Imperial College London

Imperial College London

Department of Computing

Interim Report

Author:
Romain de Spoelberch

Supervisor:
Yves-Alexandre de Montjoye

Florimond Houssiau

Contents

Chapter 1

Introduction

In our increasingly digitalized world, every step of our day leaves a trace [1]. Per minute, Venmo– a digital wallet that facilitates payments– processes over $\$160,000$, each transaction containing information on both the sender and the recipient; Uber users take over 9500 rides, once again amassing information on users' movements, and AirBnB makes over 1500 reservations. All this data can be used beneficially, optimising services, and inferring how to better meet customers' needs. However, if any of this data were to be shared, it would constitute an enormous privacy concern. For example, recently, Google's project Nightingale has raised privacy red flags. The company, having struck a deal with one of America's largest healthcare provider, is getting the personal medical data of up to fifty million Americans [2]. Their goals to further the field of medicine with data and artificial intelligence, laudable as they are, have left many suspicious. Some worry that the big tech company stands to gain a lot from using this data for alternative reasons, targeting individuals or selling the data, and committing a major privacy breach in both cases.

In an effort to remediate the privacy concerns of sharing data, researchers have developed many anonymization methods, including one called k-anonymization [3]. k-Anonymization ensures that no individual in a dataset is distinguishable from k-1 others; it's often done by generalizing records – making the information vaguer so that individuals fit in the same broader categories. Additional privacy measures have to be put in place for the dataset to be considered truly anonymized but k-anonymization forms a good basis [4].

Nevertheless, every generalization entails losing a small amount of information in the dataset. One way to quantify this information loss is to use conditional entropy [5] and while this metric gives us an idea of what's lost, it doesn't necessarily claim anything about the utility– for example, in Machine Learning tasks– of the anonymized data. Confirming entropy as a good measure for utility would allow data holders to get an estimate of the utility of an anonymized dataset with respect to the original content.

In this project, we implement several k-anonymization algorithms, and some degradation metrics, including entropy. We'll then use these to k-anonymize a dataset for different values of k, and train ML classifiers with them. In comparing the accuracies of these classifiers, we analyze the usefulness of entropy as a measure quantifying utility of anonymous datasets. Depending on the results, we either confirm the adequacy of entropy, or hope to find an alternative metric that does reflect utility.

Chapter 2

Background

The aim of this section is to give an overview of k-anonymization. We'll start with the motivation, and the theoretical definitions. We'll then cover different implementations of it, and consider a series of metrics to quantify the information loss in the anonymization process.

2.1 Motivation

When an organisation releases a dataset containing personal information, they are responsible for ensuring the privacy of all participants. This is not an easy task and a lot of public datasets fall short of this privacy expectation — whether knowingly or not. For example, naively, many organisations release datasets in which any explicit identifiers like names are removed. Nevertheless, in a lot of cases, the remaining information can be used to re-identify individuals by using unique combinations of fields in the dataset.

For instance, in a paper published in the year 2000, Latanya Sweeney found that 87% of partipicants in the 1990 U.S. Census survey were uniquely identifiable by a combination of their zipcode, gender, and age [6]. As such, if we had the U.S. Census dataset, and knew someone's age, address, and gender, chances are we could recover the rest of their personal data, in what Sweeney coined as re-identification by linking. In 2002, she published a paper demonstrating this attack [3].

Linking two datasets, Sweeney managed to re-identify the medical records of William Weld, a governor of Massachusetts at the time. The first dataset was a health insurance dataset, including information such as ethnicity, procedures undergone, medications given, etc..., for all state employees in Massachusetts. Weld, as governor, was in the dataset. This information was pseudonymised, thought safe, and sold to industry. The second dataset was the Registered Voters list for Cambridge, Massachusetts, a publicly available record of registered state voters in Cambridge (Weld's county of residence). This contained names, addresses, party affiliations, etc... The two sets had three fields in common: zipcode, birthdate, sex. Figure 2.1 represents the common fields between the health insurance set, and the voter list.

Because the Voter List included names, Sweeney found Weld's details. According to her, six people in the health insurance set had Weld's birthday, and only three of them were men. When you took the zipcode into account, he was the only one left. As such, Sweeney could guarantee that the medical record was his.

This exemplifies the danger of releasing pseudonymised datasets under the assumption they're safe. In the same paper, Sweeney proposes a measure to thwart linking attacks: k-Anonymity. Informally, in a k-anonymous dataset, no individual's record is indistinguishable from at least $k-1$ others. This ensures that even having auxiliary information on an individual, an attacker will always find at least k records with this information, creating uncertain linkings. Different methods for k-anonymization will be covered later.

For example, table 2.2 (a) displays 4 records in a dataset containing zipcodes, ages, and incomes. If we were to know someone's age and zipcode, we could unequivocally infer their income. Table (b) shows the result after a 2-anonymization. In this case, we achieved it by generalizing our records, making zipcodes less specific when necessary and giving age ranges instead of exact numbers. Even if we had auxiliary information on someone, we couldn't be certain about their income. For example, we want to find someone's salary, and we know their zipcode, and age: 79203, and 27 respectively. In (a), their income is 64k. In (b), we now can't differentiate row 2 and 4 since the

