Databases and information management

Lecture 15

Piotr Syga

2023-01-23

Why do we need privacy?

- merging information form various sources
- user tracking
- metadata

Examples of data linking

- Thelma Arnold vel AOL Searcher No. 4417749
- Dane demograficzne – P. Golle
- Privacy Risks in Recommender Systems – Ramakrishnan et al.
- Netflix experiment – Narayanan, Shmatikov
- IMBD + medical data – Dwork et al.

Countermeasures

What can we do to preserve data?

- encryption
  increases the processing time, different access rules and processing of the data in an inner and outer environment
- artificial data
  requires some way of distinguishing between real and artificial data which is easy for the owner of the database, but hard for others, the size of data is influenced
- distorting of the data
  requires a proper amount of distortion, can lead to data leakage

Idea

- We make the database available for statistical purposes.
- We allow some aggregate queries (e.g., sum).
- The results we disclose should reflect the actual data.
- We disturb the data, for privacy purposes.

Definition

We say that a function $\mathcal{A}$ provides $\varepsilon-$differential privacy if for all datasets D and D' that differ on at most 1 rowand all $A \subseteq \mathrm{Range}(\mathcal{A})$ we have:

$$\Pr[\mathcal{A}(D) \in A] \leq \exp(\varepsilon) \cdot \Pr[\mathcal{A}(D') \in A]$$

Definition

We say that a function $\mathcal{A}$ provides $(\varepsilon, \delta)-$differential privacy if for all datasets D and D' that differ on at most 1 rowand all $A \subseteq \mathrm{Range}(\mathcal{A})$ we have:

$$\Pr[\mathcal{A}(D) \in A] \leq \exp(\varepsilon) \cdot \Pr[\mathcal{A}(D') \in A] + \delta$$

Laplace Distribution

If a random variable Y is Laplace distributed with parameters μ and b: $Y \sim \text{Laplace}(\mu, b)$ then

$$\text{PDF} \quad f_{\mu,b}(x) = \frac{1}{2b} \exp\left(-\frac{|x-\mu|}{b}\right)$$

$$\text{expected value} \quad \mu$$

$$\text{variance} \quad 2b^2$$

Numerical data: Laplcian mechanism

- Idea: whenever we return some value, we add a laplacian perturbation to the result
- Sensitivity $\Delta f = \max_{D,\, D'} |\mathcal{A}(D) - \mathcal{A}(D')|$
- After retrieving the exact result we report it with $Y \sim \mathrm{Laplace}(0, \frac{\Delta f}{\varepsilon})$ added.

Numerical data: Laplcian mechanism

- Idea: whenever we return some value, we add a laplacian perturbation to the result
- Sensitivity $\Delta f = \max_{D,\, D'} |\mathcal{A}(D) - \mathcal{A}(D')|$
- After retrieving the exact result we report it with $Y \sim \mathrm{Laplace}(0, \frac{\Delta f}{\varepsilon})$ added.

What happens if we repeat the query multiple times?
For n queries we distribute out a 'joined privacy budget': $\varepsilon = \sum_{i=1}^{n} \varepsilon_i$.

Example

- Let us have a database containing some information about employees, including their earning.
  For simplicity of presentation let us assume that there are 25 employees.
- The earnings for EID 1 to 25 are:
  32563, 68654, 65873, 75366, 55391, 46721, 49931, 64273, 60704, 49321, 57317, 63200, 50170, 73106, 45311, 113500, 38642, 57410, 49210, 60040, 59247, 58210, 50367, 30245, 49874
- Let us calculate the sensitivity $\Delta f = 83255$ and assume $\varepsilon = 0.15$

Example

- Let us query for the wage of the 4th employee, we get 75366+y, probing from Laplace distribution we might get (after rounding) 34230, hence we return 109596, which seems plausible.
- Next, let us query about the 6th employee: 46721+107717=154438 – a lot above the *true maximum*, and we can notice that calculating average earnings might be troublesome, but still reasonable.
- Let us query once more, e.g. EID=14, we get 73106+(-123160)=-50054, which not only does not look plausible, but also shows a threat – is it possible to query so many times that we reduce the influence of Laplacian distortion.

