Anonymization of Set-Valued Data via Top-Down, Local Generalization

Yeye He
University of Wisconsin-Madison
1210 W. Dayton St.
Madison, WI
heyeye@cs.wisc.edu

Jeffrey F. Naughton
University of Wisconsin-Madison
1210 W. Dayton St.
Madison, WI
naughton@cs.wisc.edu

ABSTRACT

Set-valued data, in which a set of values are associated with an individual, is common in databases ranging from market basket data, to medical databases of patients' symptoms and behaviors, to query engine search logs. Anonymizing this data is important if we are to reconcile the conflicting demands arising from the desire to release the data for study and the desire to protect the privacy of individuals represented in the data. Unfortunately, the bulk of existing anonymization techniques, which were developed for scenarios in which each individual is associated with only one sensitive value, are not well-suited for set-valued data. In this paper we propose a top-down, partition-based approach to anonymizing set-valued data that scales linearly with the input size and scores well on an information-loss data quality metric. We further note that our technique can be applied to anonymize the infamous AOL query logs, and discuss the merits and challenges in anonymizing query logs using our approach.

1. INTRODUCTION

In this paper we propose a new algorithm for anonymizing set-valued data. By "set-valued data" we mean data in which a logical record has the form ($personid$, $\{item_1$, $item_2$, ..., $item_n\}$). Note that such a record could be normalized into a person record and a set of related item records in order to be stored in a relational database — the important fact is that each person is associated with a set of items, not the details of how the data is stored. The canonical example of set-valued data is transactional data, where $\{item_1$, $item_2$, ..., $item_n\}$ represents the set of items that the person with id $personid$ purchased. Anonymization of this kind of data differs from the problem studied by most previous work, in which the implicit assumption is that each individual is associated with only a single sensitive value (e.g., their medical diagnosis.) Following terminology from prior work, we will refer to the two types of data as "set-valued data" and "relational data." Unfortunately, the techniques explored for anonymizing relational data do not apply to the set-valued case.

The problem of anonymizing set-valued data can arise in two different scenarios. In the first, one retains the quasi-identifier vs. sensitive value distinction of classical database data anonymization, and develops techniques to make it difficult for adversaries to exploit these quasi-identifiers to tie sensitive values to individuals [13, 25, 40, 41]. In the second scenario, there are no quasi-identifiers, any item of the sets could be sensitive, and the items of the sets themselves are exploited to tie sets of items to individuals. This is the scenario we consider. To the best of our knowledge, the only previously published work to address this scenario is the pioneering work in [32], which presented a bottom-up global-recoding algorithm exploiting the *a priori* property that has been used so effectively in association rule mining. In this paper we present an alternative algorithm, and compare and contrast our new algorithm with this prior work.

When considering the anonymization of set-valued data, one question that immediately arises is what anonymization criteria should be used. In [32], the authors proposed a new model called k^m-anonymity. The basic idea in k^m-anonymity is that for any transaction in the data set, for any subset of m items in that transaction, there are at least $k-1$ other transactions with the same m items. The algorithm presented in [32] uses generalization of the set elements to achieve this anonymization. The intuition for this model is that it helps prevent privacy breaches arising from an adversary discovering m of the items in an individual's transaction.

For example, an adversary may observe that Alice has purchased "milk," "beer" and "diapers." The adversary could do this perhaps by seeing the top of her (physical) shopping cart, or visiting Alice's house and noting the presence of these items. Now, armed with this information, the adversary can examine the released transactional data, and find all transactions containing these m items (milk, beer, and diapers in this example). This way the adversary might be able to associate a small set of transactions with the individual. If there is only one such transaction, then the adversary knows this transaction belongs to the individual, and furthermore, the adversary now knows the entire transaction, which may include sensitive items such as various prescription drugs. k^m anonymity provides some protection against this kind of attack. For example, if the data set were anonymized using 10^3 anonymity ($k = 10$, $m = 3$), then there would be at least nine transactions other than Alices's containing milk, beer, and diapers.

VLDB '09, August 24-28, 2009, Lyon, France

This definition of anonymity highlights the fact that for set-valued data, the set of sensitive values themselves can be used to identify individuals. While k^m anonymity mitigates this kind of attack, there are scenarios in which it is not sufficient. For example, in some cases it may not be possible to determine in advance a bound on how many items in a transaction an adversary can "see," in which case it is impossible to pick a safe value for m. Another scenario arises when some of the items in the set can be used to exclude certain transactions from being associated with an individual, such that the adversary may be able to tie a small number of, perhaps even less than k, transactions to certain individuals.

As an example of this second scenario, an attacker may know that typically only people over 65 purchase some items, while only people in certain geographic areas purchase others. Returning to our previous example, the adversary, knowing Alice has purchased "milk," "beer," and "diapers," may discover that two transactions contain these three items. Now suppose that Alice is a college student living in Texas; then if one of the transactions contains "snow tires" and "Centrum silver" in addition to "milk," "beer" and "diapers," it is unlikely to be Alice's. This "general public knowledge" attack increases the possibility that transactions can be linked to their owners and makes the task of protecting an individual's privacy harder. In this example, k^m privacy with $k = 2$ and $m = 3$ will not provide protection against this attack.

To protect against the general public knowledge attack, we propose the use of k-anonymity, instead of k^m-anonymity. k-anonymity simply means that for any transaction, there are at least $k-1$ other identical transactions. As will be illustrated in Section 3.1, that if a database is k-anonymous by our definition, it is k^m-anonymous for any value of m. However, setting m to anything less than m_{max}, where m_{max} is the maximum length of any transaction in the data set, results in a weaker anonymization than k-anonymity.

Because k^m-anonymity approximates k-anonymity with increasing m, the algorithm proposed in [32] for k^m-anonymity can be used to approach k-anonymity by setting $m = m_{max}$ in anonymizing a data set. Unfortunately, while that algorithm performs well when m is small, it does not scale well as m grows.

In addition to proposing k^m anonymity, the work in [32] presented the key insight that to prevent attacks on set-valued data, one can generalize some or all of the items in the set using a generalization hierarchy. For example, their algorithm might replace "wine" or "beer" in a transaction with the generalized value "alcohol." We also adopt the approach of generalizing the items in the sets. However, we propose a new algorithm, which we call "partition," that anonymizes by recursively partitioning similar set-valued transactions into groups in a top-down manner.

There are two primary criteria by which one can evaluate anonymization algorithms — their efficiency as measured by their running time, and their precision as measured by the amount of information loss through the anonymization process. Our experiments show that our algorithm is more efficient and achieves a lower degree of information loss than the algorithm presented in [32] run with $m = m_{max}$. Perhaps surprisingly, it turns out that our top-down local recoding is faster and more precise than the algorithm from [32] even when that algorithm is allowed to run with m set to values much smaller than m_{max}. This means that our algorithm generates better anonymizations and finds them more quickly even when it is enforcing a stricter privacy guarantee than the previous approach.

The greater efficiency of our anonymization algorithm enables us to consider anonymizing large real-world data sets, which makes it possible to consider applying our algorithm to the problem of anonymizing query search logs. To do so, we simply view a query as a set of keywords, then treat each keyword as an item and the union of the keywords in the collection of queries issued by the user as that user's transaction; or, treat the set of keywords in each query as an item, and the collection of queries by the user as a transaction. Although there has been some previous work, most notably [3], on protecting user's privacy when releasing their query logs, this previous work treats anonymization as an ad-hoc privacy problem, using techniques such as "drop any rare queries from the log" or "split an individual's queries among two virtual individuals."

Releasing search engine query logs without violating individuals' privacy is a practical problem and is of real world importance. There was a high-profile AOL query log release incident [1, 2] in 2006, when search histories of AOL users were naively anonymized and released. Unfortunately, multiple instances of real world privacy breaches were reported. Most notably, reporters from the *New York Times* [7] successfully identified one user in the AOL log as a 62 year old woman living in Georgia. This quickly led to a public outcry over privacy concerns, and AOL subsequently called the release a mistake.

While our algorithm's efficiency and precision does indeed enable us to anonymize the search logs with lower information loss than the techniques presented in prior work [3], as detailed in Section 6, our results on that data set are not as encouraging as our results on the traditional market basket data sets. We suspect that ultimately anonymizing such query logs will require additional new techniques beyond generalization, and regard our contribution in this direction as highlighting the promise and limitations of applying generalization to query log anonymization.

The rest of the paper is organized as follows. Section 2 describes related work on anonymization of transactional database. Section 3 formalizes the privacy model used for set-valued data. Section 4 introduces our top-down partition based anonymization. Section 5 presents an experimental evaluation of our algorithm and compares it with existing state-of-art. Section 6 extends this technique to query log anonymization. Section 7 concludes this paper.

2. RELATED WORK

In this section we describe related work that is not covered elsewhere in the paper.

Preserving privacy in transactional databases has been acknowledged as an important problem in the privacy literature. The data mining community has focused on hiding sensitive rules generated from transactional databases. In [6], the authors address this problem by altering the database to hide a given set of sensitive rules. This, however, does not protect individuals' privacy and requires prior knowledge of the rules and mining model used. The work in [33] proposes publishing only rules instead of the underlying data. While this approach will be effective in some situations, in general publishing rules rather than anonymized

data precludes the kind of arbitrary analysis one can do in a query session over a data set.

Accordingly, recently there has been a change of research focus from hiding rules to anonymizing and releasing transactional set-valued data, with individuals' privacy in mind. The pioneering work in [13] uses a sparse matrix representation of transactional databases, and applies matrix permutation techniques to capture correlations in the underlying data and to facilitate the formation of anonymized groups. Sensitive items are then randomized within groups to achieve anonymity. As we hinted in the introduction, this works when one can *a priori* partition attributes into quasi-identifiers (which are never sensitive) and sensitive values (which are never used to identify individuals.) By contrast, we are working on a different variant of the problem, where there are no explicit quasi-identifiers, and any of the elements appearing in the sets have the potential of being considered sensitive by some owner.

Along the line of [13], there is a pair of recent papers [40, 41] proposing a more sophisticated privacy criterion (h, k, p)-coherence to anonymize set-valued data. This novel privacy criterion ensures that for any p item combination that is non-sensitive, there are at least k transactions in the database containing these items, and within which at most h percent of transactions contain some sensitive item. It uses the parameter p to model the adversary's prior knowledge, which offers flexibility in anonymization based on the power of the adversary. (h, k, p)-coherence also has the advantage of incorporating a kind of diversity (of the sort originally introduced in the l-diversity criteria proposed in [23]) in the resulting anonymization. Our k-anonymity approach does not address diversity; augmenting our k-anonymity over set-valued data with no sensitive/quasi-identifiers item distinction appears to be a somewhat subtle issue and one that warrants future work. Similar to [13], however, this attractive new model is also under the assumption that an *a priori* classification of sensitive and quasi-identifier items exists. If applied directly to the problem we are attacking where no such distinction exists, (h, k, p)-coherence may have negative implications. In particular, given that any item can be both sensitive and quasi-identifier without a sensitive vs. quasi-identifier distinction, (h, k, p)-coherence will prohibit the discovery of any association rules with high confidence and fewer than p items on the left side, since such rules with confidence greater than h violate the condition specified by h.

The work in [32] is most similar to ours in not assuming an *a priori* distinction between quasi-identifier and sensitive items. As we mentioned in the introduction, they propose a k^m-anonymity model, and generalize items in a bottom-up enumerative manner, using the *a priori* principle [4], to achieve anonymity.

3. k-ANONYMITY IN SET-VALUED DATA

We first introduce our privacy model for anonymizing set-valued data. This is similar to the k-anonymity model widely used for relational data, but different from the k^m-anonymity model used in previous work on set-valued anonymization [32].

3.1 The Privacy Model

Let $I = \{I_1, I_2, ..., I_{|I|}\}$ be the set of items from which the elements of the sets are drawn, and let $D = \{t_1, t_2, ..., t_{|D|}\}$ be a transactional database over I, where each transaction $t_i \in D$ is a non-empty subset of I.

Definition 1. Equivalence class in transactional database
A transactional database D consists of a multiset of transactions. An *equivalence class* for D is the set of all transactions with identical sets of items S, $S \subseteq I$.

Definition 2. k-anonymity in set-valued data
A transactional database D is *k-anonymous* if every transaction in D occurs at least k times, or, equivalently, the size of each equivalence class in D is at least k.

Intuitively, a transactional database is k-anonymous if each transaction is identical to at least $k-1$ others. As we mentioned in the introduction, this model is different from the k^m-anonymity model previously proposed in [32], which states that given any m or fewer items chosen from any transaction, there are at least $k-1$ other transactions containing the same set of m items.

With these definitions, we can return to the question of relating these two models. We note first that k^m-anonymity only protects individuals' privacy when the adversary knows m or fewer items, whereas k-anonymity, with the absence of the parameter m, requires no limit on the number of items the adversary can know. In general, the smaller the m in k^m-anonymity, the weaker privacy k^m-anonymity provides. Note, though, that even when $m = m_{max}$, where m_{max} is the maximum length of transaction, k^m-anonymity would only approximate the level of protection offered by k-anonymity. Second, k^m-anonymity does not prevent against other attacks in which an adversary equipped with public knowledge can associate fewer than k transactions with an individual. In comparison, the k-anonymity model always ensures that a transaction is indistinguishable from a set of $k-1$ other transactions.

To summarize, we note in the following proposition that k-anonymity subsumes the previously proposed k^m-anonymity.

Proposition 1. Every database D that satisfies k-anonymity also satisfies k^m-anonymity for all m. However, there exists a database D such that for any m, D satisfies k^m-anonymity but not k-anonymity.

It is intuitive that a k-anonymous database D is always k^m-anonymous. Suppose we know that D is k-anonymous. Any n-item set, where $n \leq m$, is either contained in some transaction t in D or not contained in any transaction in D. In the first case, given the k-anonymity property of D, the equivalence class of t is at least of size k, which ensures that this n-itemset has a support of at least k. In the latter case, since no transaction in D contains this n-itemset, the support is simply 0. In both cases, D is k^m-anonymous.

It is also easy to come up with a counterexample where a database D is k^m-anonymous for any m but not k-anonymous. We will demonstrate this in Example 1.

Definition 3. k-anonymization of transactional database
A transactional database D' with some instances of items generalized from the original database D using an item generalization hierarchy is *a k-anonymization of D* if the derived D' is k-anonymous.

Logically, k-anonymization in our setting is the process of generalizing some specific items into more generic super-categories. For market basket data, there is typically a natural generalization hierarchy over the domain of purchased

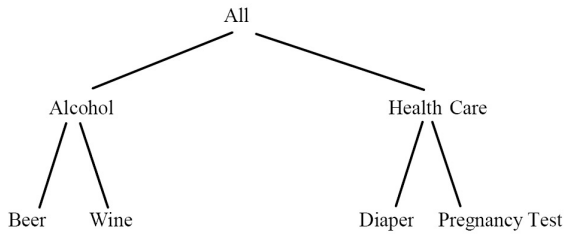


Figure 1: Item Generalization Hierarchy Example.

items. "Beer" and "wine," for instance, can be generalized to "alcohol." A sample hierarchy is illustrated in Figure 1. We assume that there is a root node, "*ALL*," that covers every possible item in the domain. This implies that as long as there are at least k transactions in the database, there is always a trivial k-anonymization, which simply generalizes everything to the root item "*ALL*." This anonymization, however, is trivial in that it only reveals the number of transactions in the database, with virtually no other information about the transactions in the original database.

Example 1. A 2-Anonymous Transactional Database Consider a transactional database with the domain generalization hierarchy from Figure 1. The original database consists of four transactions, as shown in Table 1a. Note that the "owner" column is not part of the transactional database, nor will it be published; it is included only for ease of illustration in our example.

Owner	TID	Items Purchased
Alice	T_1	{Beer, Diapers}
Bob	T_2	{Wine, Diapers, Pregnancy Test}
Chris	T_3	{Beer, Wine, Pregnancy Test}
Dan	T_4	{Beer, Wine, Diapers, Pregnancy Test}

(a) Original Database

Owner	TID	Items Purchased
Alice	T_1	{Alcohol, Health Care}
Bob	T_2	{Alcohol, Health Care}
Chris	T_3	{Beer, Wine, Health Care}
Dan	T_4	{Beer, Wine, Health Care}

(b) Generalized 2-anonymous Database

Table 1: Transactional Database Example

The original database, as presented in Table 1a, is not 2-anonymous according to our definition. Table 1b gives a possible 2-anonymization.

Note that the original dataset in Table 1a actually satisfies 2^2-anonymity in the k^m-anonymity model, and no generalization is necessary under that privacy model. Specifically, both k and m are 2, meaning any 1 or 2-itemset should have a support of either 0 or at least 2. This means that when the adversary knows that Dan bought "beer" and "diapers" he or she may be able to link Dan to T_1 and T_4 in Table 1a, but will not able to uniquely determine which transaction belongs to Dan. This provides some protection for Dan's privacy given the assumption that the adversary never knows more than 2 items. Suppose, however, that the adversary later learns that Dan bought "wine" also; then he or she can uniquely link T_4 to Dan, and thus be able to deduce that Dan also purchased a "pregnancy test."

This breach of privacy will not happen in our k-anonymity model. Even if the adversary knows that Dan bought "beer," "wine" and "diapers," he or she cannot determine which of T_3 and T_4 in Table 1b actually belongs to Dan. This intuitively demonstrates the point in Proposition 1 that k-anonymity subsumes k^m-anonymity.

To further illustrate the second part of Proposition 1 which states that there exists a database that satisfies k^m-anonymity for any m but not k-anonymity, let us assume there is an additional transaction T_5 in Table 1a which is identical to T_4 and consists of {Beer, Wine, Diapers, Pregnancy Test}. It is easy to see that this new database is 2^m-anonymous for any m given the existence of T_4 and T_5, but not 2-anonymous due to T_1, T_2 and T_3.

We now consider some subtle aspects of the guarantees provided by k^m-anonymity and k-anonymity. At first sight, the 2^2-anonymous database in Table 1a may appear more diverse than its 2-anonymous counterpart in Table 1b, since no two transactions in the former are identical. However, this is misleading; even though the 2-anonymous database in Table 1b looks more uniform, it actually reveals less information about the underlying transactions, because the transactions are identical only because some additional items have been generalized.

One possible concern of using a stronger privacy model is the quality of the resulting data and the potential loss of data utility, as it is reasonable to expect that more generalization/information loss may be necessary to meet the requirement of a stronger privacy model. As we saw in the previous example, when adopting the stronger 2-anonymity model we had to generalize some instances of items, while the original data already satisfied 2^2-anonymity. But as we will see in Section 5, in experiments over real-world data sets, our approach, while satisfying a stronger anonymization criterion, actually achieves lower information loss in most cases.

The key reason for this is that our approach performs a variant of "local recoding" as defined in the anonymization taxonomy proposed in [19]. In [19], local recoding referred to the generalization of quasi-identifiers in traditional data anonymization. Here, we apply the term to generalizing items in a set-valued data set using a hierarchy; that is, with local recoding, instances of the same item in different transactions may be generalized to different levels in the hierarchy. In Table 1b, for instance, "Beer" is generalized to "Alcohol" in T_1 but kept as is in T_3 and T_4. By contrast, with global recoding, when an item is generalized to its parent, not only are instances of this item in all transactions are generalized, so are any other child items under the same parent. One might think that for global recoding it is sufficient to merely generalize all instances of a data item and to leave remaining sibling items as is; but a moment's thought shows that this is not sufficient. For example, if we generalize "Beer" to "Alcohol" but leave "Wine" as is, then the generalization of "Beer" to "Alcohol" will serve no purpose for anonymization since there is a one-to-one correspondence between the original "Beer" and the new "Alcohol." This is why the algorithm in [32] chooses to recode all siblings whenever one sibling is generalized.

3.2 Quality Metrics

As in anonymizing relational data, set-valued data anonymization incurs information loss when a detailed item is generalized to its more generic super-category. The goal of anonymization in general is to find a transformation of the original data that satisfies a privacy model while minimizing the information loss and maximizing the utility of the resulting data. Thus a metric is necessary to measure the quality of the anonymized data.

Various metrics have been proposed in the literature. Metrics that are relevant in our setting include the *Discernability Metric* [8], and the *Normalized Certainty Penalty* [39]. The *Discernability Metric* measures the size of equivalence classes, or partitions, in the anonymization. This is not directly applicable to the k^m-anonymity model, to which we want to compare our approach, as there is no concept of equivalence class in k^m-anonymity. Thus we use the *Normalized Certainty Penalty (NCP)* information loss metric throughout this work. We note that this metric was also used in previous work [32] to measure the effectiveness of the algorithm presented in that paper.

We describe the *Normalized Certainty Penalty* for items in a generalization hierarchy, as defined in [32]. Let p be an item or its generalization. NCP is defined as

$$NCP_(p) = \begin{cases} 0, & |u_p| = 1; \\ |u_p|/|I|, & otherwise. \end{cases}$$

where u_p is the node in the hierarchy tree corresponding to p, and $|u_p|$ is the total number of leaf nodes under that node. Intuitively, the first equation states that when the item is not generalized and published as is, there is no information loss. The second equation states that the information loss for a generalized item is the number of leaves it covers divided by total number of leaves in the hierarchy. The maximum information loss is 1, when an item is generalized to the root and thus covers the entire domain.

Using T_1 of Table 1, for example, when "beer" is generalized to "alcohol," the information loss is $|u_p|/|I| = 2/4 = 1/2$. Similarly, an information loss of $1/2$ is incurred when "diapers" is generalized to "health care." "Beer" in T_3 and T_4 is not generalized, so the information loss there is 0.

Let t be a transaction in the database D, p be an item in transaction t, and C_t be the count of items in transaction t. Then the total information loss of a generalized transactional database D is defined as follows:

$$NCP(D) = \frac{\sum_{t \in D} \sum_{p \in t} NCP(p)}{\sum_{t \in D} C_t}$$

Thus the overall information loss of an anonymized database is the weighted average of the information loss of all instances of items, with a possible range from 0 to 1. Using our running example, the information loss of the four transactions when anonymized from Table 1a to 1b is: $(1/2+1/2)$ for T_1, $(1/2+1/2+1/2)$ for T_2, $(1/2)$ for T_3, and $(1/2+1/2)$ for T_4. So the total information loss of the database is $(8/2)/12 = 1/3$.

4. PARTITION BASED ANONYMIZATION

Inspired by the divide-and-conquer techniques commonly used in addressing problems involving data with multiple dimensions, such as kd-trees [12], R-tree [14] and Grid File [26], our basic idea is to partition in a top-down manner, by recursively separating set-valued data into groups where the data in each partition share a generalized representation. A similar approach was used in the Mondrian [20] anonymization algorithm. One key difference in our setting is that the generalization hierarchy has to be used in deciding which transactions are similar and should be grouped together. We propose the following top-down greedy partition algorithm.

4.1 A Greedy Partitioning Algorithm

We begin with a high-level overview of our algorithm. It starts by generalizing all transactions to the root level in the hierarchy. This, as a starting point, always produces a trivial anonymization with one partition, as long as there are at least k transactions in the database, as all transactions share the same representation ("ALL") after being generalized to the root. From this starting point, we pass the initial partition to the following *Anonymize* routine, which splits the current partition into sub-partitions, and recursively invokes *Anonymize* on all resulting sub-partitions. The partitioning process terminates when no further split is possible.

Algorithm 1 Recursive partition of set-valued data

```
Anonymize (partition)
if (no further drill down possible for partition) then
  return and put partition in global list of returned
  partitions
else
  expandNode ← pick_node (partition)
  for each data in partition do
    resultPartitions ← distribute_data(data, expandNode)
  end for
  balance_partitions (resultPartitions)
  for each subPartition in resultPartitions do
    Anonymize(subPartition)
  end for
end if
```

We now present the algorithm in more detail. At any time during the algorithm, for each partition, logically there is a generalized representation, consisting of to which level in the hierarchy transactions in the partition have been generalized. This generalized representation for each partition can be recorded using a set of hierarchy nodes, which cuts the hierarchy tree into two portions. We call this generalized representation the "hierarchy cut" (a similar notion was also described in [32]). The initial partition, which has all transactions in it all generalized to "ALL," has the root item as its hierarchy cut. As partitions are split, the nodes in the hierarchy cut associated with each partition will be expanded, with more child-level nodes replacing the expanding parent node, resulting in a hierarchy cut with more lower level hierarchy nodes, giving a more detailed representation for all transactions in this partition.

Every time when a partition is to be split, we only expand one node in the hierarchy cut at a time, for all the transactions in the partition. So for each partition, there is a choice of which node to specialize. A generalized transaction ("food," "beverage"), for instance, can be expanded by drilling down on the node "food" to get ("cheese," "beverage"), or drilling down on the node "beverage" to generate ("food," "beer"). Each choice will send the transaction to a different bucket.

This choice of node expansion for each partition is de-

Table 2: Sample database and its anonymization

TID	Original Data	Local Recoding (2-anonymity)	Global Recoding (2^2-anonymity)	Global Recoding (2-anonymity)
T_1	$\{a_1\}$	$\{A\}$	$\{a_1\}$	$\{A\}$
T_2	$\{a_1, a_2\}$	$\{A\}$	$\{a_1, a_2\}$	$\{A\}$
T_3	$\{b_1, b_2\}$	$\{b_1, b_2\}$	$\{B\}$	$\{B\}$
T_4	$\{b_1, b_2\}$	$\{b_1, b_2\}$	$\{B\}$	$\{B\}$
T_5	$\{a_1, a_2, b_2\}$	$\{a_1, a_2, B\}$	$\{a_1, a_2, B\}$	$\{A, B\}$
T_6	$\{a_1, a_2, b_2\}$	$\{a_1, a_2, B\}$	$\{a_1, a_2, B\}$	$\{A, B\}$
T_7	$\{a_1, a_2, b_1, b_2\}$	$\{a_1, a_2, B\}$	$\{a_1, a_2, B\}$	$\{A, B\}$

termined using a greedy heuristic in the *pick_node* subroutine. Various heuristics are possible in deciding which node to expand given transactions in one partition. In what follows we will only present details of the best heuristic, *Max_Global_Info_Gain*, out of several we tested in subroutine *pick_node*.

Pick_node first iterates through each non-leaf (thus expandable) node in the hierarchy cut, computing the total information gain for all the transactions in the partition should this node be expanded, where the information gain is defined as the difference between the information loss of the generalizations before and after parent node expansion. The node expansion which maximizes information gain for the whole partition will be picked and used. We will see in Section 5 that this simple heuristic performs well in practice.

The chosen node is then expanded, distributing every transaction in the partition into a bucket (sub-partition). Transactions still sharing the same representation will land into the same bucket. At the end of data distribution phase, the *balance_partitions* sub-routine handles buckets with fewer than k transactions by placing data from those buckets in a leftover partition. If necessary, transactions producing the least information gain from buckets with over k transactions will be re-distributed to the leftover partition as well to make sure the leftover partition has at least k transactions. The leftover partition will then be associated with the original hierarchy cut, as the expansion of the current node was not successful in producing sub-partitions for this leftover data.

Each sub-partition represents the specialization from the parent node to a set of different child nodes in the hierarchy cut. Thus, for each sub-partition, a new hierarchy cut with the parent node expanded and replaced by its specialization will be generated and associated with that sub-partition. This hierarchy cut, along with the transactions in the sub-partition, will be passed to recursive invocations of *Anonymize*, until leaf level is reached or no further drill down for the partition can be made without violating k-anonymity.

Example 2. A top down partitioning using Anonymize

We use the example in Table 2 to illustrate step by step how the *Anonymize* routine works in partitioning data top-down into 2-anonymous groups.

We start by generalizing all transactions to the root level of the hierarchy, generating an initial partition $P_{\{ALL\}}$ with all seven transactions T_1-T_7 represented by the root "*ALL*." The hierarchy cut associated with this partition is simply the root level of the hierarchy, {"*ALL*"}.

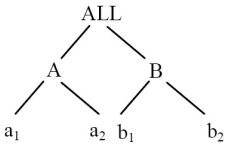


Figure 2: Domain generalization hierarchy for the sample database

The *Anonymize* routine is called for the first time to split the initial partition. The hierarchy cut of the initial partition has only one node, the root "*ALL*," in it, so there is only one way to drill down, which is $ALL \rightarrow \{A, B\}$. This drill down generates three possible buckets (sub-partitions), namely $P_{\{A\}}$, $P_{\{B\}}$ and $P_{\{A,B\}}$, where the set of items in the curly brackets of each bucket denotes the generalized representation shared by all transactions in the partition. All seven transactions are then distributed into these three buckets, based on the leaf level items in their corresponding original un-generalized transactions. We send T_1 and T_2 to bucket $P_{\{A\}}$, T_3 and T_4 to bucket $P_{\{B\}}$, and T_5-T_7 to bucket $P_{\{A,B\}}$. All three buckets have at least 2 transactions, so now we have three sub-partitions $P_{\{A\}}$, $P_{\{B\}}$ and $P_{\{A,B\}}$.

Next we recursively invoke *Anonymize* on each of the three resulting sub-partitions. Sub-partition $P_{\{A\}}$ has the two transactions T_1 and T_2, and the hierarchy cut {A}. The *pick_node* subroutine again has only one choice for drill down, which is $\{A\} \rightarrow \{a_1, a_2\}$. This generates three buckets, $P_{\{a_1,a_2\}}$, $P_{\{a_1\}}$ and $P_{\{a_2\}}$. T_1 is sent to $P_{\{a_1\}}$ and T_2 to $P_{\{a_1,a_2\}}$. Neither bucket is 2-anonymous, so the split was unsuccessful. We roll back to the previous hierarchy cut {A} and re-distribute T_1 and T_2 to the leftover partition in the *balance_partitions* subroutine. Since there is no other partitionable node, the partition will be fixed with the generalized representation {A}.

The second sub-partition $P_{\{B\}}$ with T_3 and T_4 is then passed to *Anonymize*. The choice of node to drill down is node B, giving rise to three possible buckets $P_{\{b_1,b_2\}}$, $P_{\{b_1\}}$ and $P_{\{b_2\}}$. $P_{\{b_1,b_2\}}$ has T_1 and T_2 in it while the other two buckets are empty. Note that the resulting partition $P_{\{b_1,b_2\}}$ has already reached leaf level so we stop here.

The last sub-partition, $P_{\{A,B\}}$ has three transactions, T_5-T_7, with hierarchy cut $\{A, B\}$. By only drilling down one node at a time, there are two possible choices, namely $\{A\} \rightarrow \{a_1, a_2\}$, or $\{B\} \rightarrow \{b_1, b_2\}$. The *pick_node* subroutine makes that choice using a greedy information gain heuristic. Specifically, drilling down $\{A\} \rightarrow \{a_1, a_2\}$ will give T_5, T_6 and T_7 $(2 + 2 + 2) * 2/4/17 = 3/17$ information gain, while drilling down $\{B\} \rightarrow \{b_1, b_2\}$ generates $(1 + 1 + 2) * 2/4/17 = 2/17$ information gain. Thus node A is chosen to be expanded for this partition, and T_5, T_6 and T_7 are all sent into $P_{\{a_1,a_2,B\}}$, which becomes a new partition.

We proceed to the new partition $P_{\{a_1,a_2,B\}}$, where the only choice of node to expand is $\{B\} \rightarrow \{b_1, b_2\}$. This would distribute T_5 and T_6 to bucket $P_{\{a_1,a_2,b_2\}}$, and T_7 to $P_{\{a_1,a_2,b_1,b_2\}}$. Since bucket $P_{\{a_1,a_2,b_1,b_2\}}$ has only T_7 in it, T_7 will be sent to the leftover partition in the *balance_partitions* subroutine. Note that we have less than 2 transactions in the leftover partition, meaning transactions in other buckets with least information gain will be taken and re-distributed

to the leftover partition. In our case it could either be T_5 and T_6, but that leaves $P_{\{a_1,a_2,b_2\}}$ with only one transaction. In that case all three transactions are sent to the leftover partition. We roll back and stop with the generalized representation $P_{\{a_1,a_2,B\}}$ for T_5, T_6 and T_7.

The resulting generalized transactions are presented in the "local recoding" column of Table 2. Note that due to the local recoding nature of our algorithm, we can recode "a_1," "a_2" of T_1 and T_2 into "A," while keeping "a_1," "a_2" as is in other transactions. Similarly "b_1," "b_2" in T_5, T_6 and T_7 will be generalized but other instances of "b_1," "b_2" will be left intact.

It is interesting to see how this anonymization generated by our local recoding with heuristics compares with the optimal global recoding. The optimal global recoding using 2-anonymity is presented in the last column of Table 2. Note that for global recoding, if the $\{a_1, a_2\} \rightarrow \{A\}$ generalization is necessary in some transaction, all other transactions with item a_1 or a_2 have to be generalized. This can lead to over-generalization.

As we discussed earlier, the bottom-up generalization algorithm proposed in [32] uses the 2^2-anonymity, which is a weaker privacy model compared with 2-anonymity. In our case, as a result, the optimal anonymization under 2^2-anonymity is better than the optimal global anonymization for 2-anonymity, as we can see by comparing the last two columns in Table 2. Note the generalization $\{b_1, b_2\} \rightarrow \{B\}$ is necessary because the 2-item combination $\{a_2, b_1\}$ only has a support of 1, which is below the anonymity threshold 2.

It should be noted that this optimal global recoding under the weaker 2^2-anonymity model, however, is still inferior to the local recoding we produce using the top-down partition algorithm under the stronger 2-anonymity. As we will see in Section 5, this is the key reason why our top-down local recoding outperforms the previous state-of-art, even when a stronger anonymity criteria is adopted.

5. EXPERIMENTS

5.1 Experimental Setup

5.1.1 Evaluation metrics and Platform

We evaluate our approach in terms of anonymization efficiency and effectiveness. Specifically, we measure execution time and anonymized data quality to compare our top-down partition-based anonymization (Partition) with the bottom-up Aprori-based Anonymization (AA) from [32], both using our implementations in C++. We evaluated data quality using the *NCP* information loss metric defined in [32, 39] and described in Section 3.2. Execution times are reported from experiments on an Intel Pentium4 3GHz server with 2GB memory running Linux.

5.1.2 Dataset and parameters

For the market-basket data, we used three real-world datasets *BMS-WebView-1*, *BMS-WebView-2* and *BMS-POS* [42]. *BMS-POS* is a transaction log from several years of sales of an electronics retailer, while *BMS-WebView-1* and *BMS-WebView-2* contain several months worth of clickstream data from two e-commerce web sites. All three are widely used as benchmark datasets in the knowledge discovery community. Some information about the three datasets is listed in Table 3.

Table 3: Characteristics of the three benchmark datasets

dataset	# Trans.	# Distinct items	Max trans. size	Avg. trans. size
BMS-WebView-1	59,602	497	267	2.5
BMS-WebView-2	77,512	3,340	161	5.0
BMS-POS	515,597	1,657	164	6.5

We first give an overview of how our top-down Partition algorithm performs compared with the bottom-up AA algorithm from [32] with three datasets. We then drill down to the dataset *BMS-WebView-2*, varying different parameters to see how performance changes for these two approaches. We note that this choice of dataset *BMS-WebView-2* is due to the fact that *BMS-POS* is too big for AA to run in reasonable time ($>$ 10 hours), while *BMS-WebView-1* is perhaps too small to give reliable insight into the performance of the algorithms (around 8 seconds). We report results using the medium sized *BMS-WebView-2* in our detailed performance studies, but we did observe similar trends in other datasets.

For the parameters in the privacy model, unless specified otherwise, we fixed k in k-anonymity at 10 and m in k^m-anonymity at 4. Note that this imposes a stricter condition on Partition than AA. We also varied k and m and report what effects these parameters have on the performance of these two approaches.

One difficulty in applying generalization-based anonymization to these data sets is that they do not include generalization hierarchies. Accordingly, we followed the example from [32] and artificially constructed a hierarchy on the union of all items appearing anywhere in the transactions. The node fan-out f of the hierarchy specifies how many items are generalized from one level to its parent level in the hierarchy tree. Picking a uniform node fan-out 5, for instance, will result in a hierarchy of height of 5, 7 and 6 for *BMS-WebView-1*, *BMS-WebView-2* and *BMS-POS* respectively.

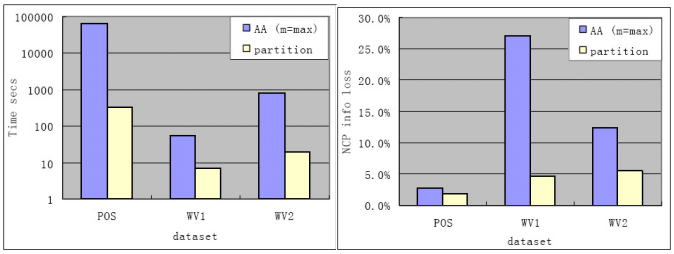
The exact choice of fan-out f, however, is somewhat tricky, as we realized later in our sensitivity analysis for f (Section 5.2.5) that compared with Partition, AA is much more sensitive to f (or alternatively, the shape of the hierarchy). Even though the previous work [32] used AA on the same data sets with $f = 5$, we observed that this is not the optimal choice for AA, which we are comparing with (we will see in Figure 7b, AA performs significantly better with $f = 4$ and $f = 6$ than $f = 5$ on *BMS-WebView-2*). To try to guard against bias from a lucky or unlucky choice for f, we ran experiments with $f = 4, 5$ and 6, and report the average of the three runs.

5.2 Experimental Results

5.2.1 Comparison of Partition and AA

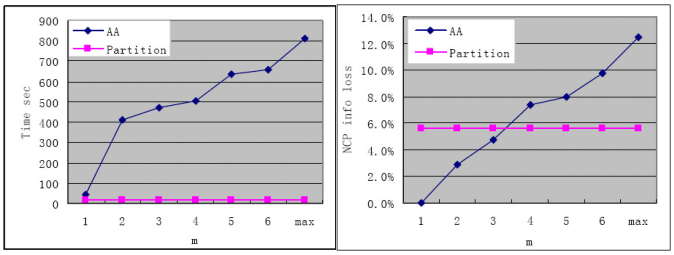
We first experiment to see how our Partition algorithm compares with AA. In this first experiment, we set m to m_{max} for AA, so that AA and Partition would have to satisfy the same privacy guarantees.

Figure 3 shows how Partition compares with AA on the three datasets. One can see from Figure 3a that Partition is faster than AA with $m = m_{max}$, ranging from 8 times faster



(a) Time comparison (b) Info loss comparison

Figure 3: Time and data quality of three real-world datasets



(a) Time comparison (b) Info loss comparison

Figure 4: Effect of m on execution time and data quality

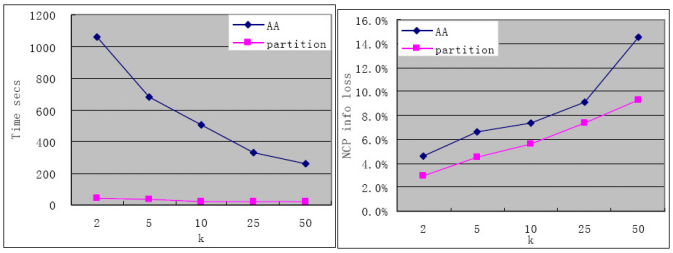
on the smallest dataset *BMS-WebView-1*, to more than 200 times on the biggest dataset *BMS-POS*. This is expected, as the computation cost of our approach grows linearly; while AA, although leveraging the *a priori* principle to prune the search space, is still exponential in m and $|I|$ in the worst case.

The quality of the resulting anonymization in Figure 3b shows that Partition also consistently outperforms AA on all three datasets, using the *NCP* metric. This is expected, because while AA uses "global recoding," our Partition algorithm adopts the more flexible "local recoding" scheme in anonymization, resulting in less generalization and thus higher data quality. Example 2 illustrates the flexibility of "local recoding" in a more concrete way.

5.2.2 *Sensitivity to Parameter* m

We note, however, that AA was not initially designed to be run with $m = m_{max}$; the idea was that AA could be run with some smaller value m', where $m' < m_{max}$, if one could really guarantee that no adversary will ever be able to associate an individual with more than m' items. In the remaining sections we will explore the performance of AA with smaller m. From this point on, we will only present results with the *BMS-WebView-2* data set, although we observed similar trends for other two datasets.

Figure 4 illustrates the effect of m in k^m-anonymity on the performance of AA. Since Partition uses k-anonymity and does not depend upon m, its performance stays constant in this experiment. We observe that setting m to smaller numbers does reduce the execution time of AA. Specifically, the time of AA in Figure 4a goes down from 809 seconds for $m = m_{max}$ to 32 seconds for $m = 1$. It is worth noting, however, that even when $m = 1$ (i.e., the adversary can only tie one item to a specific individual a priori), AA still runs slower than Partition.



(a) Time comparison (b) Info loss comparison

Figure 5: Effect of k on execution time and data quality

Figure 4b is a comparison of data quality. Setting m to smaller numbers in AA improves the quality of the anonymization as well. The information loss reduces from 12.4% for $m = m_{max}$ to 0.02% for $m = 1$. We observe that AA does provide a better information loss for $m < 4$, but it is worse for all $m \geq 4$.

5.2.3 *Sensitivity to Parameter* k

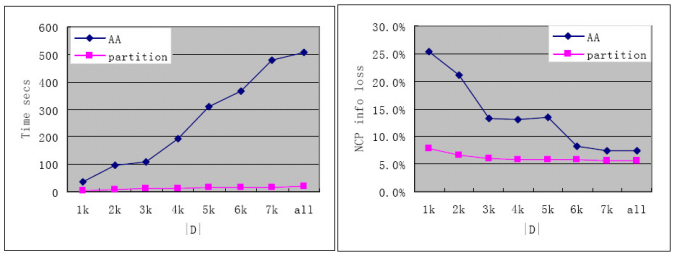
Figure 5 presents the effect of k on the performance of both approaches. The execution time of Partition in Figure 5a is consistently 10+ times faster than AA across all values of k. We observe that the execution time of AA drops quickly as k increases. The reason is that a larger k entails more generalization, and AA is very effective in leveraging generalization to reduce the size of the item domain and thus pruning the search space. The sharp increase of execution time for small k, however, also suggests AA's inefficiency in anonymization when less generalization is necessary. This is because when k is small, most item-combinations have a support of at least k, in which case not much generalization is necessary, rendering AA unable to reduce the exponential search space.

To illustrate this point, we ran both algorithms on the 10-anonymization of *BMS-WebView-2* produced by Partition. This is a trivial problem as the input data is already 10-anonymized and no generalization is necessary. While Partition detects this in 10 seconds, AA eventually runs out of virtual memory and aborts, unable to figure out that this data is already 10-anonymous and requires no generalization.

Figure 5b compares the information loss of the two approaches with increasing k. Partition consistently outperforms AA in all cases. Observe the sharp increase of information loss from $k = 25$ to $k = 50$ for AA. The explanation for this jump is that when k advances to large values, significant generalization becomes necessary, which translates to higher likelihood of over-generalization for AA, again due to global recoding.

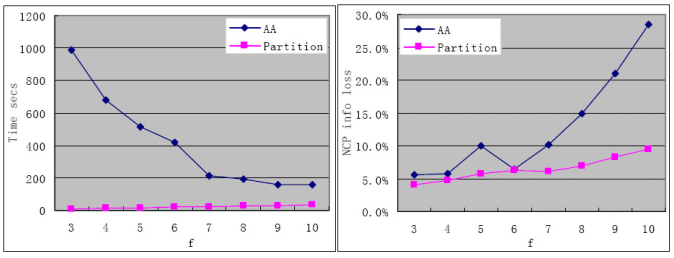
5.2.4 *Sensitivity to Parameter* $|D|$

Figure 6 shows how these two approaches scale as the number of transactions in the database grows. In Figure 6a we see that although both appear to be growing linearly, the time of AA grows much faster. Figure 6b is a comparison of information loss with increased volumes of data. For AA there is a big drop from 1k to 3k, suggesting its likelihood to over-generalize when data is "sparse," in which case again substantial generalization is necessary.



(a) Time comparison (b) Info loss comparison

Figure 6: Effect of $|D|$ on execution time and data quality



(a) Time comparison (b) Info loss comparison

Figure 7: Effect of f on execution time and data quality

5.2.5 *Sensitivity to Parameter f*

Figure 7 shows how the two approaches perform with varied node fan-out f in the balanced generalization hierarchy tree. It is interesting to note from Figure 7a that the efficiency of AA improves with larger f, while the efficiency of Partition erodes. This actually makes sense because with an increased f, the generalization tree tends to be shorter, and there are fewer levels of generalization from leaf node to the root. With fewer levels in the hierarchy tree, the search space for AA is smaller as there are fewer possible ways to generalize bottom-up, whereas for the top-down Partition, each time a partition is split, with a larger node fan-out, there are more possible sub-partitions for each split, resulting greater computational cost. This suggests that AA may be attractive in terms of execution efficiency for very wide hierarchy trees.

Unfortunately, this efficiency comes at the expense of information loss. In Figure 7b, we observe that information loss for AA is comparable to that of Partition when f is small, but increases to 28.4% at $f = 10$, whereas the information loss for Partition increases more slowly. We believe this sharp increase of information loss of AA is a direct consequence of an increased f. A larger f means more items will be generalized in a single step in AA; although this translates to more efficient execution, it also increases AA's likelihood of over-generalization.

Also note that the curve of information loss for AA has a surprising bump at $f = 5$, whereas the plot for Partition is smoother. Similar curves were observed in other fan-out sensitivity analyses we conducted. The reason for AA's variable behavior as a function of f lies in the shape of the hierarchy, and is ultimately due to AA's "global recoding" nature. In this particular case, we found that with $f = 4$ the generalizations produced by AA are mostly to the 4th level nodes in the hierarchy (from leaf and count upwards), which cover 256 (4^4) leaf nodes. Similarly with $f = 6$ AA frequently generalizes nodes to the 3rd level, which cover 216 (6^3) leaf nodes. Now this generalization is very close to what Partition produces (quantitatively, the size of item domain 3340 times Partition's around 6% information loss is about 200, which represents the average number of leaf nodes covered by the generic representation in Partition's anonymization).

The two approaches generally agree on the rough amount of generalization necessary to anonymize the data set and thus are very close in the information loss metric. But when $f = 5$, the "global recoding" AA could not generalize to the right amount, because when $f = 5$ there is no level in the hierarchy which covers about 200 leaves; instead it only has the 3rd level nodes, which cover 125 leaves, or the 4th level nodes, which cover 625 leaves. A study of the generalization produced by AA confirmed our speculation, as we found 3 instances of generalization to the 4th level (with 625 leaves.)

6. QUERY LOG ANONYMIZATION

We mentioned the applicability of set-valued anonymization techniques to privacy preserving query log publishing. In this section we discuss query log anonymization as an extended application of our top-down partition-based anonymization, and present preliminary experimental results obtained from anonymizing the infamous AOL query logs using our technique.

6.1 Background and the Problem

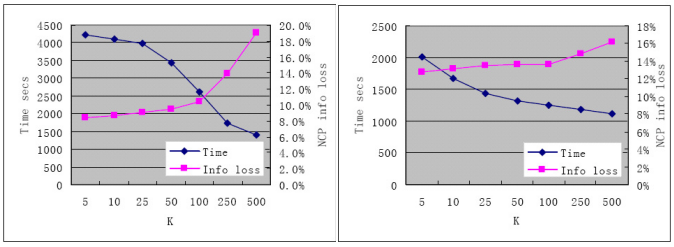
The AOL query log release incident, since it happened, has drawn a great deal of attention in the research community [3, 15, 17, 18, 27, 38]. Among this work, [3, 15, 18, 27, 38] specifically talk about privacy-preserving publishing of query logs. All of these papers illustrate the importance of anonymizing query logs, and the relative ineffectiveness of naive anonymization schemes such as token based hashing [18].

[3] is the only work that we are aware of that proposes specific measures to anonymize query logs. The authors in [3] propose two mechanisms, the first of which is to remove rare queries. This alone, however, will remove 97% of distinct queries (or 69% by query volume), which is a significant loss of information under any metric. The second step, which they call "split personality," groups queries of one user that belong to the same topic of interest together, and assigns queries on different topics issued by the same user to different fictional user ids. Assigning one user's queries to multiple fictional users, however, may limit the usability of the query log.

Instead of treating the query log anonymization as an ad-hoc problem, we address it as a set-valued anonymization problem. Specifically, we treat each search query as an item, and the set of queries issued by the same user as a transaction.

6.2 AOL Log Anonymization Experiments

This experiment is an attempt to anonymize AOL query log data so that it can be released safely without compromising any individual's privacy. The query log data, although quickly removed from AOL official website after its release, was mirrored on many sites and is still available. It has 37M instances of queries, issued by 650K users over the period of 3 months. The entire log has 10 chunks and is approximately



(a) Alphabetical Hierarchy (b) WordNet hierarchy

Figure 8: Partition performance on AOL query log data

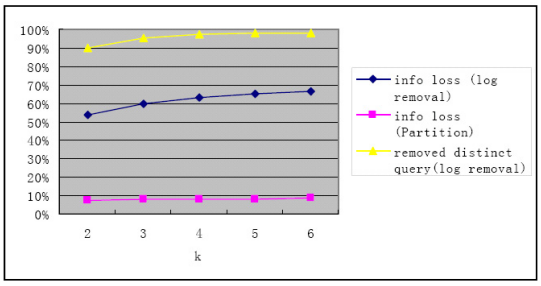


Figure 9: Quality comparison of Partition and log removal

2.2GB in size. The log is of the format:

{*AnonID, Query, QueryTime, ItemRank, ClickURL*}

where *AnonID* is the numerical ID anonymized from an actual AOL user ID, and *Query* is the query issued by the user. Although *QueryTime*, *ItemRank*, and *ClickURL* bear useful information for research purposes and may need to be anonymized before being released, we focused on anonymization of the first two fields, user ID and query text.

In our experiments we used the partition-based algorithm detailed in Section 4 to anonymize the AOL query log. To apply this approach, we need a generalization hierarchy. Unfortunately, query logs do not come with any such generalization hierarchy. The construction of a hierarchy for query logs by itself is a non-trivial problem. To tackle the absence of a generalization hierarchy, we built a quick and dirty, alphabetically-ordered generalization tree as well as a more sophisticated, semantically meaningful generalization tree derived from the WordNet dictionary [10], to assess the effectiveness of our generalization algorithm on the query log.

We pre-processed the logs by normalizing queries to lower case, with duplicate queries belonging to same user removed. In the first experiment, we built a random generalization hierarchy. We treated each query as an item, sorted all the queries in alphabetical order, and then built a balanced generalization tree on them. This hierarchy is rather arbitrary (generalizing alphabetically adjacent queries like "how to write a resume," "how to write a critique," "how to write a complaint" to one parent may make some sense, while generalizing "apple," "apple store" and "Applebee's" to the same parent does not). Nonetheless, even an arbitrary hierarchy allows some interpretation of the anonymized search logs (if only by looking at terms that are not generalized and seeing what terms are grouped in the arbitrary hierarchy.)

Figure 8a shows how partition based anonymization performs on this alphabetical hierarchy. We note that we were able to anonymize this large data set in a reasonable amount of time (about 70 minutes for $k = 5$). The information loss was significantly higher than that for the shopping basket data. This, we believe, is due to two properties of the log data. First, the domain size of the query log data (or the number of distinct queries) is around 10M, whereas the domain size of market-basket data is typically on the order of a couple of thousands. A much more diverse domain will lead to more sparsely distributed data, thus more generalization is necessary to ensure k-anonymity. The second possible factor is that transactions in the query log data are much longer than that of shopping basket data. *BMS-POS*, for instance, with an average transaction size of 6.5, already boasts the longest among the three benchmark shopping basket datasets. The number of queries issued by a user over a certain period of time, however, can easily be much larger than that; the average number of queries issued by a user is 24.7 for the AOL query log. This, we believe, also contributes to the larger information loss we see on the query logs.

Figure 9 shows how the quality of the resulting data compares between our generalization approach and the log removal approach presented in [3], again using the *NCP* information loss metric as defined in [32, 39] and described in Section 3.2. Clearly, the partition-based generalization approach outperforms the log removal approach, where all rare queries are removed. Given the diverse domain of queries that people can issue, a majority of queries have to be removed. Specifically, 97% of distinct queries (or 69% by query volume) are removed to ensure 3-anonymity.

Although the alphabetical generalization hierarchy we used was not very meaningful, it is interesting to see what queries were left un-generalized (thus still readable and semantically meaningful after anonymization). We note that quite a few popular queries were left untouched (110 such queries when we pick $k = 100$ in k-anonymity). Just to give a sense of what they are, these queries include "yahoo," "google," "mailbox," "mapquest," "myspace," "weather."

As a second step, we tried to build a more semantically meaningful hierarchy for the query logs. Building such a generalization hierarchy is an open research question and is by no means a trivial task. We attempted to build a crude hierarchy by using a semantic ontology derived from the WordNet dictionary [10]. According to the WordNet classification, nouns have a natural and deep generalization hierarchy (with every noun generalizes to the root node, which is "entity"). The hierarchy for verbs, by contrast, is short and bushy, while adjectives are grouped into clusters with similar meanings.

As a first cut of the problem, we treated each noun as an item, and the union of the nouns in the collection of queries issued by the user as the user's transaction, thus discarding all other words except nouns. In addition to the fact that verbs and adjectives were removed altogether, which discarded useful information, using words provided by WordNet is by no means a perfect approach. Many popular terms like "myspace," "mapquest" are absent from the dictionary and thus are not recognized and not processed. But we feel using WordNet suffices as an initial, simple example of a semantically meaningful hierarchy with which to experiment.

The performance of our partition-based anonymization using this WordNet hierarchy is presented in Figure 8b. Again it runs in reasonable time when anonymizing the large

query logs. Since now we have a meaningful generalization, we look into the resulting anonymized data to get a sense of what the anonymized data looks like. We see that some queries are reasonably anonymized. For example, with queries searching for a place in a state, the location is often recognized and generalized to either the state level, or to the "US" level. At the other extreme, many transactions with long and complex queries tend to be generalized to generic representations, which are not very informative. For example, we often find queries being generalized to something like "psychological state," "event," "process," or merely a "thing."

Overall, besides this anecdotal evidence that we may be generalizing too much to be meaningful in browsing the resulting anonymization, the information loss metric applied to these anonymized query logs also suggests a significantly higher information loss when anonymizing query logs than that obtained over market basket data. As we discussed earlier, this could be due to the much larger domain size and longer transaction size of query log data, compared with the market basket data.

7. CONCLUSION AND FUTURE WORK

We have proposed a top-down, partition-based technique to anonymize set-valued data. It heuristically partitions set-valued data by descending down a domain hierarchy, and in our experiments it is an order of magnitude more efficient than the current state-of-art. This partition based anonymization, by adopting a local recoding scheme, also turns out to preserve better data utility in most cases. We have illustrated how our set-valued anonymization techniques can be applied to real world data sets including retailer purchase transaction anonymization and search engine query log anonymization.

For search query logs, our algorithm is efficient enough to be applied to non-trivial real-world data, but the information loss results, while better with respect to information loss than previous work [3], are not as satisfactory as was the case when we applied our algorithm to market-basket data. It appears that anonymizing data containing large sets from large domains appears to be a very difficult problem that may require techniques beyond generalization.

In the end, all anonymization techniques, including our proposed k-anonymity for set-valued data, just serve to make privacy breaches more difficult, and none make them impossible. Just as the early work on k-anonymity in relational data sets led to follow-on work refining that model to provide stronger privacy protection, we hope that our work and the pioneering work in [13, 25, 32, 40, 41] will serve as a springboard for future work refining our anonymity models and algorithms.

In particular, the notion of k-anonymity on set-valued data, like k-anonymity on relational data, attempts to offer protection via "safety in numbers" without incorporating safeguards for diversity. This means, for example, that it is possible in an extreme case to have a data set in which more than k individuals have purchased exactly the same set of items, in which case no generalization will be performned, and it may be possible to determine exactly what is in some individual's set (with the only possible comfort being the knowledge that at least $k-1$ individuals also have exactly the same set.) Other techniques for anonymizing set-valued data have exploited the notion of sensitive vs. non-sensitive values or quasi-identifiers to prevent this, essentially requiring that for all equivalence classes of records that agree on their non-sensitive values or quasi-identifiers, the sensitive values satisfy some diversity criterion. It is not clear how this can directly be applied to the case we study, in which there is no sensitive vs. non-sensitive value distinction and no notion of quasi-identifier. Developing definitions of anonymity that guarantee some analog of "diversity" in this setting, and algorithms that perform anonymizations satisfying these definitions, is an interesting area for future work.

8. REFERENCES

[1] AOL search data scandal, http://en.wikipedia.org/wiki/aol_search_data_scandal.

[2] Chronicle of AOL search query log release incident, http://sifaka.cs.uiuc.edu/xshen/aol_querylog.html.

[3] E. Adar. User 4xxxxx9: Anonymizing query logs. In *Query Logs Workshop at WWW*, 2007.

[4] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. of VLDB*, 1994.

[5] R. Agrawal and R. Srikant. Privacy-preserving data minding. In *Proc. of ACM SIGMOD*, 2000.

[6] M. Atzori, F. Bonchi, F. Giannotti, and D. Pedreschi. Anonymity preserving pattern discovery. *VLDB Journal*, 2008.

[7] M. Barbaro and T. Zeller. A face is exposed for aol searcher no. 4417749. *New York Times*, August 9, 2006.

[8] R. Bayardo and R. Agrawal. Data privacy through optimal k-anonymization. In *Proc. of ICDE*, 2005.

[9] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *Proc. of KDD*, 2002.

[10] C. Fellbaum. *WordNet, an Electronic Lexical Database*. MIT Press, Cambridge MA, 1998.

[11] D. Frankowski, D. Cosley, S. Sen, L. Terveen, and J. Riedl. You are what you say: Privacy risks of public mentions. In *Proc. of SIGIR*, 2006.

[12] J. H. Friedman, J. H. Friedman, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software (TOMS)*, 1977.

[13] G. Ghinita, Y. Tao, and P. Kalnis. On the anonymization of sparse high-dimensional data. In *Proc.of ICDE*, 2008.

[14] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *Proc. of ACM SIGMOD*, 1984.

[15] R. Jones, R. Kumar, B. Pang, and A. Tomkins. "I know what you did last summer" - query logs and user privacy. In *Proc. of CIKM*, 2007.

[16] D. Kifer and J. Gehrke. Injecting utility into anonymized datasets. In *Proc of ACM SIGMOD*, 2006.

[17] A. Krause and E. Horvitz. Privacy, personalization, and the web: A utility-theoretic approach. In *Proc. of AAAI*, 2008.

[18] R. Kumar, J. Novak, B. Pang, and A. Tomkins. On anonymizing query logs via token-based hashing. In *WWW*, 2007.

[19] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Incognito: Efficient full-domain k-anonymity. In *Proc. of ACM SIGMOD*, 2005.

[20] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Mondrian multidimensional k-anonymity. In *Proc. of ICDE*, 2006.

[21] 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*, 2008.

[22] N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *Proc. of ICDE*, 2007.

[23] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. In *Proc. of ICDE*, 2006.

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

[25] M. Nergiz, C. Clifton, and A. Nergiz. Multirelational k-anonymity. In *Proc. of ICDE*, 2007.

[26] J. Nievergelt, H. Hinterberger, and K. C. Sevcik. The grid file: An adaptable, symmetric multikey file structure. *ACM Transactions on Database Systems*, 1984.

[27] B. Poblete, M. Spiliopoulou, and R. Baeza-Yates. Website privacy preservation for query log publishing. In *Proc. of the First International Workshop on on Privacy, Security, and Trust in KDD*, 2007.

[28] J. Quinlan. Induction of decision trees. In *Machine Learning*, 1986.

[29] S. Rizvi and J. Haritsa. Maintaining privacy in association rule mining. In *Proc. of VLDB*, 2002.

[30] R. Srikant and R. Agrawal. Mining generalized association rule. In *Proc. of VLDB*, 1995.

[31] L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570, 2002.

[32] M. Terrovitis, N. Mamoulis, and P. Kalnis. Privacy-preserving anonymization of set-valued data. In *Proc. of VLDB*, 2008.

[33] V. Verykios, A. Elmagarmid, E. bertino, Y. Saygin, and E. Dasseni. Association rule hiding. *IEEE TKDE*, 16:434–447, 2004.

[34] R. Wong, Y. Liu, J. Yin, Z. Huang, W. Fu, and J. Pei. (a, k)-anonymity: An enhanced k-anonymity model for privacy-preserving data publishing. In *Proc. of SIGKDD*, 2007.

[35] X. Xiao and Y. Tao. Anatomy: Simple and effective privacy preservation. In *Proc. of ACM SIGMOD*, 2007.

[36] X. Xiao and Y. Tao. m-invariance: Towards privacy preserving re-publication of dynamic datasets. In *Proc. of SIGMOD*, 2007.

[37] X. Xiao and Y. Tao. Dynamic anonymization: Accurate statistical analysis with privacy preservation. In *Proc. of SIGMOD*, 2008.

[38] L. Xiong and E. Agichtein. Towards privacy-preserving query log publishing. In *Query Logs Workshop at WWW*, 2007.

[39] J. Xu, W. Wang, J. Pei, X. Wang, B. Shi, and A. Fu. Utility-based anonymization using local recoding. In *Proc. of SIGKDD*, 2006.

[40] Y. Xu, B. C. M. Fung, K. Wang, A. W.-C. Fu, and J. Pei. Publishing sensitive transactions for itemset utility. In *ICDM*, 2008.

[41] Y. Xu, K. Wang, A. W. C. Fu, and P. S. Yu. Anonymizing transaction databases for publication. In *KDD*, 2008.

[42] Z. Zheng, R. Kohavi, and L. Mason. Real world performance of association rule algorithms. In *Proc. of KDD*, 2001.