IT-UNIVERSITETET I KØBENHAVN

Privacy Preserving Similarity Search

Submitted June 2019, in partial fulfillment of the conditions for the award of the degree **Msc Software Development and Technology.**

Anders Fleron Bourgeat
anfh@itu.dk

and

Jana Schmurr
jansc@itu.dk

Supervised by Martin Aumüller

IT University of Copenhagen

class code : 2722278-Spring 2019

We hereby declare that this thesis is all our own work, except as indicated in the text.

Abstract

Big data has come of age. Individuals are increasingly aware that their data is being collected. Not least because of recent privacy breaches by large multinational corporations, significant privacy concerns among the public have emerged. Local differential privacy provides ways of addressing these concerns without resorting to trust the data handlers. As similarity search is one of the key type of algorithms that lies at the heart of many data analytic services, it also presents as a compelling application of local differential privacy (LDP).

The present paper seeks to contribute to the understanding of the state of similarity search within the local differentially private setting and further develop the topic. We evaluate the only known approach which claims to provide LDP guarantees for similarity search. In continuation of this evaluation we develop the `Bucket PrivMin` which satisfies ϵ-LDP. In recognition of the theoretical drawbacks of our proposed algorithm, we then explore the prospects of designing an approach which can satisfy (ϵ, δ)-LDP. This culminates in the development of the `Noisy Secure MinHash` algorithm.

The results of our empirical analysis of the algorithms suggest that while they both eliminate or greatly reduce the risk of problematic data release, they do this at considerable expense of utility.

Contents

Abstract **i**

1 Introduction **2**

1.1 Background 2

1.2 Scope 3

1.3 Outline 4

2 Preliminaries **5**

2.1 Similarity Search 5

2.1.1 Similarity Measure 6

2.2 Differential Privacy 7

2.2.1 Local Differential Privacy 9

2.2.2 Privacy Mechanisms 9

3 Related Work **13**

3.1 Similarity Search in the Context of Differential Privacy 14

3.1.1 Private Similarity Search in the Central Setting 14

3.1.2 Private Recommender Systems in the Distributed Setting 15

3.2 Locally Private Jaccard Similarity Estimation 16

3.3 Sub-Linear Privacy-Preserving Near-Neighbor Search with Untrusted Server 17

4 The Privacy Guarantees of MinHash **18**

4.1 Existing Privacy Notions of MinHash 18

4.1.1 ϵ-security 18

4.1.2 Conditional Differentially Private Set Operations 19
4.2 MinHash in the Local Setting 20

5 Providing ϵ-LDP with MinHash 23
5.1 Utility 23
5.2 Shrinkage of the Output Space 25
5.3 Bucket PrivMin 27
5.3.1 Mechanism Selection 27
5.3.2 Bucketing 28
5.3.3 Debias of the Estimator 30

6 Providing (ϵ, δ)-LDP with MinHash 33
6.1 Altering Collision Probabilities 33
6.2 Noisy Secure MinHash 36
6.2.1 Noise Addition 36
6.2.2 Noisy Similarity Estimation 40

7 Experiments 42
7.1 Implementation 42
7.1.1 Datasets 42
7.1.2 Metrics 43
7.2 Results 44
7.2.1 Accuracy of Similarity Estimations 44
7.2.2 Utility evaluations 47

8 Discussion 50

Bibliography 51

Appendices 57

A Estimating Similarity from Noisy Signatures 57

List of Tables

5.1 Shrinkage of the output space with different $|X|$ and ϵ 26

List of Figures

4.1 Privacy budget of `MinHash` with increasing K and different τ 21

5.1 Output probabilities with utility of hash value 25

5.2 Output probabilities of the minimum hash value 30

6.1 Sensitivity with $\alpha = 1$, $\tau = 100$, increasing L and different values of δ 39

7.1 Mean Squared Error of similarity estimations with Bucket PrivMin 44

7.2 Mean Squared Error of similarity estimations on MovieLens with Noisy Secure MinHash 45

7.3 Mean Squared Error of similarity estimations on Last.FM with Noisy Secure MinHash 46

7.4 Comparison of accuracy of similarity estimations of Noisy Secure MinHash and Bucket PrivMin 47

7.5 Recall with Bucket PrivMin 48

7.6 Recall with Noisy Secure MinHash $K = 1$ 48

7.7 Comparison of utility of Noisy Secure MinHash and Bucket PrivMin 49

Chapter 1

Introduction

Big data is big business. Globally, we now generate over 3.6 exabytes of data every day [3]. In the EU alone, the market for data-related products and services is estimated at more than 50 billion Euro. It is projected to continue to grow up towards 111 billion Euro by just 2020 [22]. While some companies built their entire business models on the collection of data, all companies now need to rely on data to barely remain competitive.

1.1 Background

This growth in data collection and analysis has helped fuel an increasing awareness among the public for the privacy of their data. This concern has been further exacerbated by the continual stories of privacy breaches by some of the largest holders of data today. Although much of the data collected might not be viewed as private in a narrow sense, researchers and journalists have demonstrated the ability to identify specific individuals using anonymized data released from seemingly innocuous data sources [31][8].
After the first linkage attack in 2006 that caught broad attention [35], there is by now a general consensus that simple anonymization is not sufficient. Although people express increasing concern for the privacy of their data, this stands in contrast with their actual behaviour, which does not reflect this concern [21].

One avenue for addressing people's privacy concerns is through legislation. This approach is best exemplified by the sweeping *General Data Protection Regulation* (GDPR) imposed by the EU, which sets out a set of principles all data handlers must abide by. One can express valid concerns for the compatibility between the dynamic and fast moving world of technology and the rigid nature of such legislation [45]. With the fairly recent development of the framework of *Differential Privacy* [18], researchers and companies were provided with a common language for evaluating the privacy leakage of algorithmic systems.

1.2 Scope

Differential privacy allows data collectors to provide privacy guarantees to their users, while maintaining the ability to perform data analysis [19]. Nonetheless, have recent privacy breaches of multinational tech companies like Google and Facebook further compromised the trust of users in companies collecting their data [32]. It is hence desirable to find approaches that can evaluate privacy guarantees, even in regards to the data curator.

Local differential privacy provides such a framework. Large-scale deployed systems have shown the potential for performing effective data analysis even within this strict framework [5] [20]. However, due to the youth of this field, many algorithms used for data analysis cannot yet provide those vigorous privacy guarantees. Similarity search is one of the key type of algorithms that lies at the heart of many data analytic services. It also presents as a compelling application of differential privacy, as it by its nature involves (sensitive) user information. Combining the ubiquity of similarity search systems and data collection with the simultaneous need for data protection thus represents an attractive path for research. Hence, we in this paper aim to answer the following question:

How do existing LDP similarity search algorithms perform, and how can these results be improved upon in terms of accuracy and privacy?

1.3 Outline

The paper will proceed as follows: In chapter 2 we introduce some of the essential concepts used in the paper, including both similarity search and differential privacy. Chapter 3 proceeds to explore the current state of privacy preserving similarity search algorithms and relevant adjacent topics. Chapter 4 is used to gain a further understanding of the inherent privacy guarantees provided by the `MinHash` algorithm. In chapter 5 we analyze the only published paper claiming local differential privacy guarantees for a `MinHash` based approach called the `PrivMin` algorithm. Furthermore, we design an alternative approach termed the `Bucket PrivMin` algorithm which corrects some of the misconceptions build into the `PrivMin` algorithm. Chapter 6 is used to further explore other avenues for providing local differential privacy guarantees in similarity search, culminating in the design of the `Noisy Secure MinHash` algorithm. In chapter 7 we present the experimental results of the two algorithms and evaluate on their comparative performance. The paper is concluded by a discussion of the findings in chapter 8.

Chapter 2

Preliminaries

The current chapter will serve to introduce some of the fundamental concepts and techniques used in the fields of similarity search and differential privacy. It will hence serve as a foundation for the analysis and design of algorithms in the subsequent work.

2.1 Similarity Search

Nearest neighbour search is the optimization problem of finding the data points with highest similarity (or closeness) to a query point. It is an essential algorithmic problem in computer science, and its applications span across a variety of fields, including databases and data mining, computer vision, information retrieval, machine learning, and many more [4]. Formally, we can define the nearest neighbour $NN(q)$ to a query point q to be the data point p in the set of all data points $P = \{p_1, \ldots, p_N\}$ which satisfies $NN(q) = arg\,min_{p \in P}\, dist(q, p)$. Here, $dist(q, p)$ is the distance measure between the two points q and p [4]. The result of the algorithm is a set of N-nearest neighbours, where N represents the number of nearest neighbours. In a naive approach, this problem is easily solved by iterating through each point in the database and computing the closeness based on a similarity function, which aggregates the distances in multi-dimensional space. In the case of a large database or costly computation between the query point and other points in the database, relying on linear search will result in impractical running times.

In order to increase the efficiency of near neighbour search, *approximate near neighbour search* allows for a relaxation in the finding of the nearest neighbours. Instead of finding the exact near neighbours, it aims at finding the approximate nearest neighbours [38].

Definition 2.1 *(Approximate nearest neighbour search problem)*
To solve the $(1+\epsilon)$*-approximate nearest neighbor search problem, with a query* q *and* $\epsilon > 0$*, the goal is to find an item* p^* *so that* $dist(q,p) \leq (1+\epsilon)dist(q,p)$*, where* p *is the true near neighbour.*

Approximate near neighbour search is a well-studied and established approach for efficient and accurate similarity search [6] [26]. The approximate near neighbour search can be based upon a variety of underlying algorithmic techniques. In this work, we will focus on locality-sensitive hashing. Introduced by [23], locality-sensitive hashing (LSH) uses families of hash functions that preserve relative distances between query points in hashing. As a means, it uses hash functions which ensure that the collision probability for close objects is higher than for dissimilar data points. It allows for fast distance approximation as the hash codes are used to reduce the attributes of target items, thereby enabling efficient distance computation.

2.1.1 Similarity Measure

A similarity measure is a function which computes the similarity of two data points. The Jaccard similarity is an established similarity measure to compute the similarity of two sets, X and Y. Its result in a value $Jac(X,Y) \in [0,1]$, where 0 defines the sets to have no common items, while 1 indicates the sets to consist of two identical sets of items.

Definition 2.2 *Jaccard similarity*
Given two sets X and Y, the Jaccard similarity is the ratio between sizes of the intersection and the union of these two sets.

$$Jac(X,Y) = \frac{|X \cap Y|}{|X \cup Y|}$$

Originally introduced by Broder et al. [11], the *min-wise hashing* scheme, also referred to as the `MinHash` algorithm, provides a technique to estimate the Jaccard similarity in sub-linear running time. The similarity estimation of `MinHash` is premised on a key property: given a random permutation of elements, the probability of the sets X and Y sharing the same minimal hash value is exactly the Jaccard index.

Definition 2.3 *MinHash*

Given two sets X and Y, hashed uniformly at random and uniquely to $\mathcal{T}$, the probability of the minimum values of hash sets h(X) and h(Y) being the same is:

$$Pr\big[h_{min}(X) = h_{min}(Y)\big] = \frac{|X \cap Y|}{|X \cup Y|} = Jac(X, Y)$$

The `MinHash` algorithm is one of the key algorithmic approaches in performing efficient similarity search. Hence, we will further focus on the `MinHash` as the foundation for building a privacy preserving similarity search system.

2.2 Differential Privacy

The general aim of research in the field of privacy is to limit or eliminate the disclosure of specific information about an individual, while still gaining general, non-specific information from the population. Traditionally, the main focus of research has been on mechanisms that prevent *any* disclosure of information about the participant. Differential privacy relaxes this notion. Dwork [18] claims that perfect privacy is impractical due to poor utility. Hence, instead of providing strict protection, the notion of differential privacy helps to define the *relative* bounds on the privacy loss. The leakage of privacy is defined as the change of probability that the output of an algorithm differs due to the presence or absence of an individual in the data set [29]. Differential privacy is hence tailored towards the problem of privacy-preserving data release, and consequently, data analysis [27][19].

Formally, differential privacy is defined as:

Definition 2.4 *ϵ-Differential Privacy*

A randomized algorithm M satisfies ϵ-differential privacy iff for all databases $\|X - Y\|_1 \leq 1$, and all $S \subseteq Range(M)$:

$$Pr[M(X) \in S] \leq exp(\epsilon) \times Pr[M(Y) \in S] \tag{2.1}$$

We can imagine a randomized algorithm satisfying ϵ-differential privacy as follows: We assume two databases X and Y, differing in one element. A trusted curator receives the data and aims to release statistics to an untrusted analyst, but without disclosing any information about any individual participant. The curator ensures this by applying a randomizing mechanism to the data that guarantees that the probability of an outcome of a query on X and Y respectively will be the same, within a factor of $exp(\epsilon)$.