Example

- Let us assume, we query the same EID all the time, e.g. EID=1, we can generate 10 different distortions for 10 different queries: [41671.7392 41115.5423 986542.1257 -304956.2290 1978096.4816 -513252.5031 -113987.2574 1991133.4357 303025.9784 -1306127.0472], which averages out to 310326.2266
  The next query batch gives us completely different values, [-414492.8949 -1587658.9128 -173125.3048 638654.0050 526088.9566 124467.2006 -311528.6786 -577627.5741 -227539.2100 456071.7279], with average -154669.0685
- Even increasing batch size to larger numbers, e.g. 100, gives us the following aggregates: -146417.5938, 53160.4950, -36218.50, . . .
- Going further to 1000 (remembering we only have 25 employees in the database): -8445.9506, -11200.1687, 8037.3145, 46945.7714, . . .

So privacy does not seem to be the problem, then what is?
The usability of the data, how to get the data that can be used for statistical purposes.

Non-numerical data and preferences

- Previously we just added some (hopefully reasonable) distortion to the result
- What about queries about preferences, e.g., favorite pizza?
- How to distort data about the order?
- If we have multiple possibilities, we can use the exponential mechanism to sample form the list with some distorted probability, hence changed order.

Non-numerical data and preferences

- Define the function to be privatized (f) and the set of possible outputs (S).
- Define a utility function (u) that assigns a non-negative real value to each element of S.
- Select a privacy budget (ε) which represents the maximum amount of information that can be leaked.
- Generate a random variable (x) from the exponential distribution with mean $\frac{1}{\varepsilon}$.
- Compute the probability of each element $s \in S$:
  $\Pr[s] = exp(\varepsilon \cdot u(s) \cdot x)/Z$, where Z is a normalization factor.
- Sample an output s from S according to the computed probabilities.

Literature

- Differential Privacy
- Sub-linear queries
- Query reduction
- Publication of the data from large, public databases
- Limits with data publication
- Privacy book
- Legal aspect – GDPR

Idea

- A programming language with strong theoretical basis (Relation algebra).
- SQL is used to create, modify, and query databases, and it allows users to insert, update, and retrieve data from a database.
- SQL statements are divided into two categories: data definition language (DDL) and data manipulation language (DML).
- SQL supports data control language (DCL) statements, which are used to control access to the data stored in a database. Examples include GRANT and REVOKE.
- Basic Select query consists of 3 clauses `SELECT` <what> `FROM` <what source> `WHERE` <which conditions are met>.
- All types in SQL allow comparison and have set order.
- The query may be expanded by `JOIN`s, subqueries, using `VIEW`s or `CTE`s.
- Transactions are supported both in single-node and distributed environment (2PC).

Dynamic database objects

- SQL supports procedures, that may have input and output parameters (arbitrary number) and can modify a section of memory or even a structure of the database.
- SQL functions returns always a single, atomic value.
- A function can be used in a SELECT statement, while a procedure cannot. Functions can be used to return a value that can be used in a `SELECT` statement, while procedures are typically executed separately.
- A function can be used in a check constraint, while a procedure cannot.
- A function is executed with the rights of the calling user, while a procedure is executed with the rights of the owner of the procedure.

Dynamic database objects

- Functions are not allowed to use transaction control statements such as COMMIT or ROLLBACK, while procedures are.
- Procedures may include `PREPARE STATEMENT`, whereas functions cannot.
- SQL supports triggers, as a variant of automatically executed functions, that are invoked after a conditioning action is met.
- Keywords for manipulation of such structures: `CREATE`, `DROP`, `ALTER`.