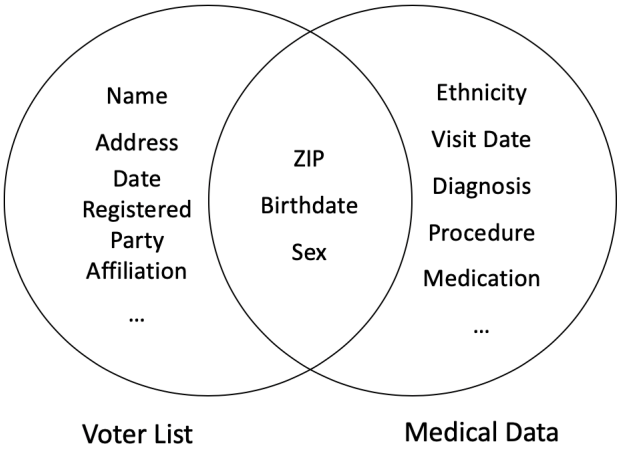


Figure 2.1: Venn diagram representing common features of the Weld example sets from [3]

ID	Zipcode	Age	Income
1	28488	33	52k
2	79203	24	98k
3	28473	39	36k
4	79203	27	64k

(a) D

ID	Zipcode	Age	Income
1	284**	30's	52k
2	79203	20's	98k
3	284**	30's	36k
4	79203	20's	64k

(b) D, 2-Anonymized

Figure 2.2: 2-Anonymous Example

ages have been generalized and the target's salary could equally be 98k or 64k. In this example, we arbitrarily used 2 as our k-value but any k bigger than one will confer some measure of security, increasing with k.

2.2 Notations and Definitions

First, we establish the notation and definitions we'll use through the report. These will be consistent with those used in Sweeney's paper originally presenting k-Anonymity [3].

We will only be dealing with structured data. From now on, any data mentioned is implicitly structured. Each column is termed an attribute, and represents a single type of categorical information. Each row is a record belonging to an individual, and is defined by an ordered tuple $< d_1, d_2, ..., d_n >$ such that every d_j, for $j = 1, 2, ..., n$, contains a value in the domain of the corresponding attribute. Whilst every attribute is unique — that is, no two columns contain the same information — a tuple can reappear multiple times throughout the data. Nevertheless, every tuple belongs to one person, and no person has two tuples in the dataset.

Definition 2.2.1. Attributes
Data denoted as $D(A_1, ..., A_n)$. The finite set of attributes of D are $\{A_1, ..., A_n\}$. Given a table $D(A_1, ..., A_n)$, and $\{A_i, ..., A_j\} \subseteq \{A_1, ..., A_n\}$, $D[A_i, ..., A_j]$ denotes the subset of attributes (columns) $A_i, ..., A_j$ in D.

Attributes can categorized into three subsets. A *direct identifier* is an attribute that, by itself, can fully identify an individual in a dataset. These could include names, or social security numbers. Next, we have *quasi-identifiers* (QI). These attributes contain personal information but are not sufficient to uniquely identify a person in the dataset. However, in certain instances, multiple QIs taken together can re-identify an individual. Finally, *sensitive attributes* are the ones containing sensitive information that we want to protect.

For example, a dataset $D(name, zipcode, gender, age, illness)$ containing a Hospital's patient information has one direct identifer: name. The set of quasi-identifiers is $QI = \{zipcode, gender, age\}$, and finally, the sensitive attribute is illness.

Definition 2.2.2. k-Anonymous
A table D, with quasi-identifiers QI_D, is k-anonymous if and only if each tuple in $D[QI_D]$ appears at least k times in $D[QI_D]$.

We have formalized what we described in the motivation. We want any combination of quasi-identifiers to be indistinguishable from $k-1$ others. That is, any auxiliary information an attacker could gather would point to at least k records, making re-identification harder.

Definition 2.2.3. Equivalence Classes
An equivalence class in D with respect to the quasi-identifiers is the set of records having identical values for $D[QI_D]$ [7].

In other words, the equivalence classes are the sets of records that have indistinguishable quasi-identifiers. In k-anonymous datasets, these classes have a cardinality of at least k. As an example, Table (a) in Figure 2.2 has two equivalence classes: $\{1, 3\}$ and $\{2, 4\}$.

2.3 Generalization and Suppression

In this section, we'll formalize the theory of generalization, hinted at in the motivation.

At a first glance, k-anonymity could be achieved by removing all records that don't have equivalent classes of sufficient size. Nevertheless, this will result in a lot of lost information. For that reason, most algorithms work with some for of generalization: instead of removing rows, tuples are replaced with more generic versions of them, allowing them to fit in broader equivalence classes [8][7]. For example, in Figure 2.2, we achieve this by stripping the rightmost digits of a zipcode, thus referencing a larger geographical area, or by giving a range of ages instead of specific values.

In the paper describing the first k-anonymization algorithms, Sweeney defines a domain.

Definition 2.3.1. Domain
A domain is the set of values an attribute can take [8]. A *ground* domain is a domain that hasn't been generalized at all. For instance, the ground domain of zipcodes, Z_0, is a string of any five digits. Domains can be generalized to make attributes less informative; say, Z_1 where $93772 \mapsto 9377*$. These build to become less and less specific $(Z_0, Z_1, Z_2...)$.