This notion of privacy can be very strong for small values of ϵ, as in such cases it can be guaranteed that any one participant will not affect the results of any query. However, for many applications this can be an untenable solution, as the amount of introduced randomness will degrade the utility of the answers to queries to such an extent as to make it virtually useless. It can therefore in some cases be necessary to relax the definition 2.4 to ensure the preservation of utility. The resulting *approximate* differential privacy notion (ADP) is defined as:

Definition 2.5 *(ϵ-δ)-Differential Privacy*

A randomized algorithm M satisfies (ϵ-δ)-differential privacy iff for all databases X, Y with $\|X - Y\|_1 \leq 1$, and all $S \subseteq Range(M)$:

$$Pr[M(X) \in S] \leq exp(\varepsilon) \times Pr[M(Y) \in S] + \delta \tag{2.2}$$

The additional parameter δ can be thought of as the probability of disclosing information which violates the privacy guarantee. This relaxation has to be made with care, so as to

avoid a *just a few* philosophy, where privacy is preserved for the majority except for a few participants [19].

2.2.1 Local Differential Privacy

If we want to eliminate the trust in the curator we can turn to the special setting of *local* differential privacy [19]. Here, the sender wants to protect the privacy of her data from the untrusted curator, making the curator simultaneously the receiver and the adversary. Local differential privacy also provides for a different notion of privacy as opposed to the central setting. In the central, each record in the database represents one participant. Hence, we aim to disguise the membership of an individual in the database. In the local setting, each individual represents his own database, thus we are mainly concerned with hiding the specific items therein. This necessitates that the data owner performs the privacy preserving randomization locally on each element v before the data release:

Definition 2.6 *Local Differential Privacy*
A randomized algorithm M satisfies (ϵ-δ)-local differential privacy iff for all inputs v,v' and all $y \in$ Range(M):

$$Pr[M(v) = y] \leq exp(\varepsilon) \times \Pr[M(v') = y] + \delta \tag{2.3}$$

Seeing as we want the owner of the data to maintain privacy even with respects to any data collector, we will hence work in the setting of local differential privacy. Its disadvantage to the central setting, however, is the need to add more noise. This stems from the fact the that randomization is applied directly on *each* individual's data, instead of the aggregated data.

2.2.2 Privacy Mechanisms

There are numerous methods for achieving differential privacy with three of the most fundamental ones being the exponential mechanism, the Laplace mechanism and randomized response. Below, we will briefly outline these approaches, their applicability and

limitations.

Laplace mechanism

The *Laplace mechanism* adds noise directly to the data. The noise is drawn from the Laplace distribution, and is determined by the sensitivity of a query. The sensitivity of a query is the maximum impact a single user can have on the outcome of the query [18]:

Definition 2.7 *l_1-sensitivity*

The l_1-sensitivity of a query q, on a database $\mathbb{N}^{|\mathcal{X}|}$ is given by:

$$\Delta q = \max_{\|X-Y\|_1=1} \|q(X) - q(Y)\|_1 \tag{2.4}$$

Intuitively, if we add enough noise so as to mask the maximum impact of a data point on the outcome of a query, we prevent distinguishability and hence achieve differential privacy. This is attained by drawing noise from the Laplace distribution where the scaling parameter λ is set to be $\frac{\Delta q}{\epsilon}$. Therefore, the privacy parameter ϵ determines how much of the sensitivity shall be masked.

Definition 2.8 *Laplace Mechanism*

The output of a query q with range $\mathbb{R}^K$ passed through the Laplace mechanism is given by:

$$q(x) = q(x) + (\Delta_1, \ldots, \Delta_k) \tag{2.5}$$

Where each Δ_i is drawn from $Lap\left(\frac{\Delta q}{\epsilon}\right)$ with $\mathbb{E}[\Delta_i] = 0$ and $Var[\Delta_i] = 2 \times \left(\frac{\Delta q}{\epsilon}\right)^2$

As a result, the added noise Δ will on average not be biased. It amounts to the value needed for concealing the impact of an individual element within the privacy budget. However, this approach of direct noise addition is not always ideal.

Exponential Mechanism

The exponential mechanism presents an alternative to the data perturbation based on adding direct noise. Its implementation is beneficial when either operating on non-

numerical data, or when adding noise to the numerical value would destroy its utility [29] [19]. The exponential mechanism applies a utility function to rank the output probabilities of values based on their utility scores u.

Definition 2.9 *Exponential Mechanism*

$M_{EXP}(X, u, R)$ *selects and outputs an element* $r \in R$ *of database* X *with probability proportional to* $exp(\frac{\varepsilon u(x,r)}{2\Delta u})$.

The utility of an element can be regarded as the informative value it has for the query, and Δu represents the maximum possible change of utility caused by a single element. We can define the mapping of utility scores as: $u : X \times R \to \mathbb{R}$, where $u(X, r)$ represents the utility of the output r for a query on database X. The output probability of the values drops exponentially with their utility, which means that the exponential mechanism can achieve high utility of query answers while still providing privacy guarantees.

Randomized Response

An early example of randomizing outcome in order to provide privacy goes back to data collection in social sciences [41] and the mechanism of randomized response. Here, the truthful response to a query is defined by chance. The standard example is the response to a "Yes" or "No" question based on a coin flip: if heads, the true answer is replied. Otherwise, the coin is flipped again, and the outcome decides which of the two options to respond. With the introduced uncertainty about the truthfulness of the answer, we have now provided the ground for privacy by the concept of *plausible deniability* [19]. This can be shown to satisfy ϵ-differential privacy, even beyond binary answers, by:

Definition 2.10 *Generalized Randomized Response*

Given a query q*, and a database* X *with* $q(X) = z$ *where* $z \in Z$*, applying generalized randomized response to* q *makes the output probabilities of q(X):*

$$q(X) = \begin{cases} z, & \textit{with probability} \quad \frac{e^{\epsilon}}{e^{\epsilon}+|Z|-1} \\ \neg z, & \textit{with probability} \quad 1 - \frac{e^{\epsilon}}{e^{\epsilon}+|Z|-1} \end{cases} \tag{2.6}$$

Composition

The previously described mechanisms provide us with a toolbox for designing differentially private algorithms. Fortunately, we are allowed to combine these mechanisms to design more sophisticated algorithms [19]. The resulting mechanism needs to account for the privacy loss of each mechanism it is composed of:

Definition 2.11 *Composition*
Given two independent mechanisms M_1 and M_2 which are (ϵ_1, δ_1)-differentially and (ϵ_2, δ_2)-differentially private, the mechanism $M_{1,2}$ is $\big((\epsilon_1 + \epsilon_2), (\delta_1 + \delta_2)\big)$-differentially private.

In terms of the current work, the `MinHash` algorithm needs to concatenate several minimum hash values to gain accuracy in similarity estimations. We can therefore predict that some form of composition of mechanisms could be needed.

We have now seen several methods for making procedures satisfy differential privacy restrictions, and how to combine them. We will use the knowledge of these tools in the review of the current state of research in the field.

Chapter 3

Related Work

In the broader sense of privacy-preserving data analysis, statistical queries (i.e. queries about the sum of entries) have been extensively studied since the late 70's [16]. The methods applied for making statistical queries privacy preserving are often categorized in three types: query restriction, data perturbation, and output perturbation [1]. As this work focuses on similarity search in the local setting, we focus on data perturbation. Generally, data perturbation for enabling privacy preserving queries on databases is well studied (see [30], [42] for example).

LDP-Systems that have been deployed in practice, though, focus on less complex count queries as opposed to similarity search. A survey of current research will show that similarity search as a complex query, within the setting of *local* differential privacy, is an open, and challenging research question. We will hence first give a broad overview of different approaches to make similarity search more privacy-preserving in general. Thereafter, we will introduce two approaches which we evaluate as most interesting in our attempt to perform similarity search in the local differentially private setting.

3.1 Similarity Search in the Context of Differential Privacy

Over the last decade, the growing popularity of recommender systems and awareness of privacy concerns, has made similarity search within a differentially private setting an increasingly researched topic [33] [12]. Nevertheless, due to the young history of differential privacy, many algorithms for data analysis are yet to be weaved into this framework. The research has especially focused on developing private recommender systems both within the central and the fully distributed model. In the following, we will investigate the existing approaches in this field, and draw conclusions in how they support us in our endeavour.

3.1.1 Private Similarity Search in the Central Setting

McSherry et al. [28] were the first researchers to apply differential privacy to recommender systems. Based in the central setting and using a selection of leading algorithms from the Netflix prize competition, they were able to show the feasibility of differential privacy guarantees without significant loss in recommendation accuracy. Since then, numerous works have been conducted on collaborative filtering techniques in the central DP-setting. In [46], the exponential mechanism used achieves a decreasing sensitivity with an increasing number of users. Based on this work, [13] claim to have improved the protocol. Here, the researchers apply the Pearson coefficient as similarity measure. The similarity is computed directly on the users. Wong et al. [43] propose a protocol directly tailored to privacy-preserving Jaccard similarity computation. The privacy guarantees are mainly derived from an encryption scheme, which relies on a semi-trusted curator. Lately, Dhaliwal et al. [14] developed an approach for performing differentially private clustering, which they claim to be immune to reconstruction attacks. Here, the curator makes use of both dimensionality reduction techniques and Laplace noise. Although demonstrating that one can retain utility in similarity computations when adding direct noise, the setting is again central and thus not directly applicable to our circumstances.

The described approaches contribute to our understanding, as they show the feasibility of preserving privacy in similarity search, in general. Nonetheless, in all the before mentioned work, the mechanisms to retain privacy rely on a (semi)trusted curator, and are hence not directly applicable in the setting of local differential privacy. As we wish to eliminate the reliance on trust in the curator, we now turn to another model of differential privacy somewhat adjacent to the local model.

3.1.2 Private Recommender Systems in the Distributed Setting

Several works have been proposed which guarantee differential privacy in the distributed model for recommender systems. Allagan et al. [2] investigate profile perturbation in a distributed model. Their `BLIP`-mechanism is based on the Bloom filter technique as opposed to `MinHash`, and as a similarity measure uses scalar product or cosine similarity instead of Jaccard resemblance. The perturbation is based on randomized response. Boutet et al. [10] propose a mechanism to protect data in a collaborative filtering system. Their approach also incorporates profile perturbation to ensure privacy guarantees through a randomizing dissemination protocol which resembles randomized response. Due to the item-based approach of the collaborative filtering, this method does not allow for similarity search between users. These approaches apply similar means to achieve differentially private similarity search as one would do in the local setting. However, their solutions rely on dissemination as the means of recommendation generation while we enquire about a system able to process similarity queries, given user profiles.

If we turn our focus to systems in the local model of differential privacy, we observe that most of the achievements within the setting of LDP have been in regards to different types of count queries. Here, notable systems have been deployed [5][15][20]. Furthermore, academia is increasingly interested in the privacy guarantees LDP can provide. Wang et al. [38] evaluated several LDP-protocols in terms of their performance. The authors concluded that randomized response outperforms other protocols, when the number of possible responses is low. On the contrary, RR was found to produce poor results with

higher dimensionality of the encoding. Similar findings have been previously produced in the study of the trade-off between privacy guarantees and utility in regards to statistical estimators [17]. This trade-off was found particularly high in certain settings, such as when working with high dimensional data. Thus, we estimate the task of similarity search in a local differential private setting to suffer from notable utility loss.

As we chose the `MinHash` algorithm for locality sensitive hashing, we are curious about the implicit differential privacy guarantees it might provide. These might be due to the noise injected by probability of random collision, and be similar to implicit privacy guarantees provided by sketching [7]. In addition, Boutet et al. [9] consider the impact of choice in similarity measure on the ability to identify values. Through evaluation of attacks on recommender systems with seven different similarity measures, they among other things conclude that the more accurate the similarity measure, the easier to attack. Jaccard similarity falls into this category of very effective but vulnerable. This shows again the accuracy-privacy trade-off. Hence, we suspect that using `MinHash` provides us with some privacy guarantees, but its vulnerability to attacks indicates that it in its pure form most likely is subject to privacy loss.

3.2 Locally Private Jaccard Similarity Estimation

To our knowledge, only Yan et al. [44] have yet published a protocol for performing Jaccard similarity estimation in the setting of local differential privacy. The foundation of their `PrivMin` algorithm is the exponential mechanism, which is applied on the min-wise hashing. The algorithm draws minimum hash values from the range of hash values based on a probability distribution derived by the exponential mechanism.
The *minimum* hash value which is the output by this hashing scheme is therefore not strictly the 'true' minimum hash value. The authors of the paper claim their algorithm to be ϵ/K -locally differentially private, and present experimental results which illustrate high accuracy in similarity search, with strict privacy budgets. However, following the acquired insights from the previously described research, achieving these levels of accuracy with

such a strict privacy budget, seems questionable. We will further analyze the approach in chapter 5.

3.3 Sub-Linear Privacy-Preserving Near-Neighbor Search with Untrusted Server

While not formally placed in the setting of local differential privacy, the research of Riazi et al. [34] might provide a solution to solving the task of LDP similarity search. The `Secure MinHash` algorithm aims at enabling privacy-preserving approximate near neighbour search (ANN) in an attack-based model. Hence, the protection of privacy consists of preventing the reconstruction of attributes of user profiles by triangulation attack.

