PRACTICAL K-ANONYMITY ON LARGE DATASETS

By

Benjamin Podgursky

Thesis

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

in

Computer Science

May, 2011

Nashville, Tennessee

Approved:

Professor Gautam Biswas

Professor Douglas H. Fisher

To *Nila*, *Mom*, *Dad*, and *Adriane*

ACKNOWLEDGMENTS

I thank my advisor Gautam Biswas for his constant support and guidance both after and long before I started writing this thesis. Over the past four years he has helped me jump headfirst into fields I would otherwise have never known about.

I thank my colleagues at Rapleaf for introducing me to this project. I am always impressed by their dedication to finding and solving challenging, important and open problems; each of them has given me some kind of insight or help on this project, for which I am grateful. Most directly I worked with Greg Poulos in originally tackling this problem when I was an intern during the summer of 2010, and without his insight and work this project could not have succeeded.

I owe gratitude to those in the Modeling and Analysis of Complex Systems lab for their patience with my (hopefully not complete) neglect of my other projects as I tackled this thesis.

Each of my professors at Vanderbilt has guided me in invaluable ways over my time here. I want to thank Doug Fisher for helping me figure out clustering algorithms, Larry Dowdy for making me think of every problem as a distributed problem, and Jerry Spinrad for showing me that everything is a graph problem at heart.

Of course, without the consistent encouragement, support, and prodding of my family and those close to me I would never have gotten this far, and this work is dedicated to them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

CHAPTER I

INTRODUCTION

As we spend more of our time online in information-rich and personalized environments, it becomes increasingly easier for details from our offline life to meld with their online presence. Through Facebook and other social networks, our preferences in friends, food, and games becomes visible to others; LinkedIn makes our employment history and professional colleagues public information; even public Pandora profiles reveal our detailed tastes in music. Records stored offline also contain just as rich a bank of data; public voter records show who is likely to vote, and purchase histories show what someone has bought (and is likely to buy again).

As this data becomes increasingly easy to access and collect, many web providers take advantage of the ability to analyze this data and use it to tailor a person's online experiences to their specific interests. The question becomes, how should one act on this data in a responsible manner? Not many companies have found a foolproof way to do this. Most recently, Facebook and Myspace have received public and media criticism for passing demographic and interest data to advertisers. An intuitive way to balance the need for privacy but provide customization is to use an anonymized data release to customize web experiences. It is reasonable to personalize a website based on a person's probable interests, as long as the website cannot determine the person's real identity (unless they intentionally log-in).

Unfortunately, releasing data and ensuring anonymity at the same time is very difficult. To keep a user's identity unknown, it is not enough to strip data of Personally Identifying Information (PII). Even non PII data can still be traced back to a single person if enough personal microdata is provided. For example, age, income, job, marital status, and movie interests can often identify a single individual[29].

A number of companies have inadvertently released insufficiently anonymized datasets in past years. Netflix, an online movie rental company, uses movie rental and ranking records to suggest movie rentals to users, and sponsored a data-mining competition where teams competed to beat Netflix's recommendation engine. The second phase of this competition was canceled out of privacy concerns when it was discovered that many individuals could be identified by a combination of movie recommendations. [14]. AOL has previously released a large set of search queries, without realizing that most of the queries could be traced back to a single individual via personalized queries[26].

As a result, many companies and privacy researchers have been forced to look for ways to ensure the anonymity of their data, while making sure the relevant information is retained, but at the same time it lends

itself to statistical inference, data mining, and online personalization applications.

Rapleaf is a startup which specializes in web personalization[2]. If a website is able to get clues about the interests or demographics of a user, it can tailor the experience to that person. To allow a website to learn about that individual, non-personally identifying microdata can be stored in a cookie on the user's browser. The cookie contains basic non-sensitive information like age, gender, and interests.

The fact that this targeting information is being served to websites and advertisers leads to concerns about user privacy; for privacy purposes, the microdata being served about an individual should not allow the receiver of the data to link a browser to an individual. There is no harm in associating a user with the fact "Enjoys basketball"; instead the fear is that if enough of these simple pieces of data are stored, the combination will uniquely identify an individual. Unfortunately, the most specific information, and, therefore, the most de-anonymizing, is often the most valuable and useful data (it's somewhat useful to know if a person is male, but very interesting to know that he drives a Ferrari.)

Privacy Preserving Data Publishing

Fortunately, the field of research on *privacy preserving data publishing* studies exactly this problem. This field studies how to publish data in a way that simultaneously maintains privacy for the individuals whose records are being published, while keeping the released dataset rich enough that it is useful for data-mining purposes.

One popular strategy for maintaining privacy in a released dataset is simply to ensure that the dataset remains anonymous. K-anonymity was the first carefully studied model for data anonymity[36]; the k-anonymity privacy assurance guarantees that a published record can be identified as one of no fewer than k individuals. The k-anonymity problem has traditionally been researched from the perspective of sensitive data disclosure–a commonly cited domain is that of medical records released for data mining purposes, where it is important to be able to link diseases to demographic data, without revealing that an individual has a particular disease.

Most research on data disclosure focuses on protecting individuals from the release of *sensitive* attributes which could be embarrassing or harmful, if released. Traditional examples of sensitive data include medical records and criminal records. However, the objective here is somewhat different; the objective is to prevent a user from being *identified* by their published data, and none of the targeted information is considered to be sensitive data.

The similarity of the data targeting problem described above to the k-anonymity problem however indicates that algorithms developed to ensure k-anonymity could be used to efficiently anonymize this targeting data. We would like to ensure for each set of targeting microdata published, $k-1$ other people have identical published microdata.

Unfortunately, the k-anonymity problem is a very difficult problem; as a provably NP-hard problem, most research has focused on developing efficient heuristic approaches. Since the original model of k-anonymity was developed, many other data anonymity models have been studied which ensure security under either more or less aggressive attack models for a variety of data models.

Goals and Contributions of Thesis

Much of this thesis work was done in collaboration with Rapleaf as one of its efforts to ensure the privacy of web users[1][1]. We study how to ensure that the microdata released in Rapleaf's personalization cookies does not violate the anonymity of the individuals in question. Rapleaf's personalization data set was invaluable here in providing a large and realistic case study with which to objectively evaluate models and algorithms.

This thesis research has three primary goals. The first objective is to investigate how the problem of serving anonymous targeting data can be phrased as a model from the body of existing Privacy-Preserving Data-Publishing literature. There is a wide body of literature on data anonymization and privacy protection, with a number of subtle differences in structure and the nature of the adversarial model. This thesis surveys existing anonymization literature and finds privacy models which are applicable to this problem. Second this paper compares published anonymity algorithm performance using data with similar properties. Last, the paper studies the scaling up properties of algorithms given to the large amount of data which must be processed here.

More specifically, this thesis proposes methods for adapting the most promising algorithms from existing literature on k-anonymization and the less studied model of $(k,1)$ anonymization to this targeting anonymization problem. In evaluating clustering algorithms Locality Sensitive Hashing (LSH) strategies are used to improve the quality of a bottom-up clustering algorithm. An incremental refinement step is proposed to improve the utility of a dataset anonymized via clustering. We also evaluate different heuristics for use with the top-down partitioning algorithm *Mondrian*[20].

[1]The author worked as a software engineering intern at Rapleaf in the summer of 2010

Organization of Thesis

Chapter II surveys the existing literature in privacy preserving data publishing, with a focus on anonymization techniques. Chapter III formally describes this targeting advertising anonymity problem in terms of the common taxonomy. Chapter IV outlines how several algorithms from literature can be adapted to this problem, and proposes modifications to boost their efficiency. Section V compares the performance of these algorithms on three datasets; a standard machine learning dataset, a synthetic dataset with properties found in the targeting dataset, and a larger in-production dataset. Last, chapter VI summarizes the findings and proposes future directions.

CHAPTER II

BACKGROUND

Given the wide variety of published anonymity models and techniques, it is important to classify anonymization algorithms by the problems they aim to solve. Here, three characteristics are used to classify a problem: *privacy model*, *data model*, and *quality metric*. Roughly speaking, the privacy model describes what type of data the records being anonymized contain; for example, numeric versus categorical data. The privacy model specifies what type of adversary the dataset must be anonymized against, and how a dataset can be made robust against attacks from that type of adversary. Last the quality metric is how an algorithm decides whether one version of an anonymized dataset is more or less desirable than another.

Privacy Model

This section discusses the different privacy objectives when anonymizing a particular dataset. Most privacy-preserving data publishing aims to protect against one of the following:

- *identity* disclosure: given a published dataset, no record contains data which could be used to identify a record as an individual person.
- *sensitive attribute* disclosure: given a published dataset which contains data deemed sensitive, an outside observer cannot learn any sensitive information about any individual in the dataset via the disclosed data.

Identity Disclosure

When anonymizing a dataset to protect against identity disclosure, it is important that no single record in the released dataset can be identified as a specific person. For example, consider Table 1 which describes a

Table 1: Example dataset vulnerable to de-anonymization

Person	Attributes
A	{Female, Biologist}
B	{Biologist, 22, Musician}
C	{Female, Biologist, 22}
D	{Male, Musician}

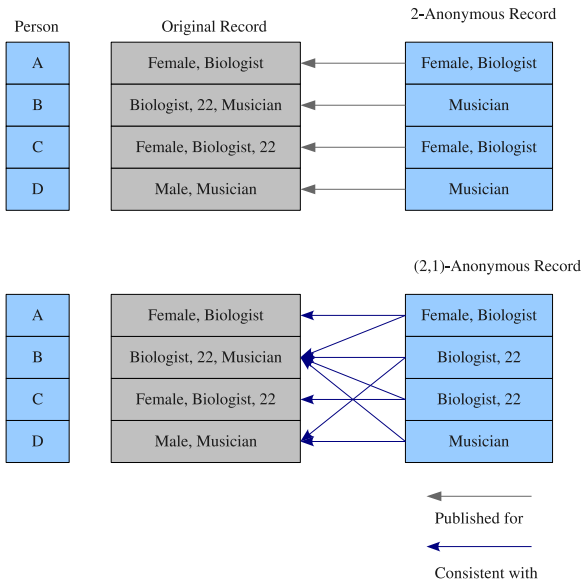

Figure 1: 2-anonymous and (2,1) anonymous views of data

population of four individuals; suppose that a non-anonymized dataset is published. If an attacker knows that an individual with attributes {Biologist, Musician} is represented in the published data, only person B could match that description. Different models are safe under different assumptions about what data an attacker has available:

k-Anonymity k-Anonymity protects against identity disclosure when the attacker knows the subset of the population represented in the dataset, knows the true attributes of all individuals in the population (including those in the dataset), and knows what data was published for each entry in the dataset.

Figure 1 shows a 2-anonymous release of the data from Table 1. Even if an attacker knew the true attributes of each individuals A, B, C and D, and knew that they were included in the dataset, it would be impossible to identity the individual with the entry {Musician}, since two people have that identical entry.

k-Anonymity assumes a strong adversary model, and it is often possible to make strong privacy assurances with a slightly more flexible model, which the next model uses.

(k, 1) anonymity For a dataset to be (k, 1) anonymous, each record in the anonymized dataset must be consistent with at least k entries in the original dataset [12]. This is the same idea as k-Unlinkability, a privacy model motivated by the re-identification of individuals based on location-visit patterns [25].

In this model, the adversary can know the true attributes associated with all individuals in the population

Table 2: Example publicly available data

Person	Attributes
A	{Female, Biologist}
B	{Biologist, 22, Musician}
C	{Female, Biologist, 22}
D	{Male, Musician}

Table 3: Published sensitive dataset vulnerable to join attack

Attributes	2-Anonymous Attributes	Sensitive Attribute
{Female, Biologist}	{Female, Biologist}	Cancer
{Biologist, 22, Musician}	{Musician}	Healthy
{Female, Biologist, 22}	{Female, Biologist}	Cancer
{Male, Musician}	{Musician}	Healthy

and the set of individuals represented by the data, but cannot know what data was released for each individual. Say that the anonymized record {Female, Biologist} from person A in Figure 1 was published. This entry is consistent with 2 entries in the original dataset–individuals A and C. So long as an attacker does not have access to what data was published for person C, they would be unable to identify the record as person A.

This model is useful when releasing data on-demand from a dataset; if the attributes to release for each query are nondeterministically selected (but still consistent with (k, 1) anonymity), and it is not logged what data is released for each individual, it remains impossible to identify an individual by the released data. An alternative adversary model under which (k, 1) anonymity holds is when the adversary knows the attributes for all individuals and the data released for each individual in the dataset, but not the set of individuals represented in the dataset.

However, this anonymity model is not as strong as k-Anonymity, and only an adversarial model which assumes that data consumers do not share information. Another weakness of (k,1) anonymity is that it does not extend well to sensitive data disclosure, discussed next. Even if k records in the original dataset are consistent with each record in the published dataset, it is possible that fewer than k published records are consistent with each record in the original dataset, leading to potential sensitive data disclosure. The tradeoff is that because of less strict adversarial assumptions, datasets can be anonymized with less information loss under this model than under straightforward k-Anonymity.

Sensitive Attribute Disclosure

The objective when making released data resistant to sensitive attribute disclosure is preventing an attacker from using publicly available data in conjunction with a published dataset to discover sensitive data about

an individual. For example, say that the data in Table 2 is considered public knowledge. If a hospital then releases the data in Table 3, an attacker could infer that individual C was treated for cancer.

Protecting against sensitive data disclosure is more difficult than protecting against identity disclosure. Even a k-anonymous dataset may be vulnerable; say that only the data in column 2 in Table 3 was published. Because both of the entries which match {Female, Biologist} were treated for cancer, even without identifying the row belonging to individual C, an attacker can deduce that person C was treated for cancer.

Furthermore, it is possible that an attacker may already have background or probabilistic knowledge about an individual, in which case one or more of the possible values may be eliminated.

P-sensitive k-Anonymity P-sensitive k-Anonymity is a simple extension of k-anonymity to prevent sensitive data disclosure; for a dataset to satisfy this property, in addition to being k-anonymous, each sensitive attribute value must appear at least p times in each equivalence class[38].

(p, α) anonymity was inspired by p sensitive anonymity with attempts to strengthen the model using the following observation: sometimes, even if an attribute is represented with two different values, the revelation of either of them is harmful[35]. If it is clear from anonymized data that a particular individual *either* has cancer or HIV, their privacy has still been violated. (p, α) anonymity defines categories of attribute values, and requires each equivalence class to contain diverse categories within a class. For example, if the categories for "disease" were {cancer, HIV} and {healthy, flu}, each of these categories would have to be represented within an equivalence class. These categories must be specified by a domain expert.

Bayes-Optimal Privacy Bayes-Optimal privacy, described in [24], is an extremely strong privacy model. Basically, the Bayes-Optimal formulation formulates the privacy loss after data release as the difference between the prior and posterior beliefs of a fact about a person; to be Bayes-Optimal, the difference between these states must be minimal.

