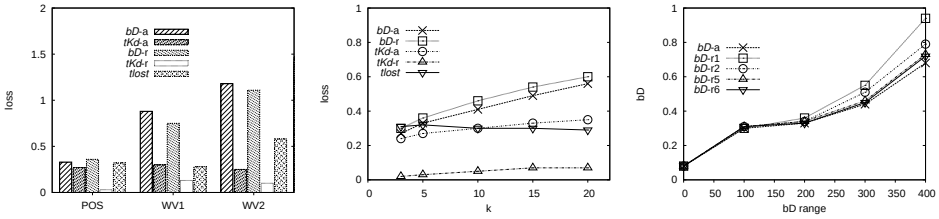


Figure 7: Behavior of Information loss

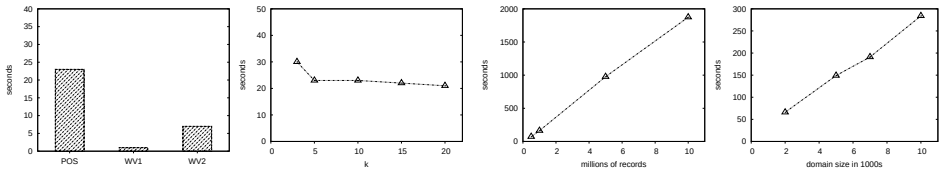


Figure 8: Performance on real datasets Figure 9: Performance on synthetic data

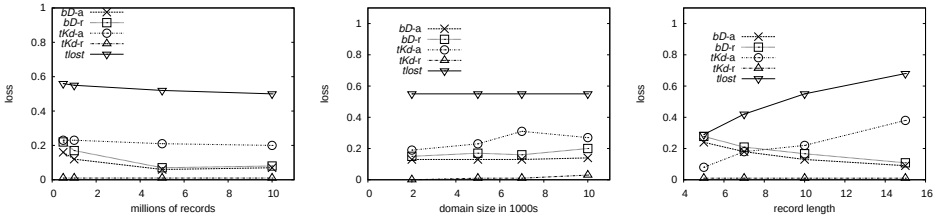


Figure 10: Behavior of Information loss on synthetic data

this is basically CPU time). We measure tKd and bD both in the anonymized data (reported as tKd-a and bD-a in figures) and in the reconstructed datasets (reported as tKd-r and bD-r in figures). In each anonymization, apart from the anonymized dataset, we have created a reconstructed dataset by randomly assigning chunk elements to records as we explained in Section 3.4. For this randomly reconstructed dataset, we measured its tKd and bD and we report it in all experiments. Moreover, we perform experiments to see how tKd and bD are affected when we measure them using the average support of itemsets in more than one reconstructed dataset. We report the average bD for all combinations we are examining, but we have avoided calculating bD on the whole domain because averaging the results of all 2-sized combinations is not very informative in cases of skewed distributions and large domains. The majority of combinations would be rare or would not exist at all in the original data and they would dominate the result. Since tKd already traces the effect of anonymization on frequent itemsets, we choose to use bD to understand what is happening with less frequent but not utterly rare or non-existent combinations. To this end, we ordered the domain of each dataset by descending term support and chose a small range of consecutive terms to trace their bD. After some testing we chose the 200th-220th most frequent terms and traced the bD of their combinations. tKd is measured for the 1000 most frequent itemsets.

Evaluation parameters. We investigate the behavior of our algorithms by varying the following parameters: *(a)* k, *(b)* the size of the dataset, *(c)* the size of the dataset's domain, *(d)* the size of the records, *(e)* the terms we use to calculate bD and *(f)* the number of reconstructed datasets we use to calculate bD and tKd. We do not present a detailed evaluation for m, because in all the available datasets its effect for values $m > 2$ is negligible. The explanation for this is that clusters that gather records with frequent terms are k^m-anonymous for any m and clusters that gather infrequent items are already partitioned enough for $m = 2$ to have chunks that are k^m-anonymous again for any m. The in-between cases seem very few to significantly affect the result. Finally, we explore the impact of abstract k^m-anonymity by creating different classifications of terms with fixed but different class sizes ($cSize$).

