Local Dampening: Differential Privacy for Non-numeric Queries via Local Sensitivity

Victor A. E. Farias*
Computer Science Department
Universidade Federal do Ceará
Fortaleza, Ceará, Brazil
victor.farias@lsbd.ufc.br

Felipe T. Brito*
Computer Science Department
Universidade Federal do Ceará
Fortaleza, Ceará, Brazil
felipe.timbo@lsbd.ufc.br

Cheryl Flynn
AT&T Labs Research
Bedminster, NJ, USA
cflynn@research.att.com

Javam C. Machado*
Computer Science Department
Universidade Federal do Ceará
Fortaleza, Ceará, Brazil
javam.machado@lsbd.ufc.br

Subhabrata Majumdar
AT&T Labs Research
New York City, NY, USA
subho@research.att.com

Divesh Srivastava
AT&T Labs Research
Bedminster, NJ, USA
divesh@research.att.com

ABSTRACT

Differential privacy is the state-of-the-art formal definition for data release under strong privacy guarantees. A variety of mechanisms have been proposed in the literature for releasing the noisy output of numeric queries (e.g., using the Laplace mechanism), based on the notions of global sensitivity and local sensitivity. However, although there has been some work on generic mechanisms for releasing the output of non-numeric queries using global sensitivity (e.g., the Exponential mechanism), the literature lacks generic mechanisms for releasing the output of non-numeric queries using local sensitivity to reduce the noise in the query output.

In this work, we remedy this shortcoming and present the *local dampening mechanism*. We adapt the notion of local sensitivity for the non-numeric setting and leverage it to design a generic non-numeric mechanism. We illustrate the effectiveness of the local dampening mechanism by applying it to two diverse problems: (i) Influential node analysis. Given an influence metric, we release the top-k most influential nodes while preserving the privacy of the relationship between nodes in the network; (ii) Decision tree induction. We provide a private adaptation to the ID3 algorithm to build decision trees from a given tabular dataset. Experimental results show that we could reduce the use of privacy budget by 3 to 4 orders of magnitude for Influential node analysis and increase accuracy up to 12% for Decision tree induction when compared to global sensitivity based approaches.

PVLDB Reference Format:
Victor A. E. Farias, Felipe T. Brito, Cheryl Flynn, Javam C. Machado, Subhabrata Majumdar, and Divesh Srivastava. Local Dampening: Differential Privacy for Non-numeric Queries via Local Sensitivity. PVLDB, 14(4): 521 - 533, 2021.
doi:10.14778/3436905.3436912

1 INTRODUCTION

Differential privacy [6, 8] is the state-of-the-art formal definition for data release under strong privacy guarantees. It imposes near-indistinguishability on the released information whether an individual belongs to a sensitive database or not. The key intuition is that the output distribution of a differentially private query should not change significantly based on the presence or absence of an individual. It provides statistical guarantees against the inference of private information through the use of auxiliary information. Algorithms can achieve differential privacy by employing output perturbation, which releases the true output of a given non-private query f with noise added. The magnitude of the noise should be large enough to cover the identity of the individuals in the input database x.

For a numeric query (i.e., query with numeric output) f, the *Laplace mechanism* [8] is a well-known output perturbing private method. It adds numeric noise to the output of f and calibrates the noise based only on f and not on x. The noise magnitude is proportional to the *global sensitivity*, a concept that measures the worst case impact on f's output of the addition or removal of an individual over the set of possible input databases. This may result in an unreasonably high amount of noise when x is far from the database with the worst case impact, which is the case for many realistic databases. To remedy this, Nissim et al. [35] proposed to add instance-based noise calibrated as a function of x. They introduced the notion of *local sensitivity*. This quantifies the impact of addition or removal of an individual for the database instance x, resulting in a lower bound to the global sensitivity. Many works use this notion to shrink the amount of noise added to release more useful statistical information [2, 22, 23, 28, 42].

For the class of non-numeric queries f, i.e. f has a non-numeric range $\mathcal{R}$, the *exponential mechanism* [33] ensures differential privacy by sampling elements from $\mathcal{R}$ using the exponential distribution. This requires a utility function $u(x, r)$ that takes as input a database x and an element $r \in \mathcal{R}$ and outputs a numeric score that measures the utility of r. The larger $u(x, r)$, the higher the probability of the exponential mechanism outputting r. The exponential mechanism uses a similar notion of global sensitivity to that found in [8] where

[*] Work was done while the author was a visiting scholar at AT&T Labs - Research, USA.

This work is licensed under the Creative Commons BY-NC-ND 4.0 International License. Visit https://creativecommons.org/licenses/by-nc-nd/4.0/ to view a copy of this license. For any use beyond those covered by this license, obtain permission by emailing info@vldb.org. Copyright is held by the owner/author(s). Publication rights licensed to the VLDB Endowment.

Proceedings of the VLDB Endowment, Vol. 14, No. 4 ISSN 2150-8097.
doi:10.14778/3436905.3436912

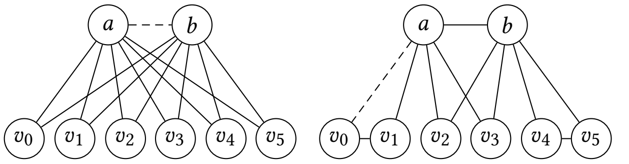
it measures the worst case impact on the utility $u(x, r)$ for all elements $r \in \mathcal{R}$ by adding or removing an individual from all databases. However, to the best of our knowledge, the literature lacks generic mechanisms that apply local sensitivity to the non-numeric setting.

Example 1.1. Consider an application where, given a graph G, the query should report the node with the largest *Egocentric Betweenness Centrality* (EBC) [9, 15, 32]. The EBC metric measures the degree to which nodes stand between each other, defined as

$$EBC(c) = \sum_{u,v \in N_c | u \neq v} \frac{p_{uv}(c)}{q_{uv}(c)},$$

where $q_{uv}(c)$ is the number of geodesic paths connecting $u \neq c$ and $v \neq c$ on the subgraph composed by c and its neighbors N_c, and $p_{uv}(c)$ is the number of those paths that include c.

For instance, let G be the graph illustrated in Figure 1a. Node a has EBC score equal to 7.5 since there are $\binom{6}{2} = 15$ pairs of neighbors of the form $\{v_i, v_j\}$, for $0 \leq i < j \leq 5$, that each contributes with 0.5 as they have two geodesic paths of length 2 from v_i to v_j, where only one contains a. Pairs of the form $\{b, v_i\}$, for $0 \leq i \leq 5$ do not contribute to the score of a since there is only one geodesic path (length 1) from b to v_i and it does not contain a. Verify that for the graph illustrated in Figure 1b, node a has EBC score equal to 6.5.



(a) EBC: Worst Case

(b) EBC: Usual case

Figure 1: Sensitivity of EBC

We use edge differential privacy where the sensitivity is measured by adding or removing one edge from the input graph. The global sensitivity for EBC is obtained from a graph of the form displayed in Figure 1a. Removing the edge (a, b) is the worst case. The new EBC score of a is 15, 1 point for each one of the 15 pairs $\{v_i, v_j\}$, $0 \leq i \leq 5$, as now there is only one geodesic path from v_i to v_j which includes a (path $< v_i, a, v_j >$). The paths of the form $< v_i, b, v_j >$ are not counted since b no longer belongs to N_a.

This gadget is formed by two nodes with high degree that share all neighbors and those neighbors do not have an edge to each other. This gadget is very unlikely to be found in real graphs. Figure 1b illustrates a more typical example. In this instance, the worst local measurement of the sensitivity is given by the removal of the edge (a, v_0) that shrinks the EBC score of a by only 3 (1 for each pair $\{v_0, b\}$, $\{v_0, v_2\}$ and $\{v_0, v_3\}$ since v_0 is no longer a neighbor of a).

Additionally, the global sensitivity of *EBC* grows quadratically with respect to the maximum degree as we discuss in Section 5. Hence, a non-numeric mechanism applying local sensitivity could add less noise to the output compared to a global sensitivity based approach like the exponential mechanism.

In this paper, we propose the *local dampening mechanism*, which adapts the notion of local sensitivity to the non-numeric setting and uses it to dampen the utility function u in order to improve accuracy. Local dampening also employs the exponential distribution, similar to the exponential mechanism [33]. Applications in which local sensitivity is significantly smaller than global sensitivity can benefit from our approach. For the scenario where local sensitivity is near the global sensitivity, the local dampening mechanism reverts to the exponential mechanism, so that the maximum addition of noise across all values of r is bounded by the exponential mechanism.

To this end, we present a new version of the local sensitivity, called *element local sensitivity*. Traditional local sensitivity measures the largest impact of the addition or deletion of an individual to the input database over all outputs $r \in \mathcal{R}$. Element local sensitivity computes this impact, but only for some given element $r \in \mathcal{R}$ (Example 1.2). This allows us to explore local measurements of the sensitivity of f even if traditional local sensitivity is near the global sensitivity, but, for most elements in $\mathcal{R}$, the element local sensitivity is low.

Example 1.2. Consider the graph in Figure 1a. The removal of edge (a, b) sets the traditional local sensitivity to 7.5 which is also the case for global sensitivity. But measurements of sensitivity per node (element) are much smaller. For instance, the sensitivity for a node v_i $(0 \leq i \leq 5)$ is 1 which is set by the removal of edge (a, b) where $EBC(v_i)$ increases from 0 to 1 (path $< a, v_i, b >$). We explore this to improve the accuracy further.

We illustrate the effectiveness of the local dampening mechanism by applying it to two problems: (i) Influential node analysis which searches for central/influential nodes in a graph database. Given a centrality/influence metric, we release the label of the top-k most central nodes while preserving the privacy of the relationships between nodes in the graph; (ii) We also provide an application on tabular data which is a private adaptation of the ID3 algorithm to build a decision tree from a given tabular dataset based on the information gain for each attribute.

The contributions of this work are summarized as follows:

- We adapt the local sensitivity definition to the non-numeric setting and introduce a new definition of local sensitivity that measures sensitivity per element (Section 3).
- We introduce the local dampening mechanism, a novel differentially private mechanism to answer non-numeric queries, that applies local sensitivity to attenuate the utility function to increase the signal-to-sensitivity ratio in order to reduce noise (Section 3).
- We explore the conditions under which the new notion of local sensitivity per element is useful and show how to tweak the local dampening mechanism to improve accuracy further (Section 4).
- We apply the local dampening mechanism to construct differentially private algorithms for a graph problem called influential node analysis. As the influence metric, we use the egocentric betweenness centrality and show how to compute local sensitivity for it (Section 5.1). Experimental results show that our approach could be as accurate as global sensitivity based mechanisms using 3 to 4 orders of magnitude less privacy budget (Section 6.1).
- As an example of a data-mining problem, we address the application of building private algorithms for decision tree

induction. We present a differentially private adaptation of the entropy based ID3 algorithm using the local dampening mechanism, and provide a way to compute local sensitivity efficiently (Section 5.2). We are able to improve accuracy up to 12% in relation to previous works (Section 6.2).