For example, say that an attacker knows that person A is in a particular hospital. They know from public records that 20% of people in the hospital are there for radiation poisoning, so before any data is released, they think there is a 20% chance person A is there for radiation poisoning. Say the hospital releases the 2-anonymized records shown in Table 4.

If the attacker knows that person A is a 22 year old female, they now have a posterior belief of $2/3$ that person A has radiation poisoning. As a result, to satisfy Bayes-Optimal privacy, the fraction of the individuals in an equivalence class with each value for each attribute must be close to that of the population as a whole.

Table 4: 3-anonymous released dataset

Attributes	Disease
{Male, 23}	Cancer
{Male, 23}	Healthy
{Male, 23}	Healthy
{Female, 22}	HIV
{Female, 22}	Radiation Poisoning
{Female, 22}	Radiation Poisoning

This is a very strong model of anonymity, and is difficult to attain in practice while retaining reasonable utility. A number of models have been proposed which provide similar privacy guarantees without as much data loss.

***l*-diversity** *l*-diversity is a relaxation of Bayes-Optimal privacy that requires that in each equivalence class, each attribute has at least *l* "well-represented" values [24]. In other words, an attacker must have at least *l* pieces of outside information to discover the value of an attribute for an individual; if an equivalence class has 4 evenly distributed values for the attribute "disease", it will take 3 pieces of negative information (learning that a person does *not* have a disease) to discover what disease they have.

The definition of an attribute being "well-represented" is left to the implementation; the original paper proposes an entropy-based definition and two recursive definitions which cover cases where positive disclosure (revealing an attribute's value), negative disclosure (revealing values an attribute is *not*), and both, are harmful.

(α, k) anonymity (α, k) anonymity is a simple model to protect sensitive values on top of k-anonymity: no sensitive attribute value may have a frequency of greater than α within any equivalence class[39]. This anonymity model does not try to protect against background or external knowledge, but instead simply prevents the posterior belief for an attribute's value from violating a threshold set by α.

Data Models

The data model of the data being anonymized is a key factor when choosing an anonymization algorithm. Broadly, data can generally be defined as *numeric* or *categorical*. Furthermore, there may already be a *utility* value associated with each data element.

Table 5: Example anonymized numeric data

Person	Age	Anonymized age
A	21	{21-22}
B	22	{21-22}
C	34	{34-36}
D	36	{34-36}

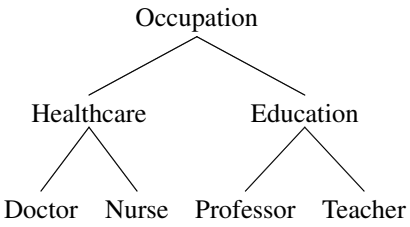

Figure 2: A simple generalization hierarchy

Numeric Data

If an attribute consists of values for which there exists a strong ordering between elements in the domain, it can be considered numeric data for anonymization purposes. Numeric data includes attributes such as *age* and *income*. It is often advantageous to represent data numerically if it makes sense to do so, as numeric data can be anonymized via *partitioning*. For example, consider the dataset in Table 5. Because age is numeric data, the range of values for age is partitioned such that the dataset is 2-anonymous without losing a significant amount of precision.

Categorical Data

An attribute is considered categorical data if its range of values is discrete, and there is no reasonable ordering among elements. For example, there is no intuitive way to map ”occupation” to numerical data, where it could be partitioned. However, often categorical data can be arranged hierarchically.

Hierarchical

Often the data that categorical data represents can be described at different levels of generality. For example, if occupation was an attribute, Figure 2 could represent its hierarchy. An individual with the attribute ”Cardiologist” could potentially be anonymized by generalizing the attribute to ”Doctor”. Anonymizing hierarchical data by generalization is conceptually similar to domain partitioning with numeric data.

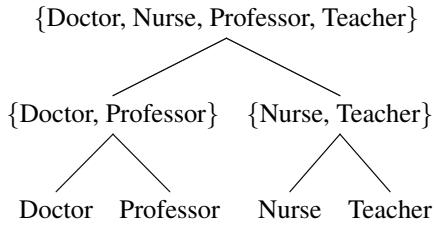

Figure 3: A possible different generalization hierarchy

Automatically generated hierarchies Usually, the hierarchy by which categorical data is generalized is defined by the user, and is based on a natural relationship between attributes. An alternative strategy is to let the anonymization algorithm define data partitions which help to preserve utility and precision[30]. For example, a section of a user-defined hierarchy for the attribute "occupation" may look like Figure 2.

When generalizing the occupations type attribute, an instance of "Doctor" could be generalized either to Doctor, Nurse, or all the way to Doctor, Nurse, Computer Engineer, Technician. This is advantageous to a human, because "Healthcare" and "Education" both hold well-understood semantic meanings. However, it is not the only way to generalize the domain. Another possible hierarchy could could look like Figure 3.

It is easy to find datasets where the latter generalization hierarchy will allow solutions with higher precision than if the former was used. The tradeoff is that the classification {Doctor, Professor} may not hold intrinsic meaning, except to indicate that a member has a 50% chance of being a doctor, and 50% chance of being a professor (depending on the global frequencies of those attributes.)

Set-Valued Data Recently the idea of anonymizing *set-valued* data has been discussed when there is not a clear distinction between non-sensitive attributes and attributes which it would be harmful to disclose (or when the sensitivity of an attribute depends on who is reading the data)[13][37]. Set-valued data is also referred to as *transactional data*, as one of the original examples of transactional data was items a person purchases. Set-valued data is conceptually easy to represent as a set: $T_{alice} = \{Coffee, Napkins\}$. While set-valued data elements can represent hierarchical data, it is also a convenient paradigm for representing non-generalizable data, where the range of values for a categorical data attribute is only $\{true, false\}$.

Data with Utility

Often, each attribute in a dataset has an associated *utility*[41]. If this is the case, the objective of the anonymization algorithm will be to maximize the utility in the anonymized dataset, rather than the raw number of attributes published.

Either hierarchical or numeric data can have associated utilities. Usually, the utility of numeric data decreases as the partition size increases; an attribute with the range [25-26] has higher utility than the range [25-30]. Hierarchical data usually has more value for leaf nodes than generalized nodes–"Cardiologist" would have higher utility than "Doctor".

Quality Metrics

The last factor to consider when designing an anonymization algorithm is how to compare the quality of two anonymized views of a dataset. A wide variety of metrics have been proposed to quantitize the information loss anonymization introduces into a dataset. Usually, the choice of quality metric is highly application specific, depending on the type of data being anonymized. This section discusses a number of quality metrics which have been reported in the literature.

Sum of suppressed entries

One of the first proposed metrics for the quality of an anonymization is simply to count the number of suppressed entries in the anonymized table [3] [27]. For example, in Table 8, the cost of the anonymization would be 1, for the one suppressed cell.

The measure is an intuitive measure for information loss for discrete data when only suppression is considered. However, the measure is not directly applicable when cells are generalized, and not simply suppressed; likewise, when the data is numeric, and is partitioned rather than suppressed, this metric is not applicable.

Domain generalization hierarchy distance

If a global recoding strategy (discussed in section II) was used to anonymize the data, a particular anonymization strategy will represent a node in the generalization lattice, like the one shown in Figure 4. The earliest cost metric for a k-anonymization strategy measures the distance traveled in the generalization lattice[33][19][7][19]. For example, in Figure 4, node $< S0, Z0 >$ has a distance vector of $height(< 0, 0 >) = 0$, while node $< S1, Z1 >$ has a distance vector of $height(< 1, 1 >) = 2$, indicating that each of the attributes had to be generalized up one level.

This cost metric only makes sense when the anonymization model is global recoding, and is not directly applicable to solutions found via local or multidimensional recoding, techniques discussed in II.

Discernibility

One of the most commonly cited metrics is the *discernibility metric*[5][20][5]. The discernibility metric penalizes each entry based on how many entries it is indistinguishable from in the anonymized dataset. In a global recoding scheme, suppressed entries are assigned a cost equal to the size of the dataset. The anonymization cost over the dataset is the sum of the anonymization costs of all entries:

$$C_{DM} = \sum_{e \in E} |e|^2 + |S||D|$$

where E is the set of all final equivalence classes (of size $\geq k$), S is the set of suppressed entries, and D is the whole dataset. The discernibility metric does not actually assign penalties based on information loss, but on the size of the final equivalence classes; the hope is that information loss will be accompanied by large equivalence classes. An equivalence class is the set of records which are indistinguishable in the published dataset because their unique attributes have been generalized or suppressed.

Classification

The classification metric assigns a penalty to each entry whose label is not the same as the majority label of the equivalence class it is in[30][5]. Like with the discernibility metric, the penalty over the dataset is the sum of the penalties of the entries:

$$C_{CM} = \frac{\sum_{t \in D} penalty(t)}{|D|}$$

where the penalty assigned to an entry t in equivalence class e is defined by:

$$penalty(t) = \begin{cases} 1 & \text{if } t \text{ is suppressed} \\ 1 & \text{if } class(t) \neq majority\_class(e) \\ 0 & \text{otherwise} \end{cases}$$

Like the discernibility metric, no actual value is assigned based on the quantity of information preserved in an equivalence class, but instead is assigned based on the similarity within classes. It is not clear how

well this metric scales to large and rich datasets, where there is a typically a high degree of diversity within equivalence classes.

Loss metric

The loss metric measures the specificity of the data cells left after an anonymization[17][30][12]. Let $f : D^*[i][j]$ be a function which, for a categorical data cell value in D^* returns the number of values this cell value represents. For example, an instance of the value "Healthcare" from the hierarchy in Figure 2 would have value 3, for itself, "Doctor", and "Nurse". Let $g(D_i)$ return the number of possible values for an attribute in D; $g(Occupation) = |\{Doctor, Nurse, Professor, Teacher\}| = 4$ in this example.

The loss metric aggregates the specificity found with these functions:

$$LM(D*) = sum_{i=1}^{n} sum_{j=1}^{|D|} fracf(D*[i][j]) - 1g(D_i) - 1$$

In essence, the LM is the sum of the "specificity" of every data cell in the dataset after anonymization.

Cluster diameter

A number of algorithms which approach k-anonymization as a clustering problem measure quality by the *diameter* of the final equivalence classes. The diameter of an equivalence class is defined as the maximum distance between two records in the class[6][22][31][23]. This metric is generated by defining distance measures over all attributes and record pairs; let δ_N be the distance between two numeric values, normalized by the domain of the attribute:

$$\delta_N(v_1, v_2) = \frac{|v_1 - v_2|}{|D|}$$

where D is the domain of the numeric attribute. Next consider the distance between two categorical values: let δ_C by the height in the taxonomy tree of two values' nearest common ancestor, normalized over the height of the taxonomy tree:

$$\delta_C = \frac{H(\Lambda(v_i, v_j)}{H(T_D)}$$

where D is a categorical domain, and T_D a taxonomy tree for D. $\Lambda(v_i, v_j)$ is the subtree at their lowest common ancestor, and $H(R)$ is the height of a tree T. The distance between two records is the sum of the distances between the records over all numeric and categorical attributes:

$$\delta(r_1, r_2) = \sum_{n \in N} \delta_N(r_1[n], r_2[n]) + \sum_{c \in C} \delta_C(r_1[c], r_2[c])$$

where N is the set of numeric attributes, and C is the set of categorical attributes. The diameter of an equivalence class is the maximum distance between any two records in the class; the cost of an anonymization is the sum of the diameters of all equivalence classes, weighted by the size of the equivalence class:

$$\sum_{e \in E} |e| Max_{r_1, r_2 \in e} \delta(r_1, r_2)$$

where E is the set of all equivalence classes.

Intuitively, this metric favors solutions with small and homogeneous equivalence classes. However, by minimizing only the maximum distance between records in a class, it does not consider information loss among interior members. As a result the metric is sensitive to datasets with a large number of outliers, whose distances usually dominate the diameter of the classes they are placed in.

Ambiguity metric

The ambiguity metric (AM) measures the number of possible combinations of input tuples that a generalized tuple could represent [30]:

$$AM(D^*) = \frac{\sum_{j=1}^{|D|} \prod_{i=1}^{n} f(D^*[i][j])}{|D|}$$

where f is defined as for the Loss metric above.

Entropy based metric

Several researchers have proposed and studied an entropy-based measure for information loss, described in [12], and shown below:

Let a public database D contain records $R_1 \cdots R_n$. If $X_j 1 \leq\leq r$ is a value of the attribute A_j in a record of D, then

$$P(X_j = a) = \frac{|\{i : R_i(j) = a, 1 \leq i \leq n\}|}{n}$$

That is, the frequency of that value of the attribute over the dataset. If B_j is a subset of A_j, then for the conditional entropy $H(X_j|B_j)$:

$$H(X_j|B_j) = -\sum_{b \in B_j} P(b|B_j) log_2 P(b|B_j)$$

where,

$$P(b|B_j) = \frac{|\{i : R_i(j) = b, 1 \leq i \leq n\}}{|\{i : R_i(j) \in B_j, 1 \leq i \leq n\}}, b \in B_j$$

If $g(D) = \{\bar{R}_1 \cdots \bar{R}_n\}$ is a generalization of D, the entropy measure cost of generalizing D to $g(D)$ is

$$\Pi_E(D, g(D)) = \frac{1}{nr} \sum_{j=1}^{n} \sum_{j=1}^{r} H(X_j|\bar{R}_i j)$$

Workload based quality metrics

Although a large number of quality metrics have been proposed and studied, it is still an open question which–if any–of these metrics accurately capture the utility of the anonymized data across use-cases. [30] measured the behavior of three quality metrics against the quality obtained from a data mining use-case. In that study, four anonymization algorithms were used to anonymize the Adult and Mushroom datasets from the UCI Machine Learning Repository [9] at different k values. The Adults dataset is based on data from the

1990 US census the quality of each of these anonymizations was measured using the cost, ambiguity and loss metrics. Then a machine learning algorithm performed classification over the attributes, and the number of classification errors was measured for the results from each anonymization algorithm.

The authors found that in fact, there was little correlation between metric quality and data mining classification quality across algorithms–the algorithms which produced anonymizations with low quality according to the cost, loss, and ambiguity metrics were not the ones which had the best classification results.

The utility of the data for use by data mining was also used as a quality metric in [20]. In that paper, an aggregate query-matching use was also considered; random aggregate queries were made on the anonymized dataset. An anonymized dataset where the results of these queries closely matched the results of the query on the original dataset was considered a high-quality anonymization. A random query might be like the one below:

SELECT $COUNT(*)$ FROM PATIENTS WHERE $Sex =$ "$Male$" and $Age \leq 26$

Anonymization which maximizes utility when attributes have prior known utilities is studied in [41]; there it is also found that anonymizations which maximize the Discernibility Metric do not necessarily maximize data utility.

Overall, it is still not clear whether there is a single generic metric which can accurately measure anonymization quality across all datasets and use-cases. If any potential uses of a dataset are known before anonymizing, it is generally advisable to tailor the quality metric on the expected uses[20]; it is quite possible that a general metric will not accurately capture the data quality as measured by that that specific use[41].