Datasets. To evaluate our proposal in a realistic setting, we have used datasets from several application areas. We used the 3 datasets, introduced in [13]: *BMS-POS*, *BMS-WebView1* and *BMS-WebView2*. Dataset *BMS-POS* is a transaction log from several years of sales of an electronics retailer. Each record represents the products bought by a customer in a single transaction. The *BMS-WebView-1* and *BMS-WebView-2* datasets contain click-stream data from two e-commerce web sites, collected over a period of several months.

The only available $\mathcal{WQL}$ dataset we know of is that of AOL data. For our anonymization purposes, we consider only the queries posed by each user (each query is one term) and we group them to one record per user, ignoring any further information. Unfortunately, the AOL data have too many unique terms. We believe this is due to the fact that many users do not use it as their main search engine (the most popular queries were google and google.com). From the 650k users in the AOL data only around 80k have asked at least one query that appears 3 or more times in the dataset, and we have used in our experiments only the records that refer to these users. Even in this case we have around 320k different terms most of them being unique. Because of the huge amount of unique or low support terms it is inevitable that many terms will end up in the term chunk. To demonstrate alternative privacy solutions we anonymized the AOL data based on abstract k^m-anonymity. The classes of similar terms were created artificially by grouping together terms with similar supports.

To generate synthetic datasets we used the IBM Quest Market-Basket Synthetic Data Generator [14]. Unless explicitly stated otherwise, the default characteristics for the synthetic datasets are $1M$ records, 5000 term domain and 10 average record length.

5.2 Experiments

In Figures 7a-7c we study the information loss caused by the anonymization procedure on real datasets. In Figure 7b we see how the information loss behaves when we increase k by setting it to 3,5,10,15 and 20. We see that we lose accuracy lin-

early to k in estimating relatively infrequent combinations as reported by bD. The support of the combinations traced by bD in *BMS-POS* does not exceed 40 (with the dataset having 500k records). Estimating which are the most frequent itemsets is also negatively affected by the increase of k as reported both by tKd-a and tKd-r, but the effect is limited. $tlost$ remains rather stable since two competitive factors affect it; on one hand, the threshold for sending a term to the term chunk is greater, on the other hand the cluster size is greater too (since we set it always as a multiple of k) so the support of each term in a single cluster can grow larger. An interesting point here is the measuring of tKd-r and bD-r on the reconstructed datasets. tKd-r takes values very close to 0, meaning that the top-1000 frequent itemsets in a reconstructed dataset and in the original are basically the same. Moreover, tKd-r is a lot small than tKd-a, which is expected since tKd-a is measured by taking account only the content of each record chunk, without taking into account the fact that two or more record chunks co-exist in the same cluster. bD-r also follows closely bD-a demonstrating that the supports of the terms tracked by bD in a random reconstruction do not diversify significantly, compared to our estimation about average support.

The experiments of Figures 7 and 8 are all performed on *BMS-POS* with $k = 5$, $m = 2$ except when explicitly stated otherwise.

To further investigate the divergence of supports in different reconstructed datasets we performed the following experiment: we created 10 random reconstructions of the anonymized *BMS-POS* dataset and we traced bD and tKd taking this time into account the average supports of the itemsets in 2, 5 and 10 of the reconstructed datasets. The tKd-r is already quite close to 0 and it does not have any substantial benefit when we measure it by taking into account the average support of frequent itemsets in more than 1 reconstructed datasets, thus we omit it in Figure 7c. bD on the other hand benefits from the average support of the traced combinations. In the experiment reported in Figure 7c we measured the bD on the combinations of the 0-20, 100-120, 200-220, 300-320 and 400-420 most frequent terms in *BMS-POS*. In Figure 7c we report the results when using 1 reconstructed dataset (bD-r), when we use the average support from 2 reconstructed datasets (rbD-r2), from 5 (rbD-r2) and from 10 (rbD-r10). In the x-axis of Figure 7c we depict the frequency order of the terms, e.g. a point over 100 refers to the bD of the combinations of the 100th-120th most frequent terms in *BMS-POS*. We observe that when the terms are frequent the support of their combinations is reported accurately in any reconstructed dataset, so taking the average does not provide any benefit. As the combinations become less frequent, using more than one reconstructed datasets allows for more accurate estimations. Finally, in Figures 8a and 8b we measure the performance of our algorithm in terms of CPU time.