Section 2 presents the relevant formal definitions for differential privacy. Section 7 surveys related work, and Section 8 concludes the paper. We defer the proofs of the lemmas and theorems to our technical report [10].

2 BACKGROUND

In this section, we describe some background concepts that we build on in subsequent sections.

2.1 Differential Privacy

Let x be a sensitive database and f a function to be evaluated on x. The database is represented as a vector $x \in \mathcal{D}^n$ where each entry represents an individual tuple. The output $f(x)$ must be released without leaking much information about the individuals. For that, we need to design a randomized algorithm $\mathcal{A}(x)$ that adds noise to $f(x)$ such that it satisfies the definition of differential privacy stated below.

Definition 2.1. (ϵ-Differential Privacy [7, 8]). A randomized algorithm $\mathcal{A}$ satisfies ϵ-differential privacy, if for any two databases x and y satisfying $d(x,y) \leq 1$ and for any possible output O of $\mathcal{A}$, we have

$$Pr[\mathcal{A}(x) = O] \leq \exp(\epsilon)\ Pr[\mathcal{A}(y) = O].$$

where $Pr[\cdot]$ denotes the probability of an event and d denotes the hamming distance between the two databases, i.e., the number of tuples of individuals that changed value, $d(x,y) = |\{i \mid x_i \neq y_i\}|$. In the following, we refer to $d(x,y)$ as the distance between two given databases x and y.

The Laplace mechanism [8] and the exponential mechanism [33] are the most well-known output perturbation methods that satisfy the definition of differential privacy. For the case where f outputs a vector of numeric values, the Laplace mechanism is able to transform f in a differentially private algorithm by adding random noise sampled from the laplace distribution to each entry of $f(x)$. When the output of f is non-numeric, the exponential mechanism applies. In the exponential mechanism setting, the data querier needs to provide a utility function $u : \mathcal{D}^n \times \mathcal{R} \rightarrow \mathbb{R}$ that takes a database x and an output $r \in \mathcal{R}$ where $\mathcal{R}$ is the set of all outputs of f and produces a score $u(x,r)$. A higher score indicates that an element is more useful to the application. The exponential mechanism privately answers f by sampling an element $r \in \mathcal{R}$ with probability proportional to its utility score $u(x,r)$.

The global sensitivity of u is defined as the maximum possible difference of utility scores at all possible pairs of database entries x, y and all possible elements r:

Definition 2.2. (Global Sensitivity Δu [33]).

$$\Delta u = \max_{r \in \mathcal{R}} \max_{x,y \mid d(x,y) \leq 1} |u(x,r) - u(y,r)|.$$

Thus, the exponential mechanism is stated as follows:

Definition 2.3. (Exponential Mechanism [33]). The exponential mechanism $\mathcal{E}(x, \epsilon, u, \mathcal{R})$ selects and outputs an element $r \in \mathcal{R}$ with probability proportional to $\exp\left(\frac{\epsilon\, u(x,r)}{2\Delta u}\right)$.

McSherry and Talwar [33] showed that the exponential mechanism satisfies ϵ-differential privacy.

2.2 Local Sensitivity

For the exponential mechanism, the larger the global sensitivity Δu is, the more it approximates a uniform random sampling. To remedy the resulting accuracy loss, we seek mechanisms with low sensitivity. The concept of local sensitivity $LS(x)$ [35] captures the sensitivity locally on the input database x instead of searching for the sensitivity in the universe of databases $\mathcal{D}^n$. Local sensitivity tends to be smaller than global sensitivity for many problems [2, 22, 23, 28, 35, 42]. We present an adapted version of the definition of local sensitivity for the non-numeric setting, as given in [35].

Definition 2.4. (Local sensitivity [35]). Given a utility function $u(x,r)$ that takes as input a database x and an element r and outputs a numeric score for x, the local sensitivity of u is defined as

$$LS^u(x) = \max_{r \in \mathcal{R}} \max_{y \mid d(x,y) \leq 1} |u(x,r) - u(y,r)|.$$

Observe that the global sensitivity is the maximum local sensitivity over the set of all databases, $\Delta u = \max_x LS^u(x)$.

Replacing Δu for $LS(x)$ in the exponential mechanism would add less noise resulting in higher accuracy. However, it does not satisfy ϵ-differential privacy (Definition 2.1). In the differential privacy setting, the noise needs to be independent of the input database, but $LS(x)$ itself contains information about the database.

To use local sensitivity on a differentially private mechanisms for queries with numeric output, Nissim et al. [35] proposed smooth sensitivity framework that injects noise sampled from a Cauchy distribution. A part of their solution is the local sensitivity at distance t, which we adapt to the non-numeric setting.

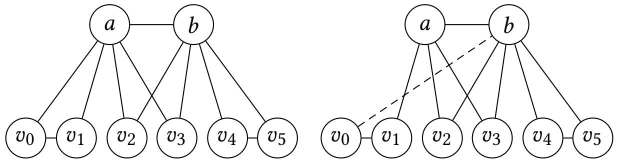
Definition 2.5. (Local sensitivity at distance t [35]). Given a utility function $u(x,r)$ that takes as input a database x and an element r and outputs a numeric score for x. The local sensitivity at distance t of u is defined as

$$LS^u(x,t) = \max_{y \mid d(x,y) \leq t} LS^u(y).$$

Local sensitivity at distance t, $LS^u(x,t)$, measures the maximum local sensitivity $LS^u(y)$ over all databases y at maximum distance t, i.e., we allow t modifications on the database before computing its local sensitivity.

Example 2.6. Consider the graph G of Figure 2a. The local sensitivity at distance t allows t extra modifications before measuring local sensitivity. As discussed in Example 1.1, the local sensitivity of G is 3 (at distance 0): $LS^{EBC}(G,0) = 3$.

Now to compute local sensitivity at distance 1, we need to find which one edge to add or remove in order to compute the maximum local sensitivity at distance 1. This case is found by removing edge (a, v_0) as shown in Figure 2b obtaining G'. Then the local sensitivity of G' is 5 where node b increases by 5 units when adding edge (b, v_0) (1 for each pair $\{v_0, v_2\}$, $\{v_0, v_3\}$, $\{v_0, v_4\}$, $\{v_0, v_5\}$ and $\{v_0, a\}$). This means that $LS^{EBC}(G,1) = 5$.



(a) Original Graph G **(b)** G' **at distance 1 from** G

Figure 2: Local sensitivity at distance t

3 LOCAL DAMPENING MECHANISM

This section presents the local dampening mechanism for answering queries with non-numeric output under differential privacy. Our approach uses the same setup of the exponential mechanism: given a query Q with range R over a database x and a utility function $u : \mathcal{D}^n \times \mathcal{R} \to \mathbb{R}$, we would like to output the element $r \in \mathcal{R}$ with highest utility score $u(x, r)$. The utility function u is application-specific and is built by the querier to give higher score to the elements in $\mathcal{R}$ that are more useful to the application.

Our approach requires the computation of any of the sensitivity notions described in the Section 2.2. Additionally, we introduce a new notion of sensitivity called element local sensitivity. It measures the worst impact on the sensitivity for a given element $r \in \mathcal{R}$ when adding or removing an individual from the input database x, i.e., the largest difference $|u(x, r) - u(y, r)|$ for all neighbors y of x.

The local dampening mechanism uses an *admissible function* to dampen the utility function u and construct its dampened version, referred to D_{u,δ^u}. Specifically, we attenuate u such that the signal-to-sensitivity ratio (i.e. u/sensitivity) is larger which results in higher accuracy. An admissible function is function that computes an upper bound on the sensitivity. This concept is specially useful when the sensitivity is not possible or efficient but computing a upper bound is simpler.

We lay the groundwork of our analysis with the definition of element local sensitivity in Section 3.1. We then define local dampening in Section 3.2, and provide a privacy guarantee for our mechanism in Section 3.3.

3.1 Element Local Sensitivity

The local sensitivity $LS^u(x, t)$ quantifies the maximum sensitivity of u over all elements $r \in \mathcal{R}$ for an input database x with t modifications (Definition 2.5). That gives a high-level description of the variation of u in neighboring databases. However, if just one element in $\mathcal{R}$ has a high value of sensitivity (close to Δu), $LS^u(x, t)$ will be large too. That is ineffective in a scenario where most of the elements have low sensitivity and just few have high sensitivity, which makes $LS^u(x, t)$ large and consequently hurts accuracy.

We introduce a more specialized definition of local sensitivity, denoted as $LS^u(x, t, r)$, which measures the sensitivity of u for given $r \in \mathcal{R}$ for an input database x at distance t (definition 3.1). This allows us to grasp the sensitivity of u for a single element.

Definition 3.1. (Element Local Sensitivity at distance t).

$$LS^u(x, t, r) = \max_{y|d(x,y) \leq t, z|d(y,z) \leq 1} |u(y, r) - u(z, r)|.$$

We can obtain $LS^u(x, t)$ from this definition: $LS^u(x, t) = \max_{r \in \mathcal{R}} LS^u(y, t, r)$ as $LS^u(x, t, r) = \max_{y|d(x,y) \leq t} LS^u(y, 0, r)$. Intuitively, to compute element local sensitivity, one needs to identify which addition or removal of an individual on the input database x causes the most impact on the utility score of a given element r, i.e., the largest difference $|u(x, r) - u(y, r)|$ for all neighbors y of x.

Example 3.2. We illustrate this definition with the same setup as example 2. Let G be the graph from Figure 2a. Suppose we want to compute the element local sensitivity for v_4, $LS^u(G, 0, v_4)$. We measure only the worst impact of the addition or removal of an edge on the value of the EBC score for v_4. This is obtained by adding the edge (v_0, v_4) (Figure 3). The EBC score increases by 2 (1 for each pair $\{b, v_0\}$ and $\{v_0, v_5\}$). Thus $LS^u(G, 0, v_4) = 2$.