As described in chapter 2.1, locality-sensitive hashing is one method of choice to achieve sub-linear running time in ANN search. The researchers claim that releasing the true LSH-embedding is unsuitable to guarantee privacy in a semi-honest setting with no trusted server. The proposed attack uses the generally desired characteristic of the LSH to preserve pairwise distances as the mechanisms vulnerability. To counteract this vulnerability, the `Secure MinHash` concatenates K minimum hash values and maps them to $\{0, 1\}$. This comes with a rapid decrease in collision probability between dissimilar elements and a corresponding increase in privacy.

Although the notion of privacy preservation differs from the LDP setting, the `Secure MinHash` algorithm proposed by Riazzi et al. might be contributing to the research question of how to make privacy-preserving similarity search in the LDP setting. The algorithm provides a protocol for data perturbation before releasing it to the untrusted server. We will therefore further investigate how the privacy guarantees for the algorithm hold when crafted into the framework of local differential privacy.

Chapter 4

The Privacy Guarantees of MinHash

In the previous chapter, we have identified two approaches that aim to make Jaccard similarity estimation privacy-preserving. As it is yet an open question if the proposed algorithms succeed in making the similarity search LDP-private, we continue in this chapter by investigating `MinHash` as the core approach based on Jaccard similarity. Does `MinHash` by itself provide some privacy? We will investigate this question by first considering the privacy guarantees in two existing privacy notions. Following, we will conduct our own evaluation of the privacy guarantees of `MinHash` in the context of (ϵ, δ)-LDP.

4.1 Existing Privacy Notions of MinHash

In the following, we will evaluate the privacy guarantees of `MinHash` through the lens of privacy as defined in the two previously identified most relevant papers [34][44].

4.1.1 ϵ-security

In the setting of local differential privacy, we are concerned about the ability to distinguish the contribution of individual elements in released data. This differs somewhat to the attack-based model of ϵ-security defined by Riazi et al. [34], where our concern is aimed at preventing an attacker from being able to infer a users values.
In order to fulfill the notion of privacy in the ϵ-security setting, this approximation shall be deemed impractical. To achieve this, we observe that unless a user is above a similarity

threshold S_0, there is no reason why a user making a similarity query should get anything but random noise in return.

Definition 4.1 *ϵ-security*

A hashing scheme mapping to $\mathbb{N}^K$ is ϵ-secure iff for any X and Y with $Jac(X,Y) \leq S_0$:

$$\frac{1}{K} \leq Pr[h(X) = h(Y)] \leq \frac{1}{K} + \epsilon \tag{4.1}$$

Within this framework, `MinHash` is obviously not secure, given that $Pr[h(X) = h(Y)] = Jac(X,Y)$. Since this function grows linearly with the Jaccard similarity, there will not be a threshold at which the definition is satisfied (except from $Jac(X,Y) = 0$). In addition, this model of privacy is not applicable in the LDP-setting, since we need a trusted curator to compute the similarity between user profiles. Furthermore, the privacy guarantee is only provided for non-similar users. Above the similarity threshold, there are no privacy guarantees.

4.1.2 Conditional Differentially Private Set Operations

Yan et al. [44] define a notion of privacy which `MinHash` can satisfy. In this notion, privacy is preserved when changing a single element in a data set does not change the similarity estimation more than some factor from the most probable range. Given two sets X and Y with $|X| = |Y| = n$, a change of a single element in either set can at most result in a change of $\frac{1}{n}$ in the Jaccard similarity between the two. Since the Jaccard similarity estimation of `MinHash` has an error rate of $\theta = \frac{1}{\sqrt{K}}$, the most probable interval of estimated similarity between two neighbouring sets is $[Jac(X,Y) - \theta - \frac{1}{n}, Jac(X,Y) + \theta + \frac{1}{n}]$:

Definition 4.2 *Conditional ϵ-DPSO*

A randomized set operation M provides conditional ϵ-differential privacy if it for all neighbouring set pairs $\{X,Y\}$ and $\{X,Y\}'$ differing in at most one element, and for the most probable outputs O satisfies:

$$Pr[M(\{X,Y\} \in O] \leq e^{\epsilon} \times Pr[M(\{X,Y\}' \in O] \tag{4.2}$$

Though placed in the local setting, this argument however, seems to be tailored to the central setting of differential privacy. It relies on a trusted curator as the privacy guarantee is given w.r.t the Jaccard similarity estimation, and not the published signatures. In LDP we are interested in providing privacy guarantees to the published data, which in this case is the signatures. So, what is needed is an evaluation of the privacy guarantees of these.

4.2 MinHash in the Local Setting

As we want to publish the signatures to an untrusted curator, the privacy guarantees of `MinHash` can be measured by the curator's disability to distinguish signatures of neighbouring databases from each other. Thus, we propose `MinHash` to satisfy the following privacy guarantees in the LDP-setting:

Theorem 4.1 *Privacy guarantee of MinHash*
Given two neighbouring databases X, Y with $\|X - Y\|_1 \leq 1$ and $|Y|, |X| \leq \tau$, the MinHash algorithm mapping X, Y to $h(X), h(Y) \in \mathbb{N}^K$ using K hash functions satisfies $\left(\frac{K}{\tau}, 1 - exp\left(-\frac{K}{\tau}\right)\right)$*-LDP.*

Proof We begin by evaluating the case of a single run of the algorithm and proceed to extend this to K runs. Consider two databases $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_n\}$ with $\|X - Y\|_1 \leq 1$. Privacy leakage will happen when it is possible to distinguish Y from X due to an added element y^*. Therefore, we now evaluate the relationship between $Pr[h(X) = z]$ and $Pr[h(Y) = z]$ where $z = h_{min}(X)$. The probability of outputting z with Y depends on the newly inserted element y^* and the event $A = \{h(y^*) > z\}$ and its complement $\neg A = \{h(y^*) \leq z\}$:

$$Pr[h(Y) = z] = Pr[A] \times Pr[h(Y) = z|A] + Pr[\neg A] \times Pr[h(Y) = z|\neg A] \tag{4.3}$$

$$= Pr[A] \times Pr[h(X) = z] + Pr[\neg A] \times Pr[h(Y) = z|\neg A] \tag{4.4}$$

$$\leq Pr[A] \times Pr[h(X) = z] + Pr[\neg A] \tag{4.5}$$

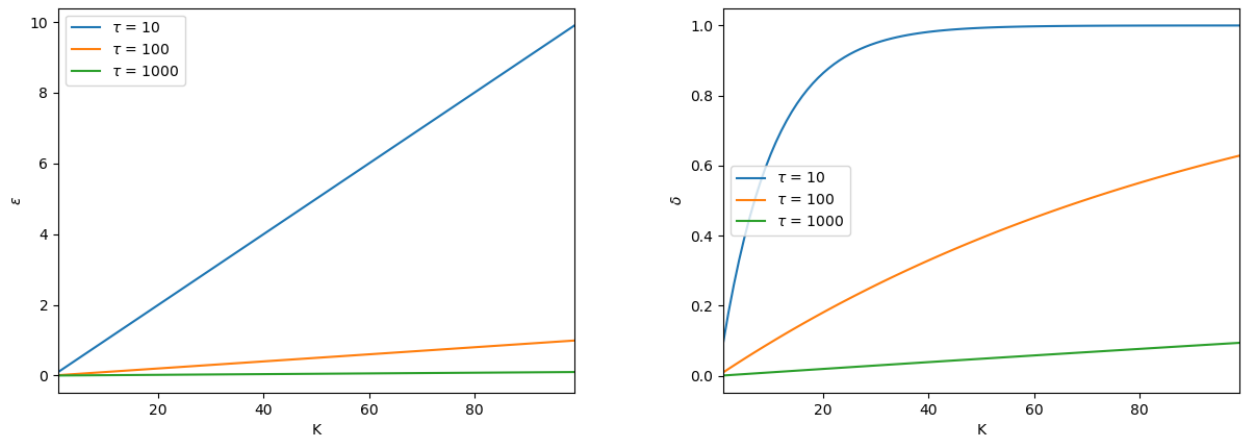
$$= e^{\epsilon} \times Pr[h(X) = z] + \delta \tag{4.6}$$

We observe that if $h(y^*) > z$, there is no change to the probability of outputting z by adding y^*. The second part of the term will be set to $Pr[\neg A] * 1$ to reflect our lack of knowledge about $Pr[h(Y)|\neg A]$. The occurrence of this worst-case scenario hence signifies the cases where we cannot provide any privacy guarantees and thus represents δ in the (ϵ, δ)-LDP setting.

Following the definition of `MinHash`, the probability of any element of X being the minimum hash value is at most $\frac{1}{\tau}$. Extending the approach above to K runs of the algorithm, we define the event $B = \{h_{mini}(X) = h_{mini}(Y)\}$, so that:

$$
\begin{aligned}
Pr[h(Y) = z_1, \ldots, z_K] &\leq Pr[B]^K \times Pr[h(X) = z_1, \ldots, z_K] + (1 - Pr[B]^K) \\
&= \left(1 - \frac{1}{\tau}\right)^K \times Pr[h(X) = z_1, \ldots, z_K] + \left(1 - \left(1 - \frac{1}{\tau}\right)^K\right) \\
&\approx exp\left(-\frac{1}{\tau}\right)^K \times Pr[h(X) = z_1, \ldots, z_K] + \left(1 - exp\left(-\frac{1}{\tau}\right)^K\right) \\
&= exp\left(-\frac{K}{\tau}\right) \times Pr[h(X) = z_1, \ldots, z_K] + \left(1 - exp\left(-\frac{K}{\tau}\right)\right)
\end{aligned}
$$

Figure 4.1: Privacy budget of `MinHash` with increasing K and different τ



(a) ϵ (b) δ

As both ϵ and δ depend on K and the size of the user set τ, the privacy guarantees provided by `MinHash` will vary widely across different values of the parameters.
The significance of τ becomes apparent from figure 4.1. It illustrates how quickly the privacy guarantees are degrading with smaller database sizes.

In conclusion, we recognize that `MinHash` alone fails to provide strong privacy guarantees. Hence, we will in the following return to the privacy enhancing mechanisms introduced in Chapter 2.2.

Chapter 5

Providing ϵ-LDP with MinHash

Given the state of privacy guarantee of `MinHash` in the local setting and the fact that Yan et al. [44] claim to provide an approach with very strong ϵ-LDP guarantees, we choose to investigate their approach further. The approach of applying the exponential mechanism on the minimum hash values initiates a consideration as to its validity.
We hence begin this chapter with an analysis of the theoretical foundations of their approach with special focus on the nature of utility in `MinHash` and techniques for increasing the output probability of minimum hash values. Then we design an alternative method, the `Bucket Privmin` algorithm, which satisfies the previous theoretical considerations. Finally, we evaluate the potential of such an approach given the needed parameters.

5.1 Utility

The `PrivMin` algorithm proposed by Yan et al. [44] selects minimum hash values with the exponential mechansim. The underlying utility function for the mechanism maps a utility score u to the set of hashed values $h_k(X)$ of X based on their index (r) in $h_k(X)$ when sorted from smallest to largest hash value:

Definition 5.1 *Utility function of the PrivMin*

Given a database $X \in \mathbb{N}^n$*, hashed to* $h_k(X) = \{h_k(x_1), \ldots, h_k(x_n)\}$*, the utility of all* $h_k(x_i)$ *given their* $index(r)$ *in* $h_k(X)$ *sorted in increasing order, is:*

$$u(X, r) = |h_k(X)| - index(r)$$

Our critique of this approach stems from the fact that the *minimum* hash value is usually the only value among the set of hash values which is of interest. It follows that although we can make an intuitive argument for the `MinHash` algorithm providing the ground for applying a utility function, there is no basis for making distinctions between the utility of remaining non-minimum hash values.

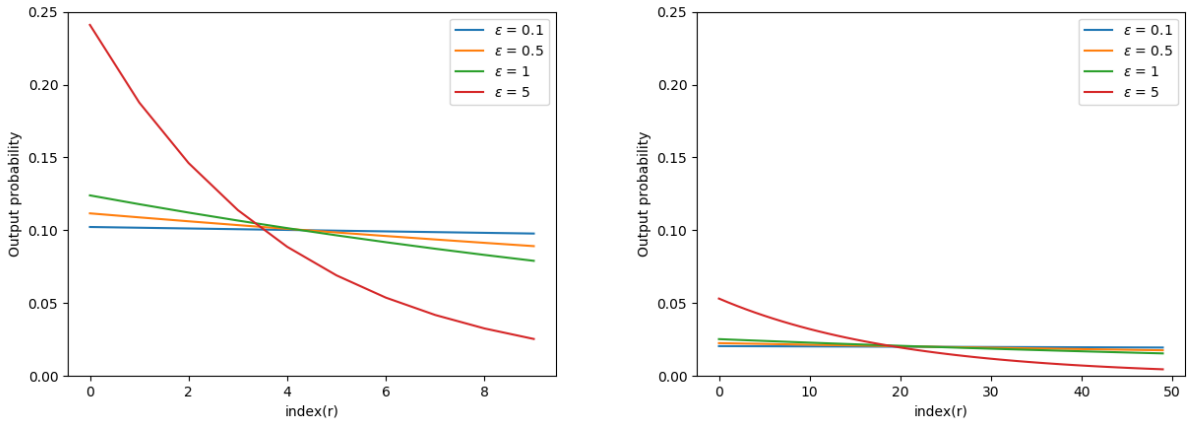
To clarify, the utility function of the `PrivMin` maps the utility scores of the hash values linearly degrading; the second smallest hash value is assigned a higher utility than the third smallest hash value, and so forth. Differentiating between hash values in such a manner should be viewed critically. If we assume to pick the minimum hash value most of the time, picking *any other* value than the minimum, and comparing it to the *presumed* minimum of another set, will yield little utility. The main property of the `MinHash` procedure for estimating Jaccard similarity is that a collision of two minimum hash values corresponds to the probability of randomly sampling a value from their union and finding it in their intersection. It is not clear that this insight holds when there are many different combinations of collision events where only one amounts to a genuine minimum hashing collision.