Algorithms

Many k-anonymity algorithms have already been published and evaluated. The previous sections all discussed how to classify an anonymization algorithm in terms of the data and the problem it solves. This section tries to classify the algorithms functionally, and investigates the body of published algorithms and evaluates their applicability to the problem described in the introduction. Existing algorithms which could be adapted to be tractable on this dataset, along with several novel methods, are then presented in detail in IV.

Since optimal k-anonymity is an NP-complete problem, the overwhelming majority of anonymity algorithms use greedy heuristics. Generally these algorithms work in one of two ways: first, starting from a fully anonymized solution, they greedily partitioning the data; or second, starting with a non-anonymized solution and cluster records until all records are anonymized.

Table 6: Another example dataset

Person	In School	Employed	Has cancer
A	1	1	1
B	1	0	0
C	1	1	1
D	0	0	0

This paper focuses on algorithms for k-anonymity and $(k, 1)$ anonymity, so none of the algorithms developed for p-sensitive k-anonymity[35][38][34], l-diversity[24], (a, k) anonymity[39] or other stronger models need to be studied in detail.

Anonymization Models

One of the main distinctions between anonymity algorithms is the model they use to recode the data to make it sufficiently anonymous. This section looks at the tools algorithms use for actually transforming a dataset into an anonymous dataset.

Almost all anonymization algorithms use two basic tools for transforming a dataset into an anonymous dataset; *generalization* and *record suppression*. Attribute generalization takes a categorical data entry for a record II and replaces it with a more general entry. Record suppression entirely removes a record from the published anonymous dataset. There are two general models for how to use these two operations to re-code a dataset: *global* recoding and *local* recoding. Global recoding maintains more consistency across a dataset, while local recoding provides higher utilities.

Global recoding When the dataset is anonymized via global recoding, an attribute is generalized or suppressed equally across all entries. For example, the dataset in Figure 6 can be globally 2-anonymized by suppressing the "In school" attribute across all , as shown in Table 7. There are several advantages to recoding attributes globally:

- The strategy is conceptually simple.
- It is often tractable to find an optimal solution via branch and bound methods[5].
- Inferences between remaining attributes remain consistent with the original data.

Global attribute suppression can also be interpreted as traversing a lattice of possible anonymization strategies[33]. Figure 4 shows such a lattice for two variables, sex and zipcode, alongside their individual generalization hierarchies.

Table 7: Data anonymized via global recoding

Person	In School	Employed	Has cancer
A	x	1	1
B	x	0	0
C	x	1	1
D	x	0	0

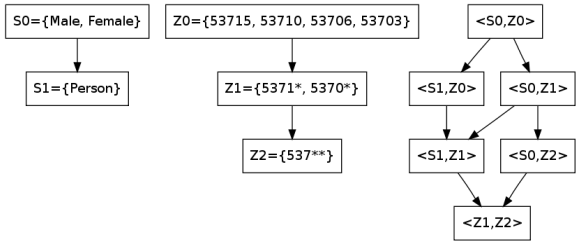

Figure 4: Example generalization lattice

The most desirable anonymization strategies on the lattice are those closest to $< S0, Z0 >$, where no generalization is performed.

To mitigate the information loss of global anonymization, usually some small number of entries are allowed to be entirely suppressed in the solution. This allows outliers to be suppressed, whose inclusion in a solution would have led to considerable information loss.

The large drawback to global recoding is that usually a huge amount of data is lost in the process. Localized recoding algorithms almost always perform much better in practice, and while many early papers on k-anonymity developed global recoding algorithms the overwhelming majority of recent research focuses on local recoding strategies. Because of the significantly lower utility of globally-recoded datasets, this thesis does not try to use global recoding algorithms, including those from [33][19][5][7].

Local recoding When a dataset is recoded locally, attributes are suppressed on a per-cell basis. Generally, the dataset can be anonymized with far less data suppressed than in global recoding.

Unfortunately, it is almost impossible to prove the optimality of a solution, or even to find reasonable bounds on solution quality: if k is the number of attributes for each of n entries, global recoding has a state space of 2^k, while local recoding has a state space of 2^{nk}. In the case of the data in Table 6, the dataset can be 2-anonymized by suppressing only one cell, as shown in Table 8.

It is argued in [10] that local recoding has additional disadvantages when machine learning techniques are used: the solution will usually contain values at multiple levels in a hierarchy (one entity may have value ”Doctor”, and another ”Cardiologist”, while ”Doctor” is a generalization of ”Cardiologist”.) The disadvan-

Table 8: Data anonymized via local recoding

Person	In School	Employed	Has cancer
A	1	1	1
B	x	0	0
C	1	1	1
D	0	0	0

tage of using an anonymized set of data for machine learning is that it is not clear how a decision tree should be constructed with records at heterogeneous generalization levels.

Multidimensional recoding

Anonymization via multidimensional partitioning [20] is functionally similar to building a classification tree over the data, with the additional constraint that each classified group is of size $\geq k$. The general idea is that instead of partitioning a numeric attribute uniformly across all the data, the partitioning will be dependent on other dimensions. Figure 5 from [20] illustrates this visually, on a space with 2 numeric attributes.

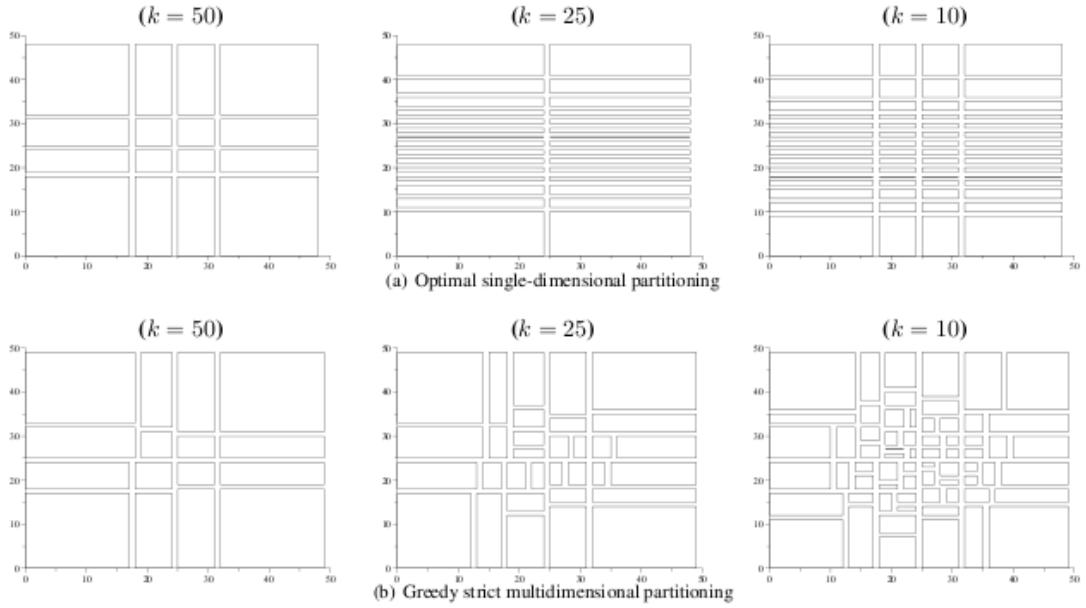

Figure 5: Figure from [20] showing multidimensional anonymizing over 2 dimensions

Complexity results and bounded approximations

A number of initial data privacy papers presented complexity analysis results and investigated ability to get bounded approximate solutions. The first paper to prove NP-hardness results of a k-anonymity problem was [27]. The paper shows the complexity of NP-hardness, under certain conditions, of both local and global recoding via reductions from the k-dimensional perfect matching problem. The paper also presents an approximation algorithm with complexity $O(n^4)$, which is not tractable here. These complexity results are extended in[3] to show that local recoding is NP-hard in any reasonable problem–specifically, any problem with more than 3 attributes. The paper also presents approximation algorithms, but without complexity analysis or implementation results.

A number of bounded approximation algorithms for k-Anonymity are presented in [31], but none of which have a lower complexity than $O(n^2)$. It is possible that the algorithms could be further approximated, but it would be at the cost of the quality boundedness which is the focus of the paper.

Algorithms which allow a data mining practitioner to obtain k-anonymous data when the data is not stored centrally but is instead distributed are discussed in [42]. The algorithm is based on the Meyerson and William algorithm [27], which has an $O(n^3)$ complexity; it is not clear that there is a lower-complexity approximation to this algorithm.

Clustering Algorithms The field of clustering studies how to partition a set of records into groups such that records which are similar generally end in the same *cluster*. Clustering is a natural way to organize and aggregate records when dealing with complex data, akin to how humans naturally group collections of objects into like clusters.

Usually the objective of a clustering algorithm is to partition the a dataset while splitting records on as few variables as possible–the goal is to be able to see a random records, and be able to bucket it in with similar records based only on a few attributes. Anonymization is a natural application of clustering. If we could partition a dataset into collections of records such that the records within each cluster are very similar, and each of these collections was of size greater than or equal to k, we have efficiently k-anonymized the dataset; under this definition an equivalence class is the closure over the attributes of the records in the cluster. A number of papers have adapted generic clustering algorithms like the k-means algorithm to achieve k-anonymity[6][23][21].

Mondrian Multidimensional k-Anonymity The model of *multidimensional k-anonymity* is introduced in [20]. The paper also presents two variations on a greedy partitioning algorithm which anonymizes data to fit this anonymity model. The basic algorithm is shown in Algorithm 1.

Algorithm 1 *Mondrian*: multidimensional partitioning anonymization

Input: partition P
Output: a set of valid partitions of P
1: **if** no allowable multidimensional cut for P **then**
2: **return** P
3: **else**
4: $dim = choose\_dimension(P)$
5: $fs = frequency\_set(P, dim)$
6: $splitVal = find\_median(fs)$
7: $lhs = \{t \in partition : t.dim \leq splitVal\}$
8: $rhs = \{t \in partition : t.dim > splitVal\}$
9: **return** $partition\_anonymize(rhs) \cup partition\_anonymize(lhs)$
10: **end if**

Essentially, this algorithm takes an existing partition, chooses a dimension upon which to create further partitions, and generates recursive calls to itself on each of these partitions. This algorithm can be adapted to categorical data by establishing an ordering among variable values, and having $find\_median$ return a mean categorical value rather than a numerical median. However, in the extreme case, when a variable has only two categorical values $(true, false)$, partitioning across a boolean value would be equivalent to splitting the partition into those records with an attribute and those without it. To allow more flexible partitioning solutions, the *relaxed multidimensional partition* is introduced, where the partitions on values for a variable may overlap. The runtime of the partitioning algorithm is $O(nlog(n))$.

k-Anonymization Revisited The $(k, 1)$ and (k, k) anonymity models are presented as alternatives to k anonymity in[12], and a greedy algorithm for $(k, 1)$ anonymity which uses a clustering-type approximation to combine records is developed. The approximation algorithm is shown in Algorithm 2.

The function $d(S_i)$ refers to the diameter of cluster S_i. The runtime of the approximation algorithm is $O(kn^2)$ because of the pairwise distance calculations in line 4 [12]. A possible relaxation of this algorithm is to simply choose a good guess for R_j instead, which is discussed in more detail in chapter IV.

Utility Based Anonymization using Local Recoding [41] studies anonymization when attributes have associated utility values. Instead of using the traditional CM and DM quality metrics, the paper uses a *weighted certainty penalty*, which takes a cluster diameter metric and weights the diameter of the cluster by the utility

Algorithm 2 $(k, 1)$ anonymization algorithm

Input: Table D, integer k
Output: Table $g(D)$ that satisfies $(k, 1)$ anonymity

```
for all 1 ≤ i ≤ n do
    Set S_i = {R_i}
    while |S_i| < k do
        Find the record R_j ∉ S_i that minimizes dist(S_i, R_j) = d(S_i ∪ {R_j}) − d(S_i)
        Set S_i = S_i ∪ {R_j}
    end while
end for
Define R̄_i to be the closure of S_i
```

associated with specificity for that attribute.

The paper proposes two algorithms, a bottom-up clustering algorithm, and a top-down partitioning algorithm; both algorithms have $O(n^2)$ runtimes. The bottom-up clustering algorithm is very similar to clustering algorithms discussed later. The top-down algorithm recursively chooses two maximally different records to partition the dataset about, until the partitions are all minimal. The top-down partitioning seems to perform well on numeric data, but it is not clear how it could be extended to categorical attributes.

Stochastic algorithms The algorithms discussed above all used a greedy heuristic to bring a dataset to k-anonymity. There are a number of advantages to a single-pass search: it is easy to analyze the runtime of these algorithms, and a good heuristic can generate a high quality solution. However, several papers have looked into performing a local search through the solution-space to improve the quality of a solution.

One way to improve solution quality is via a stochastic search. The first use of a stochastic anonymization algorithm used a genetic algorithm to evolve globally recoded k-anonymous solutions[17]. In this algorithm, the genes on a chromosome represent the level of generalization for an attribute. When evaluating the quality of a solution, tuples in equivalence classes of size $< k$ were suppressed.

The possibility of using simulated annealing as a search strategy to do global recoding is discussed in [40]. The simulated annealing strategy would traverse the global recoding solution lattice discussed in section II.

[22] discusses how genetic algorithms can be used to improve the anonymization on a k-anonymized dataset as post-processing after a clustering algorithm has been run. In this formulation of a genetic algorithm, each chromosome contains a set of rows in the dataset, which is considered a cluster. Recombination randomly swaps portions of chromosomes; if the solution quality improves, the change is accepted, and if solution quality decreases, it is accepted with some $p < 1$.

While stochastic and genetic algorithms are potentially powerful ways to explore a large solution space,

the results from [22] show only a 2-5% improvement in information retention after the initial clustering algorithm. This indicates that while search algorithms can improve solution quality, an unguided solution will not generate solutions of substantially higher quality than a clever heuristic algorithm.

Summary

This section looked at different formulations of the privacy-preserving data disclosure problem, and discussed the differences between adversarial models and the difference between protecting against data disclosure vs identity disclosure. Different metrics for measuring how much data was lost in anonymizing a dataset were discussed, and some different data models were described. After laying out this framework, the last section discussed the algorithms existing in literature and what problem formulations they were designed to work on. The next section looks at the specific problem this thesis addresses and phrases it in the terminology used in this chapter, to get an idea of which algorithms will be applicable.

CHAPTER III

PROBLEM DEFINITION

Using the vocabulary generated in section II we can now get a better idea of how the original data anonymization problem discussed in the introduction relates to existing research. This section talks about the privacy model, data model and the quality metrics which need to be used on this problem, and see what existing algorithms could generate high-quality solutions.