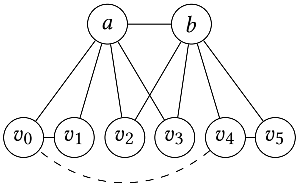


Figure 3: Element Local Sensitivity for v_4

Computing $LS^u(x, t, r)$ is not always feasible, as it can be NP-hard [35, 42]. To navigate this problem, we can relax the need for the computation of $LS^u(x, t, r)$ and build a computationally efficient function that computes an upper bound for $LS^u(x, t, r)$ that is still smaller than Δu. This function needs to have some properties to be admissible in the local dampening mechanism to guarantee differential privacy:

Definition 3.3. (Admissible function). A function $\delta^u(x, t, r)$ is *admissible* if:

(1) $\delta^u(x, 0, r) \geq LS^u(x, 0, r)$, for all $x \in \mathcal{D}^n$ and all $r \in \mathcal{R}$.
(2) $\delta^u(x, t + 1, r) \geq \delta^u(y, t, r)$, for all x, y such that $d(x, y) \leq 1$ and all $t \geq 0$.

The global sensitivity Δu is admissable since it is a constant value would trivially satisfy Definition 3.3. We also show that the function $LS^u(x, t, r)$ itself is admissible (Lemma 3.4) and, in addition, it is the minimum function in the set of all admissible functions.

Lemma 3.4. *$LS^u(x, t, r)$ is minimum admissible, i.e. $LS^u(x, t, r) \leq \delta^u(x, t, r)$ for all admissible functions δ^u.*

The accuracy of the local dampening mechanism depends on the true element local sensitivity, with a lower value translating to higher accuracy. If the element local sensitivity is bounded above by an admissible function instead, the accuracy is also proportional to how closely the admissible function approximates element local sensitivity. In Section 4.2, we discuss that Δu, $LS^u(x, t)$ and $LS^u(x, t, r)$ are also all admissible functions.

Some admissible functions, such as $LS^u(x, t)$ and $LS^u(x, t, r)$, converge to Δu by design as t grows. This is desirable since it bounds the worst-case of local dampening by the exponential mechanism in terms of accuracy. One can force this by replacing $\delta^u(x, t, r)$ by its bounded version $\min(\delta^u(x, t, r), \Delta u)$. We now show that

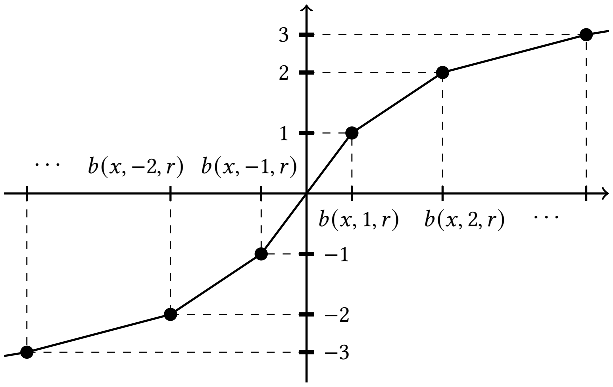


Figure 4: Dampening function D_{u,δ^u}

$\min(\delta^u(x,t,r),\Delta u)$ is admissible, and converges to Δu within n steps.

Lemma 3.5. *If $\delta^u(x,t,r)$ is admissable, then $\min(\delta^u(x,t,r),\Delta u)$ is admissible and is equal to Δu for $t > n$.*

For the rest of this paper, we shall assume that $\delta^u(x,t,r)$ converges to Δu or that we replace it by its bounded version.

3.2 Dampening Function

We now define the dampening function $D_{u,\delta^u}(x,r)$, which uses an admissible function $\delta^u(x,t,r)$ to return a dampened and scaled version of the original utility function.

Definition 3.6. (Dampening function). Given a utility function $u(x,r)$ and an admissible function $\delta^u(x,t,r)$, the dampening function $D_{u,\delta^u}(x,r)$ is defined as a piecewise linear interpolation over the points:

$$< \ldots, (b(x,-1,r),-1), (b(x,0,r),0), (b(x,1,r),1), \ldots >$$

where $b(x,i,r)$ is given by:

$$b(x,i,r) := \begin{cases} \sum_{j=0}^{i-1} \delta(x,j,r) & \text{if } i > 0, \\ 0 & \text{if } i = 0, \\ -b(x,-i,r) & \text{otherwise.} \end{cases}$$

Therefore,

$$D_{u,\delta^u}(x,r) = \frac{u(x,r) - b(x,i,r)}{b(x,i+1,r) - b(x,i,r)} + i.$$

where i is defined as the smallest integer such that $u(x,r) \in [b(x,i,r), b(x,i+1,r))$.

Figure 4 visualizes the general scheme of D_{u,δ^u}. A crucial property of D_{u,δ^u} is that it scales u so that the sensitivity of D_{u,δ^u} is bounded to 1 .

Lemma 3.7. *$|D_{u,\delta^u}(x,r) - D_{u,\delta^u}(y,r)| \le 1$ for all x, y such that $d(x,y) \le 1$ and all $r \in R$ if δ^u is admissible.*

Thus, we state the local dampening mechanism as follows:

Definition 3.8. (Local dampening mechanism). The local dampening mechanism $\mathcal{M}(x,\epsilon,u,\delta^u,\mathcal{R})$ selects and outputs an element $r \in \mathcal{R}$ with probability proportional to $\exp\left(\frac{\epsilon\, D_{u,\delta^u}(x,r)}{2}\right)$.

The accuracy of $\mathcal{M}$ depends on how well $\delta^u(x,t,r)$ is able to approximate $LS^u(x,t,r)$. A special case of local dampening is when $\delta^u(x,t,r) = \Delta u$. This is the worst-case scenario, where D_{u,δ^u} loses its attenuation power and reduces to a linear function with intercept 0 and slope $1/\Delta u$, i.e. $D_{u,\delta^u}(x,r) = u(x,r)/\Delta u$. Consequently, the local dampening mechanism selects and outputs an element $r \in \mathcal{R}$ with probability $\propto \exp((\epsilon u(x,r))/\Delta u)$, matching the definition of exponential mechanism. Thus, in the worst-case scenario local dampening reduces to the exponential mechanism. Example 3.9 illustrates a comparison of the local dampening against the exponential mechanism.

Example 3.9. This example explores the local dampening mechanism using the local sensitivity definition while the element local sensitivity is addressed in Section 4. Let G be the graph of Figure 2a. As we have discussed in Example 2.6, we have that $LS^{EBC}(G,0) = 3$ and $LS^{EBC}(G,1) = 5$. The EBC scores for the vertices are $EBC(a) = EBC(b) = 6.5$ and $EBC(v_i) = 0$, for $0 \le i \le 5$. Their dampened EBC scores are:

$$D_{EBC,LS^{EBC}}(G,a) = D_{EBC,LS^{EBC}}(G,b) = 1.7,$$
$$D_{EBC,LS^{EBC}}(G,v_i) = 0, \text{ for } 0 \le i \le 5.$$

For instance, assuming $\epsilon = 2.0$, the probability for each node to be selected is:

$$Pr[\text{a is selected}] = Pr[\text{b is selected}] \propto \exp(1.7) = 5.47,$$
$$Pr[v_i \text{ is selected}] \propto \exp(0) = 1.0, \text{ for } 0 \le i \le 5.$$

Normalizing, we have that $Pr[\text{a is selected}] = Pr[\text{b is selected}] = 0.32$ and $Pr[v_i \text{ is selected}] = 0.06$. The exponential mechanism obtained that $Pr[\text{a is selected}] = Pr[\text{b is selected}] = 0.22$ and $Pr[v_i \text{ is selected}] = 0.09$. Thus local dampening yields a higher probability of choosing the node with highest score.

3.3 Privacy Guarantee

We now prove that the local dampening mechanism $\mathcal{M}$ ensures ϵ-differential privacy (Theorem 3.10). The privacy correctness proof follows from the exponential mechanism correctness [33] and Lemma 3.7.

Theorem 3.10. *$\mathcal{M}$ satisfies ϵ-Differential Privacy if δ is admissible.*

4 TWEAKING DAMPENING MECHANISM

We dampen the utility scores of elements in $\mathcal{R}$ with different scales, i.e. $\delta^u(x,t,r)$ can be different to $\delta^u(x,t,r')$ for some $r,r' \in \mathcal{R}$. This implies that the relative order of the dampened utilities may differ from the original utility score. We call this the *inversion problem* (see Example 4.1).

Example 4.1. Consider the following setup: $\mathcal{R} = \{r_1,r_2\}$, $\delta^u(x,0,r_1) = 1$, $\delta^u(x,1,r_1) = 2$, $\delta^u(x,0,r_2) = 4$, $u(x,r_1) = 3$ and $u(x,r_2) = 4$. When applying D_{u,δ^u} to r_1 and r_2, we obtain $D_{u,\delta^u}(x,r_1) = 2$ and $D_{u,\delta^u}(x,r_2) = 1$. Originally, r_2 is more useful than r_1 but after dampening it inverts. This hurts accuracy since the local dampening mechanism will choose r_1 with higher probability.

This situation does not occur when $\delta^u(x,t,r)$ is independent of r, e.g., $\delta^u(x,t,r) = LS^u(x,t)$, where the utilities of all $r \in \mathcal{R}$ are dampened with the same scale. We propose two ways to address this problem. The first one, named shifting tweak, tackles it for the class of queries where $u(x, r)$ is correlated with $\delta^u(x, t, r)$ over r. Moreover, this tweak also takes advantage of the correlation to increase the range of the dampened utility scores to improve accuracy. The second idea is suitable for any query. It replaces $\delta^u(x, t, r)$ for $\hat{\delta}^u(x, t, r)$ where $\hat{\delta}^u_{\max}$ is not dependent on r.

4.1 Shifting tweak

We can improve accuracy further when the utility function $u(x, r)$ is correlated with element local sensitivity $LS^u(x, t, r)$ (or $\delta^u(x, t, r)$) over r. Here, shifting the utility function helps. Example 4.2 shows how shifting can increase the probability of high utility elements to be chosen (i.e. improves accuracy). We address the case of positive correlation below. The negative case can be tackled similarly.

Example 4.2. Consider the graph G from figure 2a. For nodes a and b, their measured element local sensitivities are: $LS^{EBC}(G, 0, a) = LS^{EBC}(G, 0, b) = 3$ and $LS^{EBC}(G, 1, a) = LS^{EBC}(G, 1, b) = 5$. For a node v_i, for $0 \leq i \leq 5$, its measured sensitivity is $LS^{EBC}(G, 0, v_i) = 2$. We observe a positive correlation EBC and LS^{EBC}, since the EBC scores are $EBC(a) = EBC(b) = 6.5$ and $EBC(v_i) = 0$, for $0 \leq i \leq 5$.

