Private Exploration Primitives for Data Cleaning

Chang Ge, Ihab F. Ilyas
School of Comp. Sci.
University of Waterloo
\{c4ge,ilyas\}@uwaterloo.ca

Xi He, Ashwin Machanavajjhala
Dept. of Comp. Sci.
Duke University
\{hexi88,ashwin\}@cs.duke.edu

ABSTRACT
Data cleaning, or the process of detecting and repairing inaccurate or corrupt records in the data, is inherently human-driven. State of the art systems assume cleaning experts can access the data (or a sample of it) to tune the cleaning process. However, in many cases, privacy constraints disallow unfettered access to the data. To address this challenge, we observe and provide empirical evidence that data cleaning can be achieved without access to the sensitive data, but with access to a (noisy) query interface that supports a small set of linear counting query primitives. Motivated by this, we present DPClean, a first of a kind system that allows engineers tune data cleaning workflows while ensuring differential privacy. In DPClean, a cleaning engineer can pose sequences of aggregate counting queries with error tolerances. A privacy engine translates each query into a differentially private mechanism that returns an answer with error matching the specified tolerance, and allows the data owner track the overall privacy loss. With extensive experiments using human and simulated cleaning engineers on blocking and matching tasks, we demonstrate that our approach is able to achieve high cleaning quality while ensuring a reasonable privacy loss.

1. INTRODUCTION
Data cleaning and data integration are essential tasks in most data science and data analytics applications. Due to magnitude and the diversity of the problem, most enterprises rely on externally developed tools to unify, clean, and transform their data. Examples include data wrangling tools, such as Trifacta [15], data integration tools, such as Tamr [29], and many other transformation tools [3, 10, 27]. Most of these tools are not turn-key solutions, and have a large number of tunable parameters and configurations. Self-tuned tools are often rely on field engineers examining the data, e.g., by eyeballing a sample, or by trying different configurations and design choices, and examining their effect on the data involved in the cleaning project. This interaction is guided mainly by two human roles: the data owner who has knowledge of the domain and can judge the quality of the cleaning process; and the cleaning engineer who is familiar with the cleaning tool and understands the implications of the configuration parameters. This interactive model customizes the cleaning solution and ensures the most effective deployment, and has been widely adopted in numerous systems [19, 25, 31].

This model works well if the cleaning engineer has direct access to the data, but presents serious challenges when dealing with sensitive and private data (e.g., intelligence data collected by agencies, or health records in a hospital). In these settings, enterprises often adopt one of two approaches: (i) rely on internally developed tools that can be tuned using the enterprise employees; or (ii) use commercial tools, and ask external cleaning experts who are familiar with the tool to configure it using public data sets that are close in structure to the actual data. Both approaches allow the enterprise employees or the cleaning tools experts to work directly on some data, but they suffer from major shortcomings: internally developed tools present a high cost of internal development and maintenance; and working with mockup data often results in poor tuning and sub-optimal results when the solution is deployed on the actual data. In our interaction with multiple data cleaning companies and experts, these two challenges are currently severe pain points in cleaning private and sensitive data sets.

In this work, we propose a new model for cleaning private and sensitive data sets. We observe that engineers can accurately tune cleaning workflows by combining their expertise with access to just a (noisy) aggregate query interface, rather than access to the exact data (or a sample of it). We present comprehensive empirical evidence of this observation in Section 7 and highlight our key findings in Figure 5. We asked 8 humans (real cleaning engineers) and used 200 programs that simulate human cleaners (called robots; described in Section 7) to tune workflows for two exemplar data cleaning tasks of blocking and matching in entity resolution. Figure 14 shows that humans and robots alike achieve high task quality (recall for blocking and F1-score for matching, defined in Section 5) with access to an aggregate queries interface on the database with no error. Moreover, the same is true even when aggregate queries are answered with tolerable error (Figures 13 and 14) with the highest achievable and median accuracy degrading mon-
Figure 1: Cleaning quality on blocking and matching tasks on citations dataset: (a) shows that good cleaners (including human and manually designed robots) achieve high quality using true aggregate statistics; (b) and (c) show that the cleaning quality using noisy aggregate statistics decreases as noise level (tolerance) increases. Refer to Section 7 for details.
multiple accesses in Section 6. Differential privacy has seen adoption in a number of real products at the US Census Bureau [12, 22, 30], Google [9] and Apple [11]. We did not choose semantically secure or property preserving encryption, as the former does not allow the cleaning engineer to learn anything about the data, and the latter is susceptible to attacks using side information [25].

2.2 Use Case: Entity Resolution

To illustrate the interactive process for data cleaning we consider entity resolution (also referred to as record linkage, entity matching, reference reconciliation or de-duplication) [9, 10], as an important data cleaning application, which aims to identify entities (or objects) referring to the same real-world entity. For concreteness, we will assume that the data owner has two sets of records $D_1$ and $D_2$ with the same schema of attributes $A_1, \ldots, A_k$ and the owner already posses a labeled training dataset $D_t$ with schema $A \times \Theta \times \{+,-\}$. We also assume that $D_t$ is constructed by (a) picking a sample $D'$ of records from $D_1$, (b) identifying for each $r \in D'$, another record $r' \in D_2$, and (c) for each pair $(r,r')$, labeling the pair as either a match (+) or a non-match (−) (like in [3]). Throughout this paper, we will assume that every record in $D_1$ and $D_2$ appears at most $m$ times in the training dataset $D_t$ $(m = 1$ in our experiments). Let $D^+_t$ and $D^-_t$ denote the set of positive and negative examples in the training set.

A typical entity resolution solution usually involves two tasks: **blocking** and **matching**. Both these tasks are achieved by learning a boolean formula $P$ (e.g. in DNF) over **similarity predicates**. We express a similarity predicate $p$ as a tuple $(A,t,sim,\theta)$ in $A \times T \times S \times \Theta$. First, $A \in A$ is an attribute in the schema and $t \in T$ is a transformation of the attribute value. Next, $sim \in S$ is a similarity function that usually takes a real value often restricted to $[0,1]$, and $\theta \in \Theta$ is a threshold. Given a pair of records $(r_1, r_2)$, a similarity predicate $p$ returns either ‘True’ or ‘False’ with semantics:

$$p(r_1, r_2) \equiv (sim(t(r_1,A), t(r_2,A)) > \theta) \quad (2)$$

For instance, the similarity predicate $(Name,q - gram,cosine,0.9)$ would return true whenever the cosine similarity of the character q-gram representation of the Name attribute in two records is greater than 0.9. Blocking and matching tasks typically use a rich library of attribute and context dependent transformations and similarity functions. There are an exponential number of possible predicates $A \times T \times S \times \Theta$, and an even larger number of formulas that can be constructed by combining them with boolean operators. Cleaning engineers judiciously limit the search space by their experience and domain expertise to create accurate cleaning workflows.

For blocking, our goal is to find a boolean formula that identifies a small set of candidate matching pairs that cover most of the true matches.

**Definition 2 (Blocking Task).** Discover a boolean formula $P_b$ over similarity predicates that achieves high recall, and low blocking cost. Recall is the fraction of true matching pairs that are labeled true by $P_b$.

$$recall^{P_b} = |\{(P_b(r,r') = true \mid (r,r') \in D_1 \cap D_2^+\}|/|D_2^+| \quad (3)$$

$$cost^{P_b} = |\{(P_b(r,r') = true \mid (r,r') \in D_1 \times D_2\}|/|D_1 \times D_2| \quad (4)$$

A matcher is a boolean formula $P_m$ that identifies matching records with the least false positives and false negatives.

**Definition 3 (Matching Task).** Discover a boolean formula $P_m$ over similarity predicates that achieves high recall and precision on $D_t$. Precision measures whether $P_m$ classifies true non-matches as matches, and recall measures the fraction of true matches that are captured by $P_m$.

$$precision^{P_m} = |D^+_t \cap D^+_m|/|D^+_m| \quad \text{(5)}$$

$$recall^{P_m} = |D^+_t \cap D^+_m|/|D^+_t| \quad \text{(6)}$$

$$F1^{P_m} = 2 \times \frac{precision^{P_m} \times recall^{P_m}}{precision^{P_m} + recall^{P_m}} \quad \text{(7)}$$

where $D^+_t$ denotes the set of rows in $D_t$ that satisfy $P_m$.

3. DPCLEAN: SYSTEM OVERVIEW

Figure 2 illustrates the architecture of DPCLEAN. The data owner and the cleaning engineer are the two parties involved in this system. The data owner owns the private dirty data, while the cleaning engineer is on the other side of the ‘firewall’ with no direct access to the dirty data. A cleaning engineer can interact with the sensitive data through the local exploration module, and the privacy engine answers these queries and tracks the privacy loss.

3.1 Local Exploration

Local exploration is a human-guided process, in which a cleaning engineer explores the dirty data with the goal of judiciously choosing parameters for automatic cleaning software in the data cleaning workflow.

Consider a blocking task (Def. 2) where the cleaner’s goal is to find a disjunction of predicates $p_1 \land \cdots \land p_k$ from the set of all predicates $S = P(A \times T \times S \times \Theta)$. A typical cleaning engineer interacts with the data to limit the space of predicates as follows:

**Example 1 (Data Profiling).** The cleaning engineer starts with profiling the data, for example, to quantify the number of missing values for each attribute, or to know the distribution of values of some attributes.

**Example 2 (Picking Predicates).** Based on the acquired data profile, the cleaning engineer chooses a suitable subset of attributes (e.g., ones with least number of nulls),
(a) Strategy instance 1 for blocking

\[
\text{A set of candidate predicates } P = \{ p \colon \text{sim}(t(A_i), t'(A_i)) > 0.7 \}
\]

c5a: Cleaner selects predicates in \( P \) with \( \text{sim}(2\text{grams}(1.\text{name}), 2\text{grams}(2.\text{name})) > 0.7 \).

c5b: Cleaner chooses a criterion for predicate \( p \):

- if it catches >50% of remaining matches and <10% of remaining non-matches.

\( q_3, \alpha \): What is the fraction of the remaining matches caught by \( p \)?

\( q_4, \beta \): What is the fraction of the remaining non-matches caught by \( p \)?

c6: Cleaner chooses to keep \( p \).

(b) Strategy instance 2 for blocking

\[
\text{A set of candidate predicates } P = \{ p \colon \text{sim}(t(A_i), t'(A_i)) > 0.7 \}
\]

c5a: Cleaner selects predicates in \( P \) with \( \text{sim}(2\text{grams}(1.\text{name}), 2\text{grams}(2.\text{name})) > 0.7 \).

c5b: Cleaner chooses a criterion for predicate \( p \):

- if it catches >50% of remaining matches and <10% of remaining non-matches.

\( q_3, \alpha \): What is the fraction of the remaining matches caught by \( p \)?

\( q_4, \beta \): What is the fraction of the remaining non-matches caught by \( p \)?

c6: Cleaner chooses to keep \( p \).