Data Model

The interest targeting data being studied here has a very simple model, where each attribute is a boolean identifier, where an attribute is noted as either present or absent. There is no distinction between a *negative* value for an attribute and a *lack* of a value for the attribute, because no explicit negative attribute values are stored in the dataset. This is a slight distinction between the data used here and general categorical data. While a categorical attribute with values $\{true, false\}$ would be generalized to $\{true$ or $false\}$, here we are generalizing the domain {present, false not present} to {false or not present}–there is no way to tell if a value 'not present' represents a generalization of {Present, Not Present}, or if the attribute was simply never present. Here when an attribute is discussed as *suppressed*, it means that an attribute with value 'present' was replaced with 'not present.'

Note that using only boolean data does not limit the relevance of the techniques discussed here; all categorical data can be reduced as boolean data, albeit with a cost in compactness. Given the categorical data of profession shown in figure 6, and the set of individuals in Table 9, this attribute could be represented as the boolean attributes shown in Figure 10.

Formally, let D be the dataset we wish to anonymize in order to online the data; let $D = \{R_1 \cdots R_n\}$

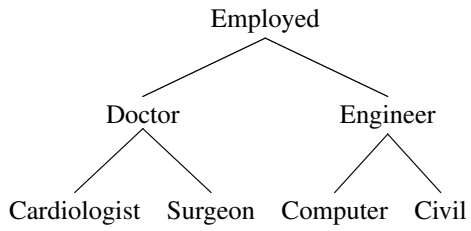

Figure 6: An example generalization hierarchy

Table 9: Hierarchical categorical professions

Person	Profession
A	Doctor
B	Surgeon
C	Civil
D	Employed

Table 10: Enumerated hierarchical data

Person	Employed	Doctor	Cardiologist	Surgeon	Engineer	Computer	Civil
A	1	1	0	0	0	0	0
B	1	1	0	1	0	0	0
C	1	0	0	0	1	0	1
D	1	0	0	0	0	0	0

where R_i is the record associated with an individual. Let $A = \{A_1 \cdots A_m\}$ be the set of all attributes potentially stored in a record; so $R_i = \{s_{i1} \cdots s_{il}\} \subset A$ is the collection of attributes stored for record i.

Adversary Model

The overall objective of anonymizing this targeting data is that the attributes stored in a cookie on a users' browser cannot be used to re-identify the user at a later date. None of the attributes are considered *sensitive* data, but instead represents publicly available information–in fact, the whole objective of placing the attributes in the first place is to let a website tailor content to a specific user *without* specifically identifying the user. Because we are only aiming to prevent identity disclosure and not sensitive data disclosure, we do not need to consider the privacy requirements of P-sensitive k-Anonymity, Bayes-Optimal privacy, *l*-diversity, (α, k) anonymity, *t*-closeness, or the other sensitive data disclosure models discussed in chapter II.

A strategy of straightforward k-anonymizing the dataset will ensure that no anonymized record can be traced back to an individual with high confidence. If the same set S of attributes is published for k individuals, seeing the set of attributes S on a browser can identify the user as nothing more than 1 of k individuals.

However, there is also a more relaxed anonymity model which could also provide an acceptable level of anonymity. The $(k, 1)$ anonymity model is a promising model for this problem. Since the targeting data is released in an on-demand fashion from the dataset, when a set of non-sensitive attributes S is released about an individual A, the user will remain anonymous so long as:

- S is a subset of the data stored for at least k individuals in the dataset, and
- S was nondeterministically selected among all the sets of segments which could be served for individ-

ual A which satisfy the above condition.

In other words, as long as there are k records for which S *could* have been published, and it is not recorded which attributes were published for each record, it suffices to $(k, 1)$ anonymize instead of k anonymizing the dataset.

Formal Models The two privacy models which seem promising for this problem are: (1) k-Anonymity, and (2) $(k, 1)$ anonymity. $(k, 1)$ anonymity is a more relaxed model, which will likely lead to higher utility datasets. However, the vast majority of research has so far focused on k-Anonymity. As a result we will study both models in this paper:

Definition 1. *An anonymized data set D is k-anonymous if $\forall i$, entry R_i = entry R_j for k distinct values of j.*

Definition 2. *A data set D is $(k, 1)$ anonymous with respect to the original data set G if for all $R_i \in D, R_i \subset R_j \in G$ for at least k distinct values of j. In other words, entry R_i could be an anonymized form of any of at least k records from G.*

Quality Metrics

In the case of anonymizing personalization data there is a well defined use-case, ie is a known monetary utility associated with each publishable attribute, and this utility is known prior to anonymization. Likewise, the utility of the anonymized dataset is simply the sum of the utilities of all publishable attributes.

Formally, we can define $u(a_i)$ as the *utility* associated with an instance of attribute i. Using the definition of quality defined above, we can define the utility of a record:

$$u(R_i) = \sum_{j=1}^{l} u(s_j)$$

and the utility of the published dataset is

$$u(D) = \sum_{i=1}^{n} u(R_i)$$

Table 11: Attributes with assigned numeric utility

i	a_i	$u(a_i)$
1	Biologist	10
2	Computer Scientist	5
3	Homeowner	3
4	Student	3

Table 12: Records with assigned numeric utility

i	R_i	$u(R_i)$
1	{Computer Scientist, Student}	8
2	{Computer Scientist}	5
3	{Homeowner}	3
4	{Biologist, Homeowner}	13

Since many recent algorithms approach anonymization as a clustering problem, it is helpful to define how this utility metric measures the utility of an equivalence of records, using the data model discussed in Chapter III. An *equivalence class* is a set of records such that the same attributes are published for every record in the class (in clustering algorithms a cluster is an equivalence class). Noting that the data here is all boolean and set-valued, the only way an attribute can be published for a record is if *every* record in the class contains that attribute. Let a equivalence class $C = \{R_1 \cdots R_m\}$ be a set of records where same set of attributes is to be published for each record R_i. If the same set of attributes $P = \{p_1 \cdots p_k\}$ is published for all records $\{R_1 \cdots R_m\}$, P must contain only attributes present in every record $R_i, i \leq m$. The maximal publishable set P is then:

$$P = \bigcap_{i=1}^{m} R_i$$

and the utility of the cluster C is

$$u(C) = |C| \sum_{i=1}^{k} u(p_i)$$

Scale

A defining characteristic of the dataset anonymization problem studied here is its size. Specifically, section V finds results on a dataset where $|D| = 20,616,891$), and $|A| = 1987$

The size of this dataset means that many published algorithms are not tractable. A naive pairwise distance calculation between all records has $O(n^2)$ complexity; a dataset in the millions of records makes the algorithm intractable. The next section takes note of which algorithms could be tuned to fit within a lower complexity bound.

The next chapter looks back at which algorithms from Chapter II will be useful on the problem as formulated here.

CHAPTER IV

ALGORITHMS

This section picks promising algorithms and presents in detail the algorithms from the set reviewed in Chapter II which will be evaluated in Chapter V. We are interested in finding and evaluating algorithms for both k-anonymity and (k, 1) anonymity, so this section outlines both the k-anonymity algorithms to be compared and a (k,1) anonymity algorithm.

Anonymity algorithms

Since the utility of the dataset anonymized here is a direct function of the number of attributes retained, and is not used for data mining purposes, the only advantage of a global recoding algorithm would be the tractability of the developed algorithms. Because the difference in the quality of datasets anonymized via local and global recoding has been extensively studied, and all results have shown significantly higher utilities with local recoding, this paper does not study global recoding

Mondrian Multidimensional k-Anonymity *Mondrian Multidimensional k-Anonymity* is a modern anonymization algorithm described in [20], and is appealing both because of its high-utility solutions and low runtime cost. The general idea behind the algorithm remains a top-down partitioning of the data; all records begin in the same equivalence class, and recursively a dimension is chosen to split the equivalence classes on, until there is no dimension which the class can be split on to produce valid k-anonymous clusters. This algorithm is shown as Figure 1.

The choice of which dimension to split on in $choose\_dimension$ was left open previously. In the implementation in [20], a heuristic is used to choose the dimension with the largest diameter in the equivalence class being split. The top-down partitioner tested in this paper is shown as Algorithm 3. The diameter of a dimension is not directly applicable here, so instead the heuristic *MOND_EVEN* chooses the attribute most frequent within an equivalence class.

In [13], a top-down partitioning algorithm very similar to one used here, has been applied to anonymize transactional data. In that algorithm an information gain heuristic was used to choose the dimension to split on. It is not immediately obvious which of these heuristics is better suited on this data, so both are evaluated

in Chapter V; in that section the algorithm *MOND_ENTROPY* chooses the dimension which minimizes the sum entropy in the split equivalence classes.

Algorithm 3 MONDRIAN: Multidimensional partitioning anonymization:

Input: partition P
Output: a set of valid partitions of P

```
1: if no allowable multidimensional cut for P then
2:     return P
3: else
4:     dim = choose_dimension(P)
5:     lhs = {t ∈ partition : t.dim = false}
6:     rhs = {t ∈ partition : t.dim = true}
7:     return partition_anonymize(rhs) ∪ partition_anonymize(lhs)
8: end if
```

Clustering As mentioned in Chapter II, clustering algorithms for k-anonymity have shown a great deal of promise. However, none of the algorithms described in [6][23]or [21] can be used without modification because of tractability problems with the nearest neighbor search each uses to select records to merge.

While a nearest-neighbor search can be done efficiently for low dimensional spaces with kd-trees or similar structures, in high dimensional spaces query times with these structures reduces to a linear search[15], giving each of these algorithms an effective runtime of $O(n^2)$. The most intuitive way to cut this runtime cost down is by using an approximate nearest neighbor search instead of an exhaustive one; instead of finding the nearest neighbor among all points, it can suffice to find the nearest among some sample of a fixed number of points and accept it. One naive implementation of this search chooses a random sample of points, and chooses the nearest neighbor among them. A more sophisticated implementation can select a sample of points bucketed via *Locality Sensitive Hashing*, so that the sampled points are more likely to share attributes with the queried point. It has been shown in previous clustering applications that runtime cost can be dramatically decreased via locality sensitive hashing without significantly impacting quality[18][32].

Algorithm 4 shows a version of the bottom-up clustering algorithm presented in [6]. Method $select\_record$ selects a record r such that $u(c \cup r)$ is high, $select\_class$ selects a cluster c such that $u(c \cup r)$ is high, and $select\_distant\_record$ selects a record such that $distance(r_{old}, r_{new})$ is high. The implementation of these procedures is discussed in more detail in this chapter. In Results in Chapter V the algorithm using a random sampling approach to find the nearest neighbor will be called *CLUSTER_RAND* and the algorithm using locality sensitive hashing is called *CLUSTER_LSH*. The original algorithm which computes the exact nearest neighbor is labeled *CLUSTER_ALL* and is used for reference on a smaller dataset.

Algorithm 4 CLUSTER: approximate clustering algorithm

Input: Data set $D = \{R_1 \cdots R_n\}, k$
Output: $S_{anon} = \{\{c_1 \cdots c_m\} : |c_i| \geq k\}$ where $\forall 1 \leq i \geq n, \exists j : R_i \in c_j$

```
S_anon = {}
r_old = random record from D
while |D| ≥ k do
    r_new = select_distant_record(D, r_old)
    D = D − {r_new}
    c = {r_new}
    while |c| < k do
        record r = select_record(D, c)
        remove r from D, add r to c
    end while
    S_anon = S_anon ∪ c
    r_old = r_new
end while
while |S| ≠ 0 do
    randomly remove a record r from D
    cluster c = select_class(S_anon, r)
    c = c ∪ r
end while
return S_anon;
```

Incremental Clustering Chapter II discussed genetic algorithms which stochastically improve the utility of a k-anonymized dataset. The local search algorithms described there all discuss ways to incrementally improve the utility of a solution by randomly moving through the solution space, retaining changes which result in higher quality solutions and discarding changes which decrease utility. The proposed changes in [17] and [22] are randomly selected, either through chromosome recombination or random bit flips. It is possible that an algorithm which tries heuristically guided changes rather than random ones could more quickly improve solution utility.

One way to guide the process of finding better solutions is to incrementally destroy "bad" clusters and reassign those records to clusters which are of higher utility, and to continue this process until the solution utility stops increasing. Algorithm 4 is an effective greedy clustering algorithm, but does not lend itself well to an incremental approach, as it builds clusters out of unassigned records rather than assigning records to clusters. Instead Algorithm 5 shows a clustering algorithm which improves the quality of a solution by iteratively breaking up the lowest-utility clusters and reassigning those records to new clusters. Within each increment are five steps: (1) de-clustering the worst $fraction$ clusters, (2) seeding new clusters to try to maintain N/k clusters (3) for each unclustered record, finding a high-utility cluster to place it in, (4) de-clustering all records in clusters of size $< k$, and (5) again for each unclustered record, finding a high-utility

cluster to place it in.

The number of clusters broken up is determined by the number of iterations specified by input l; if $l = 4$, first the lowest 75% of clusters are destroyed, then 50%, etc. Within each of these iterations, the process is repeated until the utility no longer increases. The highest-utility solution found is accepted as the anonymized dataset (this will often, but not always, be the S_{anon} at the termination of the algorithm.

The same nearest-neighbor search challenges discussed previously apply here as well; method $select\_class$ does the same approximate nearest-neighbor search as above. This incremental algorithm with an exact nearest-neighbor search here is described in the results as *ITER_CLUSTER_ALL*. When the search uses an LSH-guided search instead the algorithm is called *ITER_CLUSTER_LSH*.

Algorithm 5 ITER_CLUSTER: iterative clustering algorithm

Input: Data set $D = \{R_1 \cdots R_n\}, k, l$
Output: $S_{anon} = \{\{c_1 \cdots c_m\} : |c_i| \geq k\}$ where $\forall 1 \leq i \geq n, \exists j : R_i \in c_j$

```
1:  S_anon = {}
2:  old_utility = −1
3:  for fraction = 1 − 1/l; fraction > 0; fraction = fraction − 1/l do
4:     new_utility = u(S_anon)
5:     while new_utility > old_utility do
6:        De-cluster the fraction clusters in S_anon with highest intra-cluster distance
7:        while |S_anon| < N/k do
8:           S_anon = S_anon ∪ {random_record({r : r ∉ S_anon})
9:        end while
10:       for r ∉ S_anon do
11:          cluster c = select_class(S_anon, r)
12:          c = c ∪ r
13:       end for
14:       De-cluster all clusters c where |c| < k
15:       for r ∉ S_anon do
16:          cluster c = select_class(S_anon, r)
17:          c = c ∪ r
18:       end for
19:    end while
20: end for
21: return highest-utility solution found
```

Two-phase Aggregation A potential problem with all of the clustering algorithms described above is that for a large enough n, even an educated sampling approach will too often not find a close nearest-neighbor to the points being queried. On the other hand, two records could differ only along a single dimension, but if that dimension is chosen for the split on during the partitioning, the records will end up in different equivalence classes for a potentially high loss of utility.

Algorithm 6 proposed here is an attempt to reconcile these two potential pitfalls. On a high level, the algorithm described here works by starting each record in its own equivalence class, and incrementally generalizing each class one dimension at a time, until enough records are aggregated that each equivalence class is of size $\geq k$. In a second phase, the resulting clusters are split using a top-down partitioning like Mondrian described above uses.