Shifting the EBC scores by -7, we get that $EBC'(a) = EBC'(b) = -0.5$ and $EBC'(v_i) = -7$, for $0 \leq i \leq 5$. Then we compute their dampened EBC' scores:

$$D_{EBC',LS^{EBC}}(G, a) = D_{EBC',LS^{EBC}}(G, b) = 0.1,$$
$$D_{EBC',LS^{EBC}}(G, v_i) = -2, \text{ for } 0 \leq i \leq 5.$$

Let $\epsilon = 2.0$. The probability for each node to be selected is:

$$Pr[\text{a is selected}] = Pr[\text{b is selected}] \propto \exp(0.1) = 0.44,$$
$$Pr[v_i \text{ is selected}] \propto \exp(-2) = 0.13, \text{ for } 0 \leq i \leq 5.$$

Normalizing, we have that $Pr[\text{a is selected}] = Pr[\text{b is selected}] = 0.472$ and $Pr[v_i \text{ is selected}] = 0.0046$. Comparing to Example 3.9 the nodes with highest score increase probability compared to the unshifted local dampening and the exponential mechanism.

We design the shifting in a way that it rearranges the utility scores in a way that the distribution of the utility scores is more spread. The idea is the following: we shift left enough so that all utility scores are negative. Thus, elements with larger utility score are the elements with smallest absolute value after shifting. So these shifted score are dampened with large $\delta^u(x, t, r)$ (assuming positive correlation). This entails in more accentuate shrinkage of the dampened score compared to scores of elements with low original score, high absolute shifted score and low $\delta^u(x, t, r)$ (assuming positive correlation). This implies that large scores are dampened closer to 0 and small scores are dampened to large negative values.

Hereby we propose to replace the original utility function u with its shifted version u^s where s is the utility score shift and

$$u^s(x, r) = u(x, r) - s.$$

One could design a private query, consuming part of the privacy budget, to choose s such that it minimizes some loss function to optimize accuracy. In this work, we set s to a value that does not depend on private data, $s \longrightarrow \infty$. In what follows, the shifted local dampening mechanism is stated as follows:

Definition 4.3. (Shifted Local Dampening Mechanism). The shifted local dampening mechanism $\mathcal{M}^*(x, \epsilon, u, \delta^u, \mathcal{R})$ outputs an element $r \in \mathcal{R}$ with probability equals to

$$\lim_{s \to \infty} \left(\frac{\exp\left(\frac{\epsilon D_{u^s,\delta}(x,r)}{2}\right)}{\sum_{r' \in \mathcal{R}} \exp\left(\frac{\epsilon D_{u^s,\delta}(x,r')}{2}\right)} \right).$$

We next show that $\mathcal{M}^*$ satisfies ϵ-differential privacy.

THEOREM 4.4. *The shifted local dampening mechanism $\mathcal{M}^*_\delta$ preserves ϵ-Differential Privacy if δ is admissible.*

4.2 Function δ replacement

For any given non-numeric query with utility function u, one can construct a function $\hat{\delta}^u(x, t, r)$ from an admissible $\delta^u(x, t, r)$ that does not depend on r. Hence, replacing $\delta^u(x, t, r)$ with $\hat{\delta}^u(x, t, r)$ on the local dampening mechanism solves the aforementioned inversion problem (Example 4.1).

$$\hat{\delta}^u(x, t, r) = \max_{r' \in \mathcal{R}} \delta^u(x, t, r').$$

Basically, $\hat{\delta}^u(x, t, r)$ increases the value for a given $r \in \mathcal{R}$ and $t \geq 0$ to the maximum value for $\delta(x, t, r')$ among all $r' \in \mathcal{R}$. This results in the same value $\hat{\delta}^u(x, t, r)$ for any given r, causing r to be dampened with same scale of any other element for a fixed t. A drawback of this is that $\hat{\delta}^u(x, t, r) \geq \delta^u(x, t, r)$ meaning that $\mathcal{M}$ looses accuracy.

An intermediate result shows that $\hat{\delta}^u(x, t, r)$ is admissible.

LEMMA 4.5. *Let $\delta_1(x, t, r), \ldots, \delta_p(x, t, r)$ be admissible functions. Then $\delta(x, t, r)$ defined as $\delta(x, t, r) = \max(\delta_1(x, t, r), \ldots, \delta_p(x, t, r))$ is an admissible function.*

The proof of Lemma 4.5 is immediately given by the admissibility of $\delta_1(x, t, r), \ldots, \delta_p(x, t, r)$. Lemma 4.5 entails in some important results: (i) $\hat{\delta}^u(x, t, r)$ is admissible if δ^u is admissible and (ii) $LS^u(x, t)$ is an admissible function once $LS^u(x, t) = max_{r \in \mathcal{R}} LS^u(x, t, r)$ and $LS^u(x, t, r)$ is an admissible function (Theorem 3.4).

5 APPLICATIONS

We present two applications to demonstrate the effectiveness of local dampening: 1) Influential node analysis, where given a graph database, retrieve the label of the top-k most influential nodes based on Egocentric Betweenness Centrality (EBC); and 2) Decision tree induction where we build decision trees based on the ID3 algorithm from tabular data.

5.1 Influential Node Analysis

Identifying influential nodes in a network is an important task for social network analysis for marketing purposes [29]. This analysis has great value for making a more effective marketing campaign since influential nodes have great capacity to diffuse a message through the network. Using EBC (Definition 5.1) as an influence measure allows to identify influential nodes that are important in different loosely connected parties.

Definition 5.1. (Egocentric Betweenness Centrality (EBC) [9, 15])

$$EBC(c) = \sum_{u,v \in N_c | u \neq v} \frac{p_{uv}(c)}{q_{uv}(c)},$$

where $N_c = \{v \in V | \{c, v\} \in E\}$ is the set of neighbors of the central node c, $q_{uv}(c)$ is the number of geodesic paths connecting u and v on the induced subgraph $G[N_c \cup \{c\}]$ and $p_{uv}(c)$ is the number of those paths that include c.

Formally, influential node analysis is a query over an input graph database $G = (V, E)$ that releases the labels of k nodes that maximize a given influence metric, e.g., EBC.

Private Mechanism. We use edge differential privacy for graph databases where the goal is to protect sensitive information about the edges in G. The graph G is denoted as a vector belonging to $\{0, 1\}^{\binom{n}{2}}$ where n is the number of nodes in the input graph and each entry on this vector represents an edge in G (1 is exists, 0 otherwise). By Definition 2.1 neighboring graphs differ in exactly one edge.

Algorithm 1: PrivTopk

```
1 Procedure PrivTopk(Graph $G = (V, E)$, Privacy
    Budget $B$, Integer $k$)
2     $\epsilon = B/k$
3     $\Omega = \emptyset$
4     for $j \leftarrow 1$ to $k$ do
5         $v = MEC(G, \epsilon, EBC, V)$ // Non-numeric
              mechanism call
6         $\Omega = \Omega \cup \{v\}$
7     end
8     return $\Omega$
```

We propose *PrivTopk*, a top-k algorithm template which chooses iteratively k nodes that maximizes EBC (Algorithm 1). In each iteration, the algorithm makes a call to a non-numeric mechanism (line 5) that returns a node which maximizes EBC that was not previously chosen. We experiment with three instances of this algorithm template: 1) *GlobalPrivTopk*, where we replace line 5 with an exponential mechanism call, 2) *LocalPrivTopk* where we replace line 5 with a local dampening call and 3) *ShiftedLocalPrivTopk* where we replace line 5 by a shifted local dampening mechanism. We use EBC as the utility function.

The privacy correctness of the algorithm follows from the sequential composition property of differential privacy [33]. Our algorithm issues k calls to a private mechanism with privacy budget ϵ/k which means that the total privacy budget consumed in the entire algorithm is $k \times \epsilon/k = \epsilon$. Thus Algorithm 1 satisfies B-differential privacy.

Global Sensitivity. We need to provide the global sensitivity for EBC to the Exponential Mechanism:

Lemma 5.2. *(EBC global sensitivity)*

$$\Delta EBC = \max\left(\frac{\Delta(G)(\Delta(G) - 1)}{4}, \Delta(G)\right),$$

where $\Delta(G)$ is the maximum degree of the input graph G. In this work, we assume the maximum degree is public information or that we have an upper bound for it.

Element local sensitivity. For the local dampening call, we provide an upper bound to the element local sensitivity using the admissible function δ^{EBC}:

Definition 5.3. (Admissible function $\delta^{EBC}(G, t, v)$).

$$\delta^{EBC}(G, t, v) = \max\left(\frac{(d^G(v) + t)(d^G(v) + t - 1)}{4}, d^G(v) + t\right),$$

where $d^G(v)$ denotes the degree of v in G, i.e., $d^G(v) = |N_v^G|$. We also show that $\delta^{EBC}(G, t, v)$ is admissible (Lemma 5.4).

Lemma 5.4. *$\delta^{EBC}(G, t, v)$ is an admissible function.*

For a node v with degree $deg^G(v)$, there are $\binom{deg^G(v)}{2} = (deg^G(v) \cdot (deg^G(v) - 1))/2$ terms in the *EBC* equation for v (Definition 5.1), i.e., pairs $(u, z) \in N_v^G$. As each term contributes at most 1 to *EBC*, it suggests that there is a correlation between $EBC(v)$ and $deg^G(v) + t$ and consequently, between $EBC(v)$ and $\delta^{EBC}(G, t, v)$. Empirical observation of the datasets confirmed that correlation. For this reason, the shifted local dampening mechanism call in ShiftedLocalPrivTopk is suitable.

5.2 Decision tree induction

Classification based on decision tree is an important tool for data mining [24]. Specifically, decision trees are a set of rules that are applied to the input variables to decide to which class a given instance belongs. A decision tree induction algorithm takes as input a dataset $\mathcal{T}$ with attributes $\mathcal{A} = \{A_1, \ldots, A_d\}$ and a class attribute C and produces a decision tree. The notation for this section is summarized in Table 1. All logarithms are in base 2. When it is clear from the context, we drop the superscript $\mathcal{T}$ from the notations.