To investigate how characteristics of the dataset affect the anonymization procedure we used the IBM basket data generator to produce realist synthetic datasets. In Figure 10 we see how the information loss is affected when the dataset, domain and record size change. In Figure 9 we see how the performance of the algorithm is affected by the same factors. Since the anonymization is applied independently

on each cluster, the database size does not have a significant effect on the quality of the results. Moreover, since the distribution of terms is relatively skewed increasing the domain affects basically the tail of the distribution, thus it does not have a significant effect on frequent combinations of terms. On the other hand, both these factors affect the performance of the algorithm. Increasing the dataset size has an obvious effect on the performance, as it increases the number of clusters that have to be anonymized. The augmentation of the domain size also causes the increase of the average number of term combinations than must be checked for k^m-anonymity inside each cluster. The increase of the record length affects the quality of the anonymization in two different ways. Since the number of records in the dataset is fixed, the support of all terms increases as the records become larger. This means that the terms 200-220 traced by bD become more frequent, thus bD is reduced as the size of the records increases. On the other hand, it becomes harder to bring similar records together in clusters as the size of the records increases, thus it becomes hard for the size of the chunks to follow the increase of the record size. This explains the increase of the tKd and the $tlost$ metrics. Finally, the record length does not have a significant impact on the algorithms performance in practice (although it affects the worst case) thus we omit it from the Figures. This is again due to the fact that chunks inside the cluster do not grow significantly as the size of the record grows.

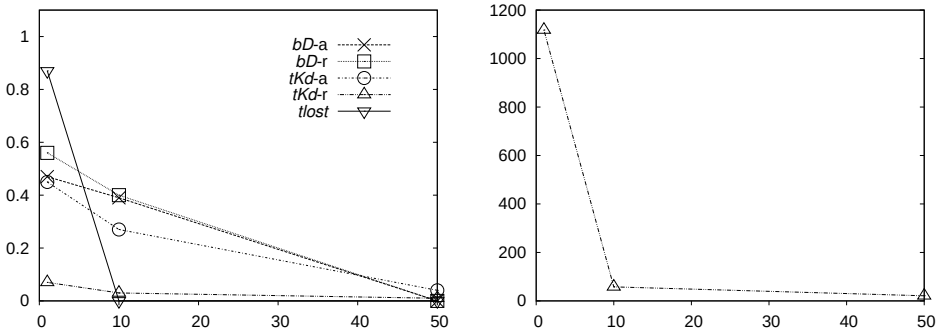


Figure 11: Abstract k^m-anonymity on the AOL data

Anonymizing the AOL dataset introduces significant information loss due its infrequent terms. To improve the quality of the result we experimented in Figure 11 with *abstract* k^m-anonymity. The x-axis of Figures 11a and 11b represents the class size ($cSize$) in each classification. For $cSize = 1$ we have the simple k^m-anonymity. We test simple and abstract k^m-anonymity with 3 different classifications for the terms, setting $cSize = 1, 10, 50, 100$. Figure 11a depicts the changes to the information loss in each case. When anonymizing according to simple k^m-anonymity, the information loss is considerable even for $k = 5$. The most frequent itemsets might be detectable as indicated by tKd, but less frequent combinations cannot be estimated accurately as implied by the high value of bD. Moreover,

many terms end up at the term chunk as shown by $tloss$. The picture changes for abstract k^m-anonymity, since, even for $cSize = 10$, all information loss metrics improve drastically. In Figure 11b we see the performance of the anonymization for each classification. The speed of the procedure is significantly enhanced by the reduction of the domain size, since the algorithm has less combinations to examine.

The experiments in both real and synthetic datasets demonstrated that our anonymization algorithm is resilient to information loss and it is able to preserve important information for a variety of different dataset characteristics and anonymization parameters. An important result is that randomly reconstructed datasets, which can be analyzed by exactly the same methods as the original ones, preserve accurately the statistical properties of the original. Finally, the experiments show that our algorithm is able to scale efficiently when the dataset, domain and record sizes increase.

6 Related Work

Privacy preservation in data publishing has attracted considerable attention due to the need of several organizations to share their data without revealing information that can be traced to real person or legal entities.