Figure 3: Two strategies for blocking with different queries

A subset of transformations (e.g., lower case, character 2-grams), a subset of similarity functions (e.g., Jaccard or Cosine similarity) along with reasonable thresholds to construct a set of candidate predicates \( S' \subseteq S \) to use to build the blocker.

Example 3 (Evaluating Predicates). Suppose \( p_1 : \text{cosine}(2\text{grams}(r_1.\text{name}), 2\text{grams}(r_2.\text{name})) > 0.7 \) and \( p_2 : \text{cosine}(2\text{grams}(r_1.\text{address}), 2\text{grams}(r_2.\text{address})) > 0.7 \) have been chosen as two predicates in \( S' \) (from Example 2).

The cleaner may want to measure the recall and blocking cost of \( p_1 \) and \( p_1 \lor p_2 \).

Example 4 (Updating Configuration). Suppose \( p_1 \) has a recall of 60% and blocking cost of 0.1%, and \( p_1 \lor p_2 \) has a recall of 65% and a blocking cost of 20%. The cleaner may choose to include \( p_1 \) in the solution and not \( p_1 \lor p_2 \) (as the latter has a high cost). On the other hand, if \( p_1 \lor p_2 \) can catch 90% of the matching pairs and 1% of the non-matching pairs, the cleaner may choose to retain \( p_1 \lor p_2 \) in the solution.

Some of these aforementioned example actions – e.g., profiling and predicate evaluation – can be achieved by the cleaning engineer posing aggregate queries over the (training) data. Other actions – e.g., selecting attributes or transformation, and setting thresholds on recall/cost – need the expertise of a human cleaning engineer. These two types of actions are formulated as follows.

Type-I: Queries with Tolerance. This type of actions allow the cleaning engineer to query aggregate statistics on the database with some error tolerance. Given an aggregate query \( q : D \to O \), we let \( \text{dist}_{q.\text{type}} : O \times O \to \mathbb{R}^+ \) denote a general distance function over the output space of the query that represent the difference between the true answer of a query \( q \) over \( D \), referred to as \( q(D) \), and a noisy answer \( a \in O \) to \( q \). We define error tolerance on query as:

\[ \text{Definition 4 ((} \alpha, \beta \text{)-q.type tolerance). Given a query } q : D \to O, \text{ and a distance function over the query output, } d : O \times O \to \mathbb{R}^+. \text{ We say a mechanism } M : D \to O \text{ satisfies } (\alpha, \beta)-q.\text{type tolerance for } \alpha \in \mathbb{R}^+ \text{ and } \beta \in [0, 1] \text{ if for any table } D \in D \]

\[ \Pr[\text{dist}_{q.\text{type}}(M(D), q(D)) \geq \alpha] \leq \beta. \quad (8) \]

We fix \( \beta \) to be a very small value \( e^{-15} \) throughout the paper. DPCEA supports the following query types: linear counting queries, linear counting queries with conditions and linear top-k queries. The query types and their tolerances are defined in Section 4. We denote a sequence of query-based actions by \( [(q_1, \alpha_1), (q_2, \alpha_2), \ldots] \) (and keep \( \beta \) the same for all queries).

Type-II: Smart Cleaning Choices. This type of actions are the smart choices made by the cleaning engineer based on the knowledge of the engineer (including the sequence query answers learned so far in addition to the initial expertise and the preference of this engineer). Consider Example 2 the space of choices on predicates are exponential in terms of the number of attributes, transformations, similarity functions, and possible thresholds. Different cleaning engineers can choose different sets of candidate predicates, \( P \) and even have different orders to evaluate these predicates. Moreover, there exist cleaner dependent criterion on whether to keep or prune a predicate in Example 4 and these criterion vary among engineers. We denote these engineer specific actions by \( C = [c_1, c_2, \ldots] \), where \( c_i \) corresponds to a set of choices made by the engineer after the \( i \)th type-I action \( q_i \) for \( i \geq 0 \).

Cleaning Strategy. During the interactive process, we observe a sequence of type-I and type-II actions interleaved \( \ldots, q_i, c_i, \ldots \). We use a strategy to denote a class of local explorations that using the same set of type-I actions (query types) but different type-II actions. Figure 3 shows two strategies iterates over example actions given in Example 4 to 4. These two strategies are different as they use different set of type-I actions (labeled by \( (q_i, \alpha_i) \)), though they share the same type-II actions (labeled by \( c_i \)). In particular, the queries \( q_1, q_5 \) in the strategy shown in Figure 3a are count queries, while the queries \( q_1', q_5' \) in Figure 3b are top-k and counting with condition queries respectively. The same engineer who applies the second strategy will see less information and may continue the exploration differently. If changing type-II actions of the strategy instance in Figure 3a (e.g., c5b changes to ‘Cleaner chooses a criterion for predicate \( p \): if it catches >60% of remaining matches and <5% of remaining non-matches’, the resulted exploration still belongs to the same strategy. The differences in type-II actions are mainly due to the differences in the expertise or preference of cleaning engineers.

3.2 Privacy Engine

The privacy engine in DPCEA ensures that every sequence of queries (with tolerances) posed by the cleaning engineer is answered while satisfying (\( B, \delta \))-differential privacy, where \( B \) and \( \delta \) are privacy budget parameters specified
by the data owner. The privacy engine serves two important functions, and its operations are summarized in Algorithm 1.

The first component of this module is an error tolerance to privacy translator (Line 4) that takes each query \( q \) and its tolerance requirement \((\alpha_i, \beta_i)\), and translates it into a differentially private (noise) mechanism \( M_i^{\alpha_i, \beta_i} \) or simply represented as \( M_i \). The translation is such that: (i) \( M_i \) satisfies \((\alpha_i, \beta_i)\)-type tolerance and \((\epsilon, \delta)\)-differential privacy with \( \epsilon \)-minimal privacy loss, i.e., the same mechanism \( M_i \) with different noise parameters cannot both simultaneously achieve \((\alpha_i, \beta_i)\)-type tolerance and \((\epsilon, \delta)\)-differential privacy. The translation is such that: (i) \( M_i \) satisfies \((\alpha, \beta)\)-type tolerance and \((\epsilon, \delta)\)-differential privacy with \( \epsilon \)-minimal privacy loss. Note that constraints (i) and (ii) can be individually ensured by releasing the true answer or releasing an extremely noisy answer, respectively. Satisfying both the constraints simultaneously is a key technical innovation in this paper, and we propose a set of translation algorithms with minimal privacy loss in Section 5. While our algorithms translate individual queries with tolerances to privacy mechanisms, translating a group of queries simultaneously can result in higher accuracy (and lower privacy loss) – we defer this interesting direction for future work.

The second component named as privacy analyzer (Line 11) analyzes the accumulated privacy cost. Given a sequence of mechanisms \((M_1, \ldots, M_{i-1})\) already executed by the privacy engine that satisfy an overall \((B_{i-1}, \delta)\)-differential privacy, the privacy engine calls function \( B_i = \text{estimateLoss}(M_1, \ldots, M_i, \delta) \) (Line 5) to estimate the privacy loss that would be incurred by running the next mechanism \( M_i \) (to answer the next query). If \( B_i \) exceeds the privacy budget \( B \), then the privacy does not answer the query, and the cleaning engine might either halt, or ask a query with a higher error tolerance. If \( B_i < B \), then the mechanism will be executed and made \( M_i(D) \) will be returned to the cleaning engine. The privacy engine then calls \( B_i = \text{analyzeLoss}(M_1, \ldots, M_i, \delta) \) to compute the actual privacy loss. In most cases, \( B_i \) will be the same as \( B_i \), but in some cases the two can be different. In some of our translated mechanisms \( M_i \), different execution paths have different privacy losses. \( B_i \) represents the worst case privacy loss across all execution paths (computed before running \( M_i \)), while \( B_i \) represents the actual privacy loss (after running \( M_i \)). The privacy analyzer guarantees that the execution of any sequence of mechanisms \((M_1, M_2, \ldots, M_i)\) before it halts is \((B, \delta)\)-differential privacy (see Section 6).

The following sections will describe the set of aggregate query primitives supported by DP-Clean, the privacy translator and the privacy analyzer.

4. PRIVACY PRIMITIVES

This section formalizes the aggregate query primitives supported in DP-Clean: (i) linear counting queries, (ii) linear counting queries with conditions, and (iii) top-k linear counting queries. We consider these primitives because: (1) the definition of tolerance is different for these queries, and (2) the differentially private algorithms that support these primitives are different.

4.1 Linear Counting Query (LC)

There is much work on answering linear counting queries under differential privacy [20, 13]. An LC query on a table \( D \), denoted by \( q_\phi() \), returns the number of rows in the table that satisfy a boolean predicate \( \phi : \Sigma \rightarrow \{0, 1\} \). An LC query \( q_\phi(D) \) can be expressed in SQL as

\[
\text{SELECT COUNT(*) FROM } D \text{ WHERE } \phi;
\]

Given an LC query \( q \), we define the distance between the true output \( q(D) \) and an approximate answer \( a \) as the absolute difference \( |q(D) - a| \). The error tolerance for a mechanism for answering an LC query is defined as:

\[
\text{Definition 5 (}(\alpha, \beta)\text{-LC TOLERANCE). Given a linear counting query } q_\phi \text{, we say a mechanism } M : D \rightarrow \mathbb{R} \text{ satisfies (} (\alpha, \beta)\text{-LC tolerance, if for any table } D \in D, \\
\Pr[|M(D) - q_\phi(D)| \geq \alpha] \leq \beta. \tag{9}
\]

LC queries can be used to guide several cleaning tasks.

Example 5 (LC Queries for NULLs). Counting the number of rows with NULL for an attribute is an LC query on a base table \( D \) which counts the number of rows in \( D \) where the given attribute has ‘NULL’ value. If the true answer is 10000, and the tolerance is \((\alpha = 10, \beta = 0.05)\), then the approximate answer should be within [9990, 10010] with 95% probability. The sensitivity of an LC query on a base table, or the max change in its output due to adding or removing a tuple in the input, is always \( 1 \). LC queries can also be posed on views over the base table, and its sensitivity is bounded by the stability of the view. Stability, defined as the number of tuples added or removed from a view due to adding or removing a single tuple in the base table, can be computed using well known methods [23, 14]. In all our experiments, queries are either on base tables or views with stability 1; hence, LC queries always have sensitivity 1.