Approaches like the `Min-Max hashing` [24] show that the utility of a hash value is not depending on it being the *minimum* value per se. However, we argue that for the locality-sensitive hashing scheme to be an unbiased estimator for the Jaccard similarity, we shall pick *consistently*. Otherwise, we need to adapt our similarity estimation so that we take into account the inconsistent value selection.

5.2 Shrinkage of the Output Space

The discussion on utility highlights the importance of choosing the *true* minimum hash value. Hence, we are interested in the output probabilities of the hash values when the exponential mechanism is applied.

Figure 5.1: Output probabilities with utility of hash value



(a) $|h_k(X)| = 10$ (b) $|h_k(X)| = 50$

Figure 5.1 shows the output probabilities of the hash values in set X, based on their position r in $h_k(X)$ in sorted order. The output ratio of the minimum hash value to other values becomes ever smaller with smaller ϵ, larger K, and larger $|X|$. It thus becomes obvious that the output distribution does not have the desired qualities of choosing the minimum hash value as output with a significantly higher possibility than other values, at least not for $\epsilon \leq 1$. We can argue that in a setting as described, the exponential mechanism will destroy any utility of `MinHash` for Jaccard similarity estimation.
From [19] we know that the two parameters to raise the utility gained from the exponential mechanism are the privacy budget ϵ and the size of the output range $\mathcal{R}$.

In acknowledgement of the diminishing utility, Yan et al. [44] propose a shrinkage of the output space as a solution. In the proposed `PrivMin` algorithm, the researchers assign $RANGE$ values that are empirically determined in conjunction with the average size of the user sets $|X_{AVG}|$.

In their implementation, this evaluates to:

$$RANGE = \left(\frac{|X_{AVG}|}{1.995} \right)^{-\epsilon}$$

This results in the output space of each set of hash values $h_k(X)$ being shrunk to $\mathcal{R} = RANGE \times |h_k(X)|$. Table 5.1 illustrates the size of the resulting shrunk output space for different combinations of set sizes $|X|$ and privacy budgets ϵ.

	ϵ				
$\|X\|$	0.1	0.5	1	2	5
10	8.51	4.47	2.00	0.40	0.00
20	15.88	6.32	2.00	0.20	0.00
50	36.23	9.99	2.00	0.08	0.00
100	67.61	14.12	2.00	0.04	0.00

Table 5.1: Shrinkage of the output space with different $|X|$ and ϵ

As shown in table 5.1, the parameter ϵ has a significant influence on the size of the output range. In practice, as soon as $\epsilon \geq 1.2$, the output space ≤ 1. If we assume the `PrivMin` algorithm to use a ceiling function, this will in conclusion mean that with $\epsilon \geq 1.2$, the protocol uses an output range $\mathcal{R} = 1$. The output range will hence only contain the minimum hash value, and consequently, we can derive that the protocol falls back to using the 'regular' `MinHash` algorithm.

Additionally, it is important to note that the value of 1.995 in the denominator of the $RANGE$ function is arbitrarily chosen. The researchers make no connection between this value and the privacy guarantees of their `PrivMin` algorithm, although it must affect the ability to keep their privacy guarantees. By decreasing the denominator in the $RANGE$ function, we can shrink the output space to our liking, without reflecting this in our privacy guarantees. Nothing prevents us from taking this further and letting the arbitrary parameter approach 0, which would ensure that the algorithm always reverts back to a regular `MinHash`.

In conclusion, we have shown that although it seems necessary to reduce the size of the output space of the exponential mechanism in order to retain utility of `MinHash`, other

approaches are needed to satisfy ϵ-LDP guarantees.

5.3 Bucket PrivMin

Based on our previous reflections, we propose the randomizing, utility-considerate algorithm `Bucket PrivMin`. It consists of the general randomized response mechanism and a shrinkage of the output range.

5.3.1 Mechanism Selection

As described in 5.1, using the exponential mechanism might destroy the utility of the `MinHash` algorithm. We further argued that assigning utility values monotonically decreasing with the position in the hash value array is to be viewed critically. Therefore, we propose the following utility function where the utility values correspond to their actual utility in the estimation of similarity:

Definition 5.2 *Utility*

Given a user X hashed to $h_k(X)$ the utility of a hash value $h(x_i)$ is given by:

$$util(h(x_i)) = \begin{cases} 1, & h(x_i) = min(h_k(X)) \\ 0, & otherwise \end{cases} \tag{5.1}$$

Given the binary nature of the choices, we do not rely on the smooth sensitivity curve of the exponential mechanism. One might be tempted to therefore replace it with the in subsection 2.2.2 introduced mechanism of randomized response. Then we could simply treat the problem as a coin flip problem. The probability of outputting the minimum hash value then corresponds to one side of the coin and lying about the value corresponds to the other. However, to implement the `MinHash` algorithm in the RR setting, we cannot assume a binary output with the two options 'True' and 'False'. We have to view the problem as one of an m sided dice, where we assign some probability p to returning each of the possible output values [40]. Therefore, in our design of an algorithm, we either report the minimum hash value, or a random value with probabilities:

$$\text{output} = \begin{cases} \text{minimum hash value, with p} = \frac{e^{\epsilon}}{e^{\epsilon}+(m-1)} \\ \text{random hash value, with q} = \frac{1}{e^{\epsilon}+(m-1)} \end{cases}$$

Following the proof in [39], we know that this randomizing algorithm satisfies ϵ-LDP.

If we now turn back to the exponential mechanism, we can observe that with a binary utility function the output probabilities, given a user profile X with $|X| = m$, evaluate to:

$$Pr[h(X) = h_{min}(X)] = \frac{e^{\epsilon/2}}{e^{\epsilon/2} + \sum^{m-1} e^{0}} = \frac{e^{\epsilon/2}}{e^{\epsilon/2} + (m-1)} \tag{5.2}$$

$$Pr[h(X) \neq h_{min}(X)] = \frac{1}{e^{\epsilon/2} + \sum^{m-1} e^{0}} = \frac{1}{e^{\epsilon/2} + (m-1)} \tag{5.3}$$

Comparing the exponential mechanism with binary utility function to the generalized randomized response, we can see that we achieve higher output probabilities of the minimum hash value using the generalized randomized response. Hence, we design our privacy-preserving algorithm with the randomized response mechanism.

5.3.2 Bucketing

In order to provide plausible deniability, the output range of the randomizer cannot be limited to the set of hash values of one user. The range of possible outputs must include all possible hash values in the universe. This has tremendous effects on the output probabilities for the `MinHash`, and consequently for the utility of the algorithm. In order to maintain utility when applying the randomizing algorithms, we shall hence explore how to shrink the output range in a valid matter. One approach to do this is to introduce the technique of *bucketing*:

Definition 5.3 *Bucketing*

Given a universe of databases $\mathcal{X}$ where the average size of databases is $|\mathcal{X}_{avg}|$ and a universe of possible hash values $\mathcal{R}$, map $\mathcal{R}$ uniformly at random to B buckets so that $B < |\mathcal{X}_{AVG}|$.

All possible hash values are allocated to the buckets and instead of reporting the minimum hash value with probability p, we will now report the ID of the bucket containing the value with probability p. Because the range of possible outputs is then the number of buckets B, we shrink our output space since $B \leq |\mathcal{R}|$. Furthermore, since $B \leq |\mathcal{X}_{avg}|$ we will gain higher output probabilities for the minimum hash values in most cases. The final algorithms for the `Bucket PrivMin` are presented in algorithms 1 and 2.

Algorithm 1: BucketPrivMin (User side)

Input : Profile X with items $X = \{x_1, \ldots, x_n\}$, K hash functions, privacy budget ϵ.

Output: perturbed MinHash signature $h_K(X)$

Initialize $h_K(X)$ as empty list

for *k in K* **do**

  Initialize $h_k(X)$ as empty list

  for *x in X* **do**

    Append $h_k(x)$ to $h_k(X)$

  end

  $h_{min^*}(x) = min(h_k(X))$ with probability $= \begin{cases} min(h_k(X)), & \frac{e^\epsilon}{e^\epsilon + |\mathcal{R}| - 1} \\ \text{Random value} \in \mathcal{R}, & \frac{1}{e^\epsilon + |\mathcal{R}| - 1} \end{cases}$

  Append $h_{min^*}(X)$ to $h_K(X)$

end

return $h_K(X)$

Algorithm 2: BucketPrivMin (Server side)

Input : Universe of users $U = \{u_1, \ldots, u_n\}$, K hash functions, privacy budget ϵ.

Output: Matrix $M^{n \times K}$

Initialize M

for *user in U* **do**

  signature $\leftarrow$ BucketPrivMin_UserSide(user, K, ϵ)

  Append signature to M

end

return M

Since multiple hash values can share a bucket, we gain plausible deniability because it is now uncertain if an observed collision of bucket b is due to a collision of hash values. On the other hand,the increase in random noise will lower the accuracy of our Jaccard similarity estimation. So, while the output probability will be higher for smaller B, those benefits might be counteracted by the introduced noise.

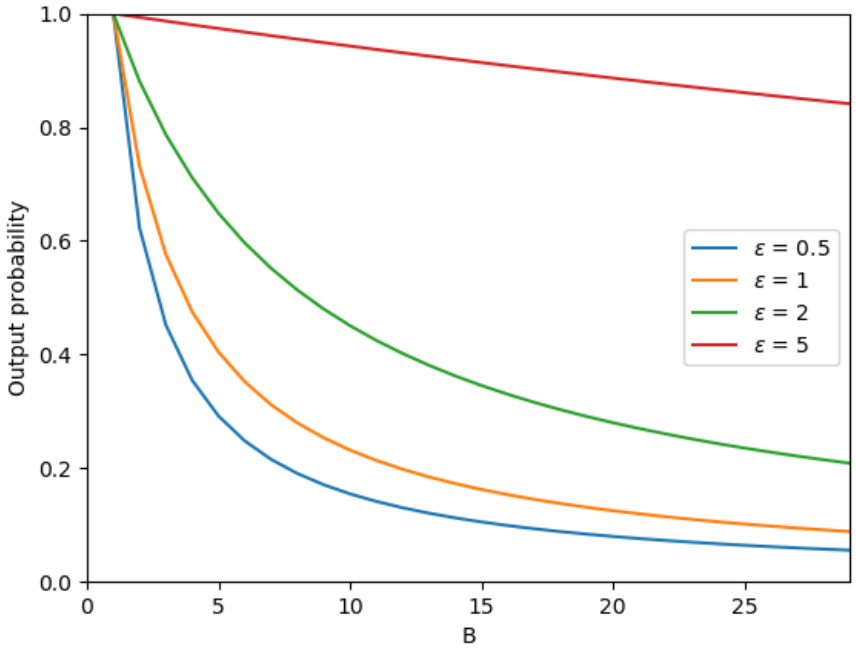


Figure 5.2: Output probabilities of the minimum hash value

As depicted in figure 5.2, the output probability of bucket b containing the minimum hash value drops exponentially with an increasing size of buckets B and is significantly influenced by the privacy budget ϵ.

5.3.3 Debias of the Estimator

For estimating the Jaccard similarity on the signature, we need to consider that the signatures have now been subjected to two levels of distortion; the randomization originating from applying generalized randomized response, and the aggregation of values into B buckets. In order to estimate the similarity from these signatures we need to take both into account. So, given two signatures $h_K(X), h_K(Y)$ with $|h_K(X)| = |h_K(Y)| = K$ mapped to B buckets, where we observe C collision of hash values, the collision probability

is:

$$Pr[h_i(X) = h_i(Y)] = \frac{C}{K} \tag{5.4}$$

$$= \left(\frac{e^\epsilon}{B-1+e^\epsilon}\right)^2 \times Jac(X,Y) \tag{5.5}$$

$$+ 2 \times \left(\frac{e^\epsilon}{B-1+e^\epsilon} \times \left(1 - \frac{e^\epsilon}{B-1+e^\epsilon}\right) \times \frac{1}{B}\right) \tag{5.6}$$

$$+ \left(1 - \frac{e^\epsilon}{B-1+e^\epsilon}\right)^2 \times \frac{1}{B} \tag{5.7}$$

In the above, three separate scenarios of causes for hashing collisions have been considered. Eq. (5.5) refers to the scenario where both of the minimum hash values values have been returned truthfully and they collide, i.e. the Jaccard similarity. In eq. (5.6) we account for the two scenarios where we observe a hash collision, but only one value is the *true* minimum hash value. Finally, in eq. (5.7) we consider the case of two false answers and a random collision. Rearranging the above we find that:

$$Jacc(X,Y) = \frac{B \times C - 2 \times K \times \left(1 - \frac{e^\epsilon}{B-1+e^\epsilon}\right) \times \left(\frac{e^\epsilon}{B-1+e^\epsilon} - \left(1 - \frac{e^\epsilon}{B-1+e^\epsilon}\right)\right)}{B \times K \times \left(\frac{e^\epsilon}{B-1+e^\epsilon}\right)^2} \tag{5.8}$$

Eq. [5.8] is then an unbiased estimator of the two signatures. Its application to nearest neighbour search can be seen in algorithm 3.