The ID3 algorithm [37] starts with the root node containing the original set. Then the algorithm greedly chooses an unused attribute to split the set and generate child nodes. The selection criterion is Information Gain (IG), given by the entropy before splitting minus the entropy after splitting. This process continues recursively for the child node until splitting does not reduce entropy or the maximum depth is reached.

Private Mechanism. We use the algorithm GlobalDiffPID3 [16] (Algorithm 2) as a template. This algorithm is the evolution of the method presented with SuLQ framework [3]. Algorithm 2 makes calls to SuLQ [3] primitives: 1) **NoisyCount** that uses Laplace mechanism to return private estimate of count queries, and 2) **Partition** that splits the dataset into disjoint subsets so that the privacy budgets for the queries over each subset do not sum up (parallel composition [34]) meaning that the privacy budget can be used more efficiently. Thus, we aim to apply the local dampening mechanism and the shifted local dampening to the Algorithm 2.

Information Gain. In line 12, Algorithm 2 we need to provide an utility function that describes the quality of the split of an attribute. In this paper, we address one of the most traditional split criterion, information gain (IG). It is given by the entropy of the class attribute C in $\mathcal{T}$ and the obtained entropy of C splitting the tuples according to an attribute $A \in \mathcal{A}$.

Table 1: Notation table for private decision tree induction

Variable	Definition
IG	Information Gain
$Gini$	Gini Coefficient
$\mathcal{T}$	Dataset
$\mathcal{A}$	Attribute set
A_i	i-th attribute
C	Class attribute
$\tau^{\mathcal{T}}$	Cardinality of a dataset $\mathcal{T}$: $\tau^{\mathcal{T}} = \lvert\mathcal{T}\rvert$
r_A	Values of an attribute A in a record r
r_C	Values of the class attribute C in a record r
$\mathcal{T}_j^A$	Set of records $r \in \mathcal{T}$ where attribute A takes value j: $\mathcal{T}_j^A = \{r \in \mathcal{T} : r_A = j\}$
$\tau_j^{A,\mathcal{T}}$	Cardinality of $\mathcal{T}_j^A$: $\tau_j^{A,\mathcal{T}} = \lvert\mathcal{T}_j^A\rvert$
$\tau_c^{\mathcal{T}}$	Number of records $r \in \mathcal{T}$ where class attribute C takes value c: $\tau_c^{\mathcal{T}} = \lvert r \in \mathcal{T} : r_C = c\rvert$
$\tau_{j,c}^{A,\mathcal{T}}$	Number of records $r \in \mathcal{T}$ where attribute A takes value j and class attribute C takes value c: $\tau_{j,c}^{A,\mathcal{T}} = \lvert r \in \mathcal{T} : r_A = j \wedge r_c = c\rvert$

Algorithm 2: GlobalDiffPID3

```
1  Procedure GlobalDiffPID3(Dataset $\mathcal{T}$, Attribute
   Set $\mathcal{A}$, Class attribute $C$, Depth $d$, Privacy
   Budget $B$)
2      $\epsilon = B/(2(d+1))$
3      return Build_DiffPID3($\mathcal{T}, \mathcal{A}, C, d, \epsilon$)
4  Procedure Build_DiffPID3($\mathcal{T}$, $\mathcal{A}$, $C$, $d$, $\epsilon$)
5      $t = \max_{A \in \mathcal{A}} |A|$
6      $N_{\mathcal{T}} = \mathrm{NoisyCount}_\epsilon(\mathcal{T})$
7      if $\mathcal{A} = \emptyset$ or $d = 0$ or $\frac{N_{\mathcal{T}}}{t|C|} < \frac{\sqrt{2}}{2}$ then
8          $\mathcal{T}_c = \mathrm{Partition}(\mathcal{T}_j, \forall c \in C : r_c = c)$
9          $\forall c \in C : N_c = \mathrm{NoisyCount}_\epsilon(\mathcal{T}_c)$
10         return a leaf labeled with $\arg\max_c(N_c)$
11     end
12     $\bar{A} = \mathcal{E}(\mathcal{T}, \epsilon, u, \mathcal{A})$ // Exp. mechanism call
13     $\mathcal{T}_i = \mathrm{Partition}(\mathcal{T}, \forall i \in \bar{A} : r_{\bar{A}} = i)$
14     $\forall i \in \bar{A}$ : Subtree$_i$ = Build_DiffPID3($\mathcal{T}_i, \mathcal{A} \setminus \bar{A}, C, d-1, \epsilon$)

15     return a tree with a root node labeled $\bar{A}$ and edges
        labeled 1 to $\bar{A}$ each going to Subtree$_i$
```

$$IG(\mathcal{T}, A) = H_C(\mathcal{T}) - H_{C|A}(\mathcal{T}).$$

Since $H_C(\mathcal{T})$ does not depend on A, we can further simplify the utility function IG:

$$IG(\mathcal{T}, A) = -\tau.H_{C|A}(\mathcal{T}) \tag{1}$$

$$= -\sum_{j \in A}\sum_{c \in C} \tau_{j,c}^A.\log(\frac{\tau_{j,c}^A}{\tau_j^A}). \tag{2}$$

Global sensitivity. The exponential mechanism requires the computation of the global sensitivity for IG. It is given by $\Delta IG = \log(N+1) + 1/\ln 2$ [16] where N is the size of the dataset $\mathcal{T}$. The global sensitivity case can be achieved by $\mathcal{T}$ and $\mathcal{T}'$ where 1) $\mathcal{T}$ has all tuples with values for A equal to a single value $j \in A$ and all tuples class attribute C are set to a value different from a given value $c \in C$ (i.e. $\tau_j^A = \tau$ and $\tau_{j,c}^A = 0$); and 2) $\mathcal{T}'$ is obtained from $\mathcal{T}$ by adding a tuple r where $r_A = j$ and $r_C = c$.

Element local sensitivity. In our experiments, we observed that this mentioned case for the global sensitivity is unusual for real datasets. For those datasets, a local measurement of the sensitivity can be about one order of magnitude lower than the global sensitivity. To this matter, we replace line 12 for a local dampening mechanism call producing the algorithm *LocalDiffPID3*.

Element Local Sensitivity at distance 0. To use local dampening mechanism, we provide means to efficiently compute the element local sensitivity for IG (Lemma 5.6). The element local sensitivity at distance t measures $LS^{IG}(\mathcal{T}', 0, A)$ for all datasets $\mathcal{T}'$ such that $d(\mathcal{T}, \mathcal{T}') \le t$. We first show how to obtain $LS^{IG}(\mathcal{T}', 0, A)$:

LEMMA 5.5. *(Element local sensitivity at distance 0 for IG). Given a dataset $\mathcal{T}$ and the attribute set A, $LS^{IG}(\mathcal{T}, 0, A)$ produces the element local sensitivity for IG at distance 0:*

$$LS^{IG}(\mathcal{T}, 0, A) = \max_{j \in A, c \in C} h(\tau_j^{A,\mathcal{T}}, \tau_{j,c}^{A}\mathcal{T}),$$

where

$$h(a, b) = \max(f(a) - f(b), g(b) - g(a)),$$
$$g(x) = x.log((x-1)/x) - log(x-1),$$
$$f(x) = x.log((x+1)/x) + log(x+1).$$

Assume that $g(x) = 0$ for $x \le 1$ and $f(x) = 0$ for $x \le 0$. Note that, the expression $g(\tau_{j,c}^{A,\mathcal{T}}) - g(\tau_j^{A,\mathcal{T}})$ measures the impact of the removal of a tuple r such that $r_A = j$ and $r_C = c$ and the expression $f(\tau_j^{A,\mathcal{T}}) - f(\tau_{j,c}^{A,\mathcal{T}})$ measures the addition of tuple r. Thus to obtain $LS^{IG}(\mathcal{T}, 0, A)$, we need to measure, for all $j \in A$ and $c \in C$, the addition or removal of the tuple r where $r_A = j$ and $r_C = c$, i.e. $h(\tau_j^{A,\mathcal{T}}, \tau_{j,c}^{A,\mathcal{T}}) = \max(f(\tau_j^{A,\mathcal{T}}) - f(\tau_{j,c}^{A,\mathcal{T}}), g(\tau_{j,c}^{A,\mathcal{T}}) - g(\tau_j^{A,\mathcal{T}}))$.

Element local sensitivity at distance t. We use a similar idea to compute $LS^{IG}(\mathcal{T}, t, A)$. $LS^{IG}(\mathcal{T}, t, A)$ searches for the largest $LS^u(\mathcal{T}', 0, A)$ over all datasets $\mathcal{T}'$ where $d(\mathcal{T}, \mathcal{T}') \le t$:

$$LS^{IG}(\mathcal{T}, t, A) = \max_{\mathcal{T}' | d(\mathcal{T}, \mathcal{T}') \le t} LS^u(\mathcal{T}', 0, A)$$
$$= \max_{c \in C, j \in A} \max_{\mathcal{T}' | d(\mathcal{T}, \mathcal{T}') \le t} h(\tau_j^{A,\mathcal{T}'}, \tau_{j,c}^{A,\mathcal{T}'}).$$

Exhaustively iterating over all $\mathcal{T}'$ to compute $h(\tau_j^{A,\mathcal{T}'}, \tau_{j,c}^{A,\mathcal{T}'})$ is not feasible since the number of datasets $\mathcal{T}'$ grows exponentially with respect to t. However, we can restrict the number of evaluations of h by discarding some of the datasets $\mathcal{T}'$.

To this end, we introduce the algorithm *Candidates*$(\mathcal{T}, t, j, c)$ (Algorithm 3) that produces a subset of the set of the pairs $(\tau_j^{A,\mathcal{T}'}$,

$\tau^{A,\mathcal{T}'}_{j,c}$) of all datasets $\mathcal{T}'$ such that $d(\mathcal{T},\mathcal{T}') = t$, i.e., *Candidates*$(\mathcal{T}, t, j, c) \subseteq \{(\tau^{A,\mathcal{T}'}_{j}, \tau^{A,\mathcal{T}'}_{j,c}) \mid d(\mathcal{T},\mathcal{T}') = t\}$.