We can see that for any domain, we need a function mapping every level of generalization to the next one – i.e., for an attribute A, we need functions f_i such that each is a generalization from a domain A_i to the next generalized domain A_{i+1}. This will allow us to move from specific values to increasingly generalized ones. More formally,

Definition 2.3.2. Domain Generalization Hierarchy
For an attribute A, a Domain Generalization Hierarchy (DGH_A) is a set of surjective functions $f_i, i = 0, ..., n-1$ such that:

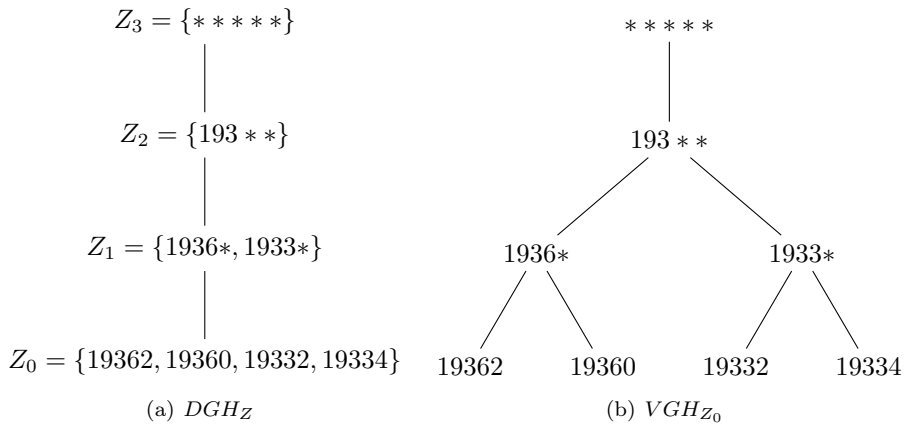
$$A_0 \xrightarrow{f_0} A_1 \xrightarrow{f_1} ... \xrightarrow{f_{n-1}} A_n$$

Where $A_0 = A$, and $|A_n| = 1$ (all values generalize to the same generalized output). Going back to our zipcode example, if we assume every further generalization strips off the rightmost digit, we see that the highest level of generalization, f_4, maps any $x * * * * * \mapsto * * * * *$ and leaves us with a suppressed value; it divulges no information.

The properties of a DGH imply we can create *value generalization hierarchies* (VGH) for attributes. These are the realization of a DGH_{A_i} on a domain A_i and are helpful to visualize the purpose of a DGH. For example, Figure 2.3 displays the DGH for on Z, the zipcode attribute, if our dataset contained $Z_0 = \{19362, 19360, 19332, 19334\}$. Notice that we go from Z_2 to total suppression. This is because Z_2 is singleton, implying further generalizations are pointless since we'll still have only one class of zipcodes.

It's important to point out that we can also create DGHs for categorical attributes, and choose to generalize them in a way that fits our purposes. An example is provided in Figure 2.4, with a level of education attribute where we generalize the level of study to whether any further studies were undertaken after High School. We could also have categorized PhD separately to give an alternative, but equally valid, VGH.

Finally, anytime we put one of these generalizations in practice on a table T, we create a generalized table, T'. We introduce the notation $T \leq T'$ as a partial ordering meaning that T' is a generalization of T.



(a) DGH_Z (b) VGH_{Z_0}

Figure 2.3: A DGH, and VGH for an example zipcode ground domain

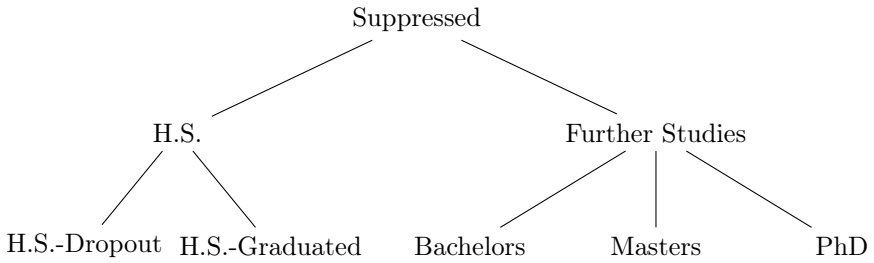


Figure 2.4: An example VGH for a level of education attribute

2.4 k-Anonymization Algorithms

In this section, we'll describe some of the proposed k-anonymization algorithms.

2.4.1 MinGen

It would be trivial– but useless– to anonymize a dataset by suppressing every quasi-identifier to the highest possible level. Ideally, we are looking to keep our data as integral as possible. So, the first algorithm we describe is MinGen. Proposed by Sweeney in 2002[8], MinGen is naive, and "makes no claim to be efficient" but guarantees k-anonymity with the minimal amount of generalization over the quasi-identifiers.

We start by assuming we're going to do all our generalizations at the attribute level, generalizing full columns to the same levels of their respective hierarchies. Given a table T with $QI = A_1, ..., A_j$, the total number of possible combinations of generalizations T is thus equal to:

$$\prod_{i=1}^{j}(|DGH_i| + 1) \tag{2.1}$$

To further reduce the information loss, we could generalize at the cell level – i.e., consider every possible generalization for individual values of every record but this increases the amount of possibilities from an already unmanageable number, (2.1), to the following

$$\prod_{i=1}^{j}(|DGH_i| + 1)^{|T|} \tag{2.2}$$

This is infeasible on any dataset of interest. So we'll restrict ourselves to generalizing at the attribute level.