Algorithm 3: BucketPrivMin (Nearest Neighbour search)

Input : User ID u, Signature matrix $M^{n \times K}$, privacy budget ϵ, number of buckets B, N number of neighbours.

Output: Set of nearest neighbours $S_N(u)$

Retrive user u's signature from M

Set output probability p to $\left(\frac{e^{\epsilon}}{e^{\epsilon}+B-1}\right)$

Set q to 1 - p

Initialize S(u)

for *other user in M* **do**

  Count number of hashing collisions C

  Set similarity to $\frac{B \times C - 2 \times K \times q \times (p-q)}{B \times K \times p^2}$

  Append (similarity, other user) to S(u)

end

return $max_N S(u)$

We conclude this chapter by recognizing the significant need to shrink the output space. Bucketing might be a way to achieve that, but by this we also introduce additional noise due to higher collision probabilities. In conjunction with figure 5.2, we observe that we need a substantial privacy budget to achieve satisfying output probability on the true minimum hash value. Hence, we will explore an additional approach of performing privacy preserving Jaccard similarity estimation in the subsequent chapter.

Chapter 6

Providing (ϵ, δ)-LDP with MinHash

As seen in chapter 4, the method of introducing plausible deniability as a way of achieving ϵ-LDP guarantees is infeasible with `MinHash`. Subsequently in chapter 5, we then explored a combination of plausible deniability and reduction of output space as an alternative. The limitations of this approach due to the introduction of substantial random noise in the form of increased collision probability, prompts us to question whether an approach exclusively focused on increased collision probability will suffice privacy-wise. We thus first evaluate the privacy guarantees of the `Secure MinHash` [34] in the LDP-setting. Based on our assessment, we design the `Noisy Secure MinHash` to mitigate the privacy shortcomings identified by the analysis.

6.1 Altering Collision Probabilities

We begin by observing from our discussion in 4.2 that the `MinHash` procedure leaks problematic information as soon as the differing element between two databases influences the signatures. A natural solution to this issue would be to ensure that the collision probability remains high, despite the databases X and Y differing in an element.
Riazi et al. [34] use an approach where K minimum hash values are concatenated and mapped to a single bit. The resulting `Secure MinHash` algorithm has an exponential drop in the collision probability with increasing K plateauing at $\frac{1}{2}$. For non-similar users, the collision probability thus quickly drops to random.

For two users X and Y with similarity $Jac(X,Y)$ this results in a collision probability of [34]:

$$Pr[h(x) = h(y)] = \frac{Jac(x,y)^K + 1}{2} \tag{6.1}$$

If the `Secure MinHash` algorithm is evaluated within the setting of local differential privacy, we propose that:

Theorem 6.1 *Given two databases X,Y with $||X - Y||_1 \leq 1$ and $|X|,|Y| \leq \tau$, the Secure MinHash algorithm mapping $\mathbb{N}^\tau \to \mathbb{N}^K \to \{0,1\}$ satisfies (ϵ,δ)-LDP with $\epsilon = ln\left(\frac{exp(-\frac{K}{\tau})+1}{2}\right)$ and $\delta = \frac{1-exp(-\frac{K}{\tau})}{2}$*

Proof Given two databases X and Y with $||X - Y||_1 \leq 1$ and $|X|,|Y| \leq \tau$ mapped to $h_K(X)$, $h_K(Y)$ mapped to single bits using h_{uni}, we begin by observing that the probability of all K signatures being the same is:

$$\begin{aligned}
&Pr\Big[\big(h_{min1}(X),\ldots,h_{mink}(X)\big) = \big(h_{min1}(Y),\ldots,h_{mink}(Y)\big)\Big] \\
&= Jac(X,Y)^K \\
&= \Big(1 - \frac{1}{\tau}\Big)^K \\
&\simeq exp\Big(-\frac{K}{\tau}\Big)
\end{aligned}$$

In addition, in half of the events when $\big(h_{min1}(X),\ldots,h_{mink}(X)\big) \neq \big(h_{min1}(Y),\ldots,h_{mink}(Y)\big)$, they will still map to the same bit. This case results in:

$$\begin{aligned}
&Pr\Big[h_{uni}(X) = h_{uni}(Y) | \big(h_{min1}(X),\ldots,h_{mink}(X)\big) \neq \big(h_{min1}(Y),\ldots,h_{mink}(Y)\big)\Big] \\
&\times Pr\Big[h_{uni}(X) = h_{uni}(Y)] = \frac{1}{2}\Big(1 - \Big(1 - \frac{1}{\tau}\Big)^K\Big) \\
&\simeq \frac{1 - exp\big(-\frac{K}{\tau}\big)}{2}
\end{aligned}$$

So we find that:

$$Pr[h_{uni}(Y) = z] \leq \frac{exp(-\frac{K}{\tau}) + 1}{2} \times Pr[h_{uni}(X) = z] + \frac{1 - exp(-\frac{K}{\tau})}{2}$$

Consequently, the `Secure MinHash` algorithm does provide approximate local differential privacy guarantees with $\epsilon = ln\left(\frac{exp(-\frac{K}{\tau})+1}{2}\right)$ and $\delta = \frac{1-exp(-\frac{K}{\tau})}{2}$. In addition, we can see that both values tend to converge with $\frac{K}{\tau}$. More specifically:

$$\lim_{\frac{K}{\tau}\to\infty} e^{-\epsilon} = \frac{1}{2} \qquad \lim_{\frac{K}{\tau}\to 0} e^{-\epsilon} \approx 2.71 \tag{6.2}$$

$$\lim_{\frac{K}{\tau}\to\infty} \epsilon = -ln\left(\frac{1}{2}\right) = 0.7 \qquad \lim_{\frac{K}{\tau}\to 0} \epsilon \approx -ln(2.71) \approx 1 \tag{6.3}$$

$$\lim_{\frac{K}{\tau}\to\infty} \delta = \frac{1}{2} \qquad \lim_{\frac{K}{\tau}\to 0} \delta \approx 1 \tag{6.4}$$

The limiting behaviour makes intuitive sense: we would expect that if the sizes of the item sets are really large, and/or K is small, the probability of the differing element having an impact on the published signature is small. This leads to better privacy guarantees. On the other hand, if the databases are relatively small, it becomes difficult to provide any privacy guarantees, because the differing element inevitably will affect the signature, even with relatively small K. Significantly, we still have the issue of a problematic data release occurring when the signatures do not match in a given location. What we gain is the same as with the bucketing approach, namely an obscuring of the contributing element to cause the change. However, the ability to tell that an element has had an impact on the released signature still poses a privacy problem. What could alleviate this problem would be an ability to mask these instances, so that even though the signatures differ in a location, the change would be concealed.

6.2 Noisy Secure MinHash

Based on the previous considerations, we shall now consider methods to extend the privacy of the `Secure MinHash`. One way of concealing differences between signatures could be to make every user add noise to their signatures in order to ensure that the impact of any individual element of their data set is concealed.

As outlined in section 2.8, this requires us to scale the noise added according to the sensitivity of the query Δq, so our consideration is as follows: Given a database X containing a set of items $\{x_1, \ldots, x_n\}$ and signature computed using the `Secure MinHash` $h(X) = \{h_{min1}(X), \ldots, h_{minl}(X)\} \in \{0,1\}^L$, what is the sensitivity of publishing $h(X)$?

Well, in the worst case adding x_{n+1} to the set of X could result in: $\forall i : h_{mini}(X) = h_i(x_{n+1})$, which results in a sensitivity $\Delta q = L$. In practical terms, this makes simple noise addition an unfeasible solution to achieving ϵ-differential privacy of `Secure MinHash` while retaining some notion of utility. The added noise would rise very quickly with L. Making noise addition a feasible approach to the problem without having a linear increase of ϵ with L therefore seems to require restricting or otherwise manipulating Δq.

6.2.1 Noise Addition

One way of approaching this problem of rising sensitivity is the relaxation of the local differential privacy notion to the approximate setting introduced in section 2.6. As a result, we will allow some problematic data release, but preferably with very low probability. The aim of noise addition therefore becomes to add noise scaled so that the *most likely* cases will be masked. What is then needed is a measure of how likely it is that problematic data is released, preferably in a way that permits us to tune the threshold, so that noise can be added according to the accuracy and privacy needs.

Therefore, we introduce a version of Chernoff bounds which we will use in our pursuit to limit the sensitivity without compromising *too* much on the privacy guarantees:

Definition 6.1 *Multiplicative Chernoff Bound*

Let $X_1, \ldots, X_i$ be independent random variables where $X_i \in \{0,1\}$. Let P be the number of observed values and $\mu = \mathbb{E}[P]$ be the expected mean. Then:

$$Pr[P > (1+\beta)\mu] \leq exp\left(\frac{-\mu\beta^2}{3}\right) \tag{6.5}$$

This provides a bound on how probable it will be that the *observed* values deviate by some factor from the *expected* values. We know that even though $\Delta q = L$, it is exceedingly unlikely for a single element to actually have this large an influence, especially when the size of the user set and/or L is large. We can therefore limit our notion of sensitivity to cases where we observe a *likely* number of differences in signatures and ignore the problematic, but very unlikely cases.

In the following, we therefore aim to estimate the expected number of differing positions in two signatures.

We know that for `Secure MinHash` with databases X and Y with $\|X - Y\|_1 \leq \alpha$ and $|X|, |Y| \leq \tau$:

$$Jac(X,Y) \geq 1 - \frac{\alpha}{\tau} \tag{6.6}$$

Thus, the expected number of differing positions P will be:

$$\mathbb{E}[P] = \mathbb{E}\Big[\sum_{i=1}^{L}\big(h_i(X) \neq h_i(Y)\big)\Big] = L \times \left(1 - \frac{Jac(X,Y)^K + 1}{2}\right) \tag{6.7}$$

Using 6.1, we can evaluate the probability of observing $(1+\beta)$ times more actual differences between two signatures than expected:

$$Pr[P > \mathbb{E}[P] \times (1+\beta)] \leq exp\Big(\frac{-\mathbb{E}[P] \times \beta^2}{3}\Big) \leq \delta \tag{6.8}$$

We are interested in determining our sensitivity given a specific δ parameter threshold, such that we can adjust the added noise based on the privacy requirements. For this, we define an expression for β in terms of δ.

We remember that the bound is on the form e^{-x} and we know that then $x \leq \ln\left(\frac{1}{x}\right)$. This leads to:

$$\ln(\delta) \leq -\frac{\mathbb{E}[P] \times \beta^2}{3} \tag{6.9}$$

$$\Updownarrow$$

$$\ln\left(\frac{1}{\delta}\right) \geq \frac{\mathbb{E}[P] \times \beta^2}{3} \tag{6.10}$$

$$\Updownarrow$$

$$\beta \leq \sqrt{\frac{3 \times \ln(\frac{1}{\delta})}{\mathbb{E}[P]}} \tag{6.11}$$

Using eq. (6.11) as a substitute for β we claim with $1 - \delta$ percent certainty that we will at most observe:

$$\mathbb{E}[P] \times (1 + \beta) = \mathbb{E}[P] \times \left(1 + \sqrt{\frac{3 \times \ln(\frac{1}{\delta})}{\mathbb{E}[P]}}\right) = \Delta q \tag{6.12}$$

differences between two signatures of neighbouring databases. Plotting eq. (6.7) into eq. (6.12) we thus have a bound on our sensitivity. As we remember, δ values $\geq \frac{1}{\tau}$ are very problematic as this becomes an expression of the *just a few* philosophy of privacy, where we *allow* a few values of the signature to be distinguishable. We choose to avoid such a scenario and explore the impact on Δq when $\delta = \frac{1}{c\tau}$ for some $c \geq 1$.

As illustrated in figure 6.1, the bounding decreases the sensitivity of the query to a very large extent. In addition, we observe that even though we become increasingly certain that no problematic data release will happen, the sensitivity only rises relatively slowly. Based on the previous analysis, we propose the `Noisy Secure MinHash` algorithm. It is comprised of two separate procedures, which are presented in algorithms 4 and 5.