The algorithm starts with each record placed in a cluster defined by its attribute vector; this also means that any two records with the same attribute sets will be in the same equivalence class. The smallest cluster $c_{smallest}$ is then selected, and the set $merged$ is built, which is the set of populated clusters (that is, classes which currently contain records) reachable by dropping one attribute from $c_{smallest}$. $select\_merge$ returns the cluster c from $mergeable$ which maximizes $u(c \cup c_{smallest})$.

If $merged$ is empty (as is often the case early in execution), $select\_drop$ selects a dimension to generalize. The algorithm chooses the dimension a which minimizes $u(a)| * \{R_i \in D|a \in R_i\}|$, a global weighted frequency measure. All records from $c_{mergeable}$ are moved to a new cluster without that attribute. This process continues until all records are in clusters of size $\geq k$. This procedure is shown in algorithm 7.

In the merging phase, because the merging is performed without a nearest-neighbor search, far more attributes will be suppressed than in an optimal solution. The second phase of the algorithm looks for clusters to which attributes can be added back. First, all records in a cluster are moved to a class which is a closure of the records in the cluster–this adds back to a cluster all segments which were not in the cluster's attribute vector. Second, the algorithm looks for attributes upon which the cluster can be *split*; it searches for an attribute which can be added to one or more record in the cluster without violating the k size threshold in either the source or destination cluster. The full logic of the algorithm is presented as Algorithm 7.

Algorithm 6 TWO_PHASE: two-phase aggregation algorithm

Input: Data set $D = \{R_1 \cdots R_n\}$
Output: $S_{anon} = \{\{c_1 \cdots c_m\} : |c_i| \geq k\}$ where $\forall 1 \geq i \leq n \exists j : R_i \in c_j$

1: initialize $S = \{\{R_1\} \cdots \{R_n\}\}$
2: merge all clusters in S with identical attribute vectors
3: $S = merge(S)$
4: $S = split(S)$
5: **return** S

Approximate (k, 1) Algorithm (k, 1) anonymity, because it protects against a weaker adversary model than k-anonymity, has not been the study of extensive research. Here we describe a modification to the original algorithm proposed in[12]. Algorithm 2 describes the original proposed (k,1) anonymity algorithm.

Algorithm 7 $merge$: merge procedure

Input: list of clusters $S = \{c_1 \cdots c_n\}$
Output: $S_{anon} = \{\{c_1 \cdots c_m\} : |c_i| \geq k\}$ where $\forall 1 \geq i \leq n \exists j : R_i \in c_j$

```
sort S such that ∀i, j|c_i| < |c_j| ↔ i < j
while ∃c_i ∈ S : |c_i| < k do
    let c_smallest = S.pop_smallest
    mergable = {c_merge ∈ S : (∃i : (c_smallest − {a_i}) = c_merge)}
    if mergable not empty then
        c_selected = select_merge(mergable, c_smallest)
        c_selected ⇐ c_smallest
    else
        a_drop = select_drop(c_smallest)
        (c_smallest − {a_drop}) ⇐ c_smallest
    end if
    update S with new sizes
end while
return S
```

The algorithm is a bottom-up agglomerative algorithm; for each record R_i in the table, it initializes a cluster $S_i = \{R_i\}$. Then until $|S_i| \geq k$, the algorithm chooses a record R_j to add to S_i which minimizes the diameter of cluster $S_i \cup R_j$.

While the original paper used the cluster diameter metric d to measure the anonymization cost, in Chapter III we developed a different utility-based quality metric; the version of algorithm 2 used for evaluation in this paper uses this utility metric.

As discussed earlier, finding a record which actually maximizes $u(S_i \cup R_j)$ reduces to a high-dimensional nearest-neighbor search, which we develop algorithms for later. This algorithm uses the same LSH sampling strategy for $select\_record$ as developed for algorithm 4. In the results, this algorithm is referred to as *K1_LSH*. For reference, the K1 algorithm which conducts a full nearest-neighbor search is labeled *K1_ALL* there.

Algorithm 9 shows the algorithm after these modifications.

Approximate Nearest Neighbor Search

The methods $select\_record$ and $select\_distant\_record$ used in Algorithms 4 and 9 rely on the ability to quickly find the nearest-neighbor to a point in a very high dimensional space. In general, this is an open problem; however, this section describes the approximation algorithm used here.

In a low-dimensional space, efficient space-partitioning structures like the kd-tree allow a nearest-neighbor search in sublinear time. On the other hand, on a small dataset, even in a high dimensional space, all these distance relationships can be easily found naively by an $O(n^2)$ search. Neither of these conditions holds on

Algorithm 8 *split*: split procedure

Input: list of clusters $S = \{c_1 \cdots c_l\}$, k
Output: set of clusters $S = \{\{c_1 \cdots c_m\} : |c_i| \geq k\}$

Let $to\_check = \{\}$
for cluster $c_{to\_close} = \{r_1 \cdots r_l\} \in S$ **do**
  Let $c_{closed} = closure(r1 \cdots r_l)$
  $c_{closed} \Leftarrow c_{to\_close}$
  $to\_check.add(c_{closed})$
end for
$S = \{\}$
while to_check not empty **do**
  Let $c_{to\_split} = to\_check.pop$
  for attribute $a \notin c_{to\_split}$ **do**
    Set $dest\_cluster = (c_{to\_split} + a)$
    $records = |(c_{to\_split})|$
    $records\_with\_attribute = \{R \in c_{to\_split} : a \in R\}$
    $count\_with\_attribute = |records\_with\_attribute|$
    $destination\_records = |dest\_cluster|$
    if $destination\_records > 0 \cap records > k \cup (records \geq 2 * k \cap count\_with\_attribute \geq k)$ **then**
      for record $R \in records\_with\_attribute$ **do**
        if $|c_{to\_split}| = k$ **then**
          break
        end if
        $dest\_cluster \Leftarrow R$
      end for
      $to\_check.push(c_{to\_split})$
      $to\_check.push(dest\_cluster)$
    else
      $S.add(c_{to\_split})$
    end if
  end for
end while
return S

Algorithm 9 K1_ANON: approximate (k,1) algorithm

Input: Table $D = \{R_1 \cdots R_n\}$, integer k
Output: Table $g(D)$ that satisfies $(k, 1)$ anonymity

for all $1 \leq i \leq n$ **do**
  Set $S_i = \{R_i\}$
  while $|S_i| < k$ **do**
    record $R_j = select\_record(D, S_i)$
    Set $S_i = S_i \cup \{R_j\}$
  end while
end for
Define $\bar{R}_i$ to be the closure of S_i

the dataset in question here, which has a dimensionality of over a thousand and millions of records. Unfortunately, when both the dimensionality and n are large, the infamous "curse of dimensionality" [15] makes it difficult to perform an efficient nearest-neighbor search. The curse of dimensionality refers to the fact that the data structures used in a low-dimension nearest-neighbor search do not scale to high-dimensional spaces[16]; for example, the kd-tree [28] take close to linear search time at high dimensions. While structures have been proposed with sub-linear query times even for high dimensions, these structures all require storage exponential with respect to the dimensionality[15]. To date, there is no known sub-linear query time algorithm with polynomial space requirements.

While no optimal algorithms have been found with this behavior, there has been considerably more success at designing algorithms which are able to efficiently query for an approximate nearest neighbor. One of the most popular families of algorithms for approximate nearest neighbor search is *locality sensitive hashing* (LSH)[4]. The idea behind LHS is that if a hashing function can be found such that collisions between neighboring objects is much more frequent than collisions between distant objects, neighbors can be easily found by hashing the point being queried and searching only through objects which collide with the query point with regards to one or more hash functions.

The standard LSH algorithm over Hamming spaces[11] defines l hash functions $g_1 \cdots g_l$. Each of these functions g_i hashes each point in the dataset into one of M buckets, in effect replicating the dataset l times. Procedure 10 generates the l hash functions and stores each of the queryable records in each of l hash tables. Algorithm 11 then queries for the nearest neighbor of a query point q by finding the closest neighbor among all points which fall into the same bucket in one of the l tables.

The original algorithm proposes building hash functions by choosing l subsets of $I_1 \cdots I_l$ with k bit positions $\{1, \cdots, d'\}$, and projecting each record onto a bucket by concatenating the value at these bit positions. The k bits are chosen "uniformly at random with replacement". For example, say a dataset has dimensionality $d = 10$, and for $k = 3$, three positions are chosen to get $I_1 = \{12, 34, 45\}$, to make hash function $g_1(p) = p_{|I_1}$. So a record $r = \{2, 10, 34, 56\}$ is hashed into bucket $g_1(r) = 010_2 = 2$.

However, this hashing algorithm leads to undesired hash function behavior when the overall frequency of many bits is $<<$ .5 or $>>$ .5, in which case the vast majority of records will fall in buckets 0 or $2^k - 1$, respectively. To prevent unbalanced hash functions, procedure 12 is used instead to build hash functions, and procedure 13 hashes a record. Essentially, the hash function, instead of splitting on a single dimension, combines bit positions until the probability that a record has *at least* one of the bit positions set is approximately .5s.

Algorithm 10 $preprocess$: generate the nearest neighbor hash

Input: number of hash tables l, set of records D, split dimensions k
Output: set of hash tables $T = \{T_i, i = 1 \cdots l\}$

for $i = 1 \cdots l$ **do**
  initialize hash table T_i with hash function $g_i = build\_hash(k)$
end for
for record $r \in D$ **do**
  for $i = 1 \cdots l$ **do**
    store record r in bucket $hash\_record(r, g_i)$ of table T_i
  end for
end for
return T

Algorithm 11 $find\_nn$: find the nearest neighbor of a point

Input: query record q, hash tables and functions $T_i, g_i, i = 1 \cdots l$
Output: an approximate nearest neighbor record r

$S =$
for $i = 1 \cdots l$ **do**
  $S = S \cup T_i(hash\_record(q, g_i))$
end for
return record r in S which minimizes $d(q, r)$

Algorithm 12 $build\_hash$: generate an LSH hash function

Input: split dimensions k
Output: a list of bitsets to use as a hash function

$bitset\_list = \{\}$
while $size(bitset\_list) < k$ **do**
  $bitset = \{\}$
  while $(1 - \prod_{b \in bitset} (1 - p(b))) < .5$ **do**
    $bitset = bitset \cup random(\{1 \cdots d\})$
  end while
  $bitset\_list = bitset\_list \cup bitset$
end while
return $bitset\_list$

Algorithm 13 $hash\_record$: hash a record into an LSH bucket

Input: Record r, hash dimensions $bitset\_list$
Output: the bucket b into which record r is hashed

$key = 0$
for $bitset$: $bitset\_list$ **do**
  $key << 1$
  for j : $bitset$ **do**
    if $j \in r$ **then**
      $key = key|1$
    end if
  end for
end for
return key

Last, while the algorithm in [11] calculates how to set l, M and k to attain certain approximation bounds, in this application the bounds can be set more directly by memory and runtime constraints. The entire dataset is replicated in memory l times, so a constant bound of $l = 5$ is used while testing here. Likewise, runtime is bounded by the number of points queried on each call to FIND_NN. For an even hash function, $l(n/2^k)$ points are queried each call. Let the desired number of points to query be $opt\_queries(n) = c * log(n)$ to keep the runtime of a query $O(log(n))$; then $k = log_2(l * n/opt\_queries(n))$. A constant $c = 50$ was used.

Now that Chapter III has formalized the anonymous personalization targeting problem, and this chapter has presented a number of algorithms designed to solve the data anonymity problem described there. The next section ties these together by evaluating the solution quality and efficiency of the algorithms on this data.

CHAPTER V

EXPERIMENTS

The algorithms outlined above were evaluated on three datasets. The first set of tests was run on the Adults dataset from the UC Irvine Machine Learning Repository[9]. The second set was on a synthetic dataset designed to emulate a dataset with a sparse, long-tail distribution. The last test is on a large subset of Rapleaf's interest targeting data, with a sparse distribution similar to the one emulated by the synthetic dataset.

The experiments in the first two sections were run on a Thinkpad W500 with 3.8 GB of memory and an Intel P8700 Core 2 Duo CPU @2.53 GHz. The last section was run on machine with two Six-Core AMD Opteron processors and 64 GB of memory.

Adults Dataset

The Adults dataset from the UC Irvine Machine Learning Repository[9] is commonly used to compare k-anonymity algorithms. This dataset contains anonymous census data on 48842 individuals from the 1990 US census. The data fields represented in the set is shown below:

- Age: continuous numeric
- Employment type: {Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.}
- Education: {Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.}
- Years of education: continuous
- Marital status: {Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse}
- Occupation: {Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspect, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.}
- Relationship: {Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried}

- Race: {White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black}
- Sex: {Female, Male}
- Capital gain: continuous
- Capital loss: continuous
- Hours worked per week: continuous.
- Native country: {United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinidad and Tobago, Peru, Hong, Holland-Netherlands}

The categorical attributes "Employment", "Education", "Marital Status", "Occupation", "Relationship", "Race", "Sex", and "Native Country" were used by enumerating each of the categorical attribute values into a distinct attribute, like discussed in Chapter III. The numeric fields "Age" and "Hours Worked per Week" were bucketed into ranges of diameter 5 and 10 respectively and treated as categorical data.

Since the original dataset does not attach any particular utility value to any of the attributes, all attributes are assumed to have the same utility in the end dataset, and the utility of a solution is just the count of all attributes in the anonymized dataset.

LSH vs Random Sampling vs Full Clustering

Intuitively the LSH sampling approach described in section IV will produce higher utility anonymizations than the a naive random sampling approach. The hope is that the quality of a solution produced via LSH sampling will also be comparable to that produced if *all* points are sampled, which is the approach taken by standard clustering algorithms; this would suggest that we can turn an $O(n^2)$ clustering algorithm into an algorithm competitive on large datasets.

Figure 7 compares the utility of the adult dataset anonymized with algorithms *CLUSTER_RAND* and *CLUSTER_LSH* (the clustering algorithm shown in algorithm 4 using random sampling and LSH guided sampling), and figure 8 shows the runtimes of the respective algorithms. *CLUSTER_ALL* shows for reference the upper bound utility with that algorithm for any sampling strategy.