Since we're not interested in overly generalized tables, we'll be searching for minimally generalized tables. These are defined as follows:

Name	Age
John Manders	21
Timothy Jacks	21
John Misler	36
Timothy Jollten	36

(a) T

Name	Age
John M.	[20-40]
Timothy J.	[20-40]
John M.	[20-40]
Timothy J.	[20-40]

(b) T'

Name	Age
Suppressed	21
Suppressed	21
Suppressed	36
Suppressed	36

(c) T''

Figure 2.5: Caption

Definition 2.4.1. k-Minimal Generalization
Let $T_l(A_1, ..., A_n)$ and $T_m(A_1, ..., A_n)$ be two tables such that $T_l[QI] \leq T_m[QI]$ where QI is the set of quasi-identifiers associated with the tables. T_m is said to be a k-minimal generalization of T over QI if and only if:

- T_m satisfies the k-anonymity property
- $\forall$ k-anonymous T_z satisfying $T_l \leq T_z, T_z \leq T_m$ then $T_z = T_m$.

In other words, this definition helps us find the first generalized table that is k-anonymous. Nevertheless, every table could have multiple k-minimally generalized tables, and they wouldn't all be as useful. For example, Table 2.5 depicts two different k-minimal generalizations of a Table T (a): we could either generalize Name twice, or both Name and Age once. They're both 2-anonymous, and any G with $G \leq T'$ or $G \leq T''$ wouldn't be, implying 2-minimal generalization. The question now becomes, which should we use?

To solve that, we define a preference criteria in the form of the precision metric (defined in Hierarchy-based Metrics) which captures the amount of information lost and we define minimal distortion:

Definition 2.4.2. k-Minimal Distortion
Let $T_l(A_1, ..., A_n)$ and $T_m(A_1, ..., A_n)$ be two tables such that $T_l[QI] \leq T_m[QI]$ where QI is the set of quasi-identifiers associated with the tables. T_m is said to be a k-minimal generalization of T over QI if and only if:

- T_m satisfies the k-anonymity property
- $\forall$ k-anonymous T_z satisfying $Prec(T_l) \geq Prec(T_z), Prec(T_z) \geq Prec(T_m)$ then $T_z = T_m$.

Furthermore, Sweeney states as a Theorem: given tables T_l and T_m such that $T_l \leq T_m$, and T_m satisfying k-anonymity. T_m is a k-minimal distortion of $T_l \implies T_m$ is a k-minimal generalization of T_l.

Puting these together, we can now limit our search to the the minimal distortion, and an algorithm starts to form. Given a Table $T(A_1, ..., A_n)$ with $QI = \{A_i, ..., A_j\} \subseteq (A_1, ..., A_n)$, a k, and a DGH_i for every i in QI, the first step is verifying if T is already k-anonymous. In that case, we're done since it will be minimally generalized. If it isn't, we store every possible generalization of T over QI. We then filter out any that don't satisfy the k-anonymity condition. From the leftover anonymous tables, we need to find the one that is minimally distorted, which we do by calculating the distortion of every generalization. If we still get multiple generalizations with the same Precision, then we let the user pick which generalization they want to use because, depending on their use for the data, they might prefer generalizing certain attributes over others. Another strategy to deal with user preferences is to assign weights to the attributes, giving them more or less importance. Assuming all attributes have equal importance, we outline the MinGen algorithm in Algorithm 1.

This is one of the algorithms we will try to use in this project. Nevertheless, in 2004, Meyerson and Williams proved that finding an optimal k-anonymous generalization is an NP-hard problem [9]. This Confirms Sweeney's claim of inefficiency, and makes it unlikely we'll be able to successfully apply this algorithm to any meaningful dataset. As such, although this algorithm produces optimal results, we'll primarily rely on the next few to get tractable problems.

Algorithm 1 MinGen: k-minimally generalized

Input: Table T; a defined DGH for every QI; a fixed k; a **preferred** function defined by user to choose one generalization amongst multiple solutions.
Output: MGT, a k-minimal distortion of T, chosen according to preference
Ensure: $|T| > k$
  if anonymous(T) **then**
    MGT $\leftarrow T$
  else
    allgen $\leftarrow \{T_i : T_i$ is a generalization of $T\}$
    anon $\leftarrow \{T_i : T_i \in$ allgen & anonymous$(T_i)\}$
    MGT $\leftarrow \{T_i : T_i \in$ anon & $\nexists T_z \in$ anon s.t. $Prec(T_z) \geq Prec(T_i)\}$
    MGT $\leftarrow$ **preferred**(MGT)
  end if
  return MGT

2.4.2 DataFly

In 1997, before formalizing the concept of k-anonymization, Sweeney developed a practical anonymization algorithm called Datafly [10]. Datafly works by greedily generalizing the individual attribute with the largest domain. It does so iteratively, until the table is k-anonymous, or close enough to warrant suppressing a few records.

For a dataset T, the algorithm revolves around a frequency table of the tuples in $T[QI_T]$. Whilst more than k records in the frequency table are not anonymous, Datafly will generalize an attribute, and update the frequency table accordingly. At the end, the small amount of leftover non-anonymous records are removed, and the dataset is rebuilt from the frequency table – i.e., a tuple with a certain count is inserted that many times in a new table. The following is outlined in Algorithm 2.

Algorithm 2 Datafly: a practical k-anonymization

Input: Table T; a defined DGH for every QI; a fixed k.
Output: GT, a k-anonymous generalization of T
Ensure: $|T| > k$
  freq $\leftarrow \{\forall t \in T[QI_T] : (t, count(t, T[QI_T]))\}$
  while more than k tuples in freq are not k-anonymous **do**
    $A_j \leftarrow$ Attribute in freq having the largest domain
    freq$[A_j] \leftarrow generalize($freq$[A_j])$
  end while
  freq $\leftarrow$ suppress tuples occuring less than k times
  $GT \leftarrow rebuild$(freq)
  return GT

This algorithm is fast but may distort data more than necessary due to the fact it makes "crude decision" in generalizing entire columns at once, even ones of tuples already k-anonymous. Furthermore, it heavily penalizes attributes with a wide range of different values, leading to unecessary generalizations. For example, if we had a table $T(ZIP, gender)$, our domain ZIP_4, containing things of the form x**** would contain up to 10 different elements whereas $gender_0$ would have two at most. This means that before Datafly even considers generalizing gender, it would fully suppress the zipcode.