Privacy preservation in relational data. Privacy preservation was first studied in the relational context. In [15, 8] the authors introduce k-anonymity and use generalization and suppression as their two basic tools for anonymizing a dataset. [16] proved that optimal k-anonymity for multidimensional QI is NP-hard, under both the generalization and suppression models. For the latter, they proposed an approximate algorithm that minimizes the number of suppressed values; the approximation bound is $O(k \cdot logk)$. [17] improved this bound to $O(k)$, while [18] further reduced it to $O(\log k)$. *Incognito* [19] and *Mondrian* [20] guarantee k-anonymity for a relation table by transforming the original data using global (full-domain) and local recoding respectively. In [2] the authors provide another local recoding approach that shows superior performance to the global recoding approach of *Incognito*. A different approach is taken in [21], where the authors propose to use *natural* domain generalization hierarchies (as opposed to user-defined ones) to reduce information loss.

The aforementioned approaches protect the published data from identity disclosure i.e., they do not allow an attacker to associate a specific record to a specific person. Still, they do not prevent an attacker from linking some sensitive value to a person. To address this problem the concept of ℓ-diversity [22] was introduced. [23, 24, 25] present various methods to solve the ℓ-diversity problem efficiently. *Anatomy* [23] lies closer to our work in the sense that it does not generalize or suppress the data, but instead it disassociates them by publishing them separately. Still, the anonymization approach is restricted to relational data with a clear distinction between sensitive values and quasi identifiers.

Privacy in sparse multidimensional data. The first works addressing privacy preservation in the context of the sparse multidimensional data were related to frequent itemset minining. [26] considered a dataset D of transactions, each of which contains a number of items. Let S be a set of association rules that can be generated by the dataset, and $S' \subset S$ be a set of association rules that should be hidden. The data transformation relies on adding or suppressing items in the original data and requires the background knowledge of S'. [27] investigates an important danger in publishing anonymized association rules, namely *inference channels*. Assume a rule $a_1a_2a_3 \Rightarrow a_4$ (a_i's are items) with support $80 \gg k$ and confidence 98.7%. By definition, we can calculate the support of itemset $a_1a_2a_3$ as $80/0.987 \simeq 81$, therefore we infer that $a_1a_2a_3\neg a_4$ appears in $81 - 80 = 1$ transactions. If the attacker has *negative* background knowledge, i.e., he knows that a person is associated with a_1, a_2 and not a_3 the privacy of related person is compromised. Considering attackers with negative knowledge is outside the scope of our paper.

[28] provides an efficient algorithm for k-anonymity in sparse multidimensional data. The authors introduce a top-down local recoding method, named *Partition* which creates equivalence classes of k identical records. [1] introduce the k^m anonymity guaranty, which is used and extended in this paper. The authors provide algorithms for anonymizing the data that, unlike our approach, are based on generalization, employing both local and global recoding. [29] introduce a similar notion of a conditional privacy guaranty, named (h, k, p)-coherence that protects both from identity and attribute leaks. Still, the anonymization methods are based solely on suppression and require the a-priori identification of sensitive values. Although suppression leads to the desired results it introduces significant information loss, whose nature is different that the one introduced by our method. In a different setting, [30] studied multirelational k-anonymity, which can be translated to a problem similar to the one studied here, but the anonymization procedure still relies on generalization.

[7] extends [23] to provide ℓ-diversity for transactional datasets with a large number of items per transaction, but they do not depart from the anonymization framework of [23]; they still have a separate set of quasi identifiers and sensitive values. The basic idea of [7] is to create equivalence classes where the quasi identifiers are published separately from the sensitive values and their supports. This setting allows for a simpler solution that ours. [4] provides a more elaborate ℓ-diversity guaranty for sparse multidimensional data, termed ρ-uncertainty, where sensitive items can act as quasi identifiers too. Still, unlike our approach they employ generalization and suppression in order to anonymize the data.

Privacy in web search query logs. The publication of AOL data and the privacy breaches that followed it [6] have attracted significant attention to the anonymization of web search query logs, although the AOL dataset is the only one, to the best of our knowledge, that has been made publicly available. Protecting privacy in the AOL dataset is hard due to the huge number of different terms and their unknown

semantics, so few solutions have been proposed. Several works have investigated the privacy vulnerabilities of the dataset [10, 31] but few propose some anonymization methods. [3] proposes an anonymization method that guarantees differential privacy by means of suppression. The authors provide a solid privacy framework, but unlike our approach the proposed anonymization procedure completely hides all terms that are infrequent, which constitute the majority of terms in AOL data. [32], on the other hand, uses a form of generalization, termed micro-aggregation to provide k-anonymity for web search query logs. Moreover, they take into account the structural information that stems from the different sessions of the users to provide a more complete protection. Still, they have not evaluated their proposal experimentally. Finally, a very recent work [33] proposes a method for protecting user privacy in query logs by adding terms to the dataset, introducing again a information loss of different nature than the one caused by disassociation.