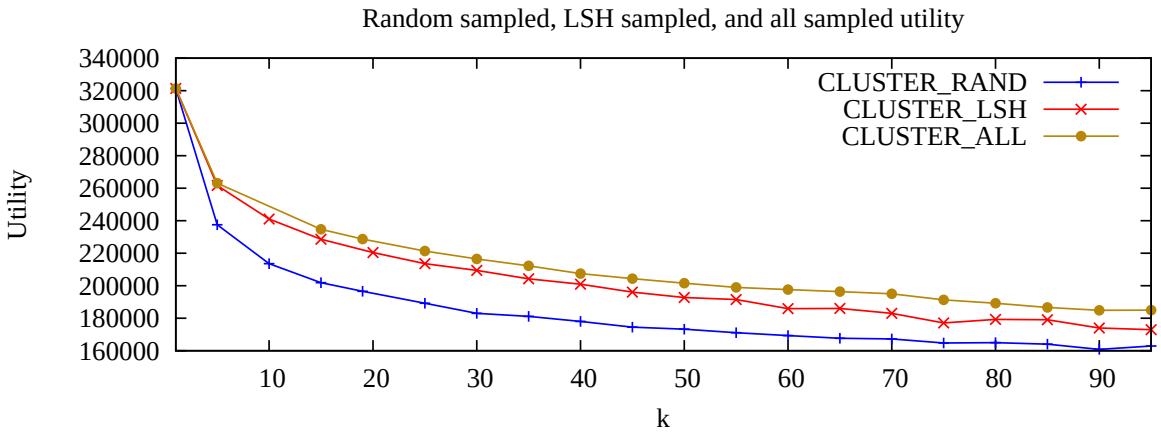

Figure 7: The utility of k-anonymized solutions using clustering with random sampling vs clustering with LSH guided sampling

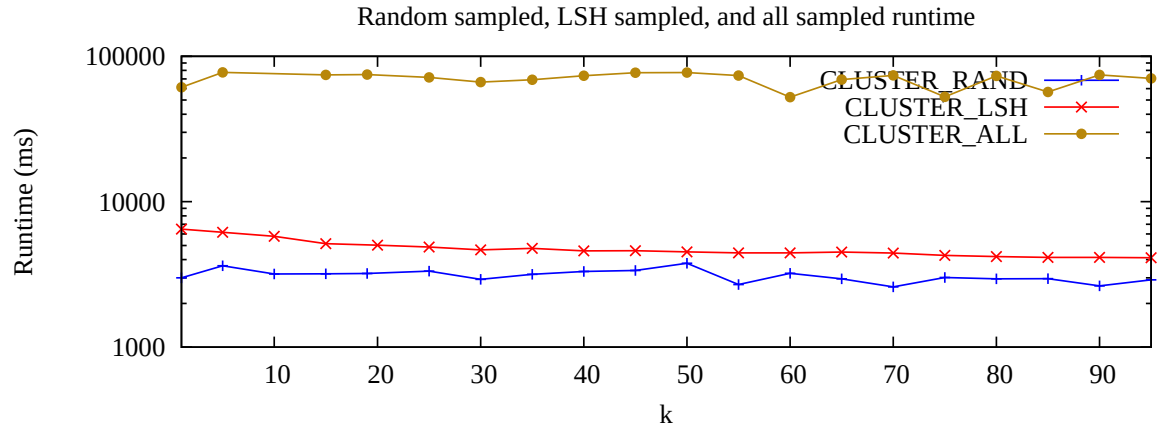

Figure 8: The runtime of clustering anonymizers using random sampling vs clustering with LSH guided sampling

Figure 7 indicates that sampling from an LSH table produces anonymized tables with significantly higher utility than that of an uninformed sampling strategy; at $k = 10$, the table has 12.8% higher utility, close to the upper-bound utility. As k increases and all strategies begin converge the boost decreases to 3.4%. The runtime of the algorithm with LSH sampling is consistently around $1500ms$ higher than random sampling; most of this can be attributed to the cost of hashing all of the records into the l hash tables. Even this higher runtime though is less than 2% of the runtime of the clustering algorithm with a full nearest-neighbor search shown by *CLUSTER_ALL*. Overall, with higher-utility solutions and a not substantially higher runtime cost, LSH guided sampling can be considered to solidly outperform unguided sampling.

Iterative vs Single-Pass Clustering

Algorithm 5 described an iterative clustering algorithm to incrementally improve the utility of a k-anonymized dataset by breaking up clusters and re-clustering those records. The hope is that at least under some conditions this algorithm will outperform a classical clustering algorithm like algorithm 4. Figure 9 compares the exact clustering algorithm *CLUSTER_ALL* with the exact iterative clustering algorithm *ITER_CLUSTER_ALL*; the runtimes are shown in figure 10.

Interestingly, on this data the classical clustering algorithm provides higher utility at k-values lower than 30, while the iterative algorithm provides higher utility at higher k values–at $k = 150$ the utility from the incremental algorithm is around 9.7% higher. The disparity at low k-values seems to be an artifact of the different base clustering strategy used by algorithm *ITER_CLUSTER_ALL* rather than a loss of utility from the iterative procedure. The runtime of the iterative algorithm at low k values is considerably higher, but converges to a runtime around twice that of the base clustering algorithm at high k values.

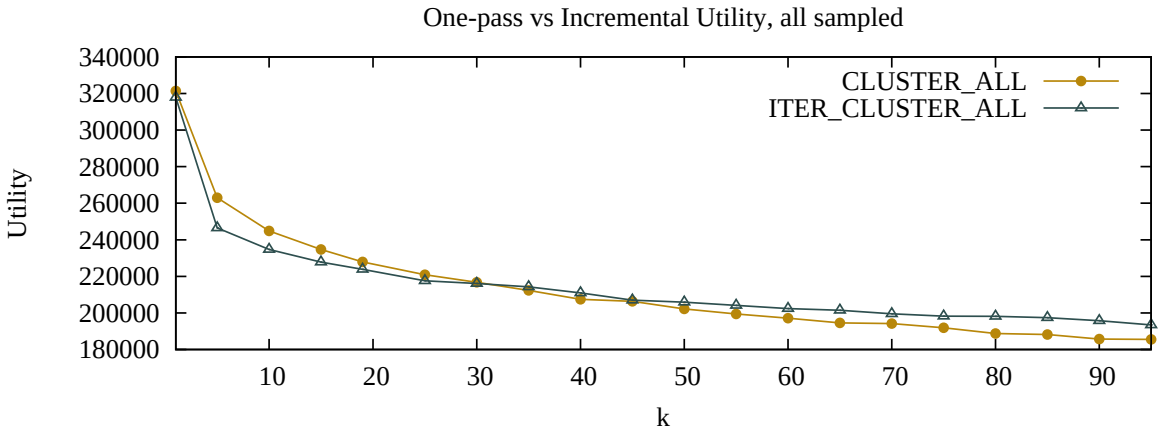

Figure 9: Utility of clustering anonymized solutions using clustering with random sampling vs clustering with LSH guided sampling

Figures 11 and 12 show the utility and runtime of these algorithms when an LSH-guided approximate nearest neighbor search is used instead of an exact search. Here the iterative algorithm performs more strongly in comparison, providing higher utility at k-values higher than 20, for 26.6% higher utility at $k = 180$. Unfortunately, the runtime is also far higher, taking more iterations to converge on a solution.

Overall neither the incremental algorithm nor the classical clustering algorithm dominates on this data; the iterative algorithm dominates for high k values so long as the runtime cost is manageable, while the classical algorithm outperforms otherwise.

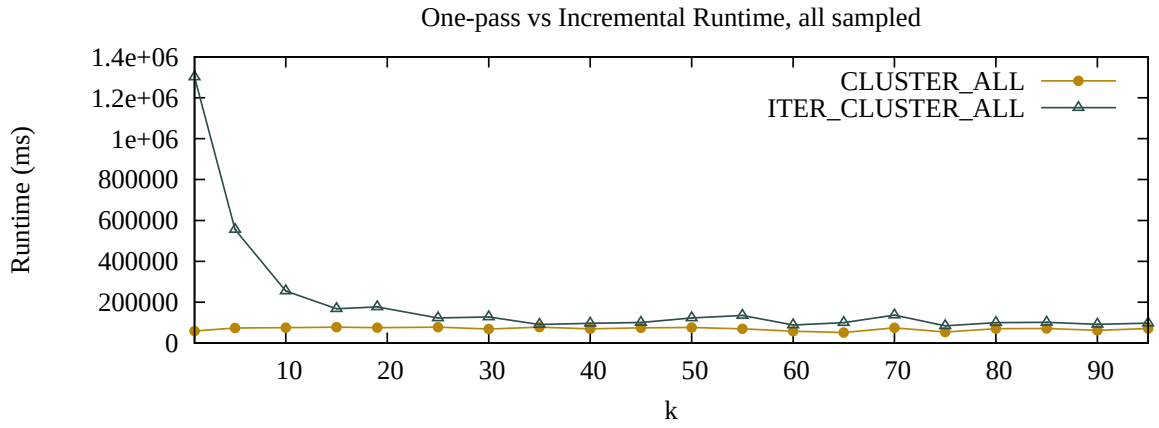

Figure 10: Runtime of clustering anonymizers using random sampling vs clustering with LSH guided sampling

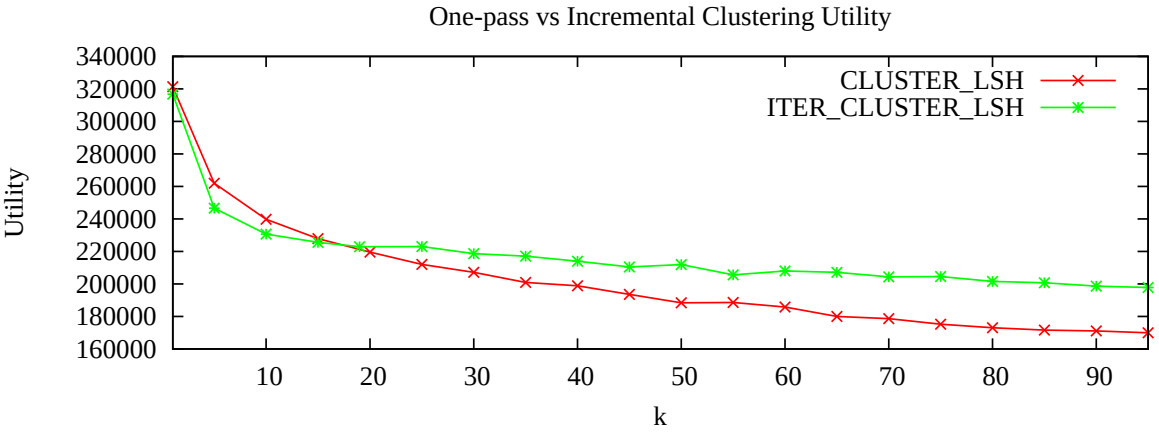

Figure 11: Utility of clustering anonymized solutions using single-pass clustering vs iterative clustering algorithm

Mondrian multidimensional heuristics

The Mondrian multidimensional k-anonymity algorithm discussed in Chapter IV leaves open the choice of dimension to split on. Two heuristics have been proposed; (1) greedy splitting on the attribute present in the most records in an equivalence class, and (2) splitting on the dimension which produced equivalence classes with the lowest entropy, with the idea that the more homogeneous classes could be further split.

Figures 13 and 14 show the utility and runtime of the algorithm with the two heuristics. Interestingly, the two produce almost identical results–at $k = 195$, the entropy-based heuristic has a utility value only $.098\%$ higher than the greedy heuristic, suggesting that the two heuristics nearly always split on the same dimensions anyway. However, the runtime of the entropy heuristic was several orders of magnitude higher

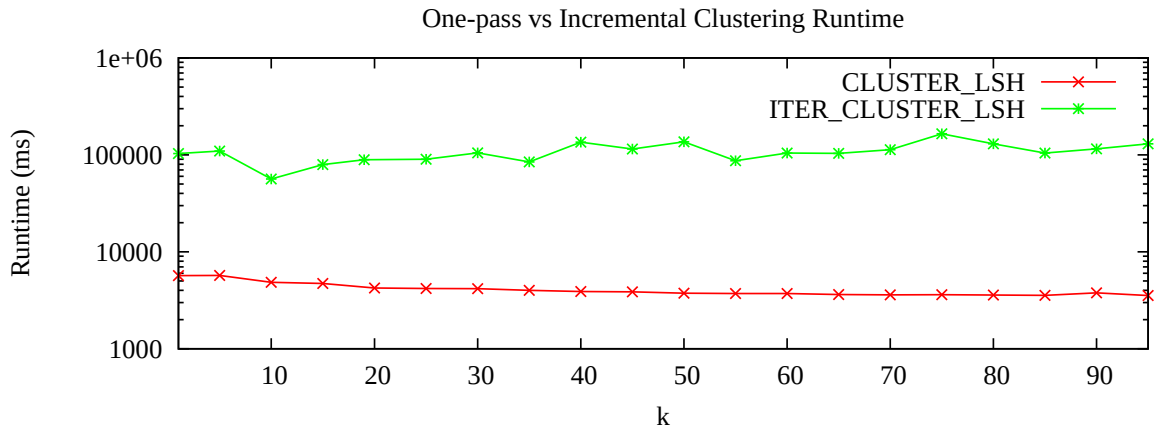

Figure 12: Runtime of clustering algorithms using single-pass clustering vs iterative clustering algorithm

than the greedy heuristic; this suggests that the greedy heuristic can reasonably be used in place of the entropy-based heuristic.

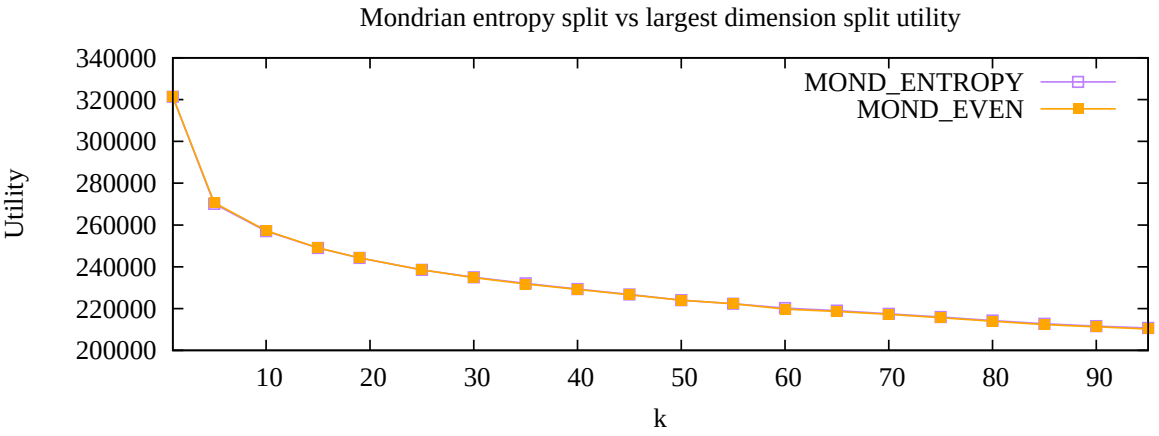

Figure 13: Utility of Mondrian anonymized solutions using an information gain heuristic vs a largest-value heuristic

(K,1) anonymity algorithms

The (k, 1) algorithms K1_LSH and K1_ALL closely resemble the bottom-up clustering algorithms CLUSTER_LSH and CLUSTER_ALL, respectively. Each algorithm starts with a cluster root and adds records into the cluster until it is of size k. However, K1 algorithms are able to reuse these records and boost the k values of many clusters, while algorithms for k-anonymity can assign a record to only a single cluster. So it is expected that these algorithms will respectively have higher utility than pure k-anonymity algorithms. This is supported by the results in Figure 15.