Algorithm 3: Candidates Algorithm

```
1  Procedure Candidates(Dataset $\mathcal{T}$, distance $t$,
     attribute value $j$, class attribute value $c$)
2      if $t = 0$ then
3          return $\{(\tau^{A}_{j}, \tau^{A}_{j,c})\}$
4      end
5      $candidates = \emptyset$
6      for each pair $(a, b) \in Candidates(\mathcal{T}, t-1, j, c)$ do
7          if $a > 0$ and $b > 0$ then
8              $candidates = candidates \cup \{(a-1, b-1)\}$
9          end
10         if $a < \tau$ then
11             $candidates = candidates \cup \{(a+1, b)\}$
12         end
13     end
14     return $candidates$
```

The Candidates algorithm has two important properties:

(1) *Candidates*$(\mathcal{T}, t, j, c)$ contains the pair $(\tau^{A,\mathcal{T}'}_{j}, \tau^{A,\mathcal{T}'}_{j,c})$ such that $h(\tau^{A,\mathcal{T}'}_{j}, \tau^{A,\mathcal{T}'}_{j,c})$ is maximum, i.e., $h(\tau^{A,\mathcal{T}'}_{j}, \tau^{A,\mathcal{T}'}_{j,c}) = \max_{\mathcal{T}' \mid d(\mathcal{T},\mathcal{T}')=t} h(\tau^{A,\mathcal{T}'}_{j}, \tau^{A,\mathcal{T}'}_{j,c})$ (Lemma 5.6);

(2) It is cacheable, when computing $D_{IG,\delta^{IG}}$, we evaluate $LS^{IG}(\mathcal{T}', t, A)$ several times in increasing order of t then one can cache calls to $Candidates(\mathcal{T}, t-1, j, c)$ to execute $Candidates(\mathcal{T}, t, j, c)$ (line 6) efficiently.

Thus $LS^{IG}(\mathcal{T}, t, A)$ is given by:

Lemma 5.6. *(Element local sensitivity at distance t for IG) Given an input table $\mathcal{T}$, a distance t and an attribute set A, $LS^{IG}(\mathcal{T}, t, A)$ produces the element local sensitivity at distance t for IG.*

$$LS^{IG}(\mathcal{T}, t, A) = \max_{\substack{j \in A, c \in C,\\ 0 \le t' \le t}} \max_{(a,b) \in Candidates(\mathcal{T}, t', j, c)} h(a, b).$$

In turn, the computation of $LS^{IG}(\mathcal{T}, t, A)$ is also cacheable. One can store a previous call to $LS^{IG}(\mathcal{T}, t-1, A)$ to compute $LS^{IG}(\mathcal{T}, t, A)$ as:

$$LS^{IG}(\mathcal{T}, t, A) = \max\left(\max_{\substack{j \in A, c \in C,\\ (a,b) \in Candidates(\mathcal{T}, t, j, c)}} h(a, b), \; LS^{IG}(\mathcal{T}, t-1, A) \right).$$

In the datasets used in our experiments, LS^{IG} and IG exhibit correlation. Consequently, we also replace the exponential mechanism call on line 12 in algorithm 2 by a call to the Shifted local dampening with LS^{IG}, this new algorithm is called *ShiftedLocalDiffPID3*.

Gini Index. Another well-known splitting criterion is Gini Index [4]. The global sensitivity for Gini Index is 2 [16] which is a relatively low sensitivity. We also calculated the local sensitivity for Gini index but its local sensitivity is not significantly lower than its global sensitivity to improve accuracy. Hence, we omit the definitions and proofs for Gini index but we report the results in the section 6.

Continuous Attributes. An important feature introduced in C4.5 algorithm [39] is the support for continuous attributes. To support continuous attributes, we use a simpler approach that performed well in our experiments. We discretize the continuous attributes on the dataset and use them as discrete attributes.

6 EXPERIMENTAL EVALUATION

We carry out experiments to compare the accuracy of local dampening on the applications outlined in Section 5.

6.1 Influential Node Analysis

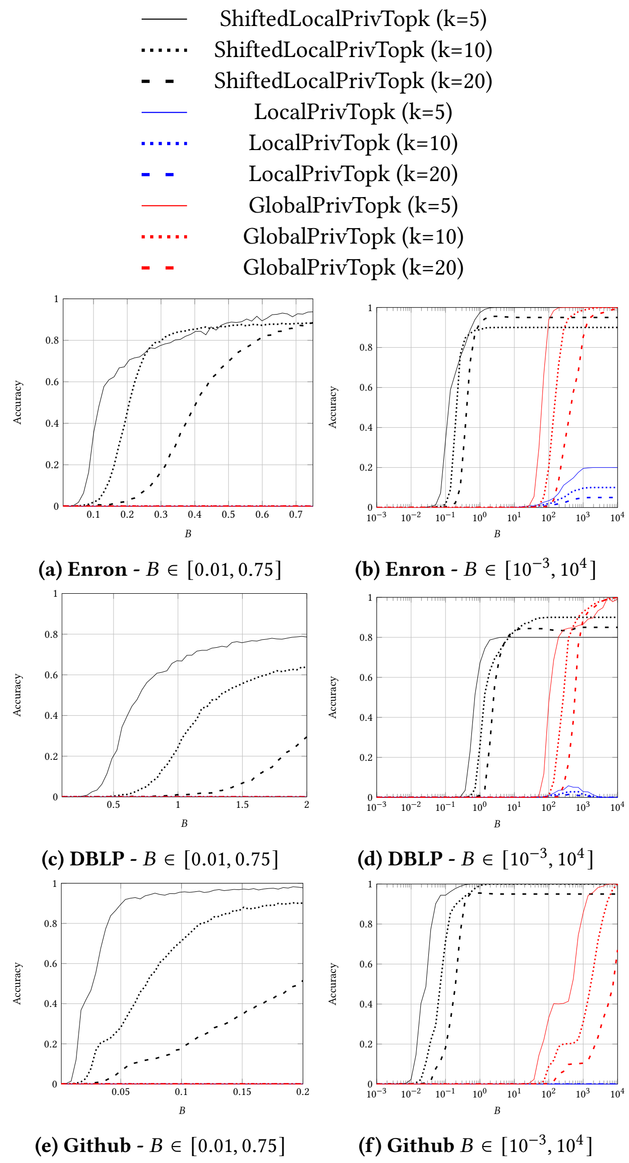
Datasets. We use three real-world graph datasets: 1) *Enron* is a network of email communication obtained from around half million emails. Each node is an email address and an edge connects a pair of email addresses that exchanges emails ($|V| = 36,692$, $|E| = 183,831$ and $\Delta G = 1,383$); 2) *DBLP* is a co-authorship network where two authors (nodes) are connected if they published at least one paper together ($|V| = 317,080$, $|E| = 1,049,866$ and $\Delta G = 343$); 3) *Github* is a network of developers with at least 10 stars on the platform. Developers are represented as nodes and an edge indicates that two developers follow each other ($|V| = 37,700$, $|E| = 289,003$ and $\Delta G = 9,458$). All datasets can be found on Stanford Network Dataset Collection [26].

Methods. We compare the three versions of *PrivTopk* (algorithm 1): 1) *GlobalPrivTopk* using exponential mechanism, 2) *LocalPrivTopk* using local dampening and 3) *ShiftedLocalPrivTopk* using shifted local dampening.

Evaluation. We evaluate the accuracy by the percentage of common nodes to the retrieved top-k set and the true top-k set, i.e., (|retrieved_topk ∩ true_topk|)$/k$. We report the mean accuracy in 100 simulations. We set $k \in \{5, 10, 20\}$ and a range for privacy budget B for each dataset: 1) Enron: $B \in [0.01, 0.75]$, 2) DBLP: $B \in [0.1, 2.0]$ and 3) Github: $B \in [001, 0.3]$. Also, we test higher values of the privacy budget for all datasets, $B \in [10^{-3}, 10^{4}]$.

Figure 5 displays the results. Note that all lines for GlobalPrivTopk and LocalPrivTopk overlap in Figures 5a, 5c and 5e. For smaller values of privacy budget (Figures 5a, 5c and 5e), we observe that ShiftedLocalPrivTopk outperforms GlobalPrivTopk and LocalPrivTopk. GlobalPrivTopk accuracy is near zero for all the tested values due to the high global sensitivity, e.g., $\Delta EBC = 22,361,076.5$ for github dataset, $\Delta EBC = 477,826.5$ for DBLP dataset and $\Delta EBC = 29,326.5$ for Enron dataset. The LocalPrivTopk algorithm suffers from the inversion problem (Section 4) while ShiftedLocalPrivTopk could exploit the correlation between EBC and δ^{EBC} to fix this problem.

In Figures 5a, 5c and 5e, we observe a clear pattern where the methods perform worse as k grows. This is explained by the fact that each call to the local dampening or exponential mechanism uses B/k of the total privacy budget B (Algorithm 1). Thus, larger k implies that less of the privacy budget is used in each local dampening call which hurts accuracy.



(a) Enron - $B \in [0.01, 0.75]$

(b) Enron - $B \in [10^{-3}, 10^{4}]$

(c) DBLP - $B \in [0.01, 0.75]$

(d) DBLP - $B \in [10^{-3}, 10^{4}]$

(e) Github - $B \in [0.01, 0.75]$

(f) Github $B \in [10^{-3}, 10^{4}]$

Figure 5: Accuracy for PrivTopk algorithm - ShiftedLocalPrivTopk outperforms LocalPrivTopk and GlobalPrivTopk by reducing the privacy budget consumption by 3 to 4 orders of magnitude.

Because of space constraints we do not present experienements for smaller values of k. However, local dampening mechanism is specially prone to suffer from the inversion problem for a small k value. Consider the case where $k = 1$ and the node in the top-1 is dampened to a lower score then the local dampening mechanism selects a non top-1 node with high probability which hurts accuracy.

For ShiftedLocalPrivTopk, we note that the datasets require different values for privacy budget to get reasonable accuracy. This is explained by a number of factors. For Github dataset, the distribution of EBC is heavy tailed thus the nodes with high EBC have a higher probability to be correctly picked with low privacy budget. On the other hand, the DBLP dataset requires more privacy budget as it has roughly 10 times more nodes than the other datasets, which dilute the probability of the nodes with higher EBC.

For larger values of privacy budget where GlobalPrivTopk achieves a higher accuracy (Figures 5b, 5d and 5f). Our approach ShiftedLocalPrivTopk achieves the same level of accuracy with privacy values 3 to 4 orders of magnitude less than GlobalPrivTopk.

6.1.1 *Comparison to related work.*

PrivateSQL [25] is an approach that can answer linear queries with cyclic joins and correlated subqueries with GROUPBY clauses. This approach identifies a set of views over the base relations that support the analyst queries and then generates private synopsis for each view. The analyst's queries are rewritten as linear queries over the private views' synopsis. The sensitivity computation for each view is based on Flex [21] augmented with truncation operators. The synopsis generation is based on non-negative least squares inference [27]. Graph databases can be modeled as a table Node(id) and a table Edge(source, target).