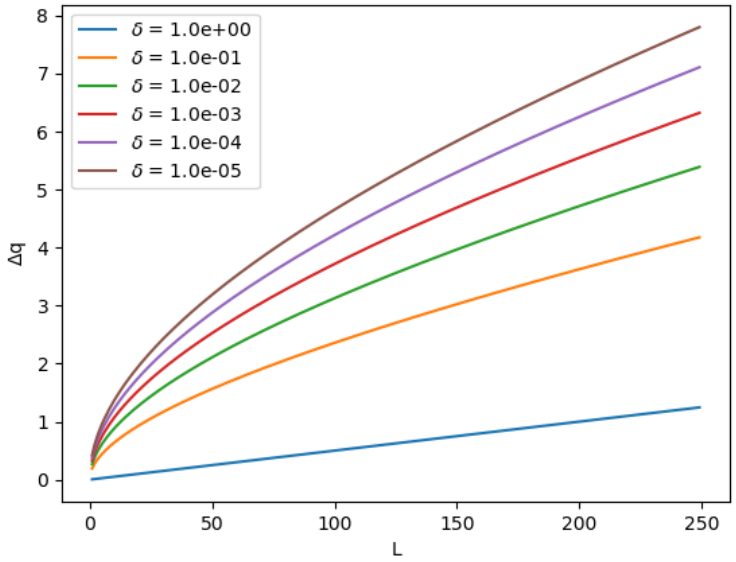


Figure 6.1: Sensitivity with $\alpha = 1$, $\tau = 100$, increasing L and different values of δ

Algorithm 4: Noisy-Secure-MinHash (Server side)

Input : Universe U with users $U = \{u_1, \ldots, u_n\}$, L hash functions, K, privacy budget ϵ, Min. user set size τ, Neighbourhood definition α, $c \geq 1$

Output: Matrix of perturbed signatures $M^{n \times L}$

Set $\delta = \frac{1}{c \times \tau}$

Set similarity threshold $Jac' = 1 - \frac{\alpha}{\tau}$

Set $\mathbb{E}[P] = L \times \left(1 - \frac{Jac'^K + 1}{2}\right)$

Set $\Delta q = \mathbb{E}[P] \times \left(1 + \sqrt{\frac{3 \times \ln(\frac{1}{\delta})}{\mathbb{E}[P]}}\right)$

Initialize *Noisy similarity matrix* $M^{n \times L}$

for *user in* U **do**

  Noisy signature ← user.User_side_algorithm(K, L, $\frac{\Delta q}{\epsilon}$) *See algorithm (5)*

  Append noisy signature to M

end

return M

Algorithm 5: Noisy-Secure-MinHash (User side)

Input : Profile X with items $X = \{x_1, \ldots, x_n\}$, L hash functions, K, noise scale parameter b.

Output: Perturbed MinHash signature $h_L(X)$

Initialize $h_L(X)$ as empty list

for *l in L* **do**
  Initialize empty hash value list
  for *x in X* **do**
    Append $h_l(x)$ to list of hash values
  end
  Append $min(h_l(X)) + Lap(b)$ to $h_L(X)$
end

return $h_L(X)$

6.2.2 Noisy Similarity Estimation

Although the `Noisy Secure MinHash` algorithm can provide LDP guarantees, it changes the inner workings of `MinHash` substantially. As we add real valued noise to our signatures, we are no longer able to count hashing collisions. To rectify this we propose that:

Theorem 6.2 *Noisy Similarity Estimation*
Given two users X and Y, their hash signatures $h(X)$ and $h(Y)$, where $|h(X)|, |h(Y)| \in \{0,1\}^L$ and their noisy hash signatures $h'(X)$,$h'(Y) \in \mathbb{R}^L$ where noise is added from $Lap(b) = Lap\left(\frac{\Delta q}{\epsilon}\right)$. The Jaccard similarity between X and Y can be estimated by:

$$Jacc(X,Y)^K = 2 \times \frac{L - \left(\sum_{i=1}^{l}(h'_i(X) - h'_i(Y))^2] - 2 \times L \times Var[Lap(b)]\right)}{L} - 1 \qquad (6.13)$$

The proof can be found in Appendix A. Based on the above, we propose an additional procedure for the `Noisy Secure MinHash` for estimating the Jaccard similarity of noisy signatures. It is presented in in algorithm 6.

Algorithm 6: Noisy-Secure-MinHash (Nearest Neighbour Search)

Input : ID of user x, Noisy similarity matrix $M^{n \times L}$, noise scale parameter b, number of neighbours N.

Output: List of N nearest neighbours $S_N(x) = [u_1, \ldots, u_N]$

Initialize $S(x)$ as empty list

Get user x's signature x' from M

for *other signature y' in M* **do**

$dist = \sum^{L}(x' - y')^2$ *Euclidean distance between x' and y'*

$col = L - (dist - 4 \times L \times b^2)$ *Estimated number of hash collisions between x' and y'*

Append $\sqrt[K]{2 \times \frac{col}{L} - 1}$ to $S(x)$

end

Sort $S(x)$ from highest to lowest and generate list of top N similar users $S_N(x)$

return $S_N(x)$

Chapter 7

Experiments

7.1 Implementation

The source code of this work contains various implementations of algorithms described in the previous chapters. The algorithms include `MinHash`, `Secur eMinHash` based on Riazi et al. [34], `PrivMin` based on Yan et al. [44], `Bucket PrivMin` and `Noisy Secure MinHash`. All experiments were run 10 times with the length of the signature being 100 to ensure more consistent results. The code is available on GitHub [1]. The experiments are collected in a Jupyter notebook.

7.1.1 Datasets

The experiments have been conducted on two real-life data sets. Both data sets contain a mapping from user profiles to a set of items. The data sets consist of the MovieLens and Last.FM data set. The data extracted from the MovieLens data consists of 2.112 users mapped to a preference set of movies (rating ≥ 4.5) with an average similarity of 6.6%. The user data extracted from the Last.FM data consists of 1.892 users mapped to their 20 highest rated songs with an average similarity of 1.7%. As opposed to the MovieLens data, the user sets here are generally of fixed size. While the average set size in the Last.FM data is 19.8 with $\sigma = 1.78$, the average set size in MovieLens is 178.1, with

[1] https://github.itu.dk/jansc/Thesis

σ = 187.46. So, the major difference between the two data sets is the variability of the sizes of their user sets. Both data sets are publicly available [2].

7.1.2 Metrics

In order to evaluate the quality of our implementation we chose metrics to evaluate our algorithms in regards to both the accuracy of the estimations of similarity, and the utility when performing similarity search. For this, we have chosen to use the mean squared error and a version of approximate recall.

Mean Squared Error

As a metric to measure the accuracy of our similarity estimation we use the mean squared error MSE. The MSE is the mean of the squared sum of differences between the true and the estimated Jaccard similarity. A MSE approaching zero indicates smaller differences between the accurate and the predicted value, and a better accuracy.

$$MSE = \frac{1}{n} \times \sum_{i=1}^{n} \Big(Jac(X_i, Y_i) - Jac(X_i', Y_i') \Big)^2$$

Approximate Recall

We use a version of *approximate* recall where we select the set M of estimated nearest neighbours and the set N of true nearest neighbours. We allow for $|M| \geq |N|$ and count it as true positives (TP) for all $m_i \in M$ if $m_i \in N$ and false negatives (FN) for all $n_i \in N$ if $n_i \notin M$. By allowing $|M| \geq |N|$ we try to capture the actual utility of a similarity search system based on the algorithms. We are less interested in returning the exact N nearest neighbours, what we are interested in is whether the returned users of a similarity query can be considered similar. We therefore in the coming section work with N = 20 and M = 100 which correspond to the respectively ~ 1% and ~ 5% most similar users to the query point in the chosen data sets.

[2] https://grouplens.org/datasets/hetrec-2011/

So, our recall metric of:

$$Rec = \frac{TP}{TP + FN}$$

measures the rate at which we find the top 20 nearest neighbours in the top 100 estimated nearest neighbours.

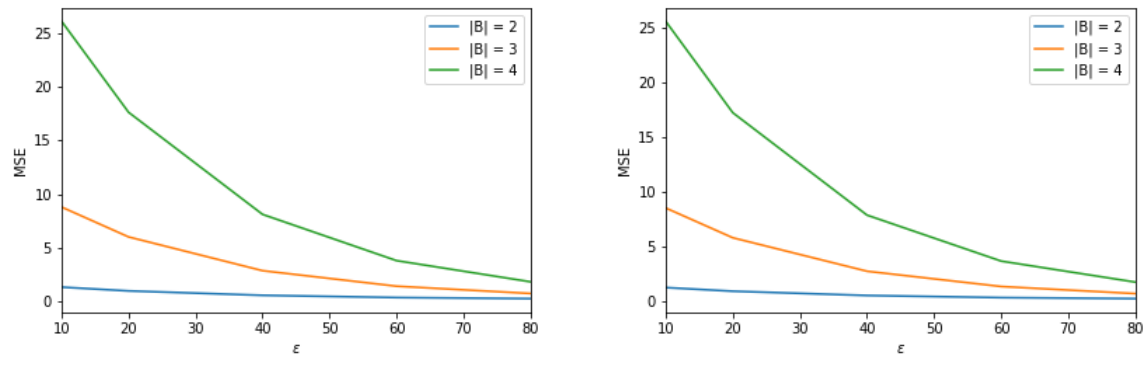
7.2 Results

We begin by presenting the results of the similarity estimations on both data sets using MSE as a metric for both algorithms. We do this to evaluate the configuration of their parameters and their impact on the accuracy of the similarity estimations. Based on the results we fix parameters for the algorithms and compare their estimations. Next, we present the results of performing nearest neighbour search with both algorithms on both data sets using approximate recall as a metric. Again, we use the results to fix parameters so as to be able to compare the performance of the algorithms.

7.2.1 Accuracy of Similarity Estimations

From figure 7.1 we see that for the `Bucket PrivMin` algorithm the MSE rises very rapidly with low ϵ and high B.

Figure 7.1: Mean Squared Error of similarity estimations with Bucket PrivMin



(a) Last.FM (b) MovieLens

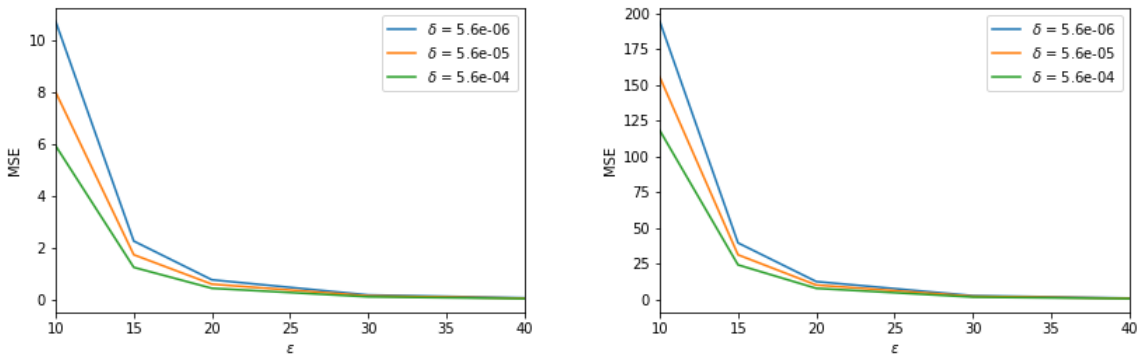
The fact that the number of buckets negatively affects the accuracy of the similarity estimations on both data sets is in line with the considerations made in chapter 5.

The benefit from the higher accuracy gained from increasing B is counteracted by the necessity for higher ϵ to increase the output probability. This confirms our previous theoretical considerations, indicating that a high output probability is essential for making accurate estimations.

The output probability is simply too low for the benefits of higher B to manifest. From this we draw the conclusion that on the given data sets, it will be necessary to have $B \leq 4$ to have any hopes for providing meaningful privacy guarantees with `Bucket PrivMin` while making accurate similarity estimations.

We next wish to evaluate the accuracy of Jaccard similarity estimation on our second proposed algorithm, the `Noisy Secure MinHash`. We compare the algorithm with different parameters of K, i.e. the number of minimum hash values mapped to a single bit. By this, we aim to reflect the algorithm's ability to provide higher privacy guarantees with larger K and the trade-off made with regards to ϵ.

Figure 7.2: Mean Squared Error of similarity estimations on MovieLens with Noisy Secure MinHash

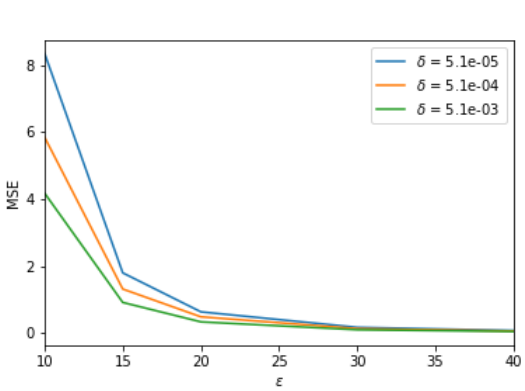


(a) K = 1

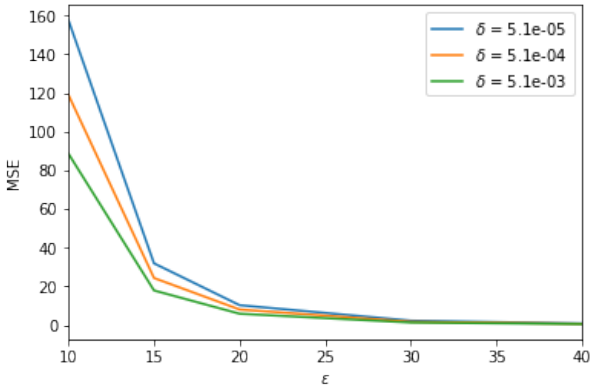
(b) K = 3

Comparing the two data sets with each other, we observe a similar behaviour of the algorithms on both MovieLens and Last.FM.