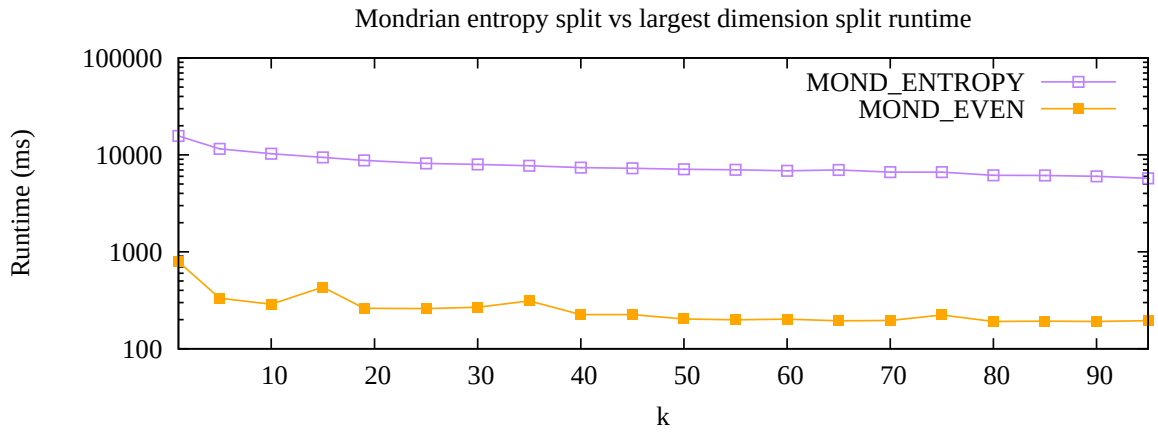

Figure 14: Runtime of Mondrian using an information gain heuristic vs a largest-value heuristic

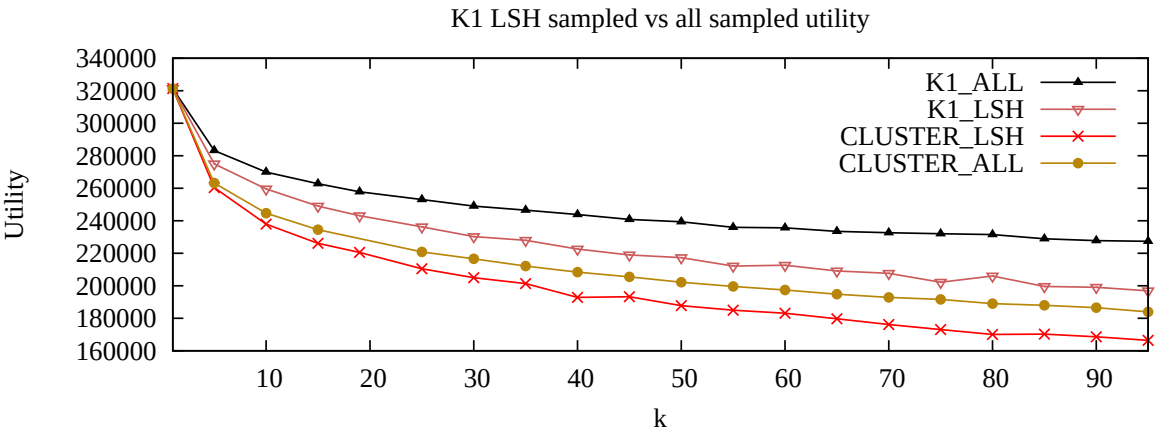

Figure 15: Utility of (k,1)-anonymized solutions

Overall comparison

The goal of the previous comparisons was to understand how the conditions affected the performance of each of the anonymization algorithms. Some of the algorithms were not competitive under any condition, namely those which used uneducated sampling rather than LSH-guided sampling. Figures 17 and 18 directly compare the utility and runtime of five of the algorithms compared above. Additionally, the two-phase approach described by Algorithm 6 is also presented.

Overall the two-phase anonymizer and Mondrian anonymizer strongly outperform all bottom-up clustering algorithms. The highest-utility clustering algorithms, the incremental algorithm with LSH hashing and the incremental algorithms with an exact search incur extremely high runtime costs. CLUSTER_LSH, while it did have reasonable runtime, did not generate very high utility solutions.

As would be expected, the full $(k, 1)$ anonymization algorithm described earlier does dominate all of the

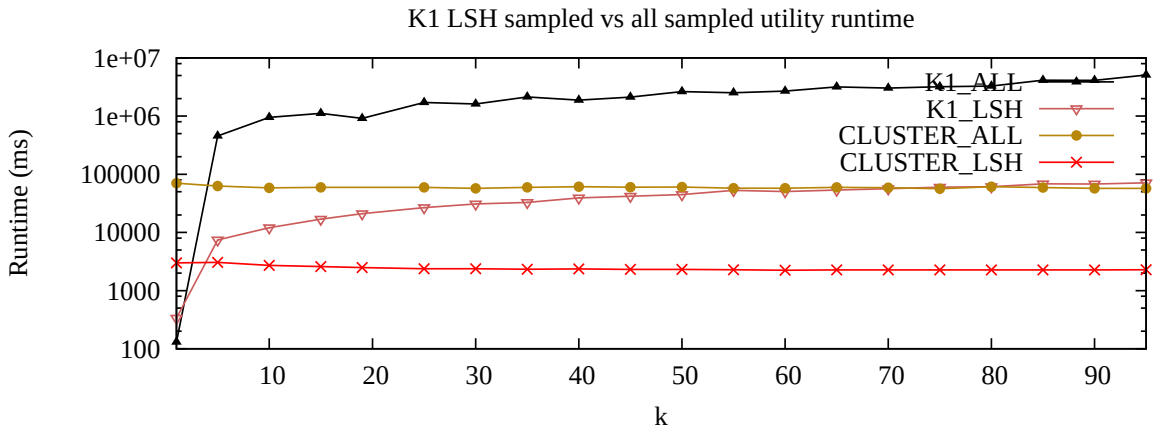

Figure 16: Runtime of (k,1)-anonymization algorithms

other algorithms in utility retained, emphasizing that that anonymity model can be attained at significantly less cost than k-anonymity. However, the attempt to modify this $O(n^2)$ algorithm to run more efficiently by sampling from an LSH hash was less successful, only roughly matching TWO_PHASE and Mondrian at low k values.

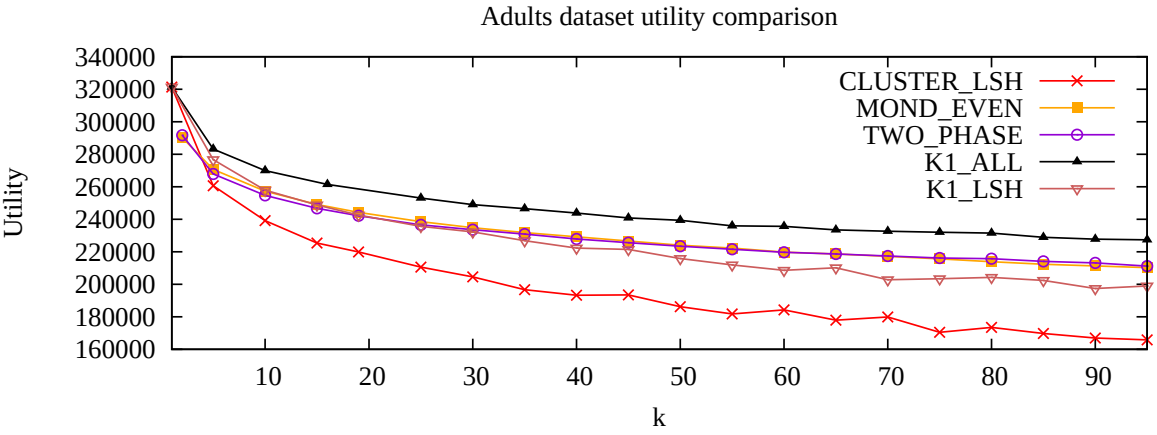

Figure 17: Side-by-side comparison of algorithm utilities on the Adults dataset

Synthetic Data

The dataset used in the previous section is a good benchmark for comparing anonymity algorithms, but for several reasons the dataset is not representative of the targeting data which is the main focus of the paper. One significant difference is the distribution of attribute frequencies in the processed Adults dataset versus the distribution of frequencies in the targeting data being studied. Figure 19 illustrates this difference, by

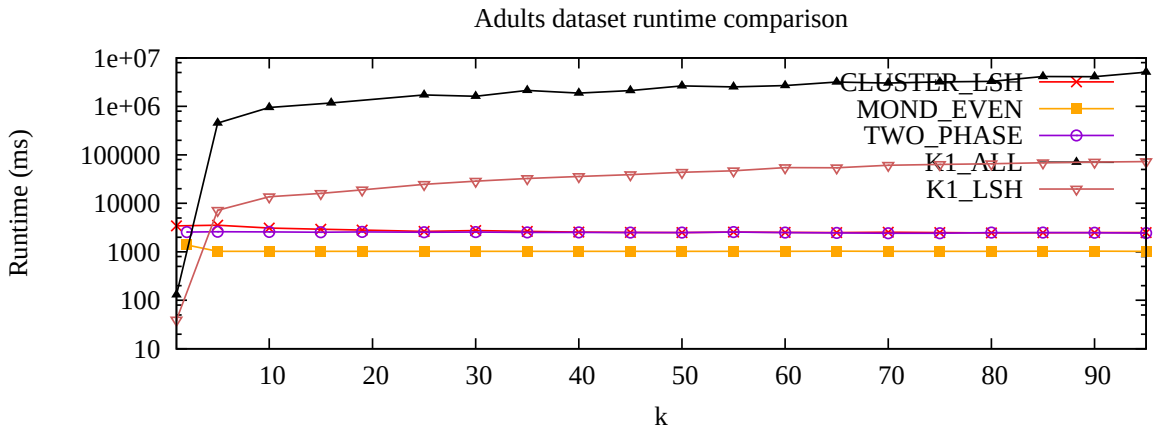

Figure 18: Side-by-side comparison of algorithm runtime on the Adults dataset

ordering the attributes by their global frequency across the dataset for the targeting set and the Adults dataset. The Adults dataset is clearly longer-tailed, with a large number of attributes appearing only at low frequency. Another difference arises from the structure of the data in the Adults dataset. While the expanded set contains 123 attributes, no record ever had more than 10 attributes set as "true" since each had only a single value for each of the original categorical attributes. Last, in the adults dataset each attribute was considered to have equal utility, while we would like to be comparing algorithms over data with explicit utility.

To address these three concerns we constructed a synthetic dataset with the previous concerns in mind. The dataset was constructed with dimensionality $d = 1000$, with frequencies between .5 and .001 to fit the logarithmic curve of the Targeting data shown in Figure 19. 100000 records were generated sampling attributes from this distribution, with a mean of 40 and a standard deviation of 24. Last, each attribute was assigned a numeric utility uniformly sampled from $[1, 100]$.

The algorithms which performed reasonably on the adult dataset at a reasonable runtime–namely, the single-pass clustering algorithm CLUSTER_LSH, heuristic-guided Mondrian MOND_EVEN, the new proposed algorithm TWO_PHASE, and the (k,1) anonymity algorithm K1_LSH–were tested on this set, and the results are shown in figures 20 and 21.

The algorithms actually perform quite differently than on the Adults dataset with respect to one another. TWO_PHASE consistently outperformed each of the other strategies, even though on the Adults dataset it matched the performance of Mondrian. This difference is likely because it operates with a stronger notion of attribute utility, allowing it to more carefully retain high-value attributes. The heuristic Mondrian uses to split equivalence classes does not factor attribute utility into its calculations, which did not matter on the Adults dataset but is a problem here.

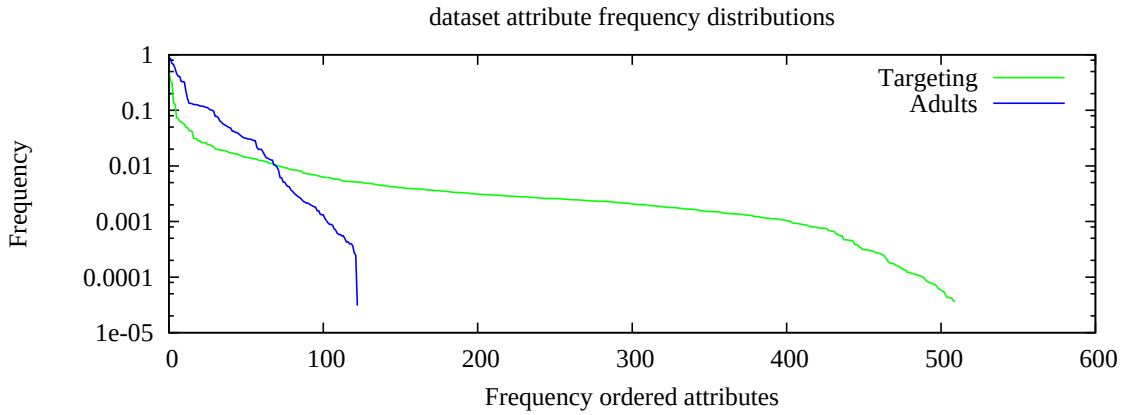

Figure 19: Attribute frequency distributions

Interestingly, although it did not perform well on the Adults dataset, at high k-values here CLUSTER_LSH is actually competitive with Mondrian. Like with the discussion above, this is likely because the distance metric used while clustering explicitly factors utility into its distance metrics, allowing it to preserve higher-value attributes.

One other surprise was that even though (K,1) anonymity is a strictly more lenient privacy model, the (k,1) anonymity algorithm performed very poorly compared to k-anonymity algorithms when $k > 5$. This indicates that even though the model allows for higher-utility solutions, the approximation algorithm proposed here is not strong enough to find these solutions. A possible avenue for further exploration that was not studied here is to approach (k,1) anonymization via top-down partitioning like strategy Mondrian uses.