Recently there has been a great deal of work in privacy preservation in social networks. The work that is most closely related to the setting of sparse multidimensional data that we study here is the one that models the social networks as graphs. Publishing information from a social network would lead to the revelation of a graph that connects each individual to his friends or other persons he socially interacts. [34] describes a series of different attack models that rely on identifying subgraphs in the published social network graph. [35] proposes a k-anonymity guaranty that protects against vertex identification from attackers who have as background knowledge the vertex degree, i.e., each vertex in the anonymized data has the same degree with other $k-1$ vertices. [36] proposes k-isomorphism as a form of extended k-anonymity for graph data that takes into account structural information. [37] anonymize graphs by generalizing graph labels to lists of possible different labels. An interesting approach for privacy preservation is take in [38] where privacy risks are examined from the user's viewpoint. The authors provide a framework for estimating the privacy risk posed by the users' information sharing activities. Privacy from the user perspective is also investigate in [39], where the authors propose a framework that helps individual users to protect their privacy in social networks.

Our work lies closer to [7, 23] in the sense that it does not suppress or generalize the data but instead it severs the links between values attributed to the same entity. Unlike [7, 23] we do not have a separation between sensitive values and quasi identifiers, thus our problem is a lot more complicated and their anonymization algorithms cannot be applied. Our basic privacy guaranty comes from [1], but we follow a completely different path with respect to the data transformation and the type of data utility we are are preserving.

7 Conclusions

In this paper we provided a novel anonymization method for sparse multidimensional data. Our method guarantees a known property, k^m-anonymity, for the pub-

lished dataset, but with a novel data transformation, termed *disassociation*. Instead of eliminating identifying information by not publishing many original terms, either by suppressing or generalizing them, we partition the records so that existence of certain terms in a record is obscured. This transformation introduces a different type of information loss from existing methods, making our approach a valuable alternative when the original terms are important. Furthermore, we complement our methods by proposing a weaker form of k^m-anonymity, termed *abstract* k^m-anonymity, which protects against attackers who do not have exact terms as background knowledge, but instead they know only the thematic areas of the terms that appear in a user record.

We show that our anonymization method is a practical solution by proposing and evaluating an algorithm that transforms the original dataset to an anonymous one, with low computational requirements. We evaluate the algorithm against real datasets and present detailed results. In the future we aim at adjusting our algorithm to parallel computation in order to improve its scalability. Additionally, we plan to investigate how the quality of the published dataset can be improved by sharing some knowledge about safe combinations of terms among the anonymization processes of each cluster.

Bibliography

[1] M. Terrovitis, N. Mamoulis, and P. Kalnis, "Privacy-preserving Anonymization of Set-valued Data," *PVLDB*, vol. 1, no. 1, 2008.

[2] J. Li, R. C.-W. Wong, A. W.-C. Fu, and J. Pei, "Anonymization by local recoding in data with attribute hierarchical taxonomies," *IEEE TKDE*, vol. 20, no. 9, pp. 1181–1194, 2008.

[3] A. Korolova, K. Kenthapadi, N. Mishra, and A. Ntoulas, "Releasing search queries and clicks privately," in *WWW*, 2009.

[4] J. Cao, P. Karras, C. Raissi, and K.-L. Tan, "ρ-uncertainty: Inference-Proof Transaction Anonymization," *PVLDB*, 2010.

[5] M. Arrington, "Aol proudly releases massive amounts of private data," 2006.

[6] M. Barbaro and T. Zeller, "A face is exposed for aol searcher no. 4417749," New York Times, 2006.

[7] G. Ghinita, Y. Tao, and P. Kalnis, "On the Anonymization of Sparse High-Dimensional Data," in *ICDE*, 2008.

[8] L. Sweeney, "k-Anonymity: A Model for Protecting Privacy," *IJUFKS*, vol. 10, no. 5, 2002.

[9] "Reference has been removed due to double blind constraints."

[10] E. Adar, "User 4xxxxx9: Anonymizing query logs," in *Workshop on Query Log Analysis at the 16th World Wide Web Conference*, 2007.

[11] K. Wang, C. Xu, and B. Liu, "Clustering transactions using large items," in *CIKM*, 1999.