Example 6 (LC Queries for Blocking). Given a dataset \( D \), let \( D_t \) be a training set that consists of pairs of records in \( D \) and whether they are matching or non-matching records (as described in Sec. 2.2). Then, recall of the blocking predicate \( P_k : \cosine(2\text{grams}(r1.name), 2\text{grams}(r2.name)) > 0.7 \) can be computed by first answering a LC \( q_{P_k} \) over the training set \( D_t \) that counts the number of matching records, \( D^{+}_t \), that satisfy the predicate \( P_k \), and then dividing by the number of matching records in \( D^{+}_t \), a known constant. Note that if each record in \( D \) appears at most \( m \) times in \( D_t \), then the stability of \( D_t \) (and the sensitivity of \( q_{P_k} \)) is at most \( m \).
4.2 LC Query with Condition (LCC)

Given a table $D$, a linear counting with condition (LCC) query $q_{\phi, \theta}$, returns a binary output ($\in \{0,1\}$) based on comparing an LC query output $q_\phi(D)$ with a given threshold $c$. The comparison operator $\theta$ can be $\{\leq, <, \geq, \leq\}$. The error tolerance for a mechanism answering an LCC query $q_{\phi, \theta}$ is:

\[
\text{Definition 6 ((\alpha, \beta)-LCC tolerance). A mechanism } M : D \rightarrow \{0,1\} \text{ satisfies (\alpha, \beta)-LCC tolerance for an LCC query } q_{\phi, \theta} \text{ if for any table } D,
\]
\[
\Pr[M(D) = 1 \mid q_\phi(D) < c - \alpha] \leq \beta \quad (10)
\]
\[
\Pr[M(D) = 0 \mid q_\phi(D) > c + \alpha] \leq \beta \quad (11)
\]

Error tolerance for other $q_{\phi, \theta}$ are defined analogously.

Example 7 (LCC Queries for NULLs). Continuing with Example 3 instead of querying for the number of NULLs for an attribute, the cleaner can ask whether the number of NULLs is greater than some threshold, say 1000. Given a tolerance of $(\alpha = 10, \beta = .05)$, then when the answer is 1, with 95% probability the true count is $> 1000 - 10$; and when the answer is 0, with 95% probability the true count is $< 1000 + 10$.

Similarly, LCC queries can be used to check if the quality of a blocking scheme or a matching predicate reaches a threshold.

4.3 LC Top-K Query (LCT)

Given a table $D$, a top-$k$ linear counting query takes in a set of boolean formulae $\Phi = \{\phi_1, \ldots, \phi_l\}$, and outputs the top-$k$ boolean formulae that are satisfied by the largest (or the least) number of rows in the table, denoted by $q_{\phi_1, \ldots, \phi_l}^{1\ldots k}$ (or $q_{\phi_1, \ldots, \phi_l}^{k\ldots l}$ respectively).

Definition 7 ((\alpha, \beta)-LCT tolerance). Given a top-$k$ linear counting query $q_{\phi_1, \ldots, \phi_l}^{1\ldots k}$, we say that a mechanism $M : D \rightarrow \Phi$ with output $a = \{\phi_1, \ldots, \phi_k\}$ satisfies (\alpha, \beta)-LCT tolerance if for any table $D$,
\[
\Pr[|\phi \in M(D) \mid q_\phi(D) < c_k - \alpha] > 0 \leq \beta \quad (12)
\]
\[
\Pr[|\phi \in (\Phi - M(D)) \mid q_\phi(D) > c_k + \alpha] = 0 \leq \beta \quad (13)
\]

where $c_k$ is the answer to the $k^{th}$ largest LC query.

This definition is equivalent to saying that with high probability all boolean formulae $\phi$ in the output of the mechanism have $q_\phi(D) > c_k - \alpha$, and every $\phi \in \{\phi_1, \ldots, \phi_l\}$ with $q_\phi(D) > c_k + \alpha$ is in the output.

Example 8 (LCT Queries for Missing Values). Again continuing with Example 3, suppose the cleaner wants to prune out $k$ attributes with the most number of NULLs. Rather than using LC queries counting the number of NULLs in each attribute, the cleaner could directly ask an LCT query. For instance, suppose $\{A_1, A_2, \ldots, A_5\}$ have $\{10000, 8000, 200, 100, 50\}$ NULLs respectively. The top $k = 2$ attributes with the most number of missing values are $A_1$, $A_2$. The number of NULLs in $A_2$, the kth largest LC query answer, is 8000. Given a tolerance of $(\alpha = 10, \beta = .05)$, then with high probability, attributes selected in the output should have a true count larger than 8000-10, and attributes with true count larger than 8000+10 should be appear in the output.

5. TRANSLATING ERROR TO PRIVACY

In this section, we provide translation algorithms for the primitives including LC, LCC, and LCT defined in the previous section. Given a query $q$, where $q.type \in \{LCC,LCT\}$ on a table $D$ with $(\alpha, \beta)$-type tolerance requirement for $q$, we propose a differentially private mechanism $M_{q, \beta}^\alpha$, or simply denoted as $M$, such that (i) $q$ is answered with mechanism $M$ that satisfies $(\alpha, \beta)$-type tolerance, (ii) mechanism $M$ satisfies $(\epsilon, \delta)$-differential privacy, with minimal privacy loss, i.e., the same mechanism $M$ with different noise parameters cannot both simultaneously achieve $(\alpha, \beta)$-type tolerance and $(\epsilon', \delta')$-differential privacy for $\epsilon' < \epsilon$. Note that this section considers mechanisms for one query at a time. The overall privacy of a sequence of queries will be presented in Section 6. We first present Laplace Mechanism based translations for all primitives. The privacy loss of mechanisms resulting from these translations depend only on the query type and tolerance, and not on the input data (Sec 5.1). We then discuss data-dependent translations for LCC queries that result in mechanisms with lower privacy loss for certain datasets and queries (Sec 5.2).

5.1 Laplace Mechanism based Translation

5.1.1 Laplace Mechanism For LC

The Laplace Mechanism is a classic differentially private algorithm that is widely used as a building block.

Definition 8 (Laplace Mechanism). Let $q$ be a LC query on database $D$. The Laplace mechanism releases $q(D) + \eta$. Noise $\eta \sim \text{Lap}(b)$ is drawn from the Laplace distribution with parameters $b$ that has a pdf $\Pr[\eta = z] \propto e^{-z/b}$.

The Laplace mechanism run with $b = s/\epsilon$ satisfies $\epsilon$-differential privacy (7), if the sensitivity of the query is bounded by $s$. For ease of presentation, we LC queries are answered on base tables or views with stability 1, resulting in $s = 1$. The Laplace distribution has a mean of 0 and a variance of $2b^2$, and $\Pr[|\eta| < |a, a|] < 1 - e^{-a/b}$. As the noise parameter $b$ increases, the noise is more likely to be outside the range $[-a, a]$.

DPClean translates LC queries $q$ with tolerance $(\alpha, \beta)$ into the Laplace mechanism with parameter $b = \alpha / \epsilon$. Based on the properties of the Laplace random variable, it is easy to see that it satisfies the tolerance requirement and has minimal privacy loss.

Theorem 1. Given a linear counting query $q_\phi(\cdot)$, for table $D \in D$, adding noise $\eta \sim \text{Lap}(b)$ to $q_\phi(D)$, where $b = \frac{1}{\ln(1/\beta)}$, denoted by $LM_{q, \beta}^\alpha(\cdot)$, can achieve $(\alpha, \beta)$-LC tolerance, i.e., $Pr[|LM_{q, \beta}^\alpha(D) - q_\phi(D)| \geq \alpha] \leq \beta$ with the minimal $\epsilon$-differential privacy cost of $\epsilon = 1/b = \frac{\ln(1/\beta)}{\alpha}$.

5.1.2 Laplace Comparison Mechanism for LCC

An LCC query can also be answered using Laplace noise, similar to LC. As shown in Algorithm 2 noise $\eta$ drawn from Laplace distribution with parameter $b = \frac{\alpha}{\ln(1/\beta)}$ is added to the difference between $q_\phi(D)$ and $c$. If the noisy difference is greater than 0, the mechanism outputs ‘True’; otherwise, ‘False’. This mechanism, referred as Laplace comparison mechanism has the following property. All proofs in this section are deferred to the Appendix.
Algorithm 2 \( \text{LCC}(q_{\phi,>c}, \alpha, \beta, D) \)

**Input:** \( LCT \ q_{\phi,>c}, \ (\alpha, \beta)\)-LCC tolerance, table \( D \)

**Output:** Answer \( a \in \{\text{True}, \text{False}\} \)

1: Draw noise \( \eta \sim \text{Lap}(b) \), where \( b = \frac{\alpha}{\ln(1/\alpha)} \)
2: Perturb difference \( \tilde{x} = q_{\phi}(D) - c + \eta \)
3: if \( \tilde{x} > 0 \) then
4: \( \text{return True} \)
5: \( \text{end if} \)
6: \( \text{return False} \)

Algorithm 3 \( \text{LTM}(q_{\phi_1,\ldots,\phi_L}, \alpha, \beta, D) \)

**Input:** \( \text{LCT} \ q_{\phi_1,\ldots,\phi_L}, \ (\alpha, \beta)\)-LCT tolerance, table \( D \)

**Output:** Answer \( a = \{\phi_1, \ldots, \phi_k\} \)

1: \((\eta_1, \ldots, \eta_L) \sim \text{Lap}(b)^L\), where \( b = \frac{\alpha}{\ln(1/\alpha) + \ln(k/\beta)} \)
2: \((\tilde{x}_1, \ldots, \tilde{x}_L) = (q_{\phi_1}(D), \ldots, q_{\phi_L}(D)) + (\eta_1, \ldots, \eta_L) \)
3: \((i_1, \ldots, i_k) = \text{Argmax}_{1 \leq i \leq L} \tilde{x}_i \)
4: \( \text{return} \{\phi_{i_1}, \ldots, \phi_{i_k}\} \)

**Theorem 2.** Given a linear counting query with condition \( q_{\phi,>c} \) for any \( D \in \mathcal{D} \), the Laplace comparison mechanism (Algorithm 2) denoted by \( \text{LCC}(q_{\phi,>c}) \), can achieve \((\alpha, \beta)\)-LCC tolerance with minimal \( \epsilon\)-differential privacy cost of \( \epsilon = \frac{\ln(1/\alpha)}{\alpha} \).

Alternatively, the cleaning engineer can pose a linear counting query \( q_{\phi} \) with \((\alpha, \beta)\)-LCT tolerance via \( \text{DPclean} \), and then use the noisy answer of \( q_{\phi}(D) \) to learn \( q_{\phi,>c} \) locally. This approach adds a smaller expected noise to \( q_{\phi}(D) \), and hence achieves \((\alpha, \beta)\)-LCC tolerance, i.e.,

**Lemma 3.** Using the output of a Laplace mechanism \( \text{LM}^{\alpha,\beta} \) to answer \( q_{\phi,>c} \) can achieve \((\alpha', \beta')\)-LCT tolerance, where \( \alpha' = (1 - \frac{\ln 2}{\ln 1/\alpha})\alpha < \alpha \).

This approach also allows a cleaning engineer to make more local decisions, for example, how much to adjust the threshold of a similarity function that consumes a larger privacy cost of \( \frac{\ln(1/\beta)}{\alpha} \) compared to \( \text{LCC}(q_{\phi,>c}) \) with cost of \( \frac{\ln(1/\alpha)}{\alpha} \).