Targeting Dataset

There were two main points to the previous two sections; first, to make sure the algorithms studied here remained competitive on a well-known benchmark, and second, to narrow down the field of algorithms to test

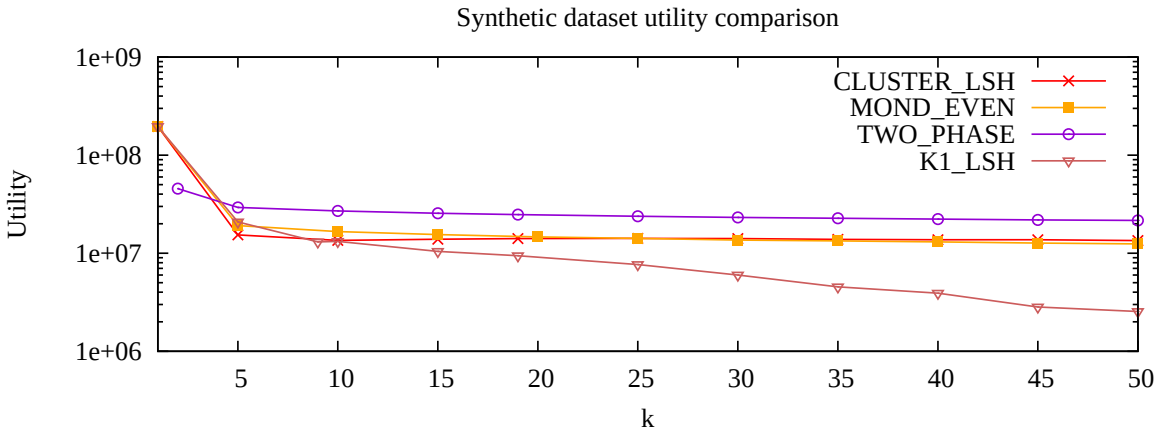

Figure 20: Side-by-side comparison of algorithm utility on a synthetic dataset

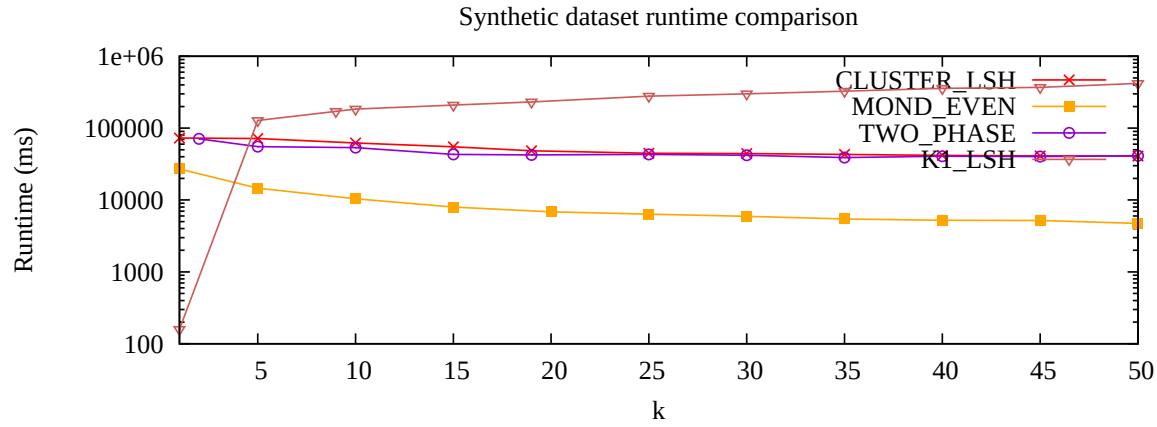

Figure 21: Side-by-side comparison of algorithm runtime on a synthetic dataset

on the large dataset mentioned in the introduction. The synthetic dataset in the previous section was modeled on properties of the real targeting data and is likely to be a good indicator of which algorithms are competitive on the full dataset. As mentioned previously, this set contained $20,616,891$ records and 1987 attributes.

The heuristic-guided Mondrian algorithm and the two-phase algorithm from section V were tested and the results are shown in Tables 13, 14, and 15. Table 13 shows the percent of the dataset's original utility which is retained post-anonymization, Table 14 shows the percent of the original attributes which are retained, independent of utility, and Table 15 shows the runtime of each of the instances. The clear strengths of

Table 13: Percent of utility retained on targeting dataset

algorithm \k	2	10	20	40
MOND_EVEN	0.434	0.242	0.227	.214
TWO_PHASE	0.454	0.273	0.255	.320

Table 14: Percent of attribute instances retained on targeting dataset

algorithm \k	2	10	20	40
MOND_EVEN	0.432	0.243	0.228	.216
TWO_PHASE	0.420	0.221	0.201	.156

Table 15: Algorithm runtime on targeting dataset

algorithm \k	2	10	20	40
MOND_EVEN	0.90	.840	1.04	.792
TWO_PHASE	5.69	5.04	4.96	5.23

Mondrian are the high attribute retention it provides and a very efficient runtime. However, we do see in Table 13 that the lack of a utility guide led it to retain lower-utility attributes than a utility-guided approach.

CHAPTER VI

DISCUSSION AND CONCLUSIONS

This thesis has discussed a practical anonymity problem in the domain of internet personalization and advertising. While access to interest and personalizing data overall leads to a better online experience for consumers and businesses, it is important that internet users are not personally identified (at least not without their knowledge). Given enough innocuous data, a person can be de-anonymized by searching with a unique combination of attributes. By studying work in the field of *privacy preserving data disclosure*, we can try to find a privacy model which prevents this from occurring.

While a wide range of privacy models have been developed to prevent *sensitive attribute* disclosure, none of the data here is considered sensitive data, and instead we can focus on preventing identity disclosure. Two promising privacy models are the *k-anonymity* and *(k, 1)-anonymity* models. The k-anonymity privacy model has been well studied in literature, and a number of algorithms have been tested. The (k,1) model is a more specialized model which has not been studied in detail, but can potentially produce higher-utility solutions. A lot of strong algorithms have been developed for k anonymity, so comparing them was the focus of this paper, but a (k, 1) anonymity algorithm was also compared to get an idea of its potential benefits.

Often k-anonymity algorithms take the form of bottom-up clustering algorithms. Unfortunately, most of these algorithms are formulated as relying on an $O(n^2)$ nearest-neighbor search. However, by using LSH hash functions and only sampling the space for a nearest neighbor, often the utility of an approximate solution can approach that of one with a full nearest neighbor search. While it has been previously shown that LSH functions can enable efficient clustering, this emphasizes that anonymization is another field which benefits from this approach. Another idea studied for improving the utility of a set anonymized with approximate methods was an incremental algorithm inspired by simulated annealing, where low-utility clusters were broken up and rebucketed until the solution stabilizes. While this does definitely lead to higher quality solutions, it comes at the cost of higher runtime.

The results clearly showed that the (k,1) anonymity model can produce higher-utility solutions than the k-anonymity model when it is applicable. The relatively naive $(k, 1)$ algorithm from [12] generated the highest-utility solutions out of all the algorithms studied, but attempts to scale this algorithm to larger datasets failed. A topic for future study is whether a top-down partitioning algorithm (which will likely be $O(nlog(n))$) can be developed instead of the inefficient bottom-up one used here, to combine the computational efficiency of

Mondrian with the utility of a more relaxed model.

An observation from the utility-oriented datasets studied in the last two parts of the results pointed out that algorithms which perform worse than Mondrian on even-utility datasets can outperform it when attributes are associated with possibly disparate utilities. A natural next step would be to develop a heuristic for Mondrian which chooses a split dimension based on maximizing utility rather than just the number of attributes.

In conclusion, it was very encouraging to see that current anonymity techniques are capable of anonymizing to a reasonable level of privacy while still retaining quite significant amounts of useful data–on the targeting dataset studied, $k = 20$ was achieved while still maintaining 26% of the set's utility. This provides hope that with enough investment in privacy-oriented research, a more personalized and information-rich online life does not always have to come at the cost of privacy or anonymity.

REFERENCES

[1] Anonymouse. http://blog.rapleaf.com/dev/2010/07/20/anonymouse/, 2010.

[2] https://www.rapleaf.com/, 2011.

[3] Gagan Aggarwal, Tomas Feder, Krishnaram Kenthapadi, Rajeev Motwani, Rina Panigrahy, Dilys Thomas, and An Zhu. Approximation algorithms for k-anonymity. In *Proceedings of the International Conference on Database Theory (ICDT 2005)*, November 2005.

[4] Alexandr Andoni. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *In FOCS06*, pages 459–468. IEEE Computer Society, 2006.

[5] Roberto J. Bayardo and Rakesh Agrawal. Data privacy through optimal k-anonymization. In *Proceedings of the 21st International Conference on Data Engineering*, ICDE '05, pages 217–228, Washington, DC, USA, 2005. IEEE Computer Society.

[6] Ji-Won Byun, Ashish Kamra, Elisa Bertino, and Ninghui Li. Efficient k-anonymization using clustering techniques. In *Proceedings of the 12th international conference on Database systems for advanced applications*, DASFAA'07, pages 188–200, Berlin, Heidelberg, 2007. Springer-Verlag.

[7] V. Ciriani, S. Capitani di Vimercati, S. Foresti, and P. Samarati. k-anonymity. In Ting Yu and Sushil Jajodia, editors, *Secure Data Management in Decentralized Systems*, volume 33 of *Advances in Information Security*, pages 323–353. Springer US, 2007. 10.1007/978-0-387-27696-0_10.

[8] Farshad Fotouhi, Li Xiong, and Traian Marius Truta, editors. *Proceedings of the 2008 International Workshop on Privacy and Anonymity in Information Society, PAIS 2008, Nantes, France, March 29, 2008*, ACM International Conference Proceeding Series. ACM, 2008.

[9] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[10] Benjamin C. M. Fung, Ke Wang, Rui Chen, and Philip S. Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Comput. Surv.*, 42:14:1–14:53, June 2010.

[11] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases*, VLDB '99, pages 518–529, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[12] Aristides Gionis, Arnon Mazza, and Tamir Tassa. k-anonymization revisited. In *ICDE '08: Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, pages 744–753, Washington, DC, USA, 2008. IEEE Computer Society.

[13] Yeye He and Jeffrey F. Naughton. Anonymization of set-valued data via top-down, local generalization. *Proc. VLDB Endow.*, 2:934–945, August 2009.

[14] Neil Hunt. Netflix prize update. http://blog.netflix.com/2010/03/this-is-neil-hunt-chief-product-officer.html, 2010.

[15] Piotr Indyk. *Handbook of Discrete and Computational Geometry*, chapter 39 Nearest Neighbors in High Dimensional Spaces. 2004.

[16] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, STOC '98, pages 604–613, New York, NY, USA, 1998. ACM.

[17] Vijay S. Iyengar. Transforming data to satisfy privacy constraints. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, pages 279–288, New York, NY, USA, 2002. ACM.

[18] Hisashi Koga, Tetsuo Ishibashi, and Toshinori Watanabe. Fast agglomerative hierarchical clustering algorithm using locality-sensitive hashing. *Knowledge and Information Systems*, 12:25–53, 2007. 10.1007/s10115-006-0027-5.

[19] Kristen LeFevre, David J. DeWitt, and Raghu Ramakrishnan. Incognito: efficient full-domain k-anonymity. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, SIGMOD '05, pages 49–60, New York, NY, USA, 2005. ACM.

[20] Kristen LeFevre, David J. DeWitt, and Raghu Ramakrishnan. Mondrian multidimensional k-anonymity. In *Proceedings of the 22nd International Conference on Data Engineering*, ICDE '06, pages 25–, Washington, DC, USA, 2006. IEEE Computer Society.

[21] Jun-Lin Lin and Meng-Cheng Wei. An efficient clustering method for k-anonymization. In Fotouhi et al. [8], pages 46–50.

[22] Jun-Lin Lin and Meng-Cheng Wei. Genetic algorithm-based clustering approach for k-anonymization. *Expert Syst. Appl.*, 36:9784–9792, August 2009.

[23] Jun-Lin Lin, Meng-Cheng Wei, Chih-Wen Li, and Kuo-Chiang Hsieh. A hybrid method for k-anonymization. In *Proceedings of the 2008 IEEE Asia-Pacific Services Computing Conference*, pages 385–390, Washington, DC, USA, 2008. IEEE Computer Society.

[24] Ashwin Machanavajjhala, Johannes Gehrke, Daniel Kifer, and Muthuramakrishnan Venkitasubramaniam. l-Diversity: Privacy Beyond k-Anonymity. In *22nd IEEE International Conference on Data Engineering*, 2006.

[25] Bradley Malin. k-unlinkability: A privacy protection model for distributed data. *Data Knowl. Eng.*, 64:294–311, January 2008.

[26] Mike Masnick. Forget the government, aol exposes search queries to everyone. `http://www.techdirt.com/articles/20060807/0219238.shtml`, 2006.

[27] Adam Meyerson and Ryan Williams. On the complexity of optimal k-anonymity. In Alin Deutsch, editor, *PODS*, pages 223–228. ACM, 2004.

[28] A. Moore. An introductory tutorial on kd-trees. Technical report, Robotics Institute, Carnegie Mellon University, 1991.

[29] Arvind Narayanan and Vitaly Shmatikov. How to break anonymity of the netflix prize dataset. *CoRR*, abs/cs/0610105, 2006.

[30] M. Ercan Nergiz and Chris Clifton. Thoughts on k-anonymization. *Data Knowl. Eng.*, 63:622–645, December 2007.

[31] Hyoungmin Park and Kyuseok Shim. Approximate algorithms for k-anonymity. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, SIGMOD '07, pages 67–78, New York, NY, USA, 2007. ACM.

[32] Deepak Ravichandran, Patrick Pantel, and Eduard Hovy. Randomized algorithms and nlp: using locality sensitive hash function for high speed noun clustering. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 622–629, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[33] P. Samarati. Protecting respondents' identities in microdata release. *IEEE Trans. on Knowl. and Data Eng.*, 13:1010–1027, November 2001.

[34] Agusti Solanas, Francesc Sebé, and Josep Domingo-Ferrer. Micro-aggregation-based heuristics for p-sensitive k-anonymity: one step beyond. In Fotouhi et al. [8], pages 61–69.

[35] Xiaoxun Sun, Hua Wang, Jiuyong Li, and Traian Marius Truta. Enhanced p-sensitive k-anonymity models for privacy preserving data publishing. *Trans. Data Privacy*, 1:53–66, August 2008.

[36] L. Sweeney. k-Anonymity: A Model for Protecting Privacy. *International journal of uncertainty, fuzziness, and knowledge-based systems*, 2002.

[37] Manolis Terrovitis, Nikos Mamoulis, and Panos Kalnis. Privacy-preserving anonymization of set-valued data. *Proc. VLDB Endow.*, 1:115–125, August 2008.

[38] Traian Marius Truta and Bindu Vinay. Privacy protection: p-sensitive k-anonymity property. In *Proceedings of the 22nd International Conference on Data Engineering Workshops*, ICDEW '06, pages 94–, Washington, DC, USA, 2006. IEEE Computer Society.

[39] Raymond Chi wing Wong, Jiuyong Li, Ada Wai chee Fu, and Ke Wang. (, k)-anonymity: an enhanced k-anonymity model for privacy preserving data publishing. In *In ACM SIGKDD*, pages 754–759, 2006.

[40] William E. Winkler. (statistics 2002-07) using simulated annealing for k-anonymity, 2002.

[41] Jian Xu, Wei Wang, Jian Pei, Xiaoyuan Wang, Baile Shi, and Ada Wai-Chee Fu. Utility-based anonymization using local recoding. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pages 785–790, New York, NY, USA, 2006. ACM.

[42] Sheng Zhong, Zhiqiang Yang, and Rebecca N. Wright. Privacy-enhancing k-anonymization of customer data. In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '05, pages 139–147, New York, NY, USA, 2005. ACM.