We carry out an experimental comparison to our Influential Node Analysis approach. For this application, PrivateSQL is not particularly scalable (as discussed further) so we performed the experiments with smaller datasets and test with fewer values for the privacy budgets.

PrivateSQL addresses the Influential Node Analysis problem by computing the counts $q_{uv}(c)$ and $p_{uv}(c)$ for all $u, v \in N_c$ (Definition 11) to compute EBC and take the top-k nodes with highest EBC score. Note that we need to account only for the terms $p_{uv}(c)/q_{uv}(c)$ where the distance from u to v in $G[N_c \cup \{c\}]$ is 2 which is the maximum possible distance as $u, v \in N_c$. If their distance is 1 (i.e. u and v are neighbors), the term $p_{uv}(c)$ is 0 since the geodesic path of length 1 from u to v ($u, v \neq c$) cannot contain c. Thus, for PrivateSQL, we pose private queries $Q(u, v, c)$ (SQL query available in [10]) that returns 1) 0 if u and v are neighbors, 2) 0 if the distance from u to v is larger than 2 and 3) $q_{uv}(c)$, otherwise.

Therefore, when $Q(u, v, c)$ is not equal to 0, it means that the term $p_{uv}(c)/q_{uv}(c)$ should be accounted. In that case, we obtain a noisy estimate for $q_{uv}(c)$ from $Q(u, v, c)$. A noisy estimate for $p_{uv}(c)$ can be derived from noisy $q_{uv}(c)$ by setting p_{uv} to 0 if $q_{uv}(c) = 0$ or to 1 if $q_{uv}(c) > 0$. The rationale is that exactly one of the paths of length 2 from u to v contains c as $u, v \in N_c$.

The set N_c is itself private information. Hence, to compute $EBC(c)$ for every $c \in V(G)$ we need to compute $Q(u, v, c)$ for every $u, v \in V(G)$. This results in a total number of $O(n^3)$ queries which poses a scalability problem. For this reason, we perform experiments with samples S of the graphs which are obtained by choosing a node sample S_n in breadth-first search fashion with a random seed node and then we set $S = G[S_n]$.

Table 2 displays the mean accuracy for 10 runs on 10 sample graphs with $|S_n| = 50$ nodes with $k \in \{1, 2, 3\}$ for each $B \in \{0.1, 0.5, 1.0, 5.0, 10.0\}$. We compare the best local dampening based algorithm ShiftedLocalPrivTopk (PTK) with the PrivateSQL based approach (PSQL).

PrivateSQL approach generated one private view for each node in the graph. Thus, the privacy budget needs to be divided by the number of nodes n which implies that accuracy is hurt as n grows. Moreover, the sensitivity for each view is high, e.g, sensitivity is

1448 when $\Delta(G) = 10$. This entails in a poor performance for the PrivateSQL based approach.

Table 2: Mean accuracy for ShiftedLocalPrivTopk (PTK) and PrivateSQL (PSQL) over 10 runs on 10 sample graphs with 50 nodes with $k \in \{1, 2, 3\}$ for each $B \in \{0.1, 0.5, 1.0, 5.0, 10.0\}$.

	Enron		DBLP		Github	
B	PTK	PSQL	PTK	PSQL	PTK	PSQL
0.1	0.06	0.01	0.05	0.02	0.07	0.02
0.5	0.45	0.01	0.27	0.02	0.49	0.02
1.0	0.60	0.16	0.44	0.05	0.69	0.02
5.0	0.84	0.20	0.87	0.11	0.86	0.03
10.0	0.88	0.21	0.92	0.21	0.91	0.07

The accuracy per k is not provided in Table 2 because of space constraints. However, as k grows, the probability of coincidence of retrieving a true top-k node increases which means accuracy increases with k.

6.2 Decision Tree Induction

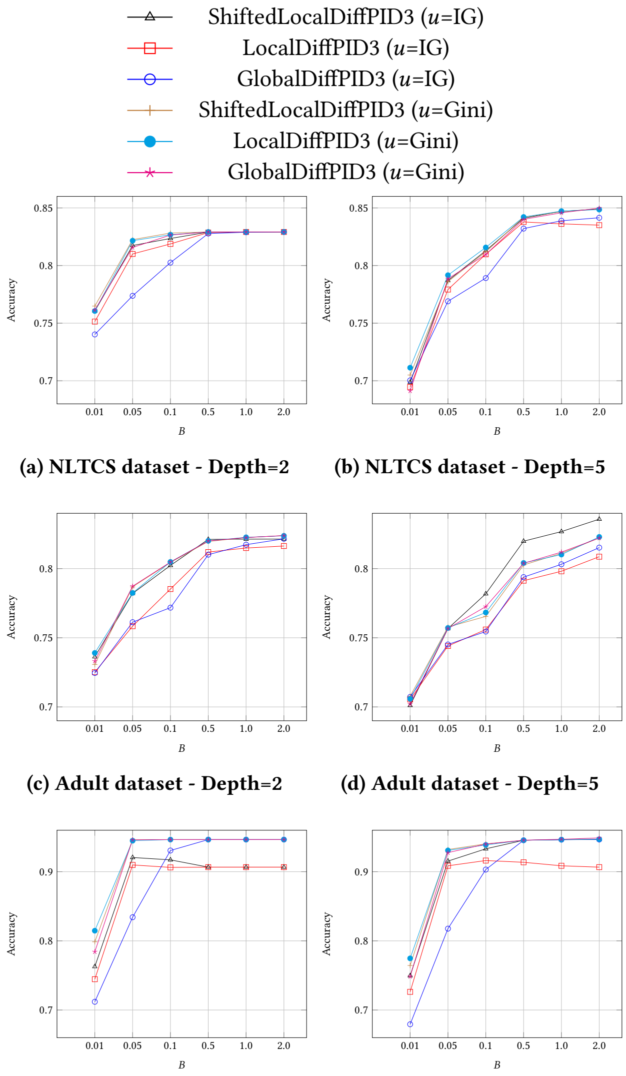
Datasets. For this application, we make use of three tabular datasets: 1) *National Long Term Care Survey (NLTCS)* [31] is a dataset that contains 16 binary attributes of 21, 574 individuals that participated in the survey, 2) *American Community Surveys (ACS)* dataset [40] includes the information of 47, 461 rows with 23 binary attributes obtained from 2013 and 2014 ACS sample sets in IPUMS-USA and 3) *Adult* dataset [1] contains 45, 222 records (excluding records with missing values) with 12 attributes where 8 are discrete and 4 are continuous.

Methods. We compare the three versions of the DiffPID3 (algorithm 2): 1) *GlobalDiffPID3* using exponential mechanism, 2) *LocalDiffPID3* using local dampening mechanism and 3) *ShiftedLocalDiffPID3* using local dampening mechanism with shifting. We test Gini index (Gini) and Information Gain (IG) as utility function.

Evaluation. We evaluate the accuracy of the approach by reporting the mean accuracy across the 10 runs of a 10-fold cross validation. We set $depth \in \{2, 5\}$ and $\epsilon \in \{0.01, 0.05, 0.1, 0.5, 1.0, 2.0\}$.

Figure 6 presents the results. For most privacy budgets, we observe LocalDiffPID3 improves the accuracy using IG as the utility function except for the Adult Dataset, up to 11%. Moreover, ShiftedLocalDiffPID3 improves on LocalDiffPID3 for every instance (further 3.5% at most), especially for the Adult Dataset where it overperforms all mechanisms using Gini for depth = 5. Also ShiftedLocalDiffPID3 has a maximum increase of 12% in accuracy compared to to GlobalPrivTopk.

The local dampening mechanism could approximate the accuracy of an IG-based algorithm to the accuracy of a low sensitivity Gini-based algorithm. An exception is for the trees built on the ACS dataset with IG in Figure 6e where the inversion problem (Section 4) appears. Specifically, the second and the third attributes with largest IG become the third and second attributes, respectively, with larger Dampened IG. As a consequence, as B grows, LocalDiffPID3 tends to pick the first and the third attributes with largest Information which is sub-optimal. This problem is mitigated for trees with larger depth since it can pick, in deeper levels, those attributes that loose rank (see Figure 6f).



(a) NLTCS dataset - Depth=2

(b) NLTCS dataset - Depth=5

(c) Adult dataset - Depth=2

(d) Adult dataset - Depth=5

(e) ACS dataset - Depth=2

(f) ACS dataset - Depth=5

Figure 6: Accuracy for DiffID3 algorithm - The results shows that, for IG, LocalDiffPID3 is more accurate in general than GlobalDiffPID3 (up to 11% improvement) and ShiftedLocalDiffPID3 improves on LocalDiffPID3 by up to 3.5%. For Gini index, local dampening mechanism shows no clear benefit.

For Gini, the local sensitivity is near the global sensitivity: the sensitivity for Gini is constant in 2 while it is 13 to 17 for IG in our datasets. Consequently local dampening does not improve much on the exponential mechanism.

7 RELATED WORK

There is a vast literature on Differential Privacy (DP) for numeric queries, and we refer the interested reader to [30] for a recent survey.

In this section, we discuss how our work fills a gap in the current literature about Differential Privacy for non-numeric queries.

Exponential mechanism: The first proposed approach for providing Differential Privacy to non-numeric queries is the Exponential Mechanism [33]. It uses a notion of global sensitivity, first proposed in [8], to sample an element from the set of possible outputs of a given query. Subsequently, the staircase mechanism [17, 18] adopts the same setting: utilization of global sensitivity for non-numeric queries. However, our early experimental evaluation with this mechanism showed that it had lower accuracy than the Exponential Mechanism for our applications.

Many DP works have tackled non-numeric problems using the exponential mechanism as part of their approaches. These works include: 1) PrivBayes, which privately releases synthetic tabular data while maintaining the correlation among its attributes by creating a k-degree Bayesian Network from the original data [43] and 2) Releasing range queries, contingency tables and data cubes [19]. Since all these problems have been addressed using the Exponential mechanism and global sensitivity of their utility function to guarantee DP, they are all candidate problems that could potentially benefit from using the local dampening mechanism instead. Doing so effectively is an interesting direction of future research.

Local sensitivity: The concept of local sensitivity was introduced in [35] for numeric queries. The authors proposed the Smooth Sensitivity framework, which is a generic approach for numeric queries. They applied it to compute the median, the cost of a minimum spanning tree, the count of triangles in a graph and k-means. Ladder functions, proposed by Zhang et al. [42], leveraged local sensitivity with the exponential mechanism to compute the counts of subgraphs as k-triangles, k-stars and k-cliques. It groups the outputs by range, where the ranges are constructed as a function of local sensitivity. Earlier work by Karwa et al. [22] addressed the same problem by using the smooth sensitivity framework [35].