[12] D. Gunopulos, R. Khardon, H. Mannila, S. Saluja, H. Toivonen, and R. S. Sharm, "Discovering all most specific sentences," *ACM Trans. Database Syst.*, vol. 28, no. 2, pp. 140–174, 2003.

[13] Z. Zheng, R. Kohavi, and L. Mason, "Real world performance of association rule algorithms," in *KDD*, 2001, pp. 401–406.

[14] http://www.almaden.ibm.com/cs/quest/syndata.html.

[15] P. Samarati, "Protecting Respondents' Identities in Microdata Release," *IEEE TKDE*, vol. 13, no. 6, pp. 1010–1027, 2001.

[16] A. Meyerson and R. Williams, "On the Complexity of Optimal K-anonymity," in *PODS*, 2004, pp. 223–228.

[17] G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu, "Approximation Algorithms for k-Anonymity," *Journal of Privacy Technology*, 2005.

[18] H. Park and K. Shim, "Approximate algorithms for k-anonymity," in *SIGMOD*, 2007, pp. 67–78.

[19] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan, "Incognito: Efficient Full-domain k-Anonymity," in *SIGMOD*, 2005, pp. 49–60.

[20] ——, "Mondrian Multidimensional k-Anonymity," in *ICDE*, 2006.

[21] M. E. Nergiz and C. Clifton, "Thoughts on k-anonymization," *Data and Knowledge Engineering*, vol. 63, no. 3, pp. 622–645, 2007.

[22] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam, "l-Diversity: Privacy Beyond k-Anonymity." in *ICDE*, 2006.

[23] X. Xiao and Y. Tao, "Anatomy: simple and effective privacy preservation," in *VLDB*, 2006, pp. 139–150.

[24] Q. Zhang, N. Koudas, D. Srivastava, and T. Yu, "Aggregate Query Answering on Anonymized Tables," in *ICDE*, 2007, pp. 116–125.

[25] G. Ghinita, P. Karras, P. Kalnis, and N. Mamoulis, "Fast Data Anonymization with Low Information Loss," in *VLDB*, 2007.

[26] V. S. Verykios, A. K. Elmagarmid, E. Bertino, Y. Saygin, and E. Dasseni, "Association Rule Hiding," *TKDE*, vol. 16, no. 4, 2004.

[27] M. Atzori, F. Bonchi, F. Giannotti, and D. Pedreschi, "Anonymity Preserving Pattern Discovery," *VLDBJ*, vol. 17, no. 4, 2008.

[28] Y. He and J. F. Naughton, "Anonymization of set-valued data via top-down, local generalization," *PVLDB*, vol. 2, no. 1, 2009.

[29] Y. Xu, K. Wang, A. W.-C. Fu, and P. S. Yu, "Anonymizing transaction databases for publication," in *KDD*, 2008, pp. 767–775.

[30] M. Nergiz, C. Clifton, and A. Nergiz, "Multirelational k-anonymity," in *ICDE*, 2007, pp. 1417 – 1421.

[31] R. Jones, R. Kumar, B. Pang, and A. Tomkins, "Vanity fair: privacy in querylog bundles," in *CIKM*, 2008.

[32] G. Navarro-Arribas and V. Torra, "Tree-based microaggregation for the anonymization of search logs," in *WI-IAT*, 2009.

[33] H. Pang, X. Ding, and X. Xiao, "Embellishing text search queries to protect user privacy," *PVLDB*, vol. 3, no. 1, 2010.

[34] L. Backstrom, C. Dwork, and J. Kleinberg, "Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography," in *WWW*, 2007.

[35] K. Liu and E. Terzi, "Towards identity anonymization on graphs," in *SIGMOD*, 2008, pp. 93–106.

[36] J. Cheng, A. W.-c. Fu, and J. Liu, "K-isomorphism: privacy preserving network publication against structural attacks," in *SIGMOD*, 2010.

[37] S. Bhagat, G. Cormode, B. Krishnamurthy, and D. Srivastava, "Class-based graph anonymization for social network data," in *VLDB*, 2009.

[38] K. Liu and E. Terzi, "A framework for computing the privacy scores of users in online social networks," *Data Mining, IEEE International Conference on*, vol. 0, pp. 288–297, 2009.

[39] L. Fang and K. LeFevre, "Privacy wizards for social networking sites," in *WWW '10*, 2010, pp. 351–360.