2.4.3 Mondrian

In 2006, LeFevre, DeWitt, and Ramakrishnan, et al. came up with a greedy algorithm for multidimensional generalization using a partition based model [7]. This is different from Sweeney's single dimension generalization algorithms, MinGen and Datafly, in the sense that it partitions the dataset in heuristically similar subsets of size at least k, and then generalizes them in the most efficient way possible. This satisfies the k-anonymity property since every subset contains at least k indistinguishable generalized records. It also allows more flexibility as we're not generalizing at

the column level, but rather at the partition level, which means we can avoid generalizing subsets that are already anonymous, distorting data for no reason.

Visualizing our data as a $|QI|$-dimensional space, we heuristically pick a dimension (i.e., an attribute), and create a cut at the median value. This cut separates our table into two subsets on which we recursively keep partitioning. We do this until we can't find a cut that would give us two regions with at least k elements in them. Once we've got our regions, we generalize them to the same tuple. The hope is that these partitions will contain similar records, and that the generalization done is minimal. Different heuristics exist to pick attributes to cut on. Usually, the attribute with the widest normalized range of values but we have flexibility on that.

Figure 2.6 shows an example of how a Mondrian partition would choose cuts on a set containing zipcodes and education levels: subfigure(a). The largest domain is education, and as such, Mondrian cuts the domain as equally as possible (b). The right and left partitioned are subsequently divided independently (c, and d). This repeats until no allowable cut exists because the subsets are too small. We now anonymize every partition. Notice that we'll anonymize different partitions differently, allowing for more fine-grained generalization:

$$\{(\text{H.S Dropout}, 34986), (\text{H.S Dropout}, 34917)\} \mapsto \{(\text{H.S. Dropout}, 34 * **), (\text{H.S. Dropout}, 34 * **)\} \quad (2.3)$$

$$\{(\text{Masters}, 34986), (\text{PhD}, 34986)\} \mapsto \{(\text{Further Studies}, 34986), (\text{Further Studies}, 34986)\} \quad (2.4)$$

Additionally, Mondrian has the additional flexibility of making DGHs for every quasi-identifier optional. Every region is bounded, and these can serve as generalization. Any numerical attribute has a clear lower and upper bound. In the absence of a DGH for the example in Figure 2.6, the generalization examples we saw in (2.3) and (2.4) could have been as shown in (2.5) and (2.6).

$$\{(\text{H.S Dropout}, 34986), (\text{H.S Dropout}, 34917)\} \mapsto \{(\text{H.S. Dropout}, [34917 - 34986]), (\text{H.S. Dropout}, [34917 - 34986])\} \quad (2.5)$$

$$\{(\text{Masters}, 34986), (\text{PhD}, 34986)\} \mapsto \{(\text{Masters} - \text{PhD}, 34986), (\text{Masters} - \text{PhD}, 34986)\} \quad (2.6)$$

The only caveat is for categorical attributes: they need to have a total ordering. Otherwise the notion of distance disintegrates. For example, in Figure 2.6, the distance between zipcodes is straightforward as it's a numerical attribute but the distances separating levels of education are abstract and have to be given. This can sometimes be a problem when the order isn't as clear, like a dataset including the Profession attribute with ground domain Prof_0 ={Tech-support, Salesman, Skydiver, Farmer}. There's no obvious order on which to base these which complicates the partitioning.