Linear queries over relational databases. Kasiviswanathan et al. [23] observed that the sensitivity of many graph problems is a function of the maximum degree of the input graph G, so they proposed a generic projection that truncates the maximum degree of G. This projection is built upon local sensitivity but the target query is answered using the global sensitivity on the truncated graphs which may still be high. Moreover, it satisfies a weaker definition of privacy: (ϵ, δ)-differential privacy.

A new notion of sensitivity called restricted sensitivity was introduced by Blocki et al. [2] to answer local profile queries and subgraph counts. In this setting, the querier may have some belief about the structure of the input graph, so the restricted sensitivity measures sensitivity only on the subset of graphs which are believed to be inputs to the algorithm. However, this work satisfies only (ϵ, δ)-differential privacy.

For releasing linear statistics over relational databases using SQL, local sensitivity has been used for answering full acyclic join queries [41]. This approach lacks generality since we cannot compute some functions as EBC without cyclic joins and GROUPBY clauses. The recursive mechanism [5] can answer linear queries with unrestricted joins with GROUPBY clauses, however it requires the target function f to be monotonic, i.e., inserting a new individual in the database always causes f to increase (or always decrease). This condition is not satified in the computation of Information Gain, Gini Index and EBC.

Private decision tree induction. A first ID3 based private algorithm is proposed by Blum et al. [3] based on Laplace mechanism. Our approach is based on Friedman and Schuster [16] which proposed an improvement on the Blum et al. [3] algorithm. It replaces a Laplace mechanism call by an exponential mechanism call. This change corresponds to line 12 in Algorithm 1. The authors also added support for continuous attributes and pruning. Our work is the first to apply local sensitivity to greedy trees which was an open question pointed out in a recent survey on private decision trees [14].

Many other works address the private construction of random forests [11–13, 20, 36, 38]. Interestingly, local sensitivity was used for building random forests [12, 13] using smooth sensitivity. This shows a promising future direction of our work which is applying local dampening to construct random forests.

To the best of our knowledge, the literature lacks a generic framework for providing Differential Privacy for non-numeric queries using local sensitivity. Our work in this paper fills this gap.

8 CONCLUSION

In this paper, we introduced the Local Dampening mechanism, a novel framework to provide Differential Privacy for non-numeric queries using local sensitivity. We have shown that using local sensitivity on non-numeric queries reduces the magnitude of the noise added to achieve Differential Privacy which makes the answer of those queries more useful. We evaluated our approach on two applications: 1) Influential node analysis which benefited greatly from the use of local sensitivity and 2) Decision Tree induction which improves on approaches that use the exponential mechanism for this task based on Information Gain.

Our paper has laid the foundations for providing DP for non-numeric queries using local sensitivity. There are many interesting directions of future work. First, as discussed in Section 7, any problem in the literature that has used the Exponential mechanism for non-numeric queries to guarantee DP is a candidate problem that could potentially benefit from using our local dampening mechanism instead, and worthy of future work. Second, it would be interesting to tackle other graph influence/centrality metrics for Influential Node analysis, such as PageRank. Third, applying the local dampening mechanism to private random forest algorithms is a promising future direction. Finally, achieving a deeper theoretical understanding of the local dampening mechanism to understand the class of problems for which it can provide significant gains over the Exponential mechanism is an important and challenging direction of future work.

ACKNOWLEDGMENTS

This work was partially support by CAPES/Brazil under grant numbers 364998/2019-00, 465652/2019-00, and 189723/2018-01.

REFERENCES

[1] Catherine L Blake and Christopher J Merz. 1998. UCI repository of machine learning databases.

[2] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. 2013. Differentially private data analysis of social networks via restricted sensitivity. In *Proceedings*

of the 4th conference on Innovations in Theoretical Computer Science. ACM, 87–96.

[3] Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. 2005. Practical privacy: the SuLQ framework. In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 128–138.

[4] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. 1984. *Classification and regression trees*. CRC press.

[5] Shixi Chen and Shuigeng Zhou. 2013. Recursive mechanism: towards node differential privacy and unrestricted joins. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. 653–664.

[6] Cynthia Dwork. 2011. Differential privacy. *Encyclopedia of Cryptography and Security* (2011), 338–340.

[7] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. 2006. Our data, ourselves: Privacy via distributed noise generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 486–503.

[8] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*. Springer, 265–284.

[9] Martin Everett and Stephen P Borgatti. 2005. Ego network betweenness. *Social networks* 27, 1 (2005), 31–38.

[10] Victor A. E. Farias, Felipe T. Brito, Chery Flynn, Javam C. Machado, Subhabrata Majumdar, and Divesh Srivastava. 2020. Local Dampening: Differential Privacy for Non-numeric Queries via Local Sensitivity. arXiv:2012.04117 [cs.CR]

[11] Sam Fletcher and Md Zahidul Islam. 2015. A Differentially Private Decision Forest. *AusDM* 15 (2015), 99–108.

[12] Sam Fletcher and Md Zahidul Islam. 2015. A differentially private random decision forest using reliable signal-to-noise ratios. In *Australasian joint conference on artificial intelligence*. Springer, 192–203.

[13] Sam Fletcher and Md Zahidul Islam. 2017. Differentially private random decision forests using smooth sensitivity. *Expert Systems with Applications* 78 (2017), 16–31.

[14] Sam Fletcher and Md Zahidul Islam. 2019. Decision tree classification with differential privacy: A survey. *ACM Computing Surveys (CSUR)* 52, 4 (2019), 1–33.

[15] Linton C Freeman. 1978. Centrality in social networks conceptual clarification. *Social networks* 1, 3 (1978), 215–239.

[16] Arik Friedman and Assaf Schuster. 2010. Data mining with differential privacy. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. 493–502.

[17] Quan Geng, Peter Kairouz, Sewoong Oh, and Pramod Viswanath. 2015. The staircase mechanism in differential privacy. *IEEE Journal of Selected Topics in Signal Processing* 9, 7 (2015), 1176–1184.

[18] Quan Geng and Pramod Viswanath. 2014. The optimal mechanism in differential privacy. In *2014 IEEE international symposium on information theory*. IEEE, 2371–2375.

[19] Moritz Hardt, Katrina Ligett, and Frank McSherry. 2012. A simple and practical algorithm for differentially private data release. In *Advances in Neural Information Processing Systems*. 2339–2347.

[20] Geetha Jagannathan, Krishnan Pillaipakkamnatt, and Rebecca N Wright. 2009. A practical differentially private random decision tree classifier. In *2009 IEEE International Conference on Data Mining Workshops*. IEEE, 114–121.

[21] Noah Johnson, Joseph P Near, and Dawn Song. 2018. Towards practical differential privacy for SQL queries. *PVLDB* 11, 5 (2018), 526–539.

[22] Vishesh Karwa, Sofya Raskhodnikova, Adam Smith, and Grigory Yaroslavtsev. 2011. Private analysis of graph structure. *PVLDB* 4, 11 (2011), 1146–1157.

[23] Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2013. Analyzing graphs with node differential privacy. In *Theory of Cryptography Conference*. Springer, 457–476.

[24] Sotiris B Kotsiantis, I Zaharakis, and P Pintelas. 2007. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering* 160 (2007), 3–24.

[25] Ios Kotsogiannis, Yuchao Tao, Xi He, Maryam Fanaeepour, Ashwin Machanavajjhala, Michael Hay, and Gerome Miklau. 2019. Privatesql: a differentially private sql query engine. *PVLDB* 12, 11 (2019), 1371–1384.

[26] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data.

[27] Chao Li, Gerome Miklau, Michael Hay, Andrew McGregor, and Vibhor Rastogi. 2015. The matrix mechanism: optimizing linear counting queries under differential privacy. *The VLDB journal* 24, 6 (2015), 757–781.

[28] Wentian Lu and Gerome Miklau. 2014. Exponential random graph estimation under differential privacy. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 921–930.

[29] Hao Ma, Haixuan Yang, Michael R Lyu, and Irwin King. 2008. Mining social networks using heat diffusion processes for marketing candidates selection. In *Proceedings of the 17th ACM conference on Information and knowledge management*. 233–242.

[30] Ashwin Machanavajjhala, Xi He, and Michael Hay. 2017. Differential Privacy in the Wild: A Tutorial on Current Practices & Open Challenges. In *Proc. of SIGMOD*. ACM, 1727–1730.

[31] Kenneth G Manton. 2010. National Long-Term Care Survey: 1982, 1984, 1989, 1994, 1999, and 2004. *Inter-university Consortium for Political and Social Research* (2010).

[32] Peter V Marsden. 2002. Egocentric and sociocentric measures of network centrality. *Social networks* 24, 4 (2002), 407–422.

[33] Frank McSherry and Kunal Talwar. 2007. Mechanism Design via Differential Privacy.. In *FOCS*, Vol. 7. 94–103.

[34] Frank D McSherry. 2009. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*. 19–30.

[35] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2007. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*. ACM, 75–84.

[36] Abhijit Patil and Sanjay Singh. 2014. Differential private random forest. In *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 2623–2630.

[37] J. Ross Quinlan. 1986. Induction of decision trees. *Machine learning* 1, 1 (1986), 81–106.

[38] Santu Rana, Sunil Kumar Gupta, and Svetha Venkatesh. 2015. Differentially private random forest with high utility. In *2015 IEEE International Conference on Data Mining*. IEEE, 955–960.

[39] Steven L Salzberg. 1993. C4. 5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc.

[40] Integrated Public Use Microdata Series. 2015. Version 6.0. *Minneapolis: University of* (2015).

[41] Yuchao Tao, Xi He, Ashwin Machanavajjhala, and Sudeepa Roy. 2020. Computing Local Sensitivities of Counting Queries with Joins. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 479–494.

[42] Jun Zhang, Graham Cormode, Cecilia M Procopiuc, Divesh Srivastava, and Xiaokui Xiao. 2015. Private release of graph statistics using ladder functions. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*. ACM, 731–745.

[43] Jun Zhang, Graham Cormode, Cecilia M Procopiuc, Divesh Srivastava, and Xiaokui Xiao. 2017. PrivBayes: Private data release via bayesian networks. *ACM Transactions on Database Systems (TODS)* 42, 4 (2017), 25.