Functional dependence

Basic notions

Let U be a finite subset of all attributes of some relation R and X, $Y \subseteq U$. Then, if the values of attributes in X determine the values of attributes in Y we say that Y *depends functionally* on X. Note that in such case $\pi_{X,Y}(R)$ is a function. Additionally, if $Y \subseteq X$ we say that Functional dependence $X \to Y$ is a trivial one.

Keys

Definitions

Let R be a relation with attributes' set U and a set of functional dependencies $\mathcal{F}$. A *key* of R is a set $K \subseteq U$, s.t.$\{K\}^+ = U$. Moreover, if there is no subset $P \subset K$ such that $\{P\}^+ = U$, then K is a *minimal key* (or a candidate key). One of the minimal keys is selected to be a primary key (it does not have to be the one with the fewest attributes).

Primary attributes

If K is a candidate key in R with set of attributes U, then all attributes $X \in K$ is a *key attribute* (also referred to as *primary attribute*). Let $\mathcal{K}$ be a set of all key attributes, then all attributes from $U \setminus \mathcal{K}$ are non-prime attributes.

1NF, 2NF, 3NF

First normal form

Relation R is in 1NF **iff** every attribute is atomic and there are no duplicate rows (i.e., there is a key).

Second normal form

Relation R is in 2NF **iff** R is in 1NF and no non-prime attribute depends functionally on a proper subset of a candidate key.

Third normal form

Relation R is in 3NF **iff** R is in 2NF and all non-prime attributes depend only on a complete candidate key.

Mnemonic

A Simple Guide to 5 Normal Forms in Relational DB Theory. W. Kent

Every non-key attribute must provide
a fact about the key (1NF),
the whole key (2NF),
and nothing but the key (3NF).
(So help me Codd.)

Characteristics

- MongoDB is a document-oriented, NoSQL database that uses JSON-like documents to store data.
- MongoDB allows for high scalability and performance by using a distributed architecture.
- MongoDB supports automatic sharding, which enables horizontal scaling and allows for large amounts of data to be spread across multiple servers.
- MongoDB supports rich and flexible data models, allowing for nested and denormalized data (arrays, embedded documents, . . .).
- MongoDB supports ACID transactions, providing support for multi-document transactions.

Characteristics

- MongoDB supports backup and recovery through the use of snapshots and backups.
- MongoDB supports a variety of programming languages and has official driver support for many popular languages.
- MongoDB supports a variety of operational and management tools, including the MongoDB Management Service (MMS) and MongoDB Compass.
- MongoDB supports a variety of cloud providers, including AWS, GCP, and Azure.
- MongoDB supports highly customizable aggregation pipeline.

Characteristics

- Cassandra is a distributed, NoSQL database that is designed to handle large amounts of data across many commodity servers.
- Cassandra is a highly available and fault-tolerant database, with no single point of failure. Utilization of ring architecture allows data to be spread and easily localized.
- Cassandra supports automatic data partitioning and replication, allowing for horizontal scalability.
- Cassandra supports tunable consistency levels, allowing for flexible trade-offs between consistency and availability.
- Cassandra supports powerful query capabilities through the use of the Cassandra Query Language (CQL). It uses Query First approach.
- The tables are organized using partition key and, optionally, aggregation columns.

Characteristics

- Cassandra supports rich data modeling, including support for collections, user-defined types, and lightweight transactions.
- Cassandra supports secondary indexes, full-text search, and geospatial indexing.
- Cassandra supports real-time analytics and reporting through the use of the Apache Spark Connector for Cassandra.
- Cassandra supports a variety of programming languages and has official driver support for many popular languages.
- Cassandra supports a variety of deployment options, including on-premises, cloud-based, and hybrid deployments.
- Cassandra supports a variety of operational and management tools, including the Apache Cassandra Management Service (CMS) and the DataStax OpsCenter.
- Cassandra supports a variety of monitoring and performance optimization tools, including the DataStax Performance Analyzer.