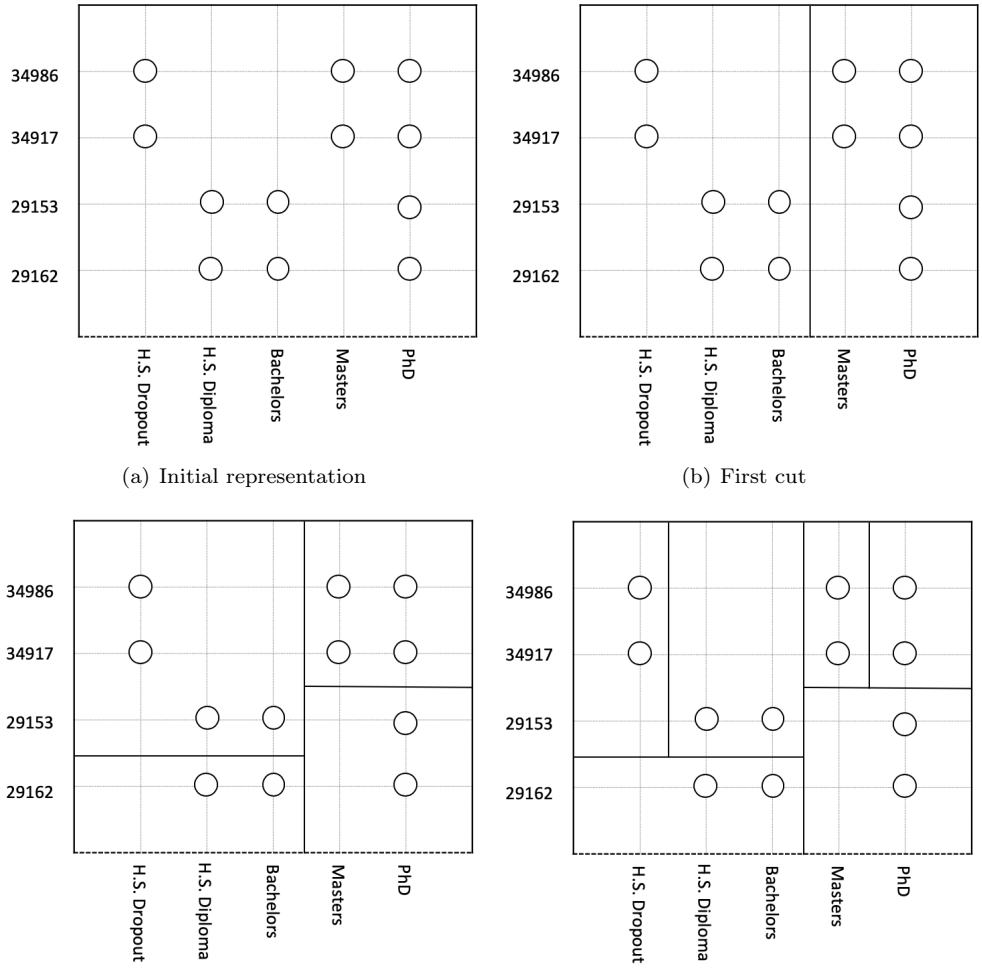
In Algorithm 3, we outline the Mondrian algorithm, assuming every attribute has been ordered correctly. 'choose_dim' gives the user a choice on the heuristic to select the attribute to cut on and 'Anonymize' returns the anonymized region.

2.5 Degradation Metrics

As explained throughout the background so far, when we generalize datasets, although we don't modify the data, we lose detail, coarsening it. In this section, we'll introduce different ways to quantify that loss of information, discussing their advantages and drawbacks.

2.5.1 Hierarchy-based Metrics

First, we try to quantify how generalized a table is. When working with DGHs, this can be done by taking into account how many times attributes are generalized (i.e., what depth of their DGH we've used):



(a) Initial representation

(b) First cut

(c) Second cuts

(d) Third and final cuts

Figure 2.6: Mondrian Example: Education and zipcodes partitioning

Algorithm 3 Mondrian: Multi-Dimensional Greedy Partitioning k-anonymization

Input: Table T; a fixed k.
Output: GT, a k-anonymous generalization of T
Ensure: $|T| > k$

```
function PARTITION(region)
    if no allowable cut on region then return Anonymize(region)
    else
        A_j ← choose_dim(region)
        med ← median(region[A_j])
        lhs ← {t ∈ region : t[A_j] ≤ med}
        rhs ← {t ∈ region : t[A_j] > med}
    end if
return PARTITION(lhs) ∪ PARTITION(rhs)
end function
```

Sweeney and Samarati first introduce the notion of distance to describe the level of generalization of a table with respect to other generalizations of the same table, based on single dimension generalization hierarchies [11]. For a table T with $QI_T = \{A_1, ..., A_n\}$, and its generalized T' we can then analyze absolute distance as follows:

$$\text{Absolute Distance}(T, T') = \sum_{i=1}^{n} d_i \tag{2.7}$$

Where d_i represents the depth of generalization for attribute i. Nevertheless, this metric has the drawback of penalizing every generalization equally, whereas that may not be the case. For example, generalizing an attribute with a DGH of depth 1 will immediately suppress the column but will account for the same distance as a small generalization on an attribute with a deep DGH. For that reason, Sweeney and Samarati then described Relative Distance [11]. Relative Distance takes into account the depth of every DGH as well:

$$\text{Relative Distance}(T, T') = \sum_{i=1}^{n} \frac{d_i}{|DGH_i|} \tag{2.8}$$

This metric gives every attribute a score between 0, and 1. The former indicating an ungeneralized column, and the latter a full suppression. Given these two distance metrics, we can start quantifying data loss in single dimensional generalization algorithms. Nevertheless, if we want metrics do analyze anything further, we need to develop other ways.

Sweeney does this by defining precision: a metric quantifying the distortion at cell level. For a table $T(A_1, ..., A_n)$, and its generalized T', precision is defined as:

$$\text{Precision}(T, T') = 1 - \frac{\sum_{i=1}^{|T|} \sum_{j=1}^{n} \frac{h_{ij}}{|DGH_{A_i}|}}{n|T|} \tag{2.9}$$

Where h_{ij} is the depth generalized at of cell $T[i, j]$ This takes the sum of the generalization level of every cell, and then normalizes it. This will return values between 0 and 1, from no generalization to full suppression. Subtracting this from 1, we get a metric determining the percentage of the table that is generalized. We use this metric to obtain minimal distortion in the MinGen algorithm (MinGen), and we could use this to measure data degradation in algorithms with domain hierarchies.

Nevertheless, precision is a crude measure of degradation as different generalizations affect the data differently. Additionally, it relies on the DGHs created by the user. This has the downside of making it impossible to compare two generalizations created using different hierarchies, and a metric that can only be used in hierarchy-based models. Furthermore, we have no way to evaluate the DGHs themselves, making this entire metric dependent on the assumption that they're well defined.

To remediate this, researchers looked for alternative solutions.