Figure 7.3: Mean Squared Error of similarity estimations on Last.FM with Noisy Secure MinHash



(a) K = 1

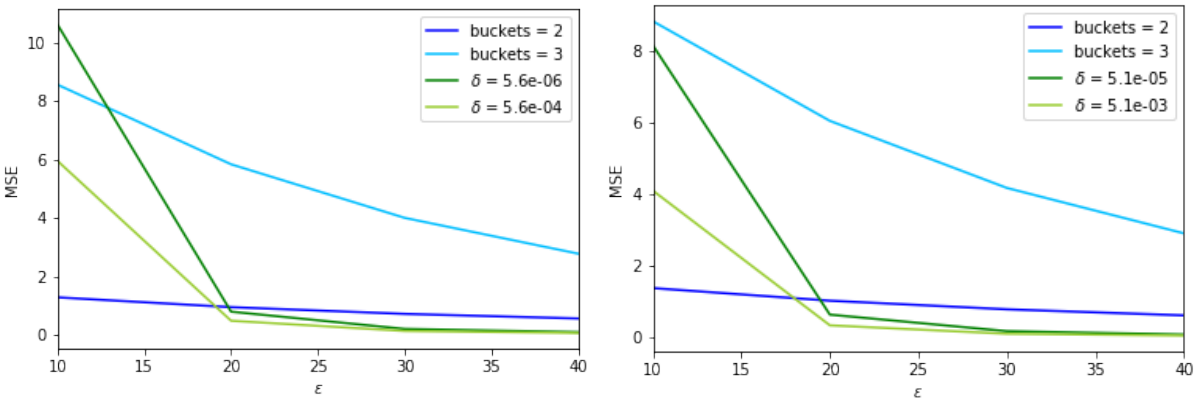


(b) K = 3

From figure 7.2 and 7.3 we see that while the `Noisy Secure MinHash` gives a high MSE with low ϵ, it decreases exponentially with increasing ϵ. This behaviour is as expected; it follows from the fact that the variance of the Laplace distribution grows by $2\left(\frac{\Delta q}{\epsilon}\right)^2$. Hence, fixing Δq and letting ϵ approach 1 makes the variance approach $2\Delta q^2$. Furthermore, it seems that the introduction of a δ allows for more accurate estimations with lower ϵ and that making δ arbitrarily low results in minuscule differences in MSE when ϵ rises.

Comparing the two algorithms, we select the two best performing configurations of the `Bucket Privmin` algorithm, as the parameters do not affect the underlying privacy guarantees. Regarding the `Noisy Secure MinHash`, selecting the best performing configurations would imply making δ so high that it hurts the privacy guarantees. Hence, we pick one configuration with very low δ and one with relatively high δ. This should allow us to make two evaluations. Firstly, how does the direct noise addition method compare to the indirect method of plausible deniability in terms of accuracy of similarity estimations. Secondly, how strong privacy guarantees can we provide while still making meaningful similarity estimations.

Figure 7.4: Comparison of accuracy of similarity estimations of Noisy Secure MinHash and Bucket PrivMin



(a) Movielens

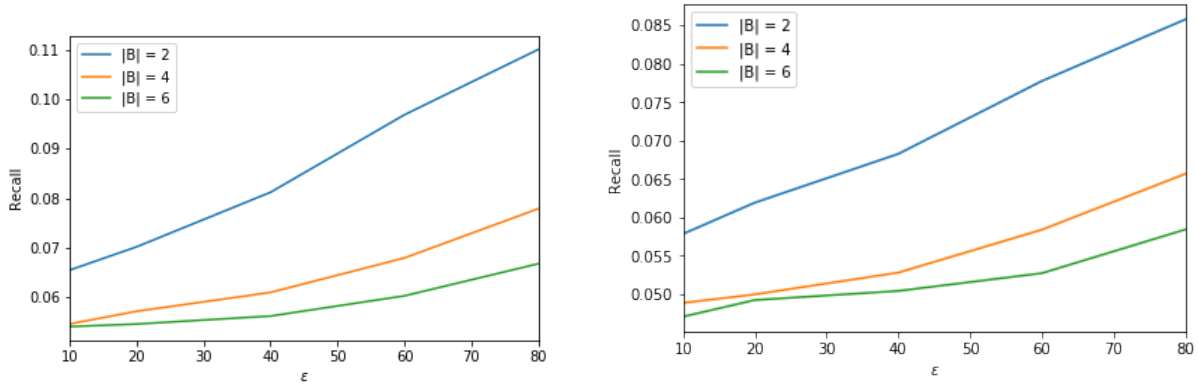
(b) Last.FM

From figure 7.4 it becomes apparent that performing accurate similarity estimations with either algorithm is infeasible with low ϵ with most configurations. However, the `Bucket PrivMin` with $B = 2$ appears to perform the best with low privacy budgets. Nonetheless, already with $B = 3$ the error rises rapidly. All configurations of `Noisy Secure MinHash`, converge with MSE between 0 and 1 with a privacy budget of ≥ 30. The results suggests that the `Noisy Secure MinHash` can achieve useful similarity estimations given high enough privacy budgets, and even surpass `Bucket PrivMin` with $B = 2$. Accuracy of similarity estimations is, however, not necessarily the most interesting metric for our algorithms. Low accuracy of estimations will matter less, if their utility is high.

7.2.2 Utility evaluations

In order to evaluate the utility of the `Bucket PrivMin` and the `Noisy Secure Minhash` algorithm in practice, we measure their recall given a scale of varying privacy budgets ϵ. As expected, the recall increases with an increasing privacy budget. Moreover, it turns out that although `Bucket PrivMin` with $B = 2$ achieves the best similarity estimations with low privacy budget, this is not translated into higher utility. This is not surprising, as the noise introduced by the random collision probability is substantial and should be reflected in the recall.

Figure 7.5: Recall with Bucket PrivMin

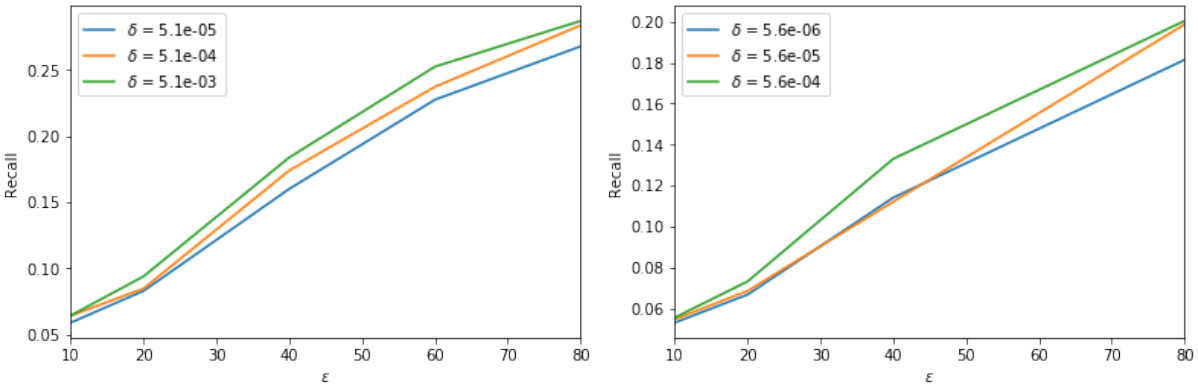


(a) Last.FM (b) MovieLens

The lower error in similarity estimations might result from the lower similarity estimation variance of `Bucket PrivMin`. Although not included in the figures, experiments with $\epsilon > 200$ confirmed that the recall for $B > 4$ does exceed the recall of $B = 2$. Nonetheless, we deem ϵ of these sizes too high to be considered. Hence, we conclude that the higher output probability trumps the reduction in random noise that we get from increasing B.

The `Noisy Secure Minhash` achieves the highest recall with a parameter $\delta = 5.6e-04$, as depicted in figure 7.6. This amounts to $\delta = \frac{1}{10 \times |X_{avg}|}$ where $|X_{avg}|$ is the average user set size in the data set, which should still be sufficiently small to avoid problematic data releases.

Figure 7.6: Recall with Noisy Secure MinHash $K = 1$

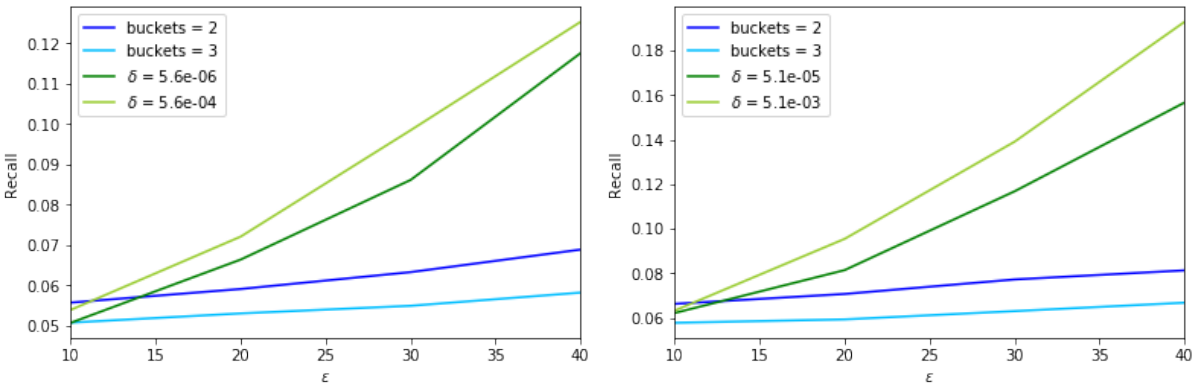


(a) Last.FM (b) MovieLens

It becomes clear that the trade-off made with increasing certainty of not publishing problematic data has minor effects on the recall. Hence, δ can be set as low as deemed necessary without significant loss of utility.

In comparing the two algorithms, we observe that the bucket parameter B in the `Bucket PrivMin` algorithm does, in fact, have a substantial influence on the utility of the algorithm. These observations are mostly in line with our theoretical analysis of the mechanisms. Allowing the existence of a δ gives our algorithm some leeway for privacy leakage which should decrease the ϵ needed for retaining utility. Meanwhile, the bucket parameter decreases the output probability too much for the benefits of high B to take effect. When inspecting figure 7.7, we can see that this is indeed the case. With half the privacy budget we can approximately get the same recall with `Noisy Secure MinHash` as we can with `Bucket Privmin`.

Figure 7.7: Comparison of utility of Noisy Secure MinHash and Bucket PrivMin



(a) Movielens

(b) Last.FM

Chapter 8

Discussion

Our investigation into the current state of similarity search providing local differential privacy guarantees revealed that there is presently no approach that solves this problem. Our analysis of `MinHash` has shown that although it can provide *some* privacy, it needs to be combined with privacy preserving mechanisms in order to avoid problematic data release. The results of our proposed `Bucket PrivMin` indicate that achieving ϵ-LDP with similarity queries requires high ϵ values to maintain the utility of the procedure. Our second proposed algorithm `Noisy Secure MinHash`, which utilizes the relaxation of LDP to the approximate setting, illustrates the benefits of performing such a relaxation in terms of utility.

Nevertheless, the results of the experiments have shown that the quality of similarity estimations of our algorithms suffer greatly by the constraints posed by our privacy ambitions. While both of our proposed approaches eliminate or greatly reduce the probability of problematic data release, they require high privacy budgets to retain utility. Although we can interpret the values of δ as the probability of problematic data releases, there is no obvious comparison for ϵ. Even though we lack strict guidelines for an acceptable amount of privacy loss ϵ [25] [36], the discussions lead us to assume that in order to give valuable privacy guarantees, the utility of both our proposed algorithms decreases rapidly. In conclusion, we find that we suffer from significant accuracy loss with both `Bucket PrivMin` and `Noisy Secure MinHash` when we aim to obtain sufficient privacy.

Apart from the design of our proposed algorithms, similarity search based on `MinHash` in the local differential private setting inherently entails some aggravating characteristics. These require us to introduce a large amount of noise, which in return discounts the quality of our similarity search. We can trace the need for high noise back to two sources: the algorithm design of `MinHash`, and the high sensitivity of the queries.

Due to the algorithm design of `MinHash`, the quality of similarity estimation is directly dependent on the number of iterations. In order to arrive at a low error rate for the Jaccard similarity estimation based on `MinHash`, we need to concatenate a number of minimum hash values to the `MinHash` signature. This leaves us no other choice than to deplete our privacy budget with every iteration of `MinHash`.