5.1.3 Laplace Top-k Mechanism for LCT

Given a set of boolean formulæ \( \{q_1, \ldots, q_k\} \), LCT aims to find top-k boolean formulæ that are satisfied by the largest number of rows in the table. This primitive can be treated as \( L \) linear counting queries \( \{q_1, \ldots, q_k\} \) and be answered as shown in Algorithm 3. Each LC query \( q_{\phi_i}(D) \) is perturbed with noise drawn from \( \text{Lap}(b), b = \frac{\alpha}{2(\ln 1/\beta + \ln(k/\beta))} \).

These boolean formulæ are then sorted based on their corresponding noisy counts in descending order and the first \( k \) boolean formulæ are outputted. This approach, referred to as Top-k Laplace Mechanism, has the following property.

**Theorem 4.** Given a top-k linear counting query, \( q_{\phi_1,\ldots,\phi_k} \), for any table \( D \in \mathcal{D} \), Laplace top-k mechanism (Algorithm 3) denoted by \( \text{LTM}^{\alpha,\beta}_{k=1,\ldots,k} \), can achieve \((\alpha, \beta)\)-LCT tolerance with minimal \( \epsilon\)-differential privacy cost, where \( \epsilon = k/b = \frac{2\ln 1/\beta}{\ln 1/\alpha} \).

Alternatively, the cleaning engineer can pose a list of linear counting query \( q_{\phi_1,\ldots,\phi_k} \) with \((\alpha, \beta)\)-LCT tolerance for each linear counting query, and then answer the top-k linear counting query \( q_{\phi_1,\ldots,\phi_k} \) locally. However, this approach does not achieve \((\alpha, \beta)\)-LCT tolerance, i.e.,

5.2 Data Dependent Translation for LCC

The translation mechanisms shown above are all data-independent as the derivation of the noise parameter and hence the privacy cost only depends on the query type and the tolerance requirement. Given the same query and tolerance requirement, it is possible to achieve smaller privacy cost using a different mechanism for certain datasets. In this section, we use linear counting query with condition LCC, \( q_{\phi,>c}(\cdot) \), as an example to explore such so-called data-dependent translations.

Intuitively, when \( q_{\phi}(D) \) is much larger (or smaller) than \( c \), then a much larger (smaller resp.) noise can be added to \( q_{\phi}(D) \) without changing the ‘True’ or ‘False’ decision of the system. Consider the following example.

**Example 9.** Consider an LCC query \( q_{\phi,>c} \), where \( c = 100 \). To achieve \((\alpha, \beta)\) tolerance for this query, where \( \alpha = 10, \beta = 0.1^{10} \), the Laplace comparison mechanism requires a privacy cost of \( \frac{\ln(1/\alpha)}{\alpha} \approx 2.23 \) by Theorem 2, regardless of input \( D \). Suppose \( q_{\phi}(D) = 1000 \). In this case, \( q_{\phi}(D) \) is much larger than the threshold \( c \), and the difference is \( \frac{1000 - 100}{100} = 90 \) times of the accuracy bound \( \alpha = 10 \). Hence, even when applying Laplace comparison mechanism with a privacy cost equals to \( \approx 2.23 \times 0.25 \) where the noise added is bounded by 90\( \alpha \) with high probability \( 1 - \beta \), the noisy difference \( q_{\phi}(D) - c + \eta_{\text{sign}} \) will still be greater than 0 with high probability.

This is an example where a different mechanism rather than \( \text{LCC}(q_{\phi,>c}) \) achieves the same LCC tolerance with a smaller privacy cost. Note that the tightening of the privacy cost in this example requires to know the value of \( q_{\phi}(D) \), and thus this privacy cost estimation is a data dependent approach. We need to ensure this step is also differentially private and propose two approaches next.

5.2.1 LCM with Poking

The first approach is summarized in Algorithm 2 named as Laplace Comparison Mechanism with Poking (LCMP). This algorithm first computes the privacy cost of \( \text{LCC}(q_{\phi,>c}) \) based on Theorem 2 denoted by \( \epsilon_{\text{LCC}} \) (Line 1). It then chooses to run LCM with a small fraction of this privacy cost.
Algorithm 5 LCMP(qφ,c,α,β,D)

Input: LCC qφ,c, (α,β)-LCC tolerance, table D, # poking steps m

Output: Answer a ∈ {True, False}
1: Compute \( \epsilon_{\text{max}} = \ln(m/(2\beta))/\alpha \)
2: Initial privacy cost \( \epsilon_0 = \epsilon_{\text{max}}/m \)
3: \( \tilde{x}_0 = q_{\phi}(D) - c + \eta_0 \), where \( \eta_0 = \text{Lap}(1/\epsilon_0) \)
4: for \( i = 0, 1, \ldots, m - 2 \) do
5: Set \( \alpha_i = \ln(m/(2\beta))/\epsilon_i \)
6: if \( (\tilde{x}_i - \alpha_i)/\alpha \geq 1 \) then
7: return True
8: else if \( (\tilde{x}_i + \alpha_i)/\alpha \leq 1 \) then
9: return False
10: else
11: Increase privacy budget \( \epsilon_{i+1} = \epsilon_i + \epsilon_{\text{max}}/m \)
12: Update noise \( \eta_{i+1} = \text{NoiseDown}(\eta_i, \epsilon_i, \epsilon_{i+1}) \)
13: New noisy difference \( \tilde{x}_{i+1} = q_{\phi}(D) - c + \eta_{i+1} \)
14: end if
15: end for
16: if \( \tilde{x}_{m-1} > 0 \) then
17: return True
18: else
19: return False
20: end if

cost: i.e., it adds \( \text{Lap}(1/\epsilon_0) \) to the difference \( q(D) - c \), with \( \epsilon_0 = f \cdot \text{LCM} \) (Line 2,3). If the noisy difference is too large (Line 5), then LCMP returns ‘True’. If it is too small (Line 6), LCMP return ‘False’. In both these cases, LCMP incurs a fraction of the privacy loss of LCM. If the noisy difference is neither too small or too large, it runs LCM (Line 10), and incurs an additional privacy loss of \( \epsilon_{\text{LCM}} \).

Theorem 6. Given a LCC query \( q_{\phi,c} \), for any table \( D \in \mathcal{D} \), LCM with Poking (Algorithm 5), denoted by \( \text{LCMP}^{\phi,c} \), achieves the privacy budget \( \epsilon_{\text{LCMP}} \).

Note that LCMP has a higher privacy loss than LCM in the worst case. However, if LCMP returns in either Line 6 or Line 8, then the privacy loss is much smaller, and it occurs often in our experiments. The privacy engine (Algorithm 5) would use the worst case privacy loss to decide whether to answer a query using LCM (in estimateLoss), but use the actual loss (which could be much smaller) to compute the overall loss (in analyzeLoss) if LCMP has been run.

5.2.2 LCM with Multi-Poking

In the Laplace Comparison Mechanism with Poking (Algorithm 5), the prepaid privacy budget \( \epsilon_0 \) needs to be specified, but it is difficult to determine a value without looking at the query answer. To tackle this challenge, we propose an alternative approach that allows of \( m \) pokes with increasing privacy cost. This approach is summarized in Algorithm 5 as Laplace Comparison Mechanism with Multi-Poking (LCMPM). This approach first computes the privacy cost if all \( m \) pokes are needed, \( \epsilon_{\text{max}} = \ln(m/(2\beta))/\alpha \). The first poke is the same as LCM which checks if the noisy difference \( \tilde{x}_0 \) is sufficiently greater (or smaller) than the tolerance \( \alpha_0 \) for the current privacy cost. If this is true (Lines 8-12), then an answer ‘True’ (or ‘False’) is returned; otherwise, the privacy budget is relaxed with additional \( \epsilon_{\text{max}}/m \). At \( i+1 \)th iteration, instead of sampling independent noise, we apply the NoiseDown Algorithm (details refer to Algorithm 6 in Appendix) to correlate the new noise \( \eta_{i+1} \) with noise \( \eta_i \) from the previous iteration. In this way, the privacy loss of the first \( i+1 \) iterations is \( \epsilon_{i+1} \), and the noise added in the \( i+1 \)th iteration is equivalent to a noise generated with Laplace distribution with privacy parameter \( b = (1/\epsilon_{i+1}) \). This approach allows the data cleaner to learn the query answer with a gradual relaxation of privacy cost. This process repeats until all \( \epsilon_{\text{max}} \) is spent. We show that Algorithm 5 achieves both tolerance and privacy requirements.

Theorem 7. Given a LCC query \( q_{\phi,c} \), for any table \( D \in \mathcal{D} \), LCM with Multi-Poking (Algorithm 5), denoted by \( \text{LCMP}^{\phi,c} \), achieves the privacy budget \( \epsilon_{\text{LCMP}} \).

Like LCMP, the worst case privacy loss of LCMPM is larger than that of LCM, but this mechanism may stop before \( \epsilon_{\text{max}} \) is used up, and hence it potentially saves privacy budget for the subsequent queries. In fact, the privacy loss of LCMPM can be a \( \frac{\ln m}{\epsilon m} \) fraction of LCM’s privacy loss, if the mechanism returns in the first iteration.

In DPClean, the default translation algorithms for LC, LCC, and LCT are respectively LM, LCM, and LTM described in Section 5.1. We empirically show these optimizations for LCC, but when to turn them on is an interesting future direction. Moreover, Laplace noise is used as an example for translation which achieves \( \epsilon \)-differential privacy per query in this section. Other type of noises are also possible, such as Gaussian noise which provides \( (\epsilon, \delta) \)-differential privacy per query, and can be adapted into this system.

6. PRIVACY ANALYZER

Privacy analyzer is the second component in the privacy engine that analyzes the accumulated privacy cost of all interactive process. There are two functions called by this component as shown in Algorithm 6: (i) \( B_1 = \text{estimateLoss}(M_1, \ldots, M_i, \delta) \) (Line 5) to estimate the privacy loss that would be incurred by running the next mechanism \( M_i \) (to answer the next query); (ii) \( B_2 = \text{analyzeLoss}(M_1, \ldots, M_i, \delta) \) to compute the actual privacy loss after running \( M_i \). The output of these two functions are the same in most cases, except when data-dependent translation algorithms are applied, such as Algorithms 3 and 4 which have different execution paths and hence different privacy loss. \( B_1 \) represents the worst case privacy loss across all execution paths (computed before running \( M_i \)) if running \( M_i \), while \( B_2 \) represents the actual privacy loss (after running \( M_i \)). We show two types of privacy composition techniques for a sequence of differentially private algorithms.

6.1 Sequential Composition

We first present sequential composition, a simple but useful composition technique.

Theorem 8 (Sequential Composition [7]). Let \( M_1(\cdot) \) and \( M_2(\cdot) \) be algorithms with independent sources of randomness that ensure \( (\epsilon_1, \delta_1) \)- and \( (\epsilon_2, \delta_2) \)-differential privacy respectively. An algorithm that outputs both \( M_1(D) = O_1 \) and \( M_2(O_1, D) = O_2 \) ensures \( (\epsilon_1 + \epsilon_2, \delta_1 + \delta_2) \)-differential privacy.