2.5.2 Entropy

Entropy, as defined by Shannon in 1948 in the context of his information theory research, is a measure to quantify information [12]. In 2007, Gionis et al. found a use for it in assessing the amount of information lost during an anonymization process [5]. More precisely, Gionis uses the conditional entropy $H(Y|X)$, quantifying the amount of information needed to describe needed to describe the outcome of a random variable Y, given the outcome of a random variable X. To understand how much information is lost, we want to find the conditional entropy of every cell in the generalized table with regards to the original, un-anonymized table.

For a table $T(A_1, ..., A_n)$, and its generalization T' where a_{ij} represents j^{th} attribute, A_j of the i^{th} record, we call the generalization function $f_{ij} : A_{j_0} \to A_{i_x}$ the function used to generalize a cell to whatever necessary level (using a DGH or not). In an abuse of notation, we call f_{ij}^{-1} the inverse returning the set of initial potential values. For example, an age value, $x = 25$, and $f(x) = (25-30)$, a generalized range. The inverse function, $f^{-1}((25-30)) = \{25, 26, 27, 28, 29, 30\}$

returns a set with the potential values that could have been. We define the conditional entropy of a random variable V_{ij} describing cell a_{ij} is given by the following equation [5]:

$$H(V_{ij}|a_{ij}) = -\sum_{v \in f^{-1}(a_{ij})} P(V_{ij} = v|a_{ij}) \log P(V_{ij} = v|a_{ij}) \tag{2.10}$$

The conditional probabilities are established by counting occurences in the initial table, T:

$$P(V_{ij}|a_{ij}) = \frac{|\{1 \leq i \leq N : c_{ij} = v\}|}{|\{1 \leq i \leq N : c_{ij} \in a_{ij}\}|} \tag{2.11}$$

Note that the conditional entropy of a cell ranges from 0 if there is absolutely no uncertainty, occurring when a_{ij} is a single definite element, to $H(V_{ij})$ when a_{ij} is fully suppressed. This implies we can create a cost function for generalizations by summing the conditional entropies of every cell in T' to obtain the entropy in generalizing T:

$$\Pi_e = \sum_{i=1}^{|T|} \sum_{j=1}^{n} H(V_{ij}|a_{ij}) \tag{2.12}$$

The higher Π_e, the higher the conditional entropy, and the more data would be needed to describe what was lost. Nevertheless, it is hard to ensure what high entropy entails, or whether it is a good measure for utility.

2.5.3 Classification Metric

A third metric that might be interesting to look at is the classification metric. This metric, defined by Iyengar in 2002, with the aim of quantifying utility of a dataset to be used in classification tasks [13]. When a dataset is k-anonymized, we would ideally want all the elements of an equivalence class to have the same class label. This would imply that the anonymization was, to some extent, justified in grouping these records together. Iyengar describes his metric as "[penalizing] impure groups"; equivalence classes that have been grouped although they represent different things.

This metric iterates through the records of an k-anonymized table T', and penalizes every row who's class label is not the same as the majority label in its equivalence class:

$$CM(T') = \sum_{t \in T'} \frac{\text{penalty}(t)}{|T'|} \tag{2.13}$$

Where the penalty is defined as follows:

$$\text{penalty}(t) = \begin{cases} 1, & \text{if } t \text{ is suppressed} \\ 1, & \text{if class}(t) \neq \text{class}(eq(t)) \\ 0, & \text{otherwise} \end{cases} \tag{2.14}$$

This metric essentially returns the accuracy of a machine learning classification task on its training set. Intuitively, it also gives a measure of coherence of the anonymization, considering equal tuples should be under the same label.

2.6 k-Anonymization Weakness

We'd be remiss not to address the fact that k-Anonymization on its own is not a sufficient privacy measure. There are plenty of ways to work around the anonymization method like homogeneity attacks[14] or skewness attacks [4]. Furthermore, private data holders have to be careful when releasing the same set anonymized in different ways as sometimes the tables can be relinked, and then de-anonymized[3].

Chapter 3

Project Plan

At present

As of the submission of this report, this is what has been done:

- Background research for literature review
- Choice of datasets to anonymize
- Set up an auto-sklearn to automatically train classifiers on a DOC cloud VM
- Found implementation of different k-anonymization algorithms

Important Milestones Left

- First anonymized dataset
- Working metrics to analyze data degradation
- Graph of entropy (hopefully telling us something about it's usefulness as a utility measure)
- Final report

Figure 3.1 shows how I see things progressing.

Obtaining Results

Obtaining results entails anonymizing the datasets, and then running the auto-sklearn tasks on them. The anonymization will involve creating Generalization trees for all the attributes, and then k-anonymizing the datasets with respect to those DGHs. As mentioned above, I found an anonymization library from the UT Dallas, a university. After that, we'll train classifiers to test their accuracy. Auto-sklearn provides a library to do automatic model selection and hyper parameter tuning. Hyper parameter tuning always takes a considerable amount of time, and these tasks even more so because they're looking for a model as well. I believe this could run up until the end of March, but I hope to finish before the easter break.

As soon as I start getting my anonymized datasets, I'll need to calculate the metrics outlined in the background on them. I just need to code them up, and I don't expect that to take too long. This segues into the analysis portion of the project.

Analysis

With a few anonymized sets in-hand, and metrics coded up, I can start analysis on them. This will involve trying to identify the patterns, and finding out if the hypothesis that entropy isn't a particularly good utility metric pans out. If our results don't work out as we hoped, there's time yet to try and either obtain more data and try other metrics, or to address any alternative problem we've found.