The high sensitivity is due to the setting of the similarity search in the local model of DP. Compared to the central setting, local differential privacy gives the stronger privacy guarantee, but the utility for data analysis is also harder to maintain [19][37].

Bibliography

[1] Adam, N. R., and Worthmann, J. C. Security-control methods for statistical databases: A comparative study. *ACM Comput. Surv. 21*, 4 (Dec. 1989), 515–556.

[2] Alaggan, M., Gambs, S., and Kermarrec, A.-M. Blip: Non-interactive differentially-private similarity computation on bloom filters. In *Stabilization, Safety, and Security of Distributed Systems* (Berlin, Heidelberg, 2012), A. W. Richa and C. Scheideler, Eds., Springer Berlin Heidelberg, pp. 202–216.

[3] Analytics, M. *Analytics Comes of Age*. 2018.

[4] Andoni, A., and Indyk, P. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM 51*, 1 (Jan. 2008), 117–122.

[5] Apple, D. P. T. Learning with privacy at scale. *Apple Machine Learning Journal 1:8* (2017).

[6] Aumüller, M., Bernhardsson, E., and Faithfull, A. Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. *CoRR abs/1807.05614* (2018).

[7] Balu, R., and Furon, T. Differentially private matrix factorization using sketching techniques. In *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security* (New York, NY, USA, 2016), IH&MMSec '16, ACM, pp. 57–62.

[8] Barbaro, M., and Zeller, T. A face is exposed for aol searcher no. 4417749, Aug 2006.

[9] Boutet, A., De Moor, F., Frey, D., Guerraoui, R., Kermarrec, A., and Rault, A. Collaborative filtering under a sybil attack: Similarity metrics do matter! In *48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2018, Luxembourg City, Luxembourg, June 25-28, 2018* (2018), pp. 466–477.

[10] Boutet, A., Frey, D., Guerraoui, R., Jégou, A., and Kermarrec, A.-M. Privacy-preserving distributed collaborative filtering. *Computing 98*, 8 (Aug. 2016), 827–846.

[11] Broder, A. Z., Charikar, M., Frieze, A. M., and Mitzenmacher, M. Min-wise independent permutations. *Journal of Computer and System Sciences 60* (1998), 327–336.

[12] Calandrino, J. A., Kilzer, A., Narayanan, A., Felten, E. W., and Shmatikov, V. "you might also like: " privacy risks of collaborative filtering. In *Proceedings of the 2011 IEEE Symposium on Security and Privacy* (Washington, DC, USA, 2011), SP '11, IEEE Computer Society, pp. 231–246.

[13] Chen, Z., Wang, Y., Zhang, S., Zhong, H., and Chen, L. Differentially private user-based collaborative filtering recommendation based on k-means clustering. *CoRR abs/1812.01782* (2018).

[14] Dhaliwal, J., So, G., Parker-Wood, A., and Beck, M. Utility preserving secure private data release. *CoRR abs/1901.09858* (2019).

[15] Ding, B., Kulkarni, J., and Yekhanin, S. Collecting telemetry data privately. *CoRR abs/1712.01524* (2017).

[16] Dinur, I., and Nissim, K. Revealing information while preserving privacy. In *Proceedings of the Twenty-second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems* (New York, NY, USA, 2003), PODS '03, ACM, pp. 202–210.

[17] Duchi, J. C., Jordan, M. I., and Wainwright, M. J. Local privacy and statistical minimax rates. In *Proceedings of the 2013 IEEE 54th Annual Symposium on Foundations of Computer Science* (Washington, DC, USA, 2013), FOCS '13, IEEE Computer Society, pp. 429–438.

[18] Dwork, C. Differential privacy. In *Proceedings of the 33rd International Conference on Automata, Languages and Programming - Volume Part II* (Berlin, Heidelberg, 2006), ICALP'06, Springer-Verlag, pp. 1–12.

[19] Dwork, C., and Roth, A. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci. 9*, 3-4 (Aug. 2014), 211–407.

[20] Erlingsson, Ú., Korolova, A., and Pihur, V. RAPPOR: randomized aggregatable privacy-preserving ordinal response. *CoRR abs/1407.6981* (2014).

[21] Gerber, N., Gerber, P., and Volkamer, M. Explaining the privacy paradox: A systematic review of literature investigating privacy attitude and behavior. *Computers Security 77* (2018), 226 – 261.

[22] IDC, Open Evidence for the European Commission (Directorate-General for Communications Networks, C., and Technology. European data market - final report, 2017.

[23] Indyk, P., and Motwani, R. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing* (New York, NY, USA, 1998), STOC '98, ACM, pp. 604–613.

[24] Ji, J., Li, J., Yan, S., Tian, Q., and Zhang, B. Min-max hash for jaccard similarity. In *2013 IEEE 13th International Conference on Data Mining* (Dec 2013), pp. 301–309.

[25] Lee, J., and Clifton, C. How much is enough? choosing ϵ for differential privacy. In *Information Security* (Berlin, Heidelberg, 2011), X. Lai, J. Zhou, and H. Li, Eds., Springer Berlin Heidelberg, pp. 325–340.

[26] Li, W., Zhang, Y., Sun, Y., Wang, W., Zhang, W., and Lin, X. Approximate nearest neighbor search on high dimensional data - experiments, analyses, and improvement (v1.0). *CoRR abs/1610.02455* (2016).

[27] Liu, J., Xiong, L., and Luo, J. Semantic security: Privacy definitions revisited. *Trans. Data Privacy 6*, 3 (Dec. 2013), 185–198.

[28] McSherry, F., and Mironov, I. Differentially private recommender systems: Building privacy into the netflix prize contenders. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2009), KDD '09, ACM, pp. 627–636.

[29] McSherry, F., and Talwar, K. Mechanism design via differential privacy. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science* (Washington, DC, USA, 2007), FOCS '07, IEEE Computer Society, pp. 94–103.

[30] Muralidhar, K., and Sarathy, R. Security of random data perturbation methods. *ACM Trans. Database Syst. 24*, 4 (Dec. 1999), 487–493.

[31] Narayanan, A., and Shmatikov, V. Robust de-anonymization of large sparse datasets. In *Proceedings of the 2008 IEEE Symposium on Security and Privacy* (Washington, DC, USA, 2008), SP '08, IEEE Computer Society, pp. 111–125.

[32] O'Flaherty, K. This is why people no longer trust google and facebook with their data, Oct 2018.

[33] Ozturk, A., and Polat, H. From existing trends to future trends in privacy-preserving collaborative filtering. *Wiley Int. Rev. Data Min. and Knowl. Disc. 5*, 6 (Nov. 2015), 276–291.

[34] Riazi, M. S., Chen, B., Shrivastava, A., Wallach, D. S., and Koushanfar, F. Sub-linear privacy-preserving search with untrusted server and semi-honest parties. *CoRR abs/1612.01835* (2016).

[35] Sweeney, L. Simple demographics often identify people uniquely. 2000.

[36] Tang, J., Korolova, A., Bai, X., Wang, X., and Wang, X. Privacy loss in apple's implementation of differential privacy on macos 10.12. *CoRR abs/1709.02753* (2017).

[37] Vadhan, S. The complexity of differential privacy, 2016.

[38] Wang, J., Shen, H. T., Song, J., and Ji, J. Hashing for similarity search: A survey. *CoRR abs/1408.2927* (2014).

[39] Wang, T., Blocki, J., Li, N., and Jha, S. Optimizing locally differentially private protocols. *CoRR abs/1705.04421* (2017).

[40] Wang, Y., Wu, X., and Hu, D. Using randomized response for differential privacy preserving data collection. In *EDBT/ICDT Workshops* (2016).

[41] Warner, S. L. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association 60*, 309 (1965), 63–69. PMID: 12261830.

[42] Wilson, R. L., and Rosen, P. A. Protecting data through perturbation techniques: The impact on knowledge discovery in databases. *J. Database Manag. 14* (2003), 14–26.

[43] Wong, K.-S., and Kim, M. H. Preserving differential privacy for similarity measurement in smart environments. *The Scientific World Journal 2014* (07 2014), 1–9.

[44] Yan, Z., Wu, Q., Ren, M., Liu, J., Liu, S., and Qiu, S. Locally private jaccard similarity estimation. *Concurrency and Computation: Practice and Experience* (09 2018).

[45] Zarsky, T. Incompatible: The gdpr in the age of big data (2017).

[46] Zhu, X., and Sun, Y. Differential privacy for collaborative filtering recommender algorithm. In *Proceedings of the 2016 ACM on International Workshop on Security And Privacy Analytics* (New York, NY, USA, 2016), IWSPA '16, ACM, pp. 9–16.

Appendix A

Estimating Similarity from Noisy Signatures

Theorem A.1 *Noisy Similarity Estimation*

Given two users X and Y, their hash signatures $h(X)$ and $h(Y)$, where $|h(X)|, |h(Y)| \in \{0,1\}^L$ and their noisy hash signatures $h'(X)$,$h'(Y) \in \mathbb{R}^L$ where noise is added from $Lap(b) = Lap\left(\frac{\Delta q}{\epsilon}\right)$. The Jaccard similarity between X and Y can be estimated by:

$$Jacc(X,Y) = 2 \times \frac{L - \left(\sum_{i=1}^{l}(h'_i(X) - h'_i(Y))^2] - 2 \times L \times Var[Lap(b)]\right)}{L} - 1 \tag{A.1}$$

Proof We begin by considering the expected euclidean distance between $h'(X)$ and $h'(Y)$:

$$\mathbb{E}\Big[\sum_{i=1}^{L}\big(h'_i(X) - h'_i(Y)\big)^2\Big] = L \times \mathbb{E}\Big[\big(h'_i(X) - h'_i(Y)\big)^2\Big] \tag{A.2}$$

$$= L \times \mathbb{E}\left[\Big((h'_i(X) - h'_i(Y)) + (\Delta_1 - \Delta_2)\Big)^2\right] \tag{A.3}$$

$$= L \times \mathbb{E}\Big[\big(h'_i(X) - h'_i(Y)\big)^2 + (\Delta_1 - \Delta_2)^2 + 2 \times (h'_i(X) - h'_i(Y)) \times (\Delta_1 - \Delta_2)\Big] \tag{A.4}$$

$$= L \times \mathbb{E}[\big(h'_i(X) - h'_i(Y)\big)^2] + \mathbb{E}[(\Delta_1 - \Delta_2)^2] + \mathbb{E}[2 \times (h'_i(X) - h'_i(Y)) \times (\Delta_1 - \Delta_2)] \tag{A.5}$$

Here, Δ signifies the noise added to the signatures. Since we know that $\mathbb{E}[(\Delta_1 - \Delta_2)] = 0$, the last expectation in eq. (A.5) is equal to 0 and since $\mathbb{E}[(h'_i(X) - h'_i(Y))^2]$ is the probability of the two signatures differing in position i:

$$\mathbb{E}\Big[\sum_{i=1}^{L}(h'_i(X) - h'_i(Y))^2\Big] = L \times \left(\frac{1 - Jacc(X,Y)^K}{2} + \mathbb{E}[(\Delta_1 - \Delta_2)^2]\right)$$
$$= L \times \left(\frac{1 - Jacc'(X,Y)^K}{2} + \mathbb{E}[\Delta_1^2] + \mathbb{E}[\Delta_2^2] - 2 \times \mathbb{E}[\Delta_1\Delta_2]\right)$$

Since we once again know that $\mathbb{E}[\Delta_1\Delta_2] = 0$ the last expectation disappears and because $Var[Lap(b)] = \mathbb{E}[Lap(b)^2] - \mathbb{E}[Lap(b)]^2 = \mathbb{E}[Lap(b)^2] - 0^2 = \mathbb{E}[\Delta^2]$:

$$\mathbb{E}\Big[\sum_{i=1}^{L}(h'(X_i) - h'(Y_i))^2\Big] = L \times \left(\frac{1 - Jacc(X,Y)^K}{2} + 2 \times Var[Lap(b)]\right) \tag{A.6}$$

Given two noisy signatures $h'(X)$ and $h'(Y)$ we can then estimate the expected number of differing positions P in two original signatures $h(X)$ and $h(Y)$ as being:

$$\mathbb{E}[P] = L \times \left(\frac{1 - Jacc(X,Y)^K}{2}\right) = \sum_{i=1}^{L}\Big(h'_i(X) - h'_i(Y)\Big)^2 - 2 \times L \times Var[Lap(b)] \tag{A.7}$$

And so, since we know that the expected number of collisions is $\mathbb{E}[C] = L - \mathbb{E}[P]$ and that we can estimate the Jaccard similarity by:

$$Jacc(X,Y)^K = 2 \times \frac{\mathbb{E}[C]}{L} - 1 \tag{A.8}$$

we will be able to estimate the Jaccard similarity of two users, given their noisy signature, by:

$$Jacc(X,Y)^K = 2 \times \frac{L - \Big(\sum_{i=1}^{L}\Big(h'_i(X) - h'_i(Y)\Big)^2 - 2 \times L \times Var[Lap(b)]\Big)}{L} - 1 \tag{A.9}$$