Cleaning by Samples:
Aggregate Queries

Data Cleaning & Integration
CompSci 590.01 Spring 2017

Some slides were based on:
Jinglin Peng’s presentation on the SIGMOD 2014 SampleClean paper
Biggest take-away points

(For Jun:)

• No need to clean the whole dataset to get a “good enough” answer

• A remarkably useful observation: cleaning a random sample = taking a random sample from the clean database (under certain assumptions)
  • Many results on sample-based approximate query processing (SAQP) can then be leveraged

• One can also think about sampling errors in data (instead of sampling data), which can sometimes give better estimates (tighter confidence intervals)
Motivation

<table>
<thead>
<tr>
<th>Solutions</th>
<th>Clean Time</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Cleaning</td>
<td>😄</td>
<td>😭</td>
</tr>
<tr>
<td>Full Cleaning</td>
<td>😭</td>
<td>😊</td>
</tr>
</tbody>
</table>

Comparison of two solutions

TB, PB data ⟷ GB even MB sample
Motivation
Foundation: CLT

(Central limit theorem)

Let $X_1, X_2, \ldots, X_K$ be sequence of i.i.d. random variables with $E[X_i] = \mu$ and finite $V[X_i] = \sigma^2$; then, their mean (“sample average”) closely follows a normal distribution $N(\mu, \sigma^2/K)$ for large enough $K$.
Applying CLT to AVG query

- Say we want `SELECT AVG(A) FROM R;`
  - Let \( \mu \) and \( \sigma^2 \) denote the true mean and variance of \( R(A) \)
- Take a random sample of size \( K \) from \( R \), just compute AVG with that
- Under the large-sample assumption, we can think of sampled tuples as i.