Ideally, I'd like to finish the analysis for the end of April so that I can focus on the write-up and/or any extension I feel the project could use.

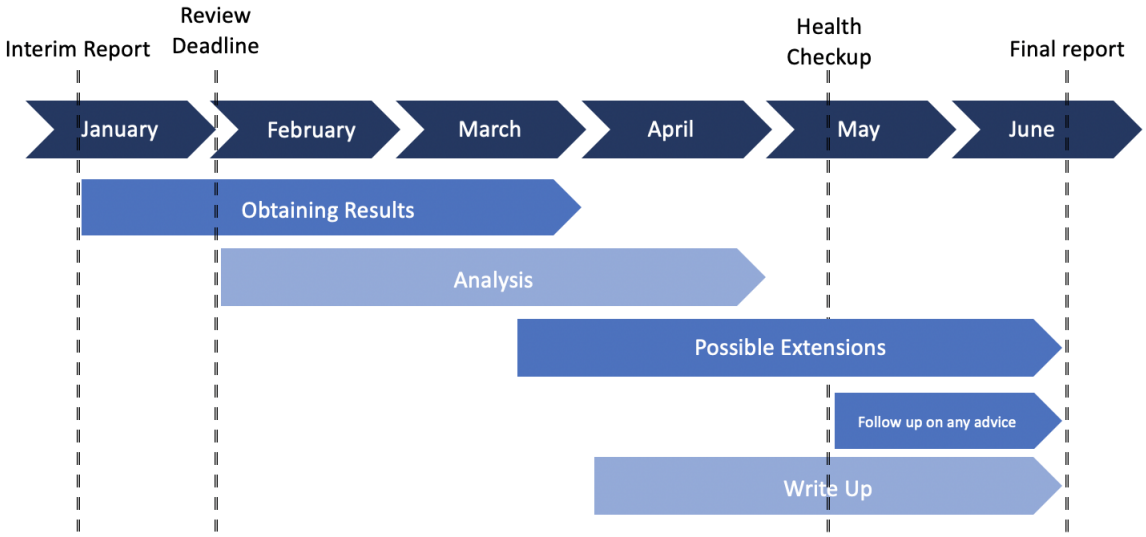


Figure 3.1: Timeline

Write Up

I'm aware of how long the write up will take. Having a good chunk of the background section done is a good start, and I want to start writing the rest in April, if not before, to ensure I have the time I need.

Chapter 4

Evaluation Plan

End Goal

The end goal for this project is to find the limitations of entropy as a measure of utility. To do that, we need to be able to calculate entropy, and anonymize datasets. We've already got an automatic ML training task in place so we'll know how useful the anonymized sets are. Once we've done that, we'll have all the tools we need to do thorough analysis on these anonymised datasets, their metrics, and how they relate to their utility.

I think a successful project is one in which we can show a definite trend either confirming that entropy worsens proportionally with utility. If we don't get satisfactory results from entropy, we would then search for a metric that could serve as a better measure of utility. This new metric could either be an existing one, or we could try our hand at designing one.

Evaluating Results

I'm hoping to work on multiple datasets to ensure the validity of my results. Hopefully, we'll find similar results for all of them, further bolstering our claims. I'll start by using two datasets, and, time permitting, add a third.

We ensure that our anonymised datasets are anonymised correctly by using an anonymization library released by the UT Dallas Data Security and Privacy Lab. We also choose to use auto-sklearn to train the classification models because it automatically does the classifier selection and hyper parameter tuning. You only need to tell it for how long to run. As such, we let every model train for an equally long time (12 to 24 hours). This allows us to compare every classifier fairly, ensures they've trained long enough to get results as good as we could expect, and removes any possible human error.

Bibliography

[1] Nicole Martin. How much data is collected every minute of the day. *Forbes*.

[2] Ed Pilkington. Google's secret cache of medical data includes names and full details of millions – whistleblower. *The Guardian*.

[3] Latanya Sweeney. k-anonymity:a model for protecting privacy. *International Journal on Uncertainty*, 10(5):557–570, 2002.

[4] Josep Domingo-Ferrer and Vicenç Torra. A critique of k-anonymity and some of its enhancements. In *2008 Third International Conference on Availability, Reliability and Security*, pages 990–993. IEEE, 2008.

[5] Aristides Gionis and Tamir Tassa. k-anonymization with minimal loss of information. In *European Symposium on Algorithms*, pages 439–450. Springer, 2007.

[6] Latanya Sweeney. Uniqueness of simple demographics in the us population. *LIDAP-WP4, 2000*, 2000.

[7] Kristen LeFevre, David J DeWitt, Raghu Ramakrishnan, et al. Mondrian multidimensional k-anonymity.

[8] Latanya Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *International Journal on Uncertainty*, 10(5):571–588, 2002.

[9] Adam Meyerson and Ryan Williams. On the complexity of optimal k-anonymity. In *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 223–228. ACM, 2004.

[10] Latanya Sweeney. Guaranteeing anonymity when sharing medical data, the datafly system. In *Proceedings of the AMIA Annual Fall Symposium*, page 51. American Medical Informatics Association, 1997.

[11] Pierangela Samaratiy and Latanya Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression.

[12] Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE mobile computing and communications review*, 5(1):3–55, 2001.

[13] Vijay S Iyengar. Transforming data to satisfy privacy constraints. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 279–288, 2002.

[14] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):3–es, 2007.