Given a sequence of data independent translation mechanisms \( M_1, M_2, \ldots, M_i \) shown in Section 5.1 that satisfy differential privacy with cost of \( \epsilon_1, \epsilon_2, \ldots, \epsilon_i \) respectively, both functions \( \text{estimateLoss} \) and \( \text{analyzeLoss} \) output \( (\epsilon_1 + \cdots + \epsilon_i) \). For data dependent translation mechanisms, \( \text{estimateLoss} \) considers the worst privacy loss of
these mechanisms. For instance, if \( M_i \) is LCM with poking (Algorithm 4), \( \text{estimateLoss} \) increments the privacy loss \( B_{i-1} \) by \( \frac{(1+\ln(1/\delta))}{\alpha} \) (the privacy loss derived in Theorem 6) to obtain \( \hat{B}_i \), while \( \text{analyzeLoss} \) considers the actual privacy loss: if LCM is not called (Line 10), then the actual privacy loss \( B_i \) increments by \( \frac{(1+\ln(1/\delta))}{\alpha} \); otherwise, the actual privacy cost \( B_i \) increments by \( \frac{(1+\ln(1/\delta))}{\alpha} \). Similarly, if \( M_i \) is LCM with multi-poking (Algorithm 5), then \( \text{estimateLoss} \) increments the privacy loss \( B_{i-1} \) by \( \epsilon \max \) while \( \text{analyzeLoss} \) considers only \( \epsilon_i \) if the algorithm stops at loop \( i \).

### 6.2 Advanced Composition

Advanced composition techniques [7] allows the privacy parameters to degrade more slowly than sequential composition by considering the privacy loss as a random variable rather than as a fixed worst case cost. DPCLEAN uses the Moments Accountant technique [2, 24] summarized below. Unlike sequential composition, which applies to black-box mechanisms, advance composition techniques requires knowledge of the mechanisms (and the noise distributions used within).

The sequence of differentially private mechanisms \( M_1, M_2, ..., M_i \) run by the privacy engine can be considered as an instance of adaptive composition which is modeled by letting the auxiliary input of the \( i \)-th mechanism \( M_i \) be the output of all the previous mechanisms, i.e., \( M_i : \prod_{i=1}^{i-1} \mathcal{O}_l \times D \to \mathcal{O}_l \). The moment accountant keeps track of a bound on the moments of the privacy loss random variable of each mechanism defined below.

For neighboring databases \( D, D' \in \mathcal{D} \), a given mechanism \( M \), an auxiliary input \( aux \), the privacy loss of \( M \) outputting \( o \in \mathcal{O} \) can be captured using the random variable:

\[
c(\alpha; M, aux, D, D') = \log \frac{\Pr[M(aux, D) = o]}{\Pr[M(aux, D') = o]}.
\]

(14)

If \( M \) satisfies \( \epsilon \)-DP, then \( c(\alpha; M, aux, D, D') < \epsilon \) with probability 1, but could be much smaller.

Let \( \mu_M(\lambda; aux, D, D') \) called the \( \lambda \)-th moment be the log of the moment generating function of \( c \) evaluated at \( \lambda \):

\[
\mu_M(\lambda; aux, D, D') = \log \mathbb{E}_{o \sim M(aux, D)}[e^{\lambda c(\alpha; M, aux, D, D')}].
\]

(15)

The privacy loss of a mechanism requires to bound all possible \( \mu_M(\lambda; aux, D, D') \), i.e.,

\[
\mu_M(\lambda) = \max_{aux, D, D'} \mu_M(\lambda; aux, D, D'),
\]

(16)

where the maximum is taken over all possible \( aux \) and all the neighboring databases \( D, D' \).

**Theorem 9** (Moments Accountant [2]). Let \( \mu_M(\lambda) \) be defined as above. Then

1. **Composability**: Suppose that a mechanism \( M \) consists of a sequence of adaptive mechanisms \( M_1, ..., M_i \), where \( M_i : \prod_{i=1}^{i-1} \mathcal{O}_l \times D \to \mathcal{O}_l \). Then, for any \( \lambda \),

\[
\mu_M(\lambda) \leq \sum_{j=1}^{i} \mu_{M_j}(\lambda).
\]

2. **Tail bound**: The mechanism \( M \) is \( (\epsilon, \delta) \)-differentially private for \( \epsilon = \min_{\lambda}(\mu_M(\lambda) - \ln \delta)/\lambda \), for any \( \delta > 0 \).

By the theorem above, the privacy analyzer only needs to bound \( \mu_M(\lambda) \) at each step and sum them to bound the moments of the sequence of mechanisms \( M \). Then the tail bound can be used to convert the moment bound to the \( (\epsilon, \delta) \)-differential privacy guarantee. The analysis of a single Laplace noise based Laplace Mechanism [24] that adds \( \eta \sim \text{Lap}(b) \) leads to a privacy loss

\[
\mu_{\text{Lap}(b)}(\lambda) = \frac{1}{\lambda - 1} \log \left( \frac{e^{b(\lambda-1)/\lambda}}{2} + \frac{(\lambda-1)e^{-b/\lambda}}{2-1} \right)
\]

(17)

when \( \lambda > 1 \); \( \mu_{\text{Lap}(b)}(\lambda) = 1/b + e^{-b/\lambda} - 1 \) when \( \lambda = 1 \). When \( \lambda \) goes to \( \infty \), \( \mu_{\text{Lap}(b)}(\lambda) \) goes to \( 1/b \). As the translation mechanisms shown in Section 5 are all based Laplace noise, a sequence of these mechanisms can be analyzed as a sequence of adaptive Laplace mechanisms that add noises to the corresponding linear counting queries. In particular, the Laplace top-\( k \) mechanism (Algorithm 3) for LCT draws a vector of noises \( (\eta_1, ..., \eta_k) \) from \( \text{Lap}(b)(L) \), where \( \mu_{\text{Lap}(b)(L)}(\lambda) \) can be shown as \( \mu_{\text{Lap}(b)(L)}(\lambda) \).

Similar to sequential composition, the outputs of the two functions \( \text{estimateLoss} \) and \( \text{analyzeLoss} \) are the same if all mechanisms are data independent. When data-dependent algorithms are run, these two functions vary in output. Suppose \( M_i \) is LCM with poking (Algorithm 4) that may run only one Laplace mechanism with \( \text{Lap}(1/\epsilon) \) or two Laplace mechanisms. Function \( \text{estimateLoss} \) considers the worst case where two Laplace mechanisms are run while function \( \text{analyzeLoss} \) depends on the execution path. If only one Laplace mechanism was run, then the moments accountant in \( \text{analyzeLoss} \) considers the privacy loss of this Laplace mechanism alone, and the privacy cost can be saved for later mechanisms. Similarly, Laplace mechanism with multi-poking (Algorithm 5) may also stops at \( \epsilon_i \) before the maximum privacy budget is used up. As the noises drawn before this mechanisms are all correlated and the last noise \( \eta_i \) follows the distribution of \( \text{Lap}(1/\epsilon) \), \( \text{analyzeLoss} \) considers the privacy loss of \( \text{Lap}(1/\epsilon) \) as the increment to the actual privacy loss, while \( \text{estimateLoss} \) considers the privacy loss of \( \text{Lap}(1/\epsilon) \) as the worst case before running \( M_i \).

### 6.3 Overall Privacy Guarantee

We show that the interactive process by DPCLEAN satisfies the privacy constraint specified by the data owner.

**Theorem 10** (Privacy Guarantee of DPCLEAN). The interactive process by DPCLEAN including privacy engine represented by Algorithm 7 and local exploration satisfies \( (B, \delta) \)-differential privacy.

**Proof.** (sketch) When \( i = 1 \), no mechanism has been executed yet, and hence \( B_0 = 0 \), therefore the interactions are definitely \( (B, \delta) \)-differentially private. Suppose \( (M_1, ..., M_{i-1}) \) is \( (B, \delta) \)-differential privacy with an actual privacy loss \( B_{i-1} \leq B \). Given a new mechanism \( M_i \), the privacy analyzer decides to run this mechanism \( M_i \), as \( B_i \leq B \), then the actual privacy loss outputted by \( \text{analyzeLoss} \) after executing \( M_i \) is \( B_i \), which should be smaller than \( B \), and hence no more than \( B \). If the privacy analyzer decides to deny the query, the actual privacy cost remains unchanged to \( B_{i-1} \). By induction, the actual privacy cost is always no more than \( B \). Moreover, the local exploration is a post-processing step, called as post-processing immunity [7], which does not consume additional privacy cost. Therefore, the interactive process by DPCLEAN satisfies \( (B, \delta) \)-differential privacy. \( \square \)
7. EMPIRICAL EVALUATIONS

This section evaluates the usability and effectiveness of DPClean with real data cleaning scenarios on real data sets. Our experiments demonstrate the following:

- Cleaning engineers can accurately tune cleaning workflows by combining their expertise with answers to aggregate query primitives (proposed in Section 4) with tolerable error. (Figure 1 and Figure 2)
- DPClean with Laplace Mechanism based translations (Section 5.1) and advanced composition techniques (Section 6) achieves high cleaning quality under reasonably privacy settings (Figure 5 and Figure 6).
- Data dependent translations (Section 5.2) are shown effective in saving privacy budget and improving cleaning quality. (Figure 7).

7.1 Experiment Setup

Cleaning Tasks & Datasets. The cleaning tasks for entity resolution defined in Section 2.2 including blocking and matching are considered in this evaluation. Two datasets [5] are used: restaurants and citations (Table 1). For each dataset, the cleaning engineer would like to identify entities from two sets of records \(D_1\) and \(D_2\) under the same schema with \(k\) attributes \((A_1, \ldots, A_k)\). The training data \(D_t\) for entity resolution tasks is sampled with the method described in Section 2.2 and half of the training record pairs are positive examples. \(D_t\) is a view over the base tables, and we make it have stability equal to 1 by ensuring that each record in the base tables appears exactly once in \(D_t\). Thus, the sensitivity of LC queries over \(D_t\) have sensitivity 1.

Local Exploration Strategies. We considered 8 humans (real cleaning engineers) and programs that simulate human cleaning (called robots) for evaluating DPClean. Each real cleaning engineer was asked to complete one matching and one blocking task under one setting (a fixed tolerance level \(\alpha\) and dataset citations) using the set of primitive templates provided by the system. Each human cleaner may use different aggregate query primitives and make different choices for type-II actions, resulting in different cleaning strategy. All robot cleaners apply one of the two blocking strategies shown in Figure 3 but vary in the choices made for type-II actions (c1 to c6 in Figure 3), which are sampled from a manually curated set of choices. These two class of strategies are respectively denoted by BS1 which uses only LC primitives and BS2 which uses only LCC and LCT primitives. Similarly, two strategies MS1 (LC only) and MS2 (LCC and LCT) are considered for matching tasks – the primary difference being that the matching rule is a conjunction of similarity predicates, while the blocking rule is a disjunction. We defer details of the user study and robot cleaning strategies to the Appendix A.

Implementation Details. We implemented DPClean using Java-1.8 and run the experiments on a Ubuntu server 16.04 with 16 cores and 64 GB memory. SimMetrics package are used for the set of similarity functions and score computations. The default DPClean applies Laplace Mechanism based translations (Section 5.1) and advanced composition techniques (Section 6). Table 2 lists all the parameter notations used in the experiments. We set the tolerance level \(\alpha = \ell \cdot |D_t|\), where \(\ell \in \{0.01, 0.02, 0.04, \ldots, 0.64\}\), a privacy budget of \(\epsilon = B\), and \(\delta = 3 \cdot 10^{-7}\).

| Table 1: Datasets for entity resolution tasks
| Datasets | \(k\) | \(|D_t \times D_r|\) | \(|D_t|\) | \(|D_r|\) |
| Restaurants | 5 | 176,423 | 100 | 50 |
| Citations | 4 | 168,112,008 | 1000 | 500 |

| Table 2: Parameter Settings
| Parameter | Values |
| Tolerance \(\alpha = \ell \cdot |D_t|\) | \(\ell \in \{0.01, 0.02, 0.04, \ldots, 0.64\}\) |
| Privacy constraint \(B\) (or \(\epsilon\)) | \(\{0.004, 0.008, 0.02, 0.1, 0.2, 0.5\}\) |
| Failing prob \(\beta\) | \(\epsilon^{-15} = 3 \cdot 10^{-7}\) |
| Poking fraction \(f\) | 0.05 |
| # Poking steps \(m\) | 5 |

7.2 Results

7.2.1 Aggregate Queries With No Privacy Constraint

We empirically establish that aggregate primitives (LC, LCC, LCT) are sufficient for engineers to author accurate cleaning workflows, even when the statistics are tolerably noisy (and in the absence of privacy \(B = \infty\)). For a given cleaning task – blocking or matching – we asked 2 humans to complete the cleaning task on citations dataset using query primitives LC, LCC, or LCT at tolerance \(\alpha = \ell \cdot |D_t|\) where \(\ell \in \{0.02, 0.08, 0.32\}\) and implemented 200 robots (100 robots for BS1/MS1 and BS2/MS1 each) that use the private primitives with tolerance \(\alpha = \ell \cdot |D_t|\) from 0 (no noise) to 0.64\(|D_t|\). Figure 1A shows the box plot for the final task quality by humans and robots when no noise is added to the query answers. We observe that humans (red points) finished the task with high quality: recall > 0.95 for blocking and F1-score > 0.9 for matching. The robot cleaners (blue-box plots) also achieved similar high quality as human cleaners. The variance across robot cleaners, attributed to the different choices each cleaner makes for type-II actions, is low since these are chosen from a manually curated set.

Figures 1B and 1C show the task quality with tolerance for blocking and matching respectively, when noises is introduced into query answers. We observe that the quality degrades when error tolerance \(\alpha\) increases from 0.01\(|D_t|\) to 0.64\(|D_t|\) and the quality is above 0.8 for most of the robot cleaners when the tolerance \(\leq 0.08\). This is evidence that cleaning tasks can be completed with good quality even with access to only noisy aggregate queries. The behavior of the human cleaners is similar to that of robot cleaners, providing further evidence of this fact and motivating the feasibility of using differentially private query answers to achieve both task quality and data privacy. Moreover, Figures 1D and 1E also show that the task quality is roughly monotonic in the noise level, i.e. the quality degrades when error tolerance increases from 0.01\(|D_t|\) to 0.64\(|D_t|\).

In Figure 2 we choose one robot cleaner each for blocking strategy BS1 and BS2 and matching strategy MS1 and MS2 from the 200 robots used in Figure 1 and report the task quality of 100 runs of the same robot cleaner for both datasets. This experiment eliminates the variance due to cleaner choices and considers only the randomness from noise. The same monotonicity property is observed that the quality decreases as tolerance increases. As tolerance increases, each query answer is noisier leading to sub-optimal cleaning choices and poorer cleaning quality. We also observe from Figure 2 (left two columns) that the blocking quality for different datasets restaurants and citations are similar at the same tolerance and the same strategy.
This is also true for matching (right two columns in Figure 4) except that the matching task is more sensitive to noise and the strategy. When tolerance increases up to 0.16|Dc|, we see large variance in matching quality for MS2, but MS1 still have reasonably quality at this tolerance level.

### 7.2.2 Varying Privacy Constraint

We now consider blocking and matching tasks using DPClean (with privacy). Given a privacy constraint \( B \) (i.e. \( \epsilon = B, \delta = e^{-15} \)) and a cleaning task, we run programs to simulate local explorations (BS1 for blocking and MS1 for matching) at a fixed tolerance \( \alpha = 0.08|Dc| \). Figure 5 shows the cleaning quality of 100 runs of BS1 and MS1 on restaurants and citations at \( B = \{0.004, 0.008, 0.02, 0.04, 0.1, 0.2, 0.5\} \). We observe that the expected task quality (median and variance) improves as the budget constraint increases and gets stable after reaching \( B \geq 0.5 \) for restaurants and \( B \geq 0.008 \) for citations. Since we fixed tolerance, the privacy loss for each aggregate query is also fixed. Thus, \( B \) directly controls the number of queries DPClean answers before halting. For small \( B \), only a few queries are answered and the cleaning quality is close to random guessing. After \( B \) reaches a certain value, the cleaning engineer has sufficient number of queries answered and be able to obtain good cleaning quality. For the same task, the smallest \( B \) to achieve high quality is higher for restaurants than citations. Since the former has a much smaller training datatize than the latter, given the same \( t \), the bound on the absolute error for queries over restaurants is smaller than that of citations and hence cost more privacy budget per query.

The privacy loss under sequential composition is much higher than under advanced composition. The sequence of mechanisms that results in \( \epsilon = 0.5, \delta = e^{-15} \) for restaurants and \( \{0.008, e^{-15}\} \) for citations has privacy loss of \( \epsilon = 176.25 \) and \( \epsilon = 15.375 \), resp., under sequential composition.

### 7.2.3 Varying Tolerance

This section shows the performance of DPClean at a fixed privacy constraint with varying tolerance for query answers. Figure 6 shows the cleaning quality of 100 runs of BS1 and MS1 on two datasets for tolerance \( \alpha = \frac{t|Dc|}{100} \), where \( t \in \{0.01, 0.02, 0.04, 0.08, 0.16, 0.32, 0.64\} \). Unlike the monotonicity between task quality and tolerance shown in Figure 4 with no privacy (\( B = \infty \)), the quality improves first as tolerance relaxes and then degrades again. This is because when there is a privacy constraint, the cleaning engineers can ask only a limited number of queries. Given a fixed privacy budget \( B = 0.1 \), when tolerance is reasonably small, increasing tolerance allows more queries can be answered. Though answers are noisy, they are still sufficient to help make decisions. As tolerance continues increasing, the answers to queries get very noisy and misleading resulting in the drop in quality, even though many more questions can be answered. Note that the optimal tolerance for both datasets are actually similar in values: \( 0.16|Dc| = 0.16 \cdot 100 = 16 \) for restaurants and \( 0.02|Dc| = 0.02 \cdot 100 = 20 \) for citations. Thus suggest an interesting direction for future work: choosing the optimal error tolerance for different questions in an interactive cleaning workflow.
7.2.4 Exploring Data Dependent Translations

This section shows that data dependent translations (Section 5.2) by switching the LCM translation (Algorithm 2) for LCC queries to LCM with poking (LCMP, Algorithm 4) or LCC with multi-poking (LCMMP, Algorithm 5) can increase cleaning accuracy. Figure 7 shows the average recall achieved by 100 runs of BS2 for blocking on restaurants and citations datasets with fixed tolerance $\alpha = 0.08|D_t|$ and as privacy ($B$) increases from 0.004 to 0.5. We choose BS2 as it uses LCC queries. DPClean attains a higher recall for both datasets at a smaller $B$ for both datasets under LCMP. In restaurants, LCMPMP achieves recall of 1.0 at $B = 0.1$, while LCMMP and LCM achieve the same recall only at $B = 0.2$. In citations, LCMMP achieves 0.75 recall with $B$ as small as 0.004, while LCM requires $B > 0.01$ to achieve the same recall. This is because more LCMMP queries are being answered for the same $B$. The improvements due to LCMP over LCM are not as significant, though we see some improvements in citation. In results not shown, we see similar results for MS2.

8. RELATED WORK

Tools for data cleaning and data integration [15, 29, 27, 3, 10] have been developed for enterprises to unify, clean, and transform their data. These tools are inapplicable for cleaning private data.

PrivateClean [15] is the most relevant prior work for privacy-preserving data cleaning, and differs from DPClean in a number of aspects. First, PrivateClean assumes a different setting, where no active data cleaner is involved. The data owner perturbs the dirty data without cleaning it, while the data analyst who wants to obtain some aggregate queries will clean the perturbed dirty data. However, to clean the private dirty data, the data analyst should have the prior knowledge on the suitable set of transformations required for the given data. In our setting, the set of transformations for cleaning the given data are the output of the exploration process by the cleaning engineer, and hence should not be known in advance. Moreover, all the privacy perturbation techniques in PrivateClean are based on record-level perturbation, which (a) only work well for attributes with small domain sizes, and (b) has asymptotically poorer utility for aggregated queries. Moreover, the randomized response technique used for sampling string attributes in PrivateClean does not satisfy differential privacy – it leaks the active domain of the attribute. In experiments not shown, we reimplement PrivateClean so that it satisfies DP. Its quality was consistently poor (0.5 recall for Blocking and < 0.8 F-1 for Matching on restaurants) for all privacy levels considered in this paper, while DPClean is able to achieve high quality for reasonable privacy levels.

There are general systems for differentially private query answering over relations such as PINQ [23], wPINQ [26] and elastic sensitivity [14]. However, DPClean is the first system where the data analyst specifies accuracy constraints on queries that are translated into privacy losses by the system. There is concurrent work [21] that considers analysts who specify accuracy constraints for machine learning tasks. DPClean differs in two aspects: (a) we use theoretical properties of the mechanisms to perform the error to accuracy translation, and (b) DPClean ensures differential privacy, while the other work ensures a variant called ex-post differential privacy. We can show that ex-post differential privacy may permit mechanisms that leak records in the table with high probability. We omit details due to space constraints.

9. CONCLUSIONS

We proposed DPClean, a system that allows data cleaners interact with and help clean sensitive datasets while ensuring that their interactions satisfy differential privacy. Using experiments with human and simulated data cleaners on blocking and matching tasks, we established that DPClean allows high quality cleaning with a reasonable privacy loss.

DPClean opens many interesting future research directions. In terms of cleaning, we believe the primitives in DPClean can be extended to support other cleaning tasks like schema matching, and related tasks like feature engineering and tuning machine learning workflows. This would also require extending our system to handle relations with multiple tables, constraints like functional dependencies, etc. In terms of privacy, we can extend DPClean’s privacy engine to use other forms of composition (like in the sparse vector technique), and consider translating groups of linear queries together that capture typical queries like histograms, CDFs and sets of range queries. DPClean turns the differentially private algorithm design problem on its head – it minimizes privacy loss given an accuracy constraint. This problem has applications beyond data cleaning and can trigger an entirely new line of research.
10. REFERENCES

[1] Simmetrics. https://github.com/Simmetrics/simmetrics

[2] M. Abadi, A. Chu, I. J. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In SIGSAC, 2016.

[3] Z. Abedjan, J. Morcos, M. N. Gubanov, I. F. Ilyas, M. Stonebraker, P. Papotti, and M. Ouzzani. Dataformer: Leveraging the web for semantic transformations. In CIDR, 2015.

[4] K. Bellare, C. Curing, A. Machanavajjhala, P. Mika, M. Rahukar, and A. Sane. Woo: A scalable and multi-tenant platform for continuous knowledge base synthesis. In Proceedings of Very Large Data Bases (VLDB) - Industrial Track, 2013.

[5] S. Das, A. Doan, P. S. G. C., C. Gokhale, and P. Konda. The magellan data repository. https://sites.google.com/site/anhaidgroup/projects/data

[6] C. Dwork, F. McSherry, K. Nissim, and A. D. Smith. Calibrating noise to sensitivity in private data analysis. In TCC, 2006.

[7] C. Dwork and A. Roth. The algorithmic foundations of differential privacy. Found. Trends Theor. Comput. Sci., 2014.

[8] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. IEEE Trans. Knowl. Data Eng., 2007.

[9] U. Erlingsson, V. Pihur, and A. Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In CCS, 2014.

[10] H. Galhardas, D. Florescu, D. E. Shasha, and E. Simon. AJAX: an extensible data cleaning tool. In SIGMOD, 2000.

[11] A. Greenberg. Apple’s ‘differential privacy’ is about collecting your data—but not your data. Wired, 2016.

[12] S. Haney, A. Machanavajjhala, J. Abowd, M. Graham, M. Kutzbach, and L. Vilhuber. Utility cost of formal privacy for releasing national employer-employee statistics. In SIGMOD, 2017.

[13] M. Hay, A. Machanavajjhala, G. Miklau, Y. Chen, and D. Zhang. Principled evaluation of differentially private algorithms using dpbench. In SIGMOD, 2016.

[14] N. M. Johnson, J. P. Near, and D. X. Song. Practical differential privacy for SQL queries using elastic sensitivity. CoRR, abs/1706.09479, 2017.

[15] S. Kandel, A. Paepcke, J. M. Hellerstein, and J. Heer. Wrangler: interactive visual specification of data transformation scripts. In CHI, 2011.

[16] P. Konda, S. Das, P. S. G. C., A. Doan, A. Ardalan, J. R. Ballard, H. Li, F. Panahi, H. Zhang, J. F. Naughton, S. Prasad, G. Krishnan, R. Deep, and V. Raghavendra. Magellan: Toward building entity matching management systems. PVLDB, 2016.

[17] F. Koufogiannis, S. Han, and G. J. Pappas. Gradual release of sensitive data under differential privacy. CoRR, abs/1504.00429, 2015.

[18] S. Krishnan, J. Wang, M. J. Franklin, K. Goldberg, and T. Kraska. Privateclean: Data cleaning and differential privacy. In SIGMOD, 2016.

[19] S. Krishnan, J. Wang, E. Wu, M. J. Franklin, and K. Goldberg. Activeclean: Interactive data cleaning for statistical modeling. PVLDB, 2016.

[20] C. Li, M. Hay, V. Rastogi, G. Miklau, and A. McGregor. Optimizing linear counting queries under differential privacy. In PODS, 2010.

[21] K. Ligett, S. Neel, A. Roth, B. Waggoner, and Z. S. Wu. Accuracy first: Selecting a differential privacy level for accuracy-constrained erm. In NIPS, 2017.

[22] A. Machanavajjhala, D. Kifer, J. Abowd, J. Gehrke, and L. Vilhuber. Privacy: Theory meets practice on the map. In IEEE ICDE, 2008.

[23] F. D. McSherry. Privacy integrated queries: An extensible platform for privacy-preserving data analysis. In SIGMOD, 2009.

[24] I. Mironov. Renyi differential privacy. CoRR, abs/1702.07476, 2017.

[25] M. Naveed, S. Kamara, and C. V. Wright. Inference attacks on property-preserving encrypted databases. In CCS, 2015.

[26] D. Proserpio, S. Goldberg, and F. McSherry. Calibrating data to sensitivity in private data analysis: A platform for differentially-private analysis of weighted datasets. Proc. VLDB Endow., 2014.

[27] V. Raman and J. M. Hellerstein. Potter’s wheel: An interactive data cleaning system. In VLDB, 2001.

[28] S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In KDD, 2002.

[29] M. Stonebraker, D. Bruckner, I. F. Ilyas, G. Beskales, M. Cherniack, S. B. Zdonik, A. Pagan, and S. Xu. Data curation at scale: The data tamzer system. In CIDR, 2013.

[30] L. Vilhuber, I. M. Schmutte, and J. M. Abowd. Proceedings from the 2016 NSF–Sloan workshop on practical privacy, Jan 2017.

[31] J. Wang, S. Krishnan, M. J. Franklin, K. Goldberg, T. Kraska, and T. Milo. A sample-and-clean framework for fast and accurate query processing on dirty data. In SIGMOD, 2014.

APPENDIX

A. ADDITIONAL EVALUATION DETAILS

A.1 User Test

This user test was conducted at the University of Waterloo by asking data cleaning experts to complete data cleaning tasks. In this test, users were first asked to conduct a blocking or matching task on restaurants dataset with direct access to the data. If a user finished the task with high quality, then this user would be asked to continue another blocking or matching task using aggregate queries on another training data set D_i generated from citations dataset. There were 8 users involved in the cleaning tasks with aggregate queries. Each user was asked to complete one matching and one blocking task at a given and fixed tolerance α = t|D_i|, where t ∈ {0, 0.02, 0.08, 0.32} and |D_i| = 1000. With the set of primitive templates provided by DPCLEAN, a user constructed a sequence of primitive

\[ \alpha = 1000. \]

1 This test has been reviewed and received ethics clearance through a University of Waterloo Research Ethics Committee (ORE#226655).
Figure 8: Two strategies for matching with different queries from DPClean. Then DPClean answered these queries within the given query tolerance and sent the query answer back to the user. In these tests, users had no privacy constraint $B = \infty$, and hence they could ask unlimited queries until they were satisfied with the exploration results.

### A.2 Strategy Instances for Matching Task

Consider the exploration space $S$ for a matcher as all possible conjunctions of predicates in the form of $p_1 \land \cdots \land p_j$. This space can also be expressed as the power set of the predicates domain, i.e., $S = \mathcal{P}(\mathcal{A} \times \mathcal{T} \times \mathcal{S} \times \Theta)$. Similar to the interactive process for blocking (Example 1 and 2), the cleaning engineer for matching will first select a small number of candidate predicates after profiling the data. However, the matching task has a different objective and exploration space. Adding a new predicate $p$ to the conjunction formed by the predicates in the current output $O$ will prune away both some matches and non-matches. The cleaner aims to limit the number of matches missed and maximize the number of non-matches pruned by adding this new predicate. Hence, the cleaning engineer typically selects a predicate $p$ if $p \lor O$ missed few matches but pruned many new non-matches with respect to $O$, where $O$ is the set of predicates already selected. Two strategy instances are shown in Figure 8 which use different primitives. These two instances have similar profiling and picking predicates as blocking strategies in Figure 9 and hence are not shown, but different in $c5b, q5a, q5b, q5a, q5b$ for different objectives.

### A.3 Modeling Cleaning Engineers

This section describes the programs simulating local exploration (called robot cleaners). Consider the blocking strategy 1 shown in Figure 9a, the corresponding cleaner model encodes the space of all the parameters involved in the cleaner’s decisions $c1$-6. Table 3 summarizes the space of all the parameters for $c1$-6 in blocking strategy.

1. From $c1$ to $c4$, the program chooses (i) a subset of attributes $x_1$ of size ranging from 2 up to the total number of attributes $|\mathcal{A}|$, (ii) a subset of transformations $x_2$ from $\mathcal{T} = \{2grams, 3grams, \text{SpaceTokenization}\}$, (iii) a subset of similarity functions $x_3$ from $\mathcal{S} = \{\text{Edit, SmithWater, Jaro, Cosine, Jaccard, Overlap, Diff}\}$, and (iv) $x_6$ thresholds from the range of $[x_4, x_5]$, where $x_4 \in (0, 0.5)$, $x_5 \in (0.5, 1)$, and $x_6 \in [2, 3, 4, 5, 6]$. The cross product of these choices forms a set of predicates $P$, and $c5a$ picks an ordering $x_7$, one of the permutation of $P$. In $c5b$, the model sets the criterion for pruning or keeping a predicate $p$ from the top list of $P$. In particular, the model sets $x_8$ and $x_9$ as the minimum fraction of the remaining matches caught and the maximum fraction of the remaining non-matches caught by $p \lor O$ respectively, where $x_8 \in [0.2, 0.5]$ and $x_9 \in [0.1, 0.2]$. These values are reset as $x_8 = x_8/x_{10}$ and $x_9 = x_9 x_{10}$ where $x_{10} \in [2, 3]$ if all predicates have been checked but $O = \emptyset$. In $c6$, the model considers three possible styles of cleaners on trusting the noisy answers: neutral style corresponds to trust the noisy answers; for optimistic (pessimistic) style, the cleaner trusts the values by adding (subtracting) $\alpha/5$ to (from) the noisy answers. If these criterion are met and the blocking cost over training data $D_t$ is less than a fixed cutoff threshold (e.g., a hardware constraint, we set 550 and 55 for Citations and Restaurants datasets respectively). An instance of all variables $C = \{x_1, \ldots, x_{11}\}$ in Table 3 forms a robot cleaner. The model for other strategies is similarly constructed.

### A.4 Optimal Error Tolerance

Section 7.2.3 shows that there exists an optimal fixed tolerance to achieve the highest possible quality at a given privacy constraint $B = 0.1$. We further investigate this optimal tolerance by reporting the empirical best tolerance for different...
ent privacy constraints $B$. For each privacy constraint $B$, we ran the same local exploration programs over different tolerance values from $0.01|D_t|$ to $0.64|D_t|$ and noted down the tolerance that achieved the highest average in quality. Figure 9 plots the means of the blocking or matching quality of 100 runs of a robot cleaner (BS1 or MS1) on two datasets respectively by setting the tolerance as the optimal value (opt) or a fixed value $t|D_t|$ where $t \in \{0.02, 0.08, 0.32\}$. The quality when setting tolerance at the optimal value is definitely better than all the other fixed tolerance values. Moreover, the optimal value at each privacy constraint is also shown in the plot. We observe that when privacy budget $B$ increases, the optimal tolerance value decreases in general. This is because a query with smaller tolerance requires higher budget. Given a more relaxed privacy constraint (larger $B$), then more questions at a smaller tolerance can be answered, which leads to good quality at smaller tolerance. The first optimal tolerance on restaurants dataset is random because none questions can be answered when $B = 0.04$. Moreover, cleaning with a larger tolerance converges faster than a smaller tolerance (e.g., $0.08|D_t|$ vs. $0.02|D_t|$) as privacy constraint increases. All these interesting results lead to an important direction for future work on how to choose optimal error tolerance for queries in an interactive cleaning workflow.

Table 4: PrivateClean Cost on Restaurants, $B = 0.1$

| Tolerance | $p$  | GRR cost / 2-gram | #pair sampled |
|-----------|-----|-------------------|---------------|
| 0.01      | 0.902 | 0.282             | 0             |
| 0.02      | 0.951 | 0.144             | 0             |
| 0.04      | 0.976 | 0.073             | 0             |
| 0.08      | 0.998 | 0.037             | 0             |
| 0.16      | 0.994 | 0.018             | 0             |
| 0.32      | 0.997 | 0.009             | 0             |
| 0.64      | 0.998 | 0.005             | 0             |

A.5 Comparisons with Baseline

In this section, we compare DPClean with prior work PrivateClean [18]. PrivateClean uses generalized randomized response (GRR) to sanitize the input training data $D_t$. The same set of queries from the cleaning strategies are answered directly on the sanitized data, unlike DPClean perturbs the answers of a query. For each record in $D_t$, a random value is sampled for each attribute of this record. To correctly ensure differential privacy, the sample space should be the entire domain including both active and inactive domain (see Section 3 of this attribute. However, given a large domain such as strings of unbounded length, the probability of sampling the true value of that record is very small and hence introduce very large error. To limit the domain, for each string attribute, we transform the first 10 characters of the string values into 5 non-overlapping 2grams. For each 2gram, we apply GRR to sample a value, and then concatenate these 5 values into a new string as the sanitized data.

Given the error tolerance for a linear counting query, we find the sampling probability to achieve this error tolerance. For instance, when the tolerance is $\alpha = 0.01|D_t|$, the corresponding probability to sampling the true 2-gram value should be above $0.902$ (Appendix E of PrivateClean [18]), and hence requires at least a budget of 0.282 (Lemma 1 of PrivateClean [18]). We reported the privacy cost to achieve tolerance values from $0.01|D_t|$ to $0.64|D_t|$ for a 2gram on the restaurants dataset by the above GRR approach and the number of pairs of true 2 grams can be sampled given a total budget of $B = 0.1$ in Table 4. Given the total privacy budget $B = 0.1$, no true training pair nor even single attribute is sampled in the output. Therefore, the cleaning process becomes random guessed and the comparison between PrivateClean and DPClean in shown in Figure 10 shows that PrivateClean has a poor cleaning quality due to random guessing, but DPClean achieves good quality with reasonable cost $B > 0.1$. 

Figure 9: Performance of DPClean for blocking (BS1) and matching (MS1) tasks on restaurants and citations with increasing privacy constraint $B$ at (i) optimal tolerances (opt) and (ii) several fixed tolerances $\alpha = t|D_t|$, where $t \in \{0.02, 0.08, 0.32\}$: the optimal value for the tolerance decreases as privacy budget increases.

Figure 10: Performance of PrivateClean and DPClean for blocking (BS1) and matching (MS1) tasks on restaurants and citations with increasing privacy budget $B$ at a fixed tolerance $0.08|D_t|$: DPClean can achieve high cleaning quality at a much smaller privacy budget.
B. THEOREMS AND PROOFS

B.1 Laplace Comparison Mechanism

We show Theorem 2 that given a linear counting query with condition \( q_{\phi} \leq c \) for any \( D \in D \), Laplace comparison mechanism (Algorithm 2) denoted by \( LCM_{\alpha, \beta}^{c, \phi}(\cdot) \), can achieve \((\alpha, \beta)\)-LCC tolerance with minimal \( \epsilon \)-differential privacy cost, where \( \epsilon = 1/b = \frac{\ln(1/(2\beta))}{\alpha} \).

Proof. Set \( b = \frac{1}{\epsilon} \). The probability to fail the tolerance requirement is (i) when \( q_{\phi}(D) < c - \alpha \),

\[
\Pr[LCM_{\alpha, \beta}^{c, \phi}(D) = \text{True} \mid q_{\phi}(D) < c - \alpha] = \Pr[q_{\phi}(D) - c + \eta > 0 \mid q_{\phi}(D) - c + \alpha < 0] < \Pr[\eta > \alpha] = e^{-\alpha/b}/2 \leq \beta
\]

and (ii) when \( q_{\phi}(D) > c + \alpha \),

\[
\Pr[LCM_{\alpha, \beta}^{c, \phi}(D) = \text{False} \mid q_{\phi}(D) > c + \alpha] = \Pr[q_{\phi}(D) - c + \eta < 0 \mid q_{\phi}(D) - c - \alpha > 0] < \Pr[\eta < -\alpha] = e^{-\alpha/b}/2 \leq \beta
\]

Hence, LCM achieves \( \alpha \)-LCC with probability \( 1 - \beta \). This mechanism also satisfies 1-differential privacy by post-processing. Hence, setting \( b = \frac{1}{\epsilon} \) gives the least cost.

B.2 Laplace Top-\( k \) Mechanism

We would like to show the privacy and tolerance requirement of Laplace Top-\( k \) Mechanism stated in Theorem 3.

We first show the proof for tolerance requirement: given a top-\( k \) linear counting query, \( q_{\phi_{i_1}, \ldots, \phi_{i_L}} \), for any table \( D \in D \), Laplace top-\( k \) mechanism (Algorithm 3) denoted by \( LTM_{\alpha, \beta}^{\phi_{i_1}, \ldots, \phi_{i_L}}(\cdot) \), can achieve \((\alpha, \beta)\)-LCT tolerance.

Proof. The output of Algorithm 3 are \( \{q_{\phi_{i_1}}, \ldots, q_{\phi_{i_L}}\} \) which have the highest noisy counts. Suppose their noisy counts are in the order of \( \hat{x}_{i_1} \geq \hat{x}_{i_2} \geq \cdots \geq \hat{x}_{i_k} \). As \( c_k \) is the true answer to the kth largest linear counting query of all \( L \) linear counting queries, the largest answer to any \( L - k + 1 \) linear counting queries should be no less than \( c_k \). Hence, we have

\[
\max_{j \in \{i_1, \ldots, i_L\}} q_{\phi_j}(D) \geq c_k.
\]

Similarly, the smallest answer to any \( k \) linear counting queries should be no greater than \( c_k \). Hence, we have

\[
\min_{j \in \{i_1, \ldots, i_k\}} q_{\phi_j}(D) \leq c_k.
\]

First we would like to show that by setting \( b \leq \frac{\alpha}{2(\ln L + \ln(1/k/\beta))} \) if a count \( q_{\phi}(D) \) is too small, \( \phi \) will be included in the answer \( a \) with small probability:

\[
\Pr[\phi \in a \mid q_{\phi}(D) < c_k - \alpha] = \Pr[q_{\phi}(D) + \eta \geq \max_{j \in \{i_{k+1}, \ldots, i_L\}} (q_{\phi_j}(D) + \eta_j) \mid q_{\phi}(D) < c_k - \alpha] \leq \Pr[q_{\phi}(D) + \eta \geq \max_{j \in \{i_{k+1}, \ldots, i_L\}} (c_k + \eta_j) \mid q_{\phi}(D) < c_k - \alpha] = \Pr[q_{\phi}(D) - c_k \geq \max_{j \in \{i_{k+1}, \ldots, i_L\}} \eta_j - \eta \mid q_{\phi}(D) < c_k - \alpha] \leq \Pr[\max_{j \in \{i_{k+1}, \ldots, i_L\}} \eta_j < -\alpha/2] + \Pr[\eta > \alpha/2] \leq (L - k + 1) e^{-\frac{\beta}{2}}/2 + e^{-\frac{\beta}{2}} < L e^{-\frac{\beta}{2}} \leq \beta/k
\]

Hence, every selected \( \phi \in a \) has \( q_{\phi}(D) > c - \alpha \) with probability \( 1 - \beta \) for \( |a| = k \).

Then, we would like to show that if a count \( q_{\phi}(D) \) is sufficiently large, then the probability of missing it in the output is small.

\[
\Pr[\phi \notin a \mid q_{\phi}(D) > c_k + \alpha] \leq \Pr[q_{\phi}(D) + \eta < \min_{j \in \{i_1, \ldots, i_k\}} (q_{\phi_j}(D) + \eta_j) \mid q_{\phi}(D) > c_k + \alpha] \leq \Pr[q_{\phi}(D) + \eta < \min_{j \in \{i_1, \ldots, i_k\}} (c_k + \eta_j) \mid q_{\phi}(D) > c_k + \alpha] = \Pr[q_{\phi}(D) - c_k < \min_{j \in \{i_1, \ldots, i_k\}} \eta_j - \eta \mid q_{\phi}(D) > c_k + \alpha] \leq \Pr[\min_{j \in \{i_1, \ldots, i_k\}} \eta_j > -\alpha/2] + \Pr[\eta < -\alpha/2] \leq ke^{-\frac{\beta}{2}}/2 + e^{-\frac{\beta}{2}} < ke^{-\frac{\beta}{2}} \leq \beta/L
\]

Hence, all boolean formulae \( \phi \in \{\phi_1, \ldots, \phi_L\} \) with \( q_{\phi}(D) > c + \alpha \) are selected with probability \( 1 - \beta \).
For the other direction, given these fixed noises, for choices of (η₁,…,ηₖ) and (ii) when ϵ∈Algorithm 4), denoted by LCMP, (25)

Hence, LCMP achieves composition of differential privacy (Theorem 8), LCMP satisfies (ϵ,β)-LCC tolerance with ϵ-differential privacy cost, where ϵ = \(\frac{\ln(m(1/β))}{α}\).

**Proof.** (sketch) The NoiseDown Algorithm shown in Algorithm 6 correlates the new noise ηnew with noise η\(_0\) from the previous iteration. In this way, the composition of the first i + 1 iterations is ϵ\(_i\), and the noise added in the i + 1th iteration is equivalent to a noise generated with Laplace distribution with privacy budget ϵ\(_i\) and the first i iterations also satisfy ϵ\(_i\)-DP for i = 0, 1,…, m – 1 (Theorem 9 [17]). This approach allows data cleaner to learn the query answer with a gradual relaxation of privacy cost. At ith iteration, the probability to fail is β/m. Hence, when outputting an answer at ith iteration, the probability to fail the requirement is iβ/m < β. The proof for the tolerance is similar to the proof of Theorem 6.