CLUSTERING-BASED WEB QUERY LOG ANONYMIZATION

by

Amin Milani Fard
B.Sc., Ferdowsi University of Mashhad, 2008

a Thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science
in the School
of
Computing Science

SIMON FRASER UNIVERSITY
Fall 2010

APPROVAL

Name: Amin Milani Fard

Degree: Master of Science

Title of Thesis: Clustering-based Web Query Log Anonymization

Examining Committee: Dr. Arthur L. Liestman, Professor, Computing Science
Chair

Dr. Ke Wang, Professor, Computing Science
Senior Supervisor

Dr. Jian Pei, Associate Professor, Computing Science
Supervisor

Dr. Martin Ester, Professor, Computing Science
SFU Examiner

Date Approved: _____

Abstract

Web query logs data contain information which can be very useful in research or marketing, however, release of such data can seriously breach the privacy of search engine users. These privacy concerns go far beyond just the identifying information in a query such as name, address, and etc., which can refer to a particular individual. It has been shown that even non-identifying personal data can be combined with external publicly available information and pinpoint to an individual as this happened after AOL query logs release in 2006. In this work we model web query logs as unstructured transaction data and present a novel transaction anonymization technique based on clustering and generalization techniques to achieve the k-anonymity privacy. We conduct extensive experiments on the AOL query log data. Our results show that this method results in a higher data utility compared to the state of-the-art transaction anonymization methods.

Keywords: Query logs data, privacy-preserving data publishing, transaction data anonymization, item generalization

To my parents, loving wife, family and friends for their support.

Acknowledgments

Here, I would like to express my deep gratitude to my senior supervisor, Prof. Ke Wang for his profound influence on my research and studies. He introduced me to the field of Privacy-Preserving Data Publishing and Mining, and taught me a great deal of valuable research skills. This project and thesis would not have been completed without his guidance. I would also like to thank the members of my thesis supervisory and examining committee: Prof. Jian Pei, Prof. Martin Ester, and Prof. Arthur L. Liestman, for their patiently reading through my thesis, and providing valuable comments.

Thanks also go to all faculty, staff, and friends in the School of Computing Science at SFU for providing such a nice academic environment. My graduate studies would not have been interesting without them. I have enjoyed the time with the members in the Database and Data Mining Laboratory, and would like to special thank Junqiang Liu for his assistance in a part of my project implementations.

I also thank the School of Graduate Studies and the Faculty of Applied Science at Simon Fraser University, for scholarship funding that helped me to focus full time on my thesis. My research was also supported by a Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant.

Finally, I would like to thank my family for their love and support over the years. And last but not the least, I am indebted to my lovely wife, Hoda, for her understanding and encouragement during my research.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

Web search engines generally store information of the queries sent by search engine users known as query logs. These data provides a valuable resource for the purpose of improving ranking algorithms, query refinement, user modeling, fraud/abuse detection, language-based applications, and sharing data for academic research or commercial needs [16]. On the other hand, the release of such data can seriously breach the privacy of search engine users. This privacy concern goes far beyond just removing the identifying information such as name, address, phone number, and etc. from a query. In a similar case for relational databases, Sweeney [56] showed that even non-identifying personal data can be combined with publicly available information, such as census or voter registration databases, to pinpoint to an individual.

In 2006 the America Online (AOL) query logs data, over a period of three months, was released to the public [8] after all explicit identifiers of searchers have been removed. Shortly after that, the searcher No. 4417749 was traced back to the 62-year-old widow Thelma Arnold who lives in Lilburn. This was not only a scandal for the AOL, but also a reason for data publishers to become reluctant in providing researchers with public anonymized query logs [33].

Since the above mentioned event, an important research problem was opened on rendering web query log data in such a way that it is difficult to link a query to a specific individual while the data is still useful to data analysis. Several recent works start to examine this problem, with [39] and [2] from web community focusing on privacy attacks, and [34], [58], and [67] from the database community focusing on anonymization techniques. These works made good progresses in protecting data privacy, however, they still face a major challenge

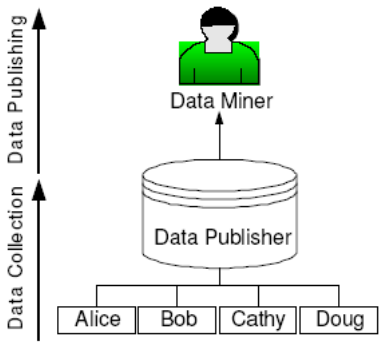

Figure 1.1: Data collection/publishing phases [27]

is reducing the significant information loss of the anonymized data.

1.1 Privacy-Preserving Data Publishing

The subject of this thesis falls into the field of *privacy-preserving data publishing* (*PPDP*) [27], which is different from *computer security* area that involves access control and authentication methods. The main issue in the area of *computer security* is to ensure the recipient of information has the authority to receive that information. While such protections can safeguard against direct disclosures, they do not address disclosures based on inferences that can be drawn from released data. The subject of *PPDP* is not much on whether the recipient can access to the information or not, but is more on what values will constitute the information the recipient will receive.

A typical scenario of data collection and publishing phases [27] is shown in Figure 1.1. The data publisher collects data from some individuals and releases them to a data miner or the public, called the data recipient, who will then apply many different data mining and analysis on them. These data mining tasks can vary from a simple count of the number of specific item to sophisticated cluster analysis, frequent pattern mining, or classification and prediction problems.

Detailed person-specific data, such as health information, often contains sensitive information about individuals. Nowadays that many of such datasets are publicly available for research purposes, there are more concern about confidentiality and privacy protection of individuals. Releasing original data can violate an individual privacy without consent that his/her identity can be disclosed.

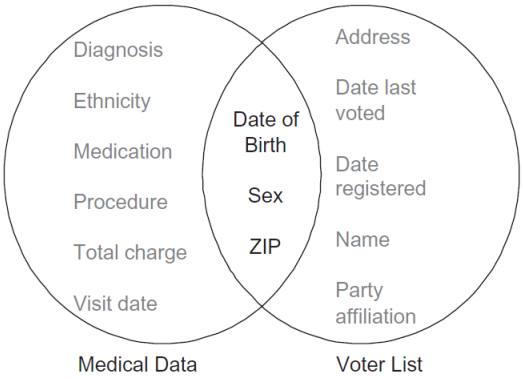

Figure 1.2: Re-identification by linking to external information [57]

The process of *anonymization* [17], [18] is a *PPDP* approach which hides the identity of individuals, and/or their sensitive information in case the sensitive data is needed for data analysis. One trivial approach for data anonymization is to remove explicit identifiers of record owners. However, Sweeney [57] showed that even non-identifying personal data can be combined with publicly available information, such as census or voter registration databases, to pinpoint to an individual.

An example is a table $T(Age, Gender, Zipcode, Disease)$ published by a hospital. Typically, the information on *Age*, *Gender* and *Zipcode* could be publicly available, say from a voter registration list. Although each of these attributes can not be used to uniquely identify an individual, however, through their combination a record owner or set of similar record owners can be identified. We call such attributes the *quasi-identifier* (QI) attributes [18]. Also we consider *Disease* as a *sensitive attribute* (SA).

The publication must prevent an adversary from inferring accurately the SA value of an individual. In case a patient has a rare combination on the QI attributes, his/her record and *Disease* can be identified with high probability. Figure 1.2 shows how a public voter list can be linked with a published medical database. Sweeney claimed that almost 87% of the U.S. population can be uniquely identified [57] as for the case of William Weld, former governor of the state of Massachusetts.

An attacker who plans for such linking attacks should not only know about the existence of the victims record in the released data, but also should know about the QI of the victim. For example, the attacker may observed that his neighbor was hospitalized in a hospital at which the patient records will be released. If he also knows about *age*, *gender* and *zipcode*,

he can launch a linking attacks.

Currently the privacy issue is considered under policies and agreements to restrict the types, uses, and storage of sensitive publishable data. This, however, has limitation of either excessive distortion of data or requiring a very high trust level that is often impractical. For example, privacy agreements cannot ensure that sensitive information will be always carefully stored and no third parties can access it [27]. Consequently researchers in *PPDP* area tries to design methods to guarantee a certain level of privacy while outsourcing the data.

Data publishers are either trustworthy or not [31]. If the data publisher is not trustworthy, he may identify sensitive information of individuals. In this case cryptographic approaches [68], anonymous communications [15], [36], or statistical methods [62], can be used to collect data from owners without revealing their identity. In this thesis, we assume that the data publisher is trustworthy and record owners are willing to provide their personal information to him.

In practice, we consider two important assumptions for data publishing scenarios [27]. Firstly, the data publisher is not necessarily a data miner and thus all data analysis is done by the data recipient. Even if he is a data miner, publishing data instead of publishing data mining results is much more useful and interesting because many other analysis can be done on such data. We also assume that the data publisher may not know about the specific data mining task performed by the recipient in advance because otherwise he could release a customized data set that preserves specific characteristics to improve the data utility. Therefore the published data should be potentially useful for many data analysis objectives. An example of such data release is the records of patients in California hospitals on the Web [14], regardless of its application.

Secondly, in *PPDP*, the data recipient could also be an attacker. In this case there is no difference between publishing data for public or for a specific trustworthy recipient. For example, the data may be published for a research center while not all its staff are trustworthy. This assumption makes the problems in *PPDP* much different from the problems in cryptography where only authorized recipients can access the data.

In this thesis, we only consider anonymization of a single release, however, in practice the same data may be published more than once and for different purposes. Some few works considered multiple release publishing [37], sequential release publishing [59], and continuous data publishing [12], [65], [28].

1.1.1 Attacks and Privacy Models

In a practical PPDP, we assume the attacker has limited background knowledge on the QI of a victim. The most common privacy threats are *record linkage*, *attribute linkage*, and *table linkage*, at which an attacker tries to link a record of an individual to a record in a published table, to a sensitive attribute in a published table, or to the published data table itself, respectively [27].

Record linkage occurs when the attacker knows the victims record is in the released table T and tries to identify such record in T. If some value on QI which matches victims QI, identifies a small number of records in the released table, the victim can be distinguished with high probability. In attribute linkage, we also assume that the attacker knows the victims record is in the released table T, and tries to identify the victims sensitive information in T. Note that, the attacker may not exactly distinguish the victims record, but could infer his/her SA values based on the set of SA values associated to the set of records matched with the victim's QI. If some SA values are much more frequent in such records, the attacker can infer the SA value with high probability.

Unlike the record linkage or attribute linkage, in table linkage the attacker does not know whether a victim exist in the released table T and thus tries to determine the presence/absence of the victims record in T. This is a privacy issue since knowing the presence of the victims record in a table containing sensitive information, such as table with a particular type of disease, could breach his/her privacy.

In some other privacy models, we do not concern about records, attributes, or tables that the attacker can link to an individual, but we concern about the change in the attacker's probabilistic belief on the SA value of a victim after seeing the published data. In such models, we try to ensure *uninformative principle* [45], which guarantees a small difference between the attacker's prior and posterior beliefs [27].

Bellow we explain some well-known approaches to prevent above mentioned attacks.

k-Anonymity

Samarati and Sweeney [55] presented the notion of k-anonymity as a solution to record linkage attacks. In a k-anonymous table, the QI of each record is exactly the same as at least k-1 other records, or equivalently each record is indistinguishable from at least k-1 other records with respect to the QI. This ensures that the probability of linking an individual to

a specific record based on QI is at most $1/k$.

In order to make records look the same with respect to the QI, generalization and suppression of values can be used which results in information loss. Many works tried to achieve k-anonymity while preserving an acceptable data distortion. One important assumption in k-anonymity model is that each record represents a "distinct" individual. This means that if several records in a table represent the same individual, a group of size k records may represent fewer than k person, and this may breach that person's privacy. An example is a patient information table in which each patient can have more than one records.

(X, Y)-Anonymity

To address the above mentioned problem in k-anonymity, (X, Y)-anonymity notion was proposed in [59], where X and Y are disjoint sets of attributes. This notion specifies that each value on set X is linked to at least k distinct values on set Y. Consider the patient records example above in which several records can represent one person. Let X to be the QI of such table and Y to be a *patientID*. by making the table (X, Y)-anonymous, each group with the same X values is linked to at least k distinct *patientID*s which ensures each group represents k distinct patients.

We can look at the k-anonymity as an special case for (X, Y)-anonymity, where X is the QI and Y is a key in table that uniquely identifies record owners. Also if each value on X describes a QI and Y represents the SA, we can use this notion to ensure that each group is associated with a diverse set of sensitive values. A similar notion of *multiRelational* k-anonymity was proposed in [49] to ensure k-anonymity on multiple relational tables. This idea is similar to (X, Y)-anonymity, where X is the set of QI and Y is the person's ID. [27].

ℓ-Diversity

The ℓ-diversity notion was proposed in [46] as a solution to attribute linkage attack. The ℓ-diversity requires each group of records with the same QI, to have at least ℓ "well-represented" SAs. Informally this means ensuring that there are at least ℓ distinct values for the SA in each such group.

This distinct ℓ values automatically satisfies k-anonymity for the ℓ-diversity notion,

where k=ℓ, since each group with unique QI has at least ℓ records. One limitation of ℓ-diversity is that it can not prevent probabilistic inference attacks. In case some SAs are much more frequent than others in a group, such as Flue versus HIV, an attacker can infer the SA value very likely. Another limitation of this approach is the assumption that the frequencies of the various SA values are some how the same. Making such tables ℓ-diverse may result in a large information loss [19].

Confidence Bounding

The problem of how to bound the confidence of inferring a SA, as another solution to attribute linkage attack, was studied in [60]. Authors specified a *privacy template* in the form $\langle QI \rightarrow s, h\rangle$, where s is a sensitive value, and h is a threshold. Their proposed privacy notion ensures that the attackers confidence of inferring the sensitive value s in any group on QI is bounded to at most h.

A general privacy model, called (X, Y)-Privacy [59], combines both (X, Y)-anonymity and *confidence bounding* which ensures each group x on X to have at least k records and $conf(x \rightarrow y) \leq h$ for any y in Y, where Y is a set of sensitive values and h is the confidence threshold as in confidence bounding model. Another similar work is the (α, k)-anonymity [63] where α is the confidence threshold. If the sensitive values are skewed both (X, Y)-privacy and $(\alpha; k)$-anonymity may result in high distortion [27].

t-Closeness

The *skewness attack* on ℓ-diverse data was explained in [43]. Authors observed that if the overall distribution of a SA is skewed, ℓ-diversity can not prevent attribute linkage attacks. For example if 95% of patients have Flu and 5% have HIV, and a group with the same QI has 50% of Flu and 50% of HIV, although this satisfies 2-diversity, any patient in the group could be inferred as having HIV with 50% confidence, while it was 5% in the overall table. As a solution to this kind of attack, authors proposed the notion of t-Closeness.

t-closeness requires the distribution of a sensitive attribute in any group on QID to be close to the distribution of the attribute in the overall table. It applies the *Earth Mover Distance* (*EMD*) function to measure the closeness between two distributions and ensures the closeness to be within t. However, this approach has some limitations including lack of flexibility in specifying different protection levels for different sensitive values, not suitable

EMD function for preventing attribute linkage on numerical sensitive attributes [42], and high information loss and removing some correlation between QI and sensitive attributes [27].

(ρ_1, ρ_2)-Privacy

As a randomization technique to satisfy data privacy, [23] proposed (ρ_1, ρ_2)-privacy notion. In this approach the adversary's knowledge before data release (prior knowledge) and after data release (posterior knowledge) are the privacy parameters. The (ρ_1, ρ_2)-privacy guarantees that if attacker's prior knowledge on a SA value is at most ρ_1 then after seeing the released data and the new perturbed SA value, his posterior knowledge is bounded by ρ_2, where $0<\rho_1 < \rho_2<1$.

For example in the patient database, a (0.01-0.1)-privacy ensures that if for an attacker, before publication the probability that a patient has HIV is 0.1, after publication this probability can increase at most to 0.1.

δ-Presence

A *table linkage* could happen if an attacker can infer the presence or the absence of the victims record in the released table with high confidence. In order to prevent such attacks, [49] proposed δ-Presence notion to bound the probability of inferring the presence of an individual's record within a range (δ_{min}, δ_{max}). This notion can indirectly prevent record and attribute linkages because if the attacker has at most $\delta\%$ of confidence that an individuals record is present in the released table, then the probability of a successful linkage to her record and sensitive attribute is at most $\delta\%$ [27]. One important assumption in this method which may not be hold in practical, is that the data publisher has access to the same external table as the attacker has.

ϵ-Differential Privacy

An interesting notion of privacy was proposed in [21] called ϵ-differential privacy which considers that participating in a statistical database should not substantially increase the risk of privacy breach for an individual. Basically, this model guarantees that the addition or removal of a "single" record in the database will not significantly change the analysis results. ϵ-differential privacy can not prevent record and attribute linkage attacks, but it

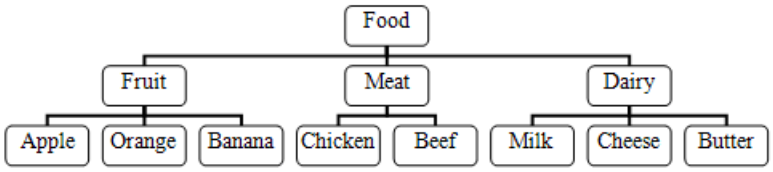

Figure 1.3: Food taxonomy tree

assures record owners that submitting their personal information to the database is very secure and almost no personal information will be discovered from the database with their information. Authors proved that this method can provide a guarantee against attackers with arbitrary background knowledge.

1.1.2 Data Anonymization Techniques

The well-known approaches for modifying a table to satisfy a privacy requirement are: *generalization*, *suppression*, *anatomization*, *permutation*, and *perturbation*.

Generalization and Suppression

In suppression we delete some values, and in generalization we replace some values with their less specific values. For the generalization, we replacement categorical attributes with respect to a given taxonomy. In Figure 1.3, the parent node *Meat* is more general than the child nodes *Beef* and the root node, *Food*, represents the most general value in a food taxonomy tree. Values in numerical attributes are usually replaced with an interval containing the original values. Both generalization and suppression produce consistent representation of original data. There are five common generalization schemes [27]:

In *full-domain generalization* scheme [40], [54], [57], all values in an attribute are generalized to the *same* level of the taxonomy tree. For example, in a full-domain generalization in Figure 1.3, if *Beef* and *Chicken* are generalized to *Meat*, then *Apple*, *Orange* and *Banana* should be generalized to *Fruit*. This method has a high information loss due to the obligatory generalization in a same level.

In *subtree generalization* scheme [9], [29], [30], [35], [40], either all child nodes or none are generalized. For example, in Figure 1.3, this scheme requires that if *Beef* is generalized to *Meat*, then the other child node, *Chicken*, would also be generalized to *Meat*, but *Apple* and *Orange*, which are child nodes of *Fruit*, can remain ungeneralized. This scheme, forms

a "cut" through the taxonomy tree.

Similar to the subtree generalization, *sibling generalization* scheme [40] also generalizes child nodes except that some siblings may remain ungeneralized. For example, in Figure 1.3, if *Apple* and *Orange* are generalized to *Fruit*, but *Banana* remains ungeneralized, *Fruit* is interpreted as all fruits covered by *Fruit* except for *Banana*.This scheme produces less distortion than subtree generalization schemes because it does not generalize all child nodes.

In *cell generalization* scheme [40], [63], [66], also known as "local recoding", only some instances of a value will be generalized compared to "global recoding" in which if a value is generalized, all its instances are generalized. For example in a transaction database the in some records *Apple* could be generalized to *Fruit*, while the in some records not. Local recoding scheme has a smaller data distortion than the global recoding. However most standard data mining methods treat *Apple* and *Fruit* as two independent values, while they are not.

Three main suppression techniques are *Record suppression* [9], [35], [40] [54],*value suppression* [60], [61], and *cell suppression* (or local suppression)[17] [47], which are processes of suppressing an entire record, all instances of a given value, and suppressing some instances of a given value in a database, respectively.

Anatomization and Permutation

Removing the association between QI and SAs by grouping and shuffling sensitive values is the main approach in *anatomization* and *permutation* techniques.

In anatomization [64] unlike generalization and suppression, the QI or the SAs are not modified and instead the QI data and the SAs data will be published in two separate tables: a QI table containing the quasi-identifier attributes, a SA table containing the sensitive attributes, and tables have one common *GroupID* attribute. With the advantage of not changing QI and SAa, authors showed that answering aggregate queries on the QI and SAs on anonymized data using this method can be more accurat than the generalization approach. On the other hand, standard data mining tools, such as classification, clustering, and association mining tools, may not be applied to such data published in two tables [27].

Authors in [69] proposed permutation approach which also removes association between a quasi-identifier and a numerical sensitive attribute. In this technique, records are partitioned into groups and then their sensitive values within each group will be shuffled.

Perturbation and Randomization

In *perturbation* technique, data is distorted by adding some noise, aggregating/swapping values, or generating synthetic data based on some statistical properties of the original data [27]. This has a long history in statistical disclosure control [1] which is simple and efficient, and at the same time preserves statistical information.

The general idea in perturbation is to replace the original data values with some synthetic data values in such a way that the statistical information is preserved. Since the perturbed data records do not correspond to an original record, an attacker cannot infer the sensitive information from the published data. However, this makes some data mining tasks difficult as the records do not correspond to a real life record owner.

In *additive noise* technique [1] [11] a sensitive numerical data such as salary is altered by adding a random value drawn from some distribution. Authors in [5] measured the privacy by how closely the original values of an altered attribute can be estimated. Studies in [26], [7], [20], [24] showed that data modified by adding random noise not only preserve some simple statistical information, like means and correlations, but also the randomized data can be useful for some data mining tasks.

The general idea of *data swapping* is to anonymize records by exchanging values of SAs among them. It was shown in [52], [51] that this method can protect numerical and categorical attributes. [23] also proposed a randomization approach based on data swapping to limit the attacker's background knowledge on inferring sensitive attributes while preserving data utility.

The *synthetic data generation* approach is also used in statistical disclosure control methods [53] by building a statistical model from the data and then publishing some sampled data from the model. In a similar approach called *condensation* [3], records are condensed into multiple groups within which some statistical information will be extracted that preserve the mean and correlations of different attributes. In the next step, based on the statistical characteristics of the group, some generated records for each group will be published.

1.2 Motivations

We study the query log anonymization problem with the focus on reducing information loss. Query log data can be seen as a special case of transaction data, where each transaction contains several "items" from an item universe I. In this thesis we also treated query logs

as transaction data and proposed our approach upon it.

In the case of query logs, each transaction represents a query and each item represents a query term. Other examples of transaction data are emails, online clicking streams, online shopping transactions, and so on. As pointed out in [58] and [67], for transaction data, typically the item universe I is very large (say thousands of items) and a transaction contains only a few items. For example, each query contains a tiny fraction of all query terms that may occur in a query log. If each item is treated as a binary attribute with 1/0 values, the transaction data is extremely high dimensional and sparse. On such data, traditional techniques suffer from extreme information loss [58] and [67].

Recently, the authors of [34] adapted the top-down *Mondrian* [41] partition algorithm originally proposed for relational data to generalize the set-valued transaction data. We refer to this algorithm as *Partition* in this thesis. They adapted the traditional k-anonymity [54] and [57] to the set valued transaction data. A transaction database is $k - anonymous$ if transactions are partitioned into equivalence classes of size at least k, where all transactions in the same equivalence class are exactly identical. This notion prevents linking attacks in the sense that the probability of linking an individual to a specific transaction is no more than $\frac{1}{k}$.

TID	**Original Data**	**Partition**
t_1	$< Orange, Chicken, Beef >$	$< Fruit, Meat >$
t_2	$< Banana, Beef, Cheese >$	$< Food >$
t_3	$< Chicken, Milk, Butter >$	$< Food >$
t_4	$< Apple, Chicken >$	$< Fruit, Meat >$
t_5	$< Chicken, Beef >$	$< Food >$

Table 1.1: The motivating example and its 2-anonymization

Our insight is that *Partition* method suffers from significant information loss on transaction data. Consider the transaction data $S = \{t_1, t_2, t_3, t_4, t_5\}$ in the second column of Table 1.1 and the item taxonomy in Figure 1.3. Assume $k = 2$. *Partition* works as follows. Initially, there is one partition $P_{\{Food\}}$ in which the items in every transaction are generalized to the top-most item food. At this point, the possible drill-down is *Food* $\rightarrow$ {*Fruit, Meat, Dairy*}, yielding 2^3-1 *disjoint* sub-partitions corresponding to the non-empty subsets of {*Fruit, Meat, Dairy*}, i.e., $P_{\{Fruit\}}$, $P_{\{Meat\}}$, ..., and $P_{\{Fruit,Meat,Dairy\}}$, where the curly

bracket of each sub-partition contains the common items for all the transactions in that sub-partition. All transactions in $P_{\{food\}}$ are then partitioned into these sub-partitions. This results in assigning transactions t_1 and t_4 to $P_{\{Fruit,Meat\}}$, t_2 to $P_{\{Fruit,Meat,Dairy\}}$, t_3 to $P_{\{Meat,Dairy\}}$, and t_5 to $P_{\{Meat\}}$. Consequently, all sub-partitions except $P_{\{fruit,meat\}}$ violate k-anonymity (for k=2) and thus are merged into one partition $P_{\{food\}}$. Further partitioning of $P_{\{fruit,meat\}}$ also violates k-anonymity. Therefore, the algorithm stops with the result shown in the last column of Table 1.1.

One drawback of Partition is that it stops partitioning the data at a high level of the item taxonomy. Indeed, specializing an item with n children will generate 2^n-1 possible sub-partitions. This exponential branching, even for a small value of n, quickly diminishes the size of a sub-partition and causes violation of k-anonymity. This is especially true for query logs data where query terms are drawn from a large universe and are from a diverse section of the taxonomy.

Moreover, the *Partition* does not deal with item duplication. As an example, the generalized t_3 in the third column of Table 1.1 contains only one occurrence of food, which clearly has more information loss than the generalized transaction <*Food, Food, Food*> because the latter tells more truthfully that the original transaction purchases at least three items. Indeed, the TFIDF used by many ranking algorithms critically depends on the term frequency of a term in a query or document. Preserving the occurrences of items (as much as possible) would enable a wide range of data analysis and applications.

1.3 Contributions

To render the input transaction data k-anonymous, our observation is: if "similar" transactions are grouped together, less generalization and suppression will be needed to render them identical. As an example, grouping two transactions <*Apple*> and <*Milk*> (each having only one item) entails more information loss than grouping two transactions <*Apple*> and <*Orange*>, because the former results in the more generalized transaction <*Food*> whereas the latter results in the less generalized transaction <*Fruit*>. Therefore, with a proper notion of transaction similarity, we can treat the transaction anonymization as a clustering problem such that each cluster must contain at least k transactions and these transactions should be "similar". Our main contributions are as follows:

Contribution 1. For a given item taxonomy, we introduce the notion of the *Least*

Common Generalization (LCG) as the generalized representation of a subset of transactions, and as a way to measure the similarity of a subset of transactions. The distortion of LCG models the information loss caused by both item generalization and item suppression. We devise a linear-time algorithm to compute LCG.

Contribution 2. We formulate the transaction anonymization as the problem of clustering a given set of transactions into clusters of size at least k such that the sum of LCG distortion of all clusters is minimized.

Contribution 3. We present a heuristic linear-time solution to the transaction anonymization problem.

Contribution 4. We evaluate our method on the AOL query logs data.

1.4 Thesis Outline

The rest of this thesis is organized as follows:

- Chapter 2 studies the related works in the literature of privacy-preserving web query-log data publishing from web community with focus on privacy concern attacks, and from the database community with focus on transaction database anonymization.
- Chapter 3 describes basic definitions, specifically definitions of *Least Common Generalization* (LCG), information loss metric, and the problem statements.
- Chapter 4 gives our clustering algorithm *Clump*, as an effective heuristic linear-time solution to the anonymization problem defined in chapter 3. This anonymization method was also published in [48].
- Chapter 5 presents the detailed algorithm for computing LCG which is part of the *Clump* algorithm.
- Chapter 6 presents the results of extensive experiments on the AOL query log data. Results shows the superiority of the proposed method with respect to the utility of data compared to the state of-the-art transaction anonymization methods.
- Chapter 7 concludes the thesis, and suggests some future work.

Chapter 2

Related Work

A recent survey [16] discussed seven query log privacy-enhancing techniques from a policy perspective, including deleting entire query logs, hashing query log content, deleting user identifiers, scrubbing personal information from query content, hashing user identifiers, shortening sessions, and deleting infrequent queries.

Log deletion is the most privacy-enhancing technique; however, the utility of data drops to zero. Hashing queries is also a vulnerable technique since other publicly available data sets, such as previously released query logs, or search engine statistics about queries in unhashed form, can be used to pinpoint an individual. Similarly, hashing identifiers cannot guarantee eliminating the risk of breaching individual privacy.

Even after removing identifying information it may still be possible to link queries back to individuals by using other publicly available information. Although shortening sessions can be highly privacy-protective, due to removal of the link between a user and his/her entire query history, the query content may still contain identifying information, and thus the risks from accidental and malicious disclosure will not be totally resolved. In addition, query logs with short sessions are less useful for analysis.

Deleting queries that appear infrequently in the logs was suggested in [2] as an effective way of removing queries that contain identifying information. Setting a threshold for being "infrequent" is however very challenging. Also studies showed that a large number of queries in huge query log datasets, occur a small number of times [10]. Consequently, this approach may lead to deletion of a remarkable amounts of non-identifying queries.

Although the above mentioned techniques protect privacy to some extent, there is a lack of formal privacy guarantees. For example, the release of the AOL query log data still

leads to the re-identification of a search engine user even after hashing users identifiers [8]. The challenge is that the query content itself may be used together with publicly available information for linking attacks.

The problem of web query-log anonymization have been examined with [39] and [2] from web community with focus on privacy concern attacks, and [34], [58], [67], and [44] from the database community with focus on transaction database anonymization. The major challenge for all query-log anonymization is reducing the significant information loss of the anonymized data.

2.1 Token based Hashing

In *token based hashing* [39] a query log is anonymized by tokenizing each query term and securely hashing each token to an identifier. However, if an unanonymized reference query log has been released previously, the adversary could apply co-occurrence analysis and frequency analysis on the reference query log to extract statistical properties of query terms and then processes the anonymized log to invert the hash function based on co-occurrences of tokens within queries. For example, if an adversary knows how often the query "Tom Cruise" appears in a previously released log can use the statistics to decipher the separate hashes for "Tom" and "Cruise".

2.2 Secret Sharing and Split Personality

Secret sharing [2] is another query anonymization method which splits a query into k random shares and publishes a new share for each distinct user issuing the same query. This technique guarantees k-anonymity because each share is useless on its own and all the k shares are required to decode the secret. This means that a query can be decoded only when there are at least k users issuing that query. The result is equivalent to suppressing all queries issued by less than k users. Since queries are typically sparse, many queries will be suppressed as a result.

Split personality, also proposed in [2], focus on reducing the possibility of reconstructing search history of a user by splitting the logs of each user based on "interests". For example, if a user is interested in both Sport and Art, then he will have two different profiles, one for the queries about Sport, and the other for the queries related to Art. In this way, the

users become dissimilar to themselves, however the distortion makes it more difficult for researchers to correlate different facets.

2.3 $(h; k; p)$-coherence Method

The *coherence* method proposed in [67] eliminates both record linkage attacks and attribute linkage attacks. The (h, k, p)-coherence privacy criterion ensures that at least k transactions must have any subset of at most p non-sensitive items and at most h percent of these transactions have some sensitive item. The parameter p, models the power of the adversarys knowledge. In other words, (h, k, p)-coherence ensures that, for an attacker with the power p, the probability of linking an individual to a transaction is limited to $1/k$ and the probability of linking an individual to a sensitive item is limited to h. The following example is borrowed from [67].

Example 2.3.1 *Consider the database D in Table 2.2 for life styles and illnesses research. "Activities" can be drinking, smoking, etc., which are non-sensitive, and "Medical History" is the persons major illness, which is considered sensitive. For k=2, p=2, and h=80%, D violates (h,k,p)-coherence. Since only one transaction T2 is $\{a, b\}$-cohort, if an adversary has the background knowledge $\{a, b\}$, T2 can be uniquely identified. Two transactions T2 and T3 are $\{b, f\}$-cohort. Since both T2 and T3 have "Hepatitis", an adversary with background knowledge $\{b, f\}$, can be 100% sure about the medical history. However, for p=1, k=2, and h=80%, D is (h,k,p)-coherent.*

TID	Activities	Medical History
t_1	a, c, d, f, g	$Diabetes$
t_2	a, b, c, f	$Hepatitis$
t_3	b, d, f, x	$Hepatitis$
t_4	b, c, g, y, z	HIV
t_5	a, c, f, g	HIV

Table 2.1: Sample database of life styles and illnesses

Let β denote the adversary's background knowledge that a transaction contains some non-sensitive items. An attack is modeled in the form of $\beta \rightarrow e$, where e is a sensitive item.

Let $Sup(\beta)$ denote the support of β i.e., the number of such transactions. $P(\beta \rightarrow e) = Sup(\beta \cup \{e\})/Sup(\beta)$ is the probability that a transaction contains e given that it contains β. The *breach probability* of β, denoted by $P_{breach}(\beta)$ is the maximum $P(\beta \rightarrow e)$ for any private item e. Assume an adversary's background knowledge is up to p non-sensitive items, i.e., $|\beta| \leq p$. If $Sup(\beta) < k$, the adversary is able to link an individual to a transaction (record linkage attack) and if $P_{breach}(\beta) > h$, the adversary is able to link an individual to a sensitive item (attribute linkage attack). A *mole*, is any background knowledge (at most to the size p) that can result in a linking attack. *Coherence* aim at eliminating all moles.

Definition 2.3.1 (Coherence) *For a setting of $(h; k; p)$, a public itemset β with $|\beta| \leq p$ and $Sup(\beta) > 0$ is called a mole if either $Sup(\beta) < k$ or $P_{breach}(\beta) > h$. The data D is $(h; k; p)$-coherent if D contains no moles.*

A *mole* can be removed by suppressing any item in the mole. Authors in [67] applied the *total item suppression* technique to enforce (h, k, p)-coherence. Total suppression of an item refers to deleting the item from *all* transactions containing it. Although total suppression results in a high information loss when the data is sparse, it has two nice properties: (1) eliminating all moles containing the suppressed item, and (2) keeping the support of any remaining itemset, equal to the support in the original data. The latter one implies that any result derived from the modified data, also holds on the original one. This is not hold for *partial suppression*, and that is the reason authors applied total item suppression.

The information loss is measured by the amount of items suppressed. Since an optimal solution to (h, k, p)-coherence, i.e. (h, k, p)-coherence with minimum information loss, was shown to be *NP*-hard [67], authors proposed a heuristic solution.They defined *minimal moles* as those moles that contain no proper subset as a mole in which removing them is sufficient for removing all moles. Their algorithm greedily suppresses the next item e with the maximum $Score(e)$ until all moles are removed, where $Score(e)$ measures the number of minimal moles eliminated per unit of information loss. Algorithm 2.1 shows this greedy algorithm.

In order to find all minimal moles, authors proposed an algorithm similar to the well-known Apriori [6] algorithm for mining frequent itemsets. It examines i-itemsets in the increasing size i until an itemset becomes a mole for the first time, at which time it must be a minimal mole. If an examined i-itemset β is not a mole, we then extend β by one more item.

Algorithm 2.1 Greedy Suppression Algorithm

Input: Transaction database: D, Privacy parameters: h, k, p
Output: Anonymized transaction database: D
1. Suppress all size-1 moles from D
2. **while** there are minimal moles in D **do**
3. Suppress the public item e with the maximum $Score(e)$ from D
4. **end while**

2.4 Band Matrix Approach

Authors in [32] proposed a data anonymization method to prevent attribute linkage attacks for high-dimensional data with sensitive items, such as transaction data, using a *band matrix* technique. In a band matrix, non-zero entries are confined to a diagonal band and zero entries on either side. In such a matrix, rows correspond to transactions and columns correspond to items, with the 0/1 value in each entry. In their method, items are divided into *sensitive items* (private items), and *non-sensitive items* (public items). A non-sensitive transaction, is a transaction with no sensitive items and sensitive transactions are those with at least one sensitive item.

A transaction set T has *privacy degree* of p if the probability of associating any transaction $t \in T$ with a particular sensitive item does not exceed $\frac{1}{p}$. To achieve this privacy requirement, [32] suggested applying two phases: (1) transforming the data to a band matrix with respect to non-sensitive attributes, and (2) grouping each sensitive transaction with non-sensitive transactions or sensitive ones with different sensitive items. The intuition why such band matrix formation is helpful, is that it organizes data such that consecutive transactions are very likely to share many common non-sensitive items and this results in a smaller reconstruction error.

In the first phase of the anonymization [32], a band matrix is built such that the total bandwidth is minimized. This problem was shown to be *NP*-complete, and thus authors applied heuristic *Reverse Cuthill-McKee* (*RCM*) algorithm approach which is one of the most effective algorithms to transform a general matrix into a band matrix by permuting rows and columns. Since band matrix concept is originally defined for square matrix and the transaction database matrix is a non-square one, we add some columns corresponding to fake item and apply padding with zero entries. *RCM* algorithm was designed for symmetric

Algorithm 2.2 Correlation-Aware Anonymization of High-dimensional Data

Input: Transaction database: D, Privacy degree: p
Output: Anonymized transaction database: D

1. Initialize histogram H for each sensitive item $s \in S$
2. *remaining*$=|T|$
3. **while** ($\exists t \in T | t$ is sensitive) **do**
4. $t \leftarrow$ next sensitive transaction in T
5. $CL(t) \leftarrow$ non-conflicting αp pred. and αp succ. of t
6. $G \leftarrow \{t\} \cup p$ 1 trans. in $CL(t)$ with closest QI to t
7. Update H for each sensitive item in G
8. **if** ($\nexists s | H[s] \cdot p >$ *remaining*) **then**
9. *remaining* $\leftarrow$ *remaining* - $|G|$
10. **else**
11. Roll back G and continue
12. **end while**
13. Output remaining transactions as a single group

matrices, while transaction database matrix is unsymmetric. Given an unsymmetric matrix M, one good solution is performing RCM over $A{=}M{\times}M^T$ and then applying the output permutation to M. Intuitively each elements A_{ij} corresponds to number of common items between row i and j in M.

In the second phase each sensitive transaction will be grouped with non-sensitive transactions or sensitive ones with different sensitive items. A greedy algorithm based on the "one-occurrence-per-group" heuristic, called *CAHD* (Correlation-Aware Anonymization of High-dimensional Data), was proposed in [32] which allows only one occurrence of each sensitive item in a group. Algorithm 2.2, shows the *CAHD* group formation heuristic.

Given a sensitive transaction t, it forms a *candidate list* $CL(t)$ by adding all transactions in a window of size $2\alpha p{-}1$ centered at t, such that those transactions are either non-sensitive or has different sensitive item, and α is a system parameter. Then, out of the transactions in $CL(t)$, the $p-1$ of them that have the largest number of non-sensitive items in common with t are chosen to form the anonymized group with t. The process continues with the next sensitive transaction in the order, after the previous selected transactions are then removed from D.

A histogram of the number of remaining occurrences for each sensitive item is also used to ensure forming a group will not lead to having remaining set of transactions that can not

	Apple	Beef	Milk	Pregnancy Test	Mental Drug
u1		X		X	
u2	X				
u3	X				X
u4		X	X		

(a) Original Data

	Apple	Beef	Milk	Pregnancy Test	Mental Drug
u2	X				
u3	X				X
u1		X		X	
u4		X	X		

(b) Band matrix on non-sensitive attributes

	Apple	Beef	Milk	Sensitive Items
u2	X			Mental Drug: 1
u3	X			
u1		X		Pregnancy Test: 1
u4		X	X	

(c) Published groups

Figure 2.1: Applying band matrix method on sample transaction log

be anonymized. If such case happened, the current group is rolled back and a new group formation is attempted starting from the next sensitive transaction in the sequence.

Example 2.4.1 *Figure 2.1(a) shows the original matrix with the shaded area for sensitive items. Figure 2.1(b) shows the result of band matrix formation with respect to non-sensitive attributes. Figure 2.1(c) shows two groups that have privacy degree 2. From the original data, we can infer that every person who bought "Beef" but not "Milk" have also bought a "Pregnancy test" (with 100% certainty). However from the anonymized data, we can only say that half of such people have bought a "Pregnancy test".*

2.5 k^m-Anonymity

To address the record linkage attacks in transaction data, authors in [58] proposed the k^m-anonymity approach which assumes that any subset of items can be used as background knowledge. It removes record linkage attacks by generalizing items with respect to an item taxonomy.

Unlike *coherence* and *band matrix* approach, data is not distinguished as sensitive and non-sensitive in this method, but it is considered both as potential quasi-identifiers and potential sensitive data. In the k^m-anonymity method, like the *coherence* method, we assume that an adversary knows at most m number of items as background knowledge.

Definition 2.5.1 (k^m-anonymity) *A transaction database D is k^m-anonymous if an adversary who knows at most m items of a transaction $t \in D$, can not use this background knowledge to identify less than k transactions in D.*

In other words, for any set of up to m items, there should be at least k transactions that contain those items in the published database. We can consider this privacy notion as

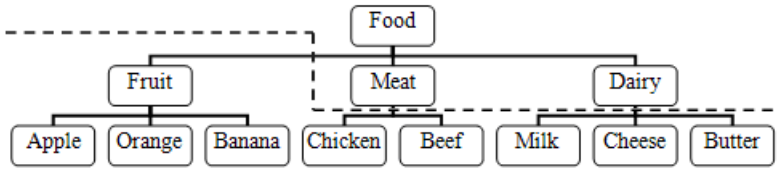

Figure 2.2: A sample cut in domain generalization

an special case of $(h; k; p)$-coherence with $h = 100\%$ and $p = m$, meaning that a subset of items that causes violation of k^m-anonymity is a mole under the *coherence* model.

The anonymization method in [58] applies generalization, in contrast to the suppression operation for *coherence*. This generalization follows the *global recoding* scheme in which when a child node is generalized to its parent, all its sibling nodes will also be generalized to their parent node, and the generalization process is applies to all transactions in the database. Each generalization corresponds to a possible horizontal *cut* of the taxonomy tree.

Example 2.5.1 *Figure 2.2 shows a generalization rule* $\{Apple, Orange, Banana\} \rightarrow Fruit$ *and the cut is shown by a dashed line. A transaction <Orange, Beef, Cheese> under this rule, will be generalized to <Fruit, Beef, Cheese>.*

The information loss of a cut is measured using the *Normalized Certainty Penalty* (*NCP*) loss metric proposed in [66]. The *NCP* captures the degree of generalization of an item i, by considering the the percentage of leaf nodes under i in the item taxonomy. The *NCP* for the whole database weights the information loss of each generalized item according to the ratio of the item appearances that are affected to the total items in the database. Therefore, if a cut c' is more general than the cut c, the information loss of c' will be larger than the one for c.

If the hierarchy cut c results in a k^m-anonymous database, then all its more general cuts c', also result in a k^m-anonymous database. This is called the *monotonicity property* of cuts [58]. The k^m-anonymization problem is to find a k^m-anonymous transformation with the minimum information loss. Based on the monotonicity property and in order to prevent higher information loss, as soon as we find a cut that satisfies the k^m-anonymity constraint, we do not have to find a more general cut.

Generating the set of all possible cuts and checking the anonymity violation for every subset of up to m items is not scalable. [58] presented *Optimal Anonymization* (*OA*) that

Algorithm 2.3 Apriori-based Anonymization

Input: Transaction database: D, Domain of items: I, Anonymity parameters: k, m
Output: k^m-anonymous transaction database: D

1. Initialize *GenRules* to the empty set
2. **for** $i = 1$ to m **do**
3. Initialize a new count-tree
4. **for all** $t \in D$ **do**
5. Extend t according to *GenRules*
6. Add all i-subsets of extended t to count-tree
7. **end for**
8. Run *DA* on count-tree for $m = i$ and update *GenRules*
9. **end for**

explores in a bottom-up fashion all possible cuts and picks the best one. Since the number of such subsets can be very large, optimal solution is not applicable for large, realistic problems. Therefore, authors proposed a greedy algorithm to find a good but not an optimal cut called *Apriori anonymization* (*AA*) which is based on the apriori principle: if an itemset J of size i, violates the anonymity requirement, then each superset of J also violates the anonymity requirement.

In the *Apriori* anonymization, the space of itemsets is explored in an apriori, bottom-up scheme. Meaning that before checking if ℓ-itemsets ($\ell = 2,\ldots, m$) violates the anonymity requirement, we first eliminate the possible anonymity violation caused by (ℓ-1)-itemsets. Following this idea, there will be a great reduction in the number of itemsets that must be checked at a higher level, since detailed items could have been generalized. Algorithm 2.3, shows the Apriori-based anonymization.

The algorithm first initializes the set of generalization rules *GenRules* to the empty set. A count-tree data structure keeps track of all i-itemsets and their support in the i^{th} iteration and paths from root to leaf in the count-tree corresponds to a i-itemset. Each transaction $t \in D$ will be generalized according to the generalization rules *GenRules* (line 5). Next, all i-subsets of each generalized transaction of t, will be added to a new count-tree and the support of each i-subset will be increased (line 6). Then all i-subsets with a support less than k and a set of generalization rules to eliminate such i-subsets will be found. This is done using the *Direct Anonymization* (*DA*) algorithm [58] on the count-tree. *DA* is performed directly on i-itemsets which violates the anonymity requirement (line 8).

All i-subsets of items with support less than k will be removed after m iterations and the final set of generalization rules in *GenRules* will be used to generalize transactions in D and make it k^m-anonymous.

2.6 Transactional k-Anonymity

In both $(h; k; p)$-coherence and k^m-anonymity methods, we assume that the background knowledge of an adversary is bounded by a given parameter. This assumption has two limitations. Firstly, in many cases it is not be possible to determine this bound in advance. Secondly, $(h; k; p)$-coherence and k^m-anonymity can ensure k-anonymity privacy by setting p or m to the maximum transaction length in the database, only if all background knowledge of the attacker is limited to the *presence* of items. The following example shows if background knowledge is on the "absence" of items and not the "presence" of items, the attacker may exclude transactions using this knowledge and focus on fewer than k transactions. The k-anonymity approach in [34], which we refer to it as *Partition*, avoids this problem since all transactions in the same equivalence class are identical.

Example 2.6.1 *An adversary knows that Bob has bought "Orange" and "Chicken", but has not bought "Milk". Suppose that three transactions contain "Orange", and "Chicken", but only two of them contain "Milk". The adversary can exclude the two transaction containing "Milk" and link the remaining transaction to Bob. Here, k^m privacy with $k = 2$ and $m = 3$ is violated, even by setting m to the maximum transaction length.*

As we also briefly explained in section 1.2, the *Partition* algorithm [34] addresses this issue by making each transaction indistinguishable from k-1 other transactions. Authors extended the original k-anonymity for relational data in [54] and [57], to the transactional k-anonymity for "set-valued data", in which a set of values are associated with an individual. This notion prevents linking attacks in the sense that the probability of linking an individual to a specific transaction is no more than $\frac{1}{k}$.

Definition 2.6.1 (Transactional k-anonymity) *A transaction database D is k-anonymous if every transaction in D occurs at least k times. A transaction database D' is a k-anonymization of D, if D' is k-anonymous and is generalized from D using a taxonomy.*

Algorithm 2.4 *Partition*: Top-down, Local Generalization

Input: Transaction database: D, Anonymity parameter: k
Output: k-anonymous transaction database: D
Anonymize(*partition*)

```
1.  if no further drill down possible for partition then
2.      return and put partition in global list of returned partitions
3.  else
4.      expandNode ← pickNode(partition)
5.      for each transaction t in partition do
6.          Distribute t to a proper subpartition induced by expandNode
7.      end for
8.      Merge small subpartitions
9.      for each subpartition do
10.         Anonymize(subpartition)
11.     end for
12. end if
```

Authors in [34] showed that every database which satisfies k-anonymity, also satisfies k^m-anonymity for all m values. However, there exists a database D which satisfies k^m-anonymity for any m, but not k-anonymity. The *Partition* method [34], which is a generalization approach to k-anonymization, partitions transactions into equivalence classes where all transactions in the same equivalence class are exactly identical. In this method, if several items are generalized to the same item, only one occurrence of the generalized item will be kept in the generalized transaction. Consequently, the information loss not only comes from item generalization, but also comes from eliminating duplicate generalized item.

The *Partition* method is basically the extended version of the top-down *Mondrian* [41] partition algorithm originally proposed for relational data. As shown in algorithm 2.4, we starts with the single partition containing all transactions with all items generalized to the *root* item. Since in this method we do not allow duplication, all transactions at this step contains only the single *root* item. Then it recursively splits a partition by specializing a node in the taxonomy for all the transactions in the partition. There is a criterion to choose which node to specialize. Then all the transactions in the partition with the same specialized item are distributed to the same sub-partition. At the end of distribution, some small sub-partitions with less than k transactions are merged into a special leftover sub-partition, and some large transactions with more than k transactions will be re-distributed

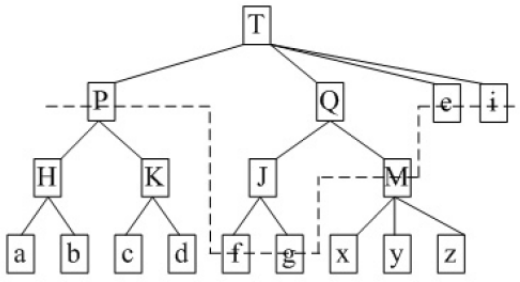

Figure 2.3: Domain generalization hierarchy

to the leftover partition to make sure that the leftover partition has at least k transactions. This partitioning stops in the case of violating k-anonymity condition.

Unlike the *Apriori* anonymization [58], the *Partition* approach follows a *local recoding* scheme since specialization is done independently for each partition.

2.7 Heuristic generalization with heuristic suppression

A very recent work also studied transaction data anonymization using suppression and generalization [44]. Authors applied *full subtree generalization* technique, meaning that a generalization solution Cut is defined by a cut on a taxonomy tree with exactly one item on every root-to-leaf path. They also applied *total item suppression* technique, which removes some items of Cut from all transactions. The set of such items to be eliminated is called a suppression scenario denoted by SS. The loss metric is the aggregate of both generalization and suppression defined as $cost(Cut,SS) = cost_G(Cut) + cost_S(SS)$.

Authors were motivated by the limitations of the k^m-anonymity, and proposed to integrate the global generalization technique in [58] with the total item suppression technique in [67] for enforcing k^m-anonymity. The *full subtree generalization* in [58] can suffer from excessive distortion in the presence of outliers. For example consider the taxonomy in figure 2.3, to achieve 2^∞-anonymity by the approach in [58] for transaction database in the first column of table 2.2, all items must be generalized to the root item, T, due to the outlier, $\{e, i\}$.

In such cases, suppressing a few outlier items will reduce information loss caused by high amount of generalization. For example, the second column of table 2.2 shows that item i is suppressed by the approach in [44]. The anonymized data is derived in two steps: first the

TID	Transaction database D	2^∞-anonymous database
t_1	b, c, d	P
t_2	a, f, g	P, f, g
t_3	d, f, y, z	P, f, M
t_4	c, d, f, x	P, f, M
t_5	a, b, c, f, g	P, f, g
t_6	e, i	$e, *$
t_7	e	e
t_8	i	$*$

Table 2.2: Sample database of life styles and illnesses

items are generalized with respect to the Cut and then items in SS are suppressed. The following example is borrowed from [44]

Example 2.7.1 *Consider the transaction database D in the first column of table 2.2, and the taxonomy in Figure 2.3. To enforce 2^∞-anonymity, D is first generalized according to the cut $\{P, f, g, M, e, i\}$. Since the number of transactions in the generalized data that support $\{e, i\}$ is more than 2, there is a privacy threat. If we suppress item i, there would be no more privacy threat.*

Since the number of cuts for a taxonomy is exponential in the number of items, enumerating suppression scenarios for a cut is also intractable. Consequently, authors provided a heuristic approach to address this issues. The outer loop of $HgHs$ algorithm enumerates cuts by a top-down greedy search of a lattice of all possible cuts and the inner loop greedily searches the suppression scenario enumeration tree, which is built per cut.

2.8 Other works

Two very recent works [22], and [13], also studied query/transaction anonymization. Authors in [13] provided a transaction anonymization approach which result in an *inference-proof* version for preventing the association of individuals to sensitive items, while truthful association rules can still be derived. Like the works in [32], and [67], they distinguish between public (non-sensitive) and private (sensitive) items.

They proposed the privacy notion *ρ-uncertainty* which ensures that the confidence of

any sensitive association rule is at most ρ. Formally, a ρ-uncertain transaction set D does not allow an attacker knowing any subset of a transaction $t \in D$ to infer a sensitive item in t with confidence higher than ρ. To meet so, they applied a technique that combines generalization and suppression.

The anonymization method in [22], clusters the queries and then replace the original queries by the centroids of the corresponding clusters considering the *semantics* of the queries. Their semantic microaggregation uses the *Open Directory Project* (ODP)[1]. Authors argued that creating a cluster with queries from users with different "interests" can result in useless protected logs and thus queries of users with common interests between them should be grouped in the same cluster.

They used ODP to compute the semantic distances between users and partition queries into groups of k users with similar interests. In the aggregation phase, they compute a new user as the representative (or centroid) of the cluster, which summarizes the queries of all the users of the cluster. The query items for the centroid are selected by a probabilistic approach based on the contribution of the user with respect to number of transactions in the cluster.

2.9 Discussion

In this thesis we model web query logs as unstructured transaction data and therefore focus on query-log anonymization from transaction database anonymization point of view. Among the previous works the *coherence* approach can both prevent record linkage attacks and attribute linkage attacks. *Band matrix* and [13] approach can prevent attribute linkage attack, and both k^m-anonymization and k-anonymization prevent record linkage attack.

The adversary's background knowledge is bounded in *coherence* and k^m-anonymization, while in *band matrix* and k-anonymization we do not limit the attacker's knowledge. A security issue about bounded knowledge in *coherence* and k^m-anonymization was explained by [34] that if background knowledge is on the "absence" of items, the attacker may exclude transactions using this knowledge and focus on fewer than k transactions. The $HgHs$ approach also has this privacy issue.

Semantic microaggregation technique [22] needs a costly computation of a *classification*

[1] http://www.dmoz.org/

matrix (containing the number of queries for each user and category at level in ODP hierarchy) and a *incidence matrix* (the addition of all coincidences between two users in the classification matrix). Also computation of the centroid for the clusters does not guarantee a minimum data distortion. They did not consider item generalization and its cost in their model.

Both k^m-anonymization and k-anonymization, do not distinguish data as sensitive and non-sensitive but as potential QI and SA. In fact, determining which items are sensitive is not always possible in many real applications considering huge size of the item universe.

The *coherence* approach and k^m-anonymity approach guarantees truthful analysis w.r.t the original data. If the transaction database is too sparse, then the item suppression of the coherence may cause a large information loss. The item generalization of the k^m-anonymization approach and the transactional k-anonymity approach could work better if the data is sparse and the taxonomy is "slim" and "tall", but have a large information loss if the taxonomy is "short" and "wide".

The local generalization of transactional k-anonymity approach has a smaller information loss than global generalization, however, the anonymized data does not have the value exclusiveness, a property assumed by most existing data mining algorithms. This means that either existing algorithms must be modified or new algorithms must be designed to analyze such data.

All the previous work in transaction data anonymization do not deal with item duplication meaning that the frequency of a term in a query can not be preserved well and will affect utilities such as count query results. The information loss of *Partition* algorithm (k-anonymity method), can be high due to item generalization, and eliminating duplicate generalized item. The latter reason of information loss was not measured by an usual information loss metric for relational data where no attribute value will be eliminated by generalization.

Chapter 3

Problem Statements

This section defines our problems. We use the terms "transaction" and "item". In the context of web query logs, a transaction corresponds to a query and an item corresponds to a query term.

3.1 Item Generalization

We assume that there is a taxonomy tree T over the item universe I, with the parent being more general than all children. This assumption was also made in the literature [54], [57], [34], [58]. For example, $WordNet$ [25] could be a source to obtain the item taxonomy.

The process of generalization refers to replacing a special item with a more general item (i.e., an ancestor), and the process of specialization refers to the exact reverse operation. In this work, an item is its own ancestor and descendant.

Definition 3.1.1 (Transactions and generalization) *A transaction is a bag of items from I (thus allowing duplicate items). A transaction t' is a Generalized Transaction of a transaction t, if for every item $i' \in t'$ there exists one "distinct" item $i \in t$ such that i' is an ancestor of i. In this case, t is the Specialized Transaction of t'.*

The above transaction model is different from [34] in several ways. First, it allows duplicate items in a transaction and in its generalized transaction. For example if t'=<$Fruit$, $Fruit$> is a generalized transaction of t, t' represents two leaf items under $Fruit$ in t. Second, it allows items in a transaction to be on the same path in the item taxonomy, in which case, each item represents a distinct leaf item. For example, we interpret the transaction

<*Fruit*, *Food*> as: *Fruit* represents a leaf item under *Fruit* and *Food* represents a leaf item under *Food* that is not represented by *Fruit*.

Also, if t' is a generalized transaction of t, each item i'$\in t$' represents (the generalization of) one "distinct" item $i \in t$. We say that an item $i \in t$ is *suppressed* in t' if no i'$\in t$' represents the item i. Hence, our *generalization* also models item *suppression*.

Example 3.1.1 *Consider the taxonomy tree in Figure 1.3 and the transaction t=<Orange, Beef>. All possible generalized transactions of t are <>, <Orange>, <Beef>, <Orange, Beef>, <Fruit, Beef>, <Orange, Meat>, <Fruit, Meat>, <Fruit>, <Meat>, <Food>, <Orange, Food>, <Food, Beef>, <Fruit, Food>, <Food, Meat>, and <Food, Food>.*

For t=<>, no item in t represents Orange or Beef, meaning that both of them are suppressed. For t=<Fruit>, Fruit represents (the generalization) of some item under the category Fruit (i.e., Orange), and Beef is suppressed. For t=<Food>, Food represents one item under Food, therefore, one of Orange and Beef in t is suppressed. For t=<Food, Food>, each occurrence of Food represents a different item in t.

3.2 Least Common Generalization

The main idea of transaction anonymization is to build groups of identical transactions through generalization. We introduce the following notion to capture such generalizations.

Definition 3.2.1 (Least Common Generalization - *LCG*) *The Least Common Generalization of a set of transactions S, denoted by LCG(S), is a common generalized transaction for all of the transactions in S, and there is no other more special common generalized transaction.*

We should note that although an empty transaction <>, is the common generalization of any set of transactions S, but it is not the least common generalization for transactions in S since at least the transaction with item *root*, is more specialized than <>. Bellow we give some examples for such least common generalization.

Example 3.2.1 *Consider the taxonomy tree in Figure 1.3. Let S1 = {<Orange, Beef>, <Apple, Chicken, Beef>}, LCG(S1) = <Fruit, Beef>. LCG(S1) cannot be <Fruit, Meat> since there is a more specialized common transaction <Fruit, Beef>.*

For S2 = {<Orange, Milk>, <Apple, Cheese, Butter>}, LCG(S2)=<Fruit, Dairy>. Dairy represents Milk in the first transaction and represents one of Cheese and Butter in the second transaction. Thus one of Cheese or Butter is considered as a suppressed item.

For S3 = {<Orange, Apple>, <Orange, Banana, Milk>, <Banana, Apple, Beef>}, LCG(S3)=<Fruit, Fruit>, which represents that all three transactions contain at least two items under Fruit. Milk and Beef are suppressed items.

For S4 = {<Orange, Beef>, <Apple, Milk>}, LCG(S4) = <Fruit, Food>, where Food represents Beef in the first transaction and Milk in the second transaction. Here LCG contains both a parent and a child item.

The following properties follow from Definition 3.2.1.

Property 1 *LCG(S) is unique for a given S.*

Property 2 *The length of LCG(S) (i.e. the number of items in it) is equal to the length of the shortest transaction in S.*

Proof: For property 1, we apply proof by contradiction: If two different transactions A and B are LCG(S), by definition each can not be specialized more into a common transaction for transactions in S. Now if A and B contain different items this means there are items with more than one parent which is contrary to fact and thus A and B are the same. As a proof for property 2, let t' be the shortest transaction in S. Each item in $LCG(S)$ should represent a "distinct" item in each transaction in S, thus $|LCG(S)|$ can not exceed the length of t'. Also $LCG(S)$ can have multiple *root* item to represent some items in t' instead of item suppression. As we discussed earlier transaction with item *root*, is more specialized than <>. Consequently $|LCG(S)|$ is equal to the size of t'. □

Various metrics have been proposed in the literature to measure the quality of generalized data including *Classification Metric* (CM), *Generalized Loss Metric* (LM) [35], and *Discernibility Metric* (DM) [9]. In this work we use *Generalized Loss Metric* to measure item generalization distortion. The similar notion of NCP (*Normalized Centrality Penalty*) has also been employed for set-valued data [58] and [34].

Definition 3.2.2 (Loss Metric - LM) *Let M be the total number of leaf nodes in the taxonomy tree T, and let Mp be the number of leaf nodes in the subtree rooted at a node p. The Loss Metric for an item p, denoted by $LM(p)$, is defined as $\frac{(Mp-1)}{(M-1)}$. For the root item p,*

LM(p)=1. In words, LM captures the degree of generalization of an item by the percentage of the leaf items in the domain that are indistinguishable from it after the generalization.

Example 3.2.2 *Considering taxonomy in Figure 1.3, LM(Fruit)=$\frac{2}{7}$.*

Suppose that we generalize every transaction in a subset of transactions S to a common generalized transaction t, and we want to measure the distortion of this generalization. Recall that every item in t represents one *distinct* item in each transaction in S (Definition 3.1.1). Therefore, each item in t generalizes exactly $|S|$ items, one from each transaction in S, where $|S|$ is the number of transactions in S. The remaining items in a transaction (that are not generalized by any item in t) are suppressed items. Therefore, the distortion of this generalization is the sum of the distortion for generalized items, $|S|\times\sum_{i\in t}LM(i)$, and the distortion for suppressed items. For each suppressed item, we charge the same distortion as if it is generalized to the *root* item, i.e., 1.

Definition 3.2.3 (Group Generalization Distortion - *GGD*) *Suppose that we generalize every transaction in a set of transactions S to a common generalized transaction t. The Group Generalization Distortion of the generalization is defined as GGD(S, t) = $|S|\times\sum_{i\in t}LM(i)$ + N_s, where N_s is the number of occurrences of suppressed items.*

To minimize the distortion, we shall generalize S to the least common generalization $LCG(S)$, which has the distortion $GGD(S, LCG(S))$.

Example 3.2.3 *Consider the taxonomy in Figure 1.3 and S1={<Orange, Beef>, <Apple, Chicken, Beef>}. We have LCG(S1) = <Fruit, Beef>. LM(Fruit)=$\frac{2}{7}$, LM(Beef)=0, and |S1|=2. Since Chicken is the only suppressed item, N_s=1. Thus GGD(S1, LCG(S1)) = 2×($\frac{2}{7}$+0) + 1 = $\frac{11}{7}$.*

3.3 Problem Definition

We adopt the transactional k-anonymity in [34] as our privacy notion. A transaction database D is k-anonymous if for every transaction in D, there are at least k-1 other identical transactions in D. Therefore, for a k-anonymous D, if one transaction is linked to an individual, so are at least k-1 other transactions, so the adversary has at most $\frac{1}{k}$ probability to link a specific transaction to the individual.

For example, the last column in Table 1.1 is a 2-anonymous transaction database.

Definition 3.3.1 (Transaction k-anonymization problem) *Given a transaction database D, a taxonomy of items, and a privacy parameter k, we want to find the clustering $C=\{S_1,\ldots,S_n\}$ of D such that $S_1,\ldots,S_n$ are pair-wise disjoint subsets of D with each S_i containing at least k transactions from D, and $\sum_{i=1}^{|C|}$ $GGD(S_i, LCG(S_i))$ is minimized.*

Let $C=\{S_1,\ldots,S_n\}$ be a solution to the above anonymization problem. A k-anonymized database of D can be obtained by generalizing every transaction in S_i to $LCG(S_i)$, $i=1,\ldots,n$.

Theorem 3.3.2 *Transactional k-anonymization problem is NP-hard.*

Sketch of Proof: Optimal k-anonymity by suppressing certain number of cells is NP-hard for $k\geq 3$, using a reduction from the *Edge Partition Into Triangles* problem [4]. Our problem deals with both generalization and suppresion, thus if we only apply the suppression and turn off the generalization, it would turn into the work in [4] with the same loss metric (number of eliminated items). □

Chapter 4

Clustering Approach

In this chapter we present our algorithm *Clump* [48] for solving the problem defined in Definition 3.3.1. In general, the problem of finding optimal k-anonymization for $k \geqslant 3$ is NP-hard [47], [4], and so is our problem. Thus, we focus on an efficient heuristic solution to this problem and evaluate its effectiveness empirically. In this section, we assume that the functions $LCG(S)$ and $GGD(S, LCG(S))$ are given. We will discuss the detail of computing these functions in Chapter 5.

The basic idea of our algorithm is to group transactions in order to reduce $\sum_{i=1}^{|C|} GGD(S_i, LCG(S_i))$, subject to the constraint that S_i contains at least k transactions. Recall that $GGD(S, LCG(S)) = |S| \times \sum_{i \in LCG(S)} LM(i) + N_s$, and from Property 2, $LCG(S)$ has the length equal to the minimum length of transactions in S. All "extra" items in a transaction that do not have a generalization in $LCG(S)$ are suppressed and contributes to the suppression distortion N_s. Since the distortion of suppressing an item is more than the distortion of generalizing an item, one heuristic is to consider length of transactions in the grouping process in order to minimize the suppression distortion N_s. We use this idea in selecting initial groups as we discuss later.

4.1 Transaction Clustering

Based on the above idea, we present our transaction clustering algorithm *Clump* [48] (Algorithm 4.5). Our clustering is similar to some general clustering algorithms, like k-means, which consider predetermined number of clusters, but is different considering the constraint of the number of members within each cluster.

The standard k-means uses an iterative refinement technique which alternate between two steps: assignment step (assigning data points to the cluster with the nearest mean) and update step (calculating the new means in clusters). This process is supposed to stop when the assignments no longer change. This algorithm is, however, not suitable for our clustering purpose. Consider taxonomy in Figure 1.3 and t_1=<$Orange, Beef$> as an initial member of a cluster. Two transactions t_2=<$Orange$> and t_3=<$Beef$> are each close to the transaction t_1 (in term of small GGD) i.e, $LCG(\{t_1\}\cup\{t_2\})$=<$Orange$>, $LCG(\{t_1\}\cup\{t_3\})$=<$Beef$> and GGD of both is 1. However t_2 and t_3 are not close to each other as $LCG(\{t_1\}\cup\{t_2\})$=<$Food$> and GGD is 2. In such cases if we assign t_2 and t_3 to the cluster containing t_1, LCG of the cluster would be <$Food$> with GGD equal to 3+1=4. This example shows that k-means is not suitable since the update step which calculates the new mean for a cluster, is done after assigning transactions to the cluster with minimum distance with the cluster mean (in our work LCG). Also reassigning the membership can be very time consuming which is not acceptable considering very large databases.

Let D be the input transaction database and let n=$\lfloor\frac{|D|}{k}\rfloor$ be the number of clusters, where $|D|$ denotes the number of transactions in D:

Step 1 (line 2-5): We arrange the transactions in D in the decreasing order of the transaction length, and we initialize the i^{th} cluster S_i, i=1,...,n, with the transaction at the position (i-1)k+1 in the ordered list. Since earlier transactions in the arranged order have longer length, earlier clusters in this order tend to contain longer transactions.

For the comparison purpose, we also implement other transaction assignment orders, such as random assignment order and the increasing transaction length order (i.e., the exact reverse order of the above algorithm). Our experiments found that the decreasing order by transaction length produced better results.

Step 2 (line 6-12): For each remaining transaction t_i in the arranged order, we assign t_i to the cluster S_j such that $|S_j|<k$ and $GGD(S_j\cup\{t_i\}, LCG(S_j\cup\{t_i\}))$ is minimized. Since this step requires computing $GGD(S_j\cup\{t_i\}, LCG(S_j\cup\{t_i\}))$, for speed up we can restrict the search to r clusters with size less than k, where r is a pruning parameter.

One possible way is to search among r clusters with respect to their LCG length. Finding r clusters with minimum difference between LCG size and transaction size, again imposes $O(n)$ complexity overhead to our algorithm. Also our greedy selection approach does not guarantee that we come up with a *globally optimal* choice. We can select different r clusters considering different ideas. In our work we only consider evaluating with the *first* r clusters

Algorithm 4.5 *Clump*: Transaction Clustering

Input: Transaction database: D, Taxonomy: T, Anonymity parameter:k, n=$\lfloor \frac{|D|}{k} \rfloor$
Output: k-anonymous transaction database: D^*

1. Initialize $S_i \leftarrow \varnothing$ for i=1, ...,$|D|$;
2. Sort the transactions in D in the descending order of length
3. **for** $i = 1$ to n **do**
4. assign the transaction at the position (i-1)k+1 to S_i
5. **end for**
6. **while** $|S_j| < k$ for some S_j **do**
7. **for** each unassigned transaction t_i in sorted order **do**
8. Let S_j be the cluster such that $|S_j|<k$ and $GGD(S_j \cup \{t_i\}, LCG(S_j \cup \{t_i\}))$ is minimized
9. $LCG(S_j) \leftarrow LCG(S_j \cup \{t_i\})$
10. $S_j \leftarrow S_j \cup \{t_i\}$
11. **end for**
12. **end while**
13. **for** each unassigned transaction t_i **do**
14. Let S_j be the cluster such that $GGD(S_j \cup \{t_i\}, LCG(S_j \cup \{t_i\}))$ is minimized
15. $LCG(S_j) \leftarrow LCG(S_j \cup \{t_i\})$
16. $S_j \leftarrow S_j \cup \{t_i\}$
17. **end for**
18. **return** $LCG(S_i)$ and S_i, i=1, ..., n

S_j with $|S_j|<k$, according to their initially sorted order. This order implies that longer transactions tend to be assigned to earlier clusters since the initial cluster assignments in step 1 was based on transaction length. Our insight is that after some transactions were assigned to the earlier clusters, the capacity of those early clusters will reach to k and thus we do not consider them anymore in step 2.

Step 3 (line 13-17): after all of the n clusters contain k number of transactions, for each remaining transaction t_i in the sorted order, we assign it to the cluster S_j with the minimum $GGD(S_j \cup \{t_i\}, LCG(S_j \cup \{t_i\}))$.

The major work of the algorithm is computing $GGD(S_j \cup \{t_i\}, LCG(S_j \cup \{t_i\}))$, which requires the $LCG(S_j \cup \{t_i\})$. We will present an algorithm for computing $LCG(S_i)$ in time $O(|T| \times |S_i|)$ in the next chapter, where $|T|$ is the size of the taxonomy tree T and $|S_i|$ is the number of transactions in S_i. It is important to note that each cluster S_i has a size at most 2k-1. Since k is small, LCG can be computed efficiently. In fact, the next lemma says that

$LCG(S_j \cup \{t_i\})$ can be computed incrementally from $LCG(S_j)$.

Lemma 4.1.1 *Let t be a transaction, S be a subset of transactions, and S'={LCG(S),t} consist of two transactions. Then LCG(S∪{t}) =LCG(S').*

Proof: First note that $LCG(S')$ is a common generalization of all transactions in $S \cup \{t\}$ because $LCG(S)$ generalizes all transactions in S. Suppose that there is a more special common generalization g of $S \cup \{t\}$. g is also a common generalization of S', contradicting that $LCG(S')$ is the LCG of S' because g is more special than $LCG(S')$. Therefore, such g does not exist. This shows that $LCG(S')$ is the LCG of $S \cup \{t\}$. □

In words, the lemma says that the LCG of $S_j \cup \{t_i\}$ is equal to the LCG of two transactions, $LCG(S_j)$ and t_i. Thus if we maintain $LCG(S_j)$ for each cluster S_j, the computation of $LCG(S_j \cup \{t\})$ involves only two transactions and takes the time $O(|T|)$.

Theorem 4.1.2 *For a database D and a taxonomy tree T, Algorithm 4.1 runs in time O(|D|×r×|T|), where r is the pruning parameter used by the algorithm.*

Proof: We apply *CountingSort* [38] which takes $O(|D|)$ time to sort all transactions in D by their length. Subsequently, the algorithm examines each transaction once to insert it to a cluster. To insert a transaction t_i, the algorithm examines r clusters and, for each cluster S_j, it computes $LCG(S_j \cup \{t_i\})$ and $GGD(S_j \cup \{t_i\}, LCG(S_j \cup \{t_i\}))$, which takes $O(|T| \times |S_j|)$ according to Theorem 5.1.1 in Chapter 5, where $|S_j|$ is the number of transactions in S_j. With the incremental computing of $LCG(S_j \cup \{t_i\})$ in Lemma 4.1.1, computing $LCG(S_j \cup \{t_i\})$ takes time proportional to $|T|$. Therefore the algorithm overally runs is in $O(|D| \times r \times |T|)$. □

Sine $|T|$ and r are constants, the algorithm takes a linear time in the database size $|D|$.

Chapter 5

Computing LCG

In the Chapter 4, we make use of the functions $LCG(S)$ and $GGD(S, LCG(S))$ to determine the cluster for a transaction. Since these functions are frequently called, an efficient implementation is crucial. In this chapter, we present a linear time algorithm for computing LCG and GGD. We focus on LCG because computing GGD is straightforward once LCG is found.

Our insight is that creating the LCG of a set of transactions has a lower cost if we apply a bottom-up generalization fashion compared to a top-down specialization approach. Intuitively, we try to find a common ancestor for each item in every transactions.

5.1 Bottom-Up Item Generalization

We present a bottom-up item generalization ($BUIG$) algorithm to build $LCG(S)$ for a set S of transactions. First, we initialize $LCG(S)$ with the empty set of items. Then, we examine the items in the taxonomy tree T in the bottom-up fashion: examine a parent only after examining all its children. For the current item i examined, if i is an ancestor of some item in *every* transaction in S, we add i to $LCG(S)$. In this case, i is the least common generalization of these items. If i is not an ancestor of any item in some transaction in S, we need to examine the parent of i.

This algorithm is described in Algorithm 2. Let $S = <t_1, \ldots, t_m>$. For an item i, we use an array $R_i[1..m]$ to store the number of items in a transaction of which i is an ancestor. Specifically, $R_i[j]$ is set to the number of items in the transaction t_j of which i is an ancestor. $MinCount(R_i)$ returns the minimum entry in R_i, i.e., $min_{j=1..m}\ R_i[j]$. If $MinCount(R_i)>0$,

i is an ancestor of at least $MinCount(R_i)$ distinct items in *every* transaction in S, so we will add $MinCount(R_i)$ copies of the item i to $LCG(S)$.

Algorithm 2 is a call to the recursive procedure $BUIG(root)$ with the *root* of T. Line 1-6 in the main procedure initializes LCG and R_i. Consider $BUIG(i)$ for an item i. If i is a leaf in T, it returns. Otherwise, line 4-9 examines recursively the children i' of i, by the call $BUIG(i')$. On return from $BUIG(i')$, if $MinCount(R_{i'})>0$, i' is an ancestor of at least $MinCount(R_{i'})$ items in *every* transaction in S, so $MinCount(R_{i'})$ copies of i' are added to LCG. If $MinCount(R_{i'})=0$, i' does not represent any item for some transaction in S, so the examination moves up to the parent item i; in this case, line 8 computes R_i by aggregating $R_{i'}$ for all child items i' such that $MinCount(R_{i'})=0$. Note that, by not aggregating $R_{i'}$ with $MinCount(R_{i'})>0$, we stop generalizing such child items. If i is the *root*, line 10-11 adds $MinTranSize(S)$-$|LCG|$ copies of the *root* item to LCG, where $MinTranSize(S)$ returns the minimum transaction length of S. This step ensures that LCG has the same length as the minimum transaction length of S (Property 2).

Example 5.1.1 *Let S={<Orange, Apple>, <Orange, Banana, Milk>, <Banana, Apple, Beef>} and consider the taxonomy in Figure 1.3. BUIG(Food) recurs until reaching leaf items. Then the processing proceeds bottom-up as depicted in Figure 5.1. Next to each item i, we show o:R_i, where o is the sequence order in which i is examined and R_i stores the number of items in each transaction of which i is an ancestor.*

The first three items examined are Apple, Orange, and Banana. R_{Apple} = [1,0,1] (since Apple appeared in transactions 1 and 3), R_{Orange} = [1,1,0], and R_{Banana} = [0,1,1]. MinCount(R_i)=0 for these items i. Next, the parent Fruit is examined and R_{Fruit} = R_{Apple} + R_{Orange}+ R_{Banana}=[2,2,2]. With MinCount(R_{Fruit}) = 2, two copies of Fruit are added to LCG, i.e., LCG(S)=<Fruit, Fruit> and we stop generalizing Fruit.

A similar processing applies to the sub-trees at Meat and Dairy, but no item i is added to LCG because MinCount(R_i)=0. Finally, at the root Food, R_{Food} = R_{Meat} + R_{Dairy} = [0,1,1]. Note that we do not add R_{Fruit} because MinCount(R_{Fruit})>2, which signals that the generalization has stopped at Fruit. Since |LCG|=MinTranSize(S), no Food is added to LCG. So the final LCG(S)=<Fruit, Fruit>. As mentioned in Example 2, the two occurrences of Fruit indicate that all three transactions contain at least two items under Fruit.

Theorem 5.1.1 *Given a set of transactions S and a taxonomy tree T of items, BUIG*

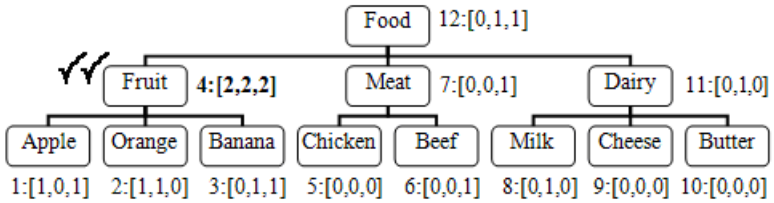

Figure 5.1: BUIG's processing order

Algorithm 5.6 Bottom-up Item Generalization

Input: Taxonomy: T, Set of m transactions: $S = < t_1, \ldots, t_m >$
Output: $LCG(S)$;

1. $LCG \leftarrow \varnothing$
2. **for each** item $i \in T$ **do**
3. **for each** $t_j \in S$ **do**
4. **if** t_j contains i **then** $R_i[j] \leftarrow 1$ **else** $R_i[j] \leftarrow 0$
5. **end for**
6. **end for**
7. $BUIG(root)$;
8. **return** LCG;

******

***BUIG*(i):**

1. **if** i is a leaf in T **then**
2. **return**
3. **else**
4. **for each** child i' of i **do**
5. $BUIG(i')$;
6. **if** $MinCount(R_{i'})>0$ **then**
7. Add $MinCount(R_{i'})$ copies of i' to LCG
8. **else** $R_i \leftarrow R_i + R_{i'}$ /* examining the parent i */
9. **end for**
10. **if** $i = root$ **then**
11. Add $MinTranSize(S)$-$|LCG|$ copies of $root$ to LCG
12. **return**

produces LCG(S) and takes time O(|T|×|S|), where |S| is the number of transactions in S and |T| is the number of items in taxonomy tree T.

Proof: First, *BUIG* generalizes transactions by examining the items in T in the bottom-up order and stops generalization whenever encountering an item that is a common ancestor of some unrepresented item in every transaction in S. This property ensures that each item added to *LCG* is the earliest possible common ancestor of some unrepresented item in every transaction. Second, *BUIG* visits each node in T once, and at each node i, it examines $R_{i'}$ and R_j of size $|S|$, where i' is a child of i. So the complexity is $O(|T|\times|S|)$. □

5.2 A Complete Example

Let us illustrate the complete run of *Clump* using the motivating example in Section 1.1. We reproduce the five transactions t_1 to t_5 in Table 5.1, arranged by the descending order of transaction length. Let k=2. First, the number of clusters is $m = \lfloor \frac{5}{2} \rfloor = 2$, and the first cluster S_1 is initialized to the first transaction t_1 and the second cluster S_2 is initialized to the third transaction t_3. Next, we assign the remaining transactions t_2, t_4, and t_5 in that order. Consider t_2. If we assign t_2 to S_1, $LCG(S_1 \cup \{t_2\})$=<*Fruit*, *Beef*, *Food*>, and *GGD* = $2\times(\frac{2}{7}+0+1) = 2.57$. If we assign t_2 to S_2, $LCG(S_2 \cup \{t_2\})$=<*Meat*, *Dairy*, *Food*> and $GGD = 2\times(\frac{1}{7}+\frac{2}{7}+1) = 2.85$. Thus the decision is assigning t_2 to S_1 resulting in S_1={t_1,t_2} and $LCG(S_1)$=<*Fruit*, *Beef*, *Food*>.

Next, we assign t_4 to S_2 because S_1 has contained k=2 transactions. So S_2={t_3, t_4} and $LCG(S_2)$= <*Chicken*, *Food*>. Next, we have the choice of assigning t_5 to S_1 or S_2 because both have contained 2 transactions. The decision is assigning t_5 to S_2 because it results in a smaller *GGD*, and $LCG(S_2)$ = <*Chicken*, *Food*>. So the final clustering is S_1={t_1, t_2} and S_2={t_3, t_4, t_5}. The last column of Table 5.1 shows the final generalized transactions.

Let us compare this result of *Clump* with the result of *Partition* in the third column (which has been derived in Section 1.1). For *Clump*, we measure the distortion by $\sum GGD(S_i,\ LCG(S_i))$ over all clusters S_i. For *Partition*, we measure the distortion by $\sum GGD(S_i,\ t_j)$ over all sub-partitions S_i where t_j is the generalized transaction for S_i. The *GGD* for *Clump* is $2\times(\frac{2}{7}+0+1)$ + [3×(0+1)+1] = 6.57, compared to [$2\times(\frac{2}{7}+\frac{1}{7})$+1] + [3×(1)+5] = 8.85 for the *Partition*.

ID	**Original Data**	**Partition**	**Clump**
t_1	$< Orange, Chicken, Beef >$	$< Fruit, Meat >$	$< Fruit, Beef, Food >$
t_2	$< Banana, Beef, Cheese >$	$< Food >$	$< Fruit, Beef, Food >$
t_3	$< Chicken, Milk, Butter >$	$< Food >$	$< Chicken, Food >$
t_4	$< Apple, Chicken >$	$< Fruit, Meat >$	$< Chicken, Food >$
t_5	$< Chicken, Beef >$	$< Food >$	$< Chicken, Food >$

Table 5.1: Comparison of *Partition* and *Clump* in sample 2-anonymization

Chapter 6

Experiments and Results

In this chapter we evaluate our approach using the real AOL query logs [50]. We could also perform experiments on synthetic data, however, such results can not be realistic and does not indicate our method is effective if applied for query log data since such generated data can be suitable for our algorithm and challenging for the competitor approach.

We compared our method *Clump* with the state-of-the-art transaction anonymization method *Partition* [34]. The implementation was done in Visual C++ and the experiments were performed on a system with core-2 Duo 2.99GHz CPU with 3.83 GB memory.

6.1 Experiment Setup

6.1.1 Dataset information

The AOL query log collection dataset consists of 20M web queries collected from 650k users over three months in form of $\{AnonID, QueryContent, QueryTime, ItemRank, ClickURL\}$ and are sorted by anonymous $AnonID$ (user ID). Our experiments focused on anonymizing QueryContent. The dataset has a size of 2.2GB and is divided into 10 subsets, each of which has similar characteristics and size. In our experiment, we used the first subset.

In addition, we merged the queries issued by the same user ($AnonID$) into one transaction because each query is too short, and removed duplicate items, resulting in 53,058 queries or transactions with the average transaction length of 20.93. Merging short queries make sense in our work since longer transactions arise privacy issues not the short ones. Besides, merged queries still have many applications such as web search personalization,

query spelling correction, advertisement, and etc. Also short queries can result in a huge information loss due to suppression and also generalizing level (close to the taxonomy root) which is not suitable for our anonymization purpose. This merging, however, affect some query log applications which deal with timestamps and session of queries.

We generated the item taxonomy T using the $WordNet$ dictionary [25]. According to the $WordNet$, each noun has multiple senses. A sense is represented by a synset, i.e., a set of words with the same meaning. We used the first word to represent a synset. In pre-processing the AOL dataset, we discarded words that are not in the $WordNet$ dictionary. We treated each noun as an item and interpreted each noun by its most frequently used sense i.e., the first synset. Therefore, nouns together with the *is-a-kind-of* links among them comprise a tree. The generated taxonomy tree contains 25,645 items and has the height 18.

We investigate four quality indicators: a) distortion (i.e., information loss), b) average generalized transaction length, which reflects the number of items suppressed, c) average level of generalized items (with the root at level 1), and d) execution time. The distortion is measured by $\sum GGD(S_i, LCG(S_i))$ over the clusters S_i for $Clump$, and by $\sum GGD(S_i, t_j)$ over the sub-partitions S_i for $Partition$ where t_j is the generalized transaction.

6.1.2 Parameters Setting

The first parameter is the anonymity parameter k. We set k to 5, 7, 10, and 15. Another parameter is the database size $|D|$ (i.e., the number of transactions). In our experiments, we used the first 1000, 10000, and 53,058 transactions to evaluate the runtime and the effect of "transaction density" on our algorithm performance.

$\|D\|$	1,000	10,000	20,000	30,000	40,000	53,058
Density	0.28%	0.25%	0.20%	0.16%	0.14%	0.11%

Table 6.1: Transaction database density

The transaction density is measured by the ratio $\frac{N_{total}}{(|D|\times|L|)}$, where N_{total} is the sum of number of items in all transactions, $|D|$ is the number of transactions, and $|L|$ is the number of leaf items in our taxonomy. $|D| \times |L|$ is the maximum possible number of items that can occur in $|D|$ transactions. Table 6.1 shows the density of the first $|D|$ transactions. Clearly, a database gets sparser as $|D|$ grows. Unless otherwise stated, we set the parameter r=10.

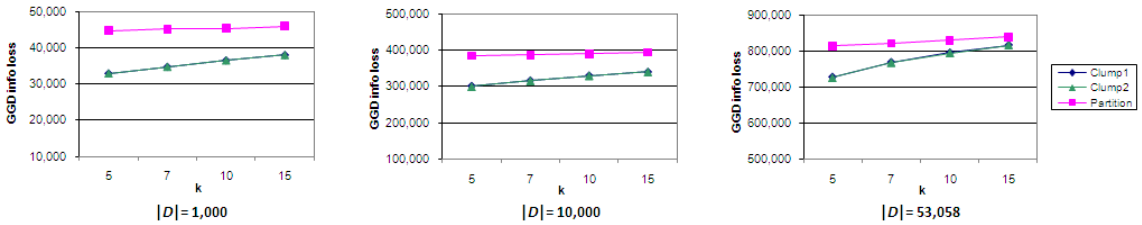

Figure 6.1: Comparison of information loss

6.2 Results

As discussed in Section 1.1, one of our goals is to preserve duplicate items after generalization because duplication of items tells some information about the number of items in an original transaction, which is useful to data analysis. To study the effectiveness of achieving this goal, we consider two versions of the result produced by *Clump*, denoted by *Clump*1 and *Clump*2. *Clump*1 represents the result produced by *Clump* as discussed in Section 4, thus, preserves duplicate items in *LCG*. *Clump*2 represents the result after removing all duplicate items from *LCG*.

6.2.1 Information loss

Figure 6.1 clearly shows that the information loss is reduced by the proposed *Clump* compared with *Partition*. The reduction is as much as 30%. As we shall see shortly, this reduction comes from the lower generalization level of the generalized items in *LCG*, which comes from the effectiveness of grouping similar transactions in our clustering algorithm. However, the difference between *Clump*1 and *Clump*2 is very small.

A close look reveals that many duplicate items preserved by *Clump*1 are at a high level of the taxonomy tree. For such items, generalization has a *GGD* close to that of suppressing an item. However, this does not mean that such duplicate items carry no information. Indeed, duplicates of items tell some information about the quantity or frequency of an item in an original transaction. Such information is not modelled by the *GGD* metric.

As the database gets larger, the data gets sparser, the improvement of *Clump* over *Partition* gets smaller. In fact, when data is too sparse, no algorithm is expected to perform well. As the privacy parameter k increases, the improvement reduces. This is because each cluster contains more transactions, possibly of different lengths; therefore, more generalization and more suppression are required for the *LCG* of such clusters. Typically, k in

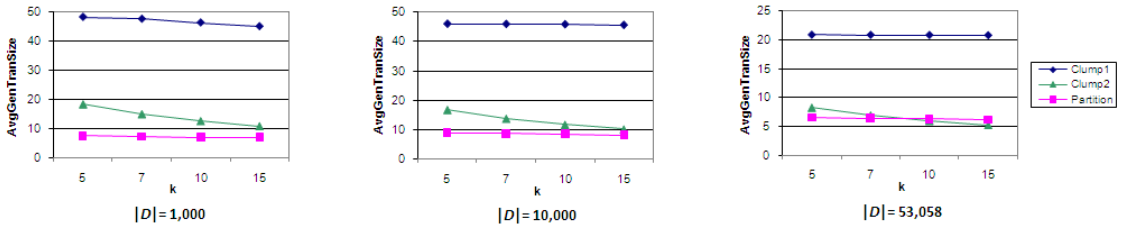

Figure 6.2: Comparison of average generalized transaction length

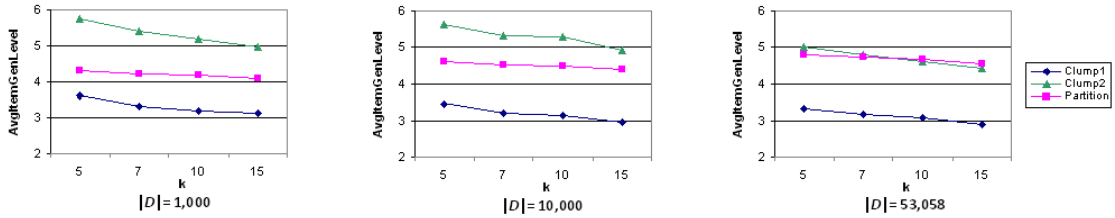

Figure 6.3: Comparison of average level of generalized item

the range of [5,10] would provide adequate protection.

6.2.2 Average generalized transaction length

Figure 6.2 shows the average length of generalized transactions. *Clump*1 has significantly larger length than *Clump*2 and *Partition*. This longer transaction length is mainly the consequence of preserving duplicate items in *LCG* by *Clump*1. As discussed above, duplicate items carry useful information about the quantity or frequency of items in an original transaction. The proposed *Clump* preserves better such information than *Partition*.

6.2.3 Average level of generalized items

Figure 6.3 shows that the average level of generalized items for *Clump*2 is lower than that for *Partition* which is lower than that for *Clump*1 (recall that the *root* item is at level 1). This is due to the fact that many duplicate items preserved by *Clump*1 are at a levelclose to the *root*. When such duplicates are removed (i.e., *Clump*2), the remaining items have a lower average level than Partition.

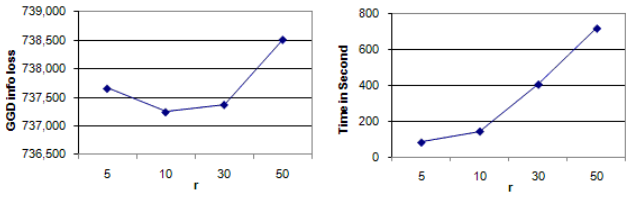

Figure 6.4: Effect of r on Clump1

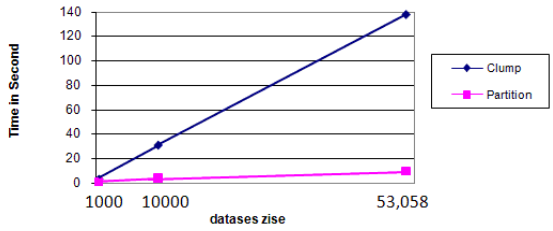

Figure 6.5: Comparison of running time

6.2.4 Sensitivity to the parameter r

This is the number of top clusters examined for assigning each transaction. A larger r means that more clusters will be examined to assign a transaction, thus, a better local optimal cluster but a longer runtime. In this experiment, we set $|D|$=53,058 and k=5. As shown in Figure 6.4, we set r to 5, 10, 30, 50, and 100.

This experiment shows that a larger r does not always give a better result since *Clump* works in a greedy manner and by increasing the number of clusters to examine, we may come up with a locally optimal choice that later increases the overall information loss. Our experiments show that r=10 achieves a good result.

6.2.5 Runtime

Figure 6.5 depicts the runtime comparison for k=5 and r=10. *Clump* takes longer time than *Partition* does. In fact, the small runtime of *Partition* is largely due to the fact that the top-down algorithm stops partitioning the data at a high level of the taxonomy because a sub-partition contains less than k transactions. Thus, this small runtime is in fact at the costly information loss. *Clump* takes a longer runtime but is still linearly scalable with respect to the data size. Considering the notably less information loss, the longer runtime of *Clump* is justified.

Chapter 7

Conclusion

The objective of publishing query logs for research is constrained by privacy concerns and it is a challenging problem to achieve a good tradeoff between privacy and utility of query log data. In this work, we proposed a novel solution to this problem by casting it as a special clustering problem and generalizing all transactions in each cluster to their least common generalization (LCG). The goal of clustering is to group transactions into clusters so that the overall distortion is minimized and each cluster has at least the size k. We devised efficient algorithms to find a good clustering. Our studies showed that the proposed algorithm retains a better data utility in terms of less data generalization and preserving more items, compared to the state-of-theart transaction anonymization approaches.

Bibliography

[1] N. R. Adam and J. C. Wortman. Security conrol methods for statistical databases. *ACM Computer Surveys*, 21, 1989.

[2] E. Adar. User 4xxxxx9: Anonymizing query logs. In *Query Log Workshop, In WWW*, 2007.

[3] C. C. Aggarwal and P. S. Yu. On static and dynamic methods for condensation-based privacy preserving data mining. *ACM Transactions on Database Systems (TODS)*, 2008.

[4] G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu. Anonymizing tables. In *ICDT*, pages 246–258, 2005.

[5] D. Agrawal and C. C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *ACM PODS*, 2001.

[6] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD*, 1993.

[7] R. Agrawal and R. Srikant. Privacy preserving data mining. In *ACM SIGMOD*, 2000.

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

[9] R. J. Bayardo and R. Agrawal. Data privacy through optimal k-anonymization. In *IEEE ICDE*, 2005.

[10] S. Beitzel, E. Jensen, A. Chowdhury, Grossman D., and O. Frieder. Hourly analysis of a very large topically categorized web query log. In *ACM SIGIR*, 2004.

[11] R. Brand. Microdata protection through noise addition. *Inference Control in Statistical Databases, From Theory to Practice*, pages 97–116, 2002.

[12] J.-W. Byun, Y. Sohn, E. Bertino, and N. Li. Secure anonymization for incremental datasets. In *VLDB Workshop on Secure Data Management (SDM)*, 2006.

[13] J. Cao, Karras P., C. Raissi, and K. Tan. ρ-uncertainty: Inference-proof transaction anonymization. In *VLDB*, 2010.

[14] D. M. Carlisle, M. L. Rodrian, and C. L. Diamond. California inpatient data reporting manual, medical information reporting for california. *Tech. rep., Office of Statewide Health Planning and Development, 5th edition.*, 2007.

[15] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24:84–88, 1981.

[16] A. Cooper. A survey of query log privacy enhancing techniques from a policy perspective. In *ACM Transactions on the Web*, volume 2, No. 4, pages 1–27. ACM, 2008.

[17] L. H. Cox. Suppression methodology and statistical disclosure control. In *Journal of the American Statistical Association*, volume 75, 370 (June), pages 377–385, 1980.

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

[19] J. Domingo-Ferrer and V. Torra. A critique of k-anonymity and some of its enhancements. In *International Conference on Availability, Reliability and Security (ARES)*, 2008.

[20] W. Du and Z. Zhang. Using randomized response techniques for privacy-preserving data mining. In *ACM SIGKDD*, 2003.

[21] C. Dwork. Differential privacy. In *the 33rd International Colloquium on Automata, Languages and Programming (ICALP)*, 2007.

[22] A. Erola, J. Castell-Roca, G. Navarro-Arribas, and V. Torra. Semantic microaggregation for the anonymization of query logs. In *Privacy in Statistical Databases*, 2010.

[23] A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *ACM PODS*, 2003.

[24] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *ACM SIGKDD*, 2002.

[25] C. Fellbaum. *WordNet: An electronic lexical database*. MIT Press, Cambridge, MA, 1998.

[26] W. A. Fuller. Masking procedures for microdata disclosure limitation. *Official Statistics*, pages 383–406, 1993.

[27] B. C. M. Fung, K. Wang, Rui C., and P. S. Yu. Privacy preserving data publishing: a survey on recent developments. In *ACM Computing Surveys*, volume 42, Issue No 4, 2010.

[28] B. C. M. Fung, K. Wang, A. W. C. Fu, and J. Pei. Anonymity for continuous data publishing. In *the 11th International Conference on Extending Database Technology (EDBT)*, 2008.

[29] B. C. M. Fung, K. Wang, and P. S. Yu. Top-down specialization for information and privacy preservation. In *IEEE ICDE*, 2005.

[30] B. C. M. Fung, K. Wang, and P. S. Yu. Anonymizing classification data for privacy preservation. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 19:711–725, 2007.

[31] J. Gehrke. Models and methods for privacy-preserving data publishing and analysis. In *Tutorial at ACM SIGKDD*, 2006.

[32] G. Ghinita, Y. Tao, and P. Kalnis. On the anonymization of sparse high-dimensional data. In *IEEE International Conference on Data Engineering (ICDE)*, 2008.

[33] K. Hafner and T. Zeller. Tempting data, privacy concerns; researchers yearn to use aol logs, but they hesitate. *The New York Times*, 2006.

[34] Y. He and J. F. Naughton. Anonymization of set valued data via top-down, local generalization. In *VLDB*, 2009.

[35] V. S. Iyengar. Transforming data to satisfy privacy constraints. In *ACM SIGKDD*, 2002.

[36] M. Jakobsson, A. Juels, and R. L. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *Proc. of the 11th USENIX Security Symposium*, 2002.

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

[38] D. Knuth. *The art of computer programming*, volume 3. Addison-Wesley, 1998.

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

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

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

[42] J. Li, Y. Tao, and X Xiao. Preservation of proximity privacy in publishing numerical sensitive data. In *ACM SIGMOD*, 2008.

[43] N. Li, T. Li, and S. Venkitasubramaniam. t-closeness: Privacy beyond k-anonymity and ℓ-diversity. In *IEEE ICDE*, 2007.

[44] J. Liu and K. Wang. Anonymizing transaction data by integrating suppression and generalization. In *PAKDD*, 2010.

[45] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. ℓ-diversity: Privacy beyond k-anonymity. In *22nd IEEE International Conference on Data Engineering (ICDE)*, 2006.

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

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

[48] A. Milani Fard and K. Wang. An effective clustering approach to web query log anonymization. In *Intl. Conf. on Security and Cryptography, (SECRYPT'10)*, 2010.

[49] M. E. Nergiz, M. Atzori, and C. W. Clifton. Hiding the presence of individuals from shared databases. In *ACM SIGMOD*, 2007.

[50] G. Pass, A. Chowdhury, and C. Torgeson. A picture of search. In *The 1st International Conference on Scalable Information Systems*, 2006.

[51] S. P. Reiss. Practical data-swapping: the first steps. *ACM Transactions on Database Systems (TODS)*, pages 20–37, 1984.

[52] S. P. Reiss, M. J. Post, and T. Dalenius. Non-reversible privacy transformations. In *ACM PODS*, 1982.

[53] D. B. Rubin. Discussion statistical disclosure limitation. *Journal of Official Statistics*, pages 461–468, 1993.

[54] P. Samarati. Protecting respondents' identities in microdata releases. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 13:1010–1027, 2001.

[55] P. Samarati and L. Sweeney. Generalizing data to provide anonymity when disclosing information. In *ACM SIGACT-SIGMOD-SIGART PODS*, 1998.

[56] L. Sweeney. Uniqueness of simple demographics in the u.s. population. *LIDAP-WP4 CMU, Laboratory for International Data Privacy*, 10:571–588, 2000.

[57] L. Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10:571–588, 2002.

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

[59] K. Wang and B. C. M. Fung. Anonymizing sequential releases. In *ACM SIGKDD*, 2006.

[60] K. Wang, B. C. M. Fung, and P. S. Yu. Template-based privacy preservation in classification problems. In *IEEE ICDM*, 2005.

[61] K. Wang, B. C. M. Fung, and P. S. Yu. Handicapping attackers confidence: An alternative to k-anonymization. *Knowledge and Information Systems (KAIS)*, 11, 2007.

[62] S. L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60:63–69, 1965.

[63] R. C. W. Wong, Li. J., A. W. C. Fu, and K. Wang. (α,k)-anonymity: An enhanced k-anonymity model for privacy preserving data publishing. In *ACM SIGKDD*, 2006.

[64] X. Xiao and Y. Tao. Anatomy: Simple and effective privacy preservation. In *VLDB*, 2006.

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

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

[67] Y. Xu, K. Wang, A. W. Fu, and P. S. Yu. Anonymizing transaction databases for publication. In *ACM SIGKDD*, 2008.

[68] Z. Yang, S. Zhong, and R. N. Wright. Anonymity-preserving data collection. In *ACM SIGKDD*, 2005.

[69] Q. Zhang, N. Koudas, D. Srivastava, and T. Yu. Aggregate query answering on anonymized tables. In *IEEE ICDE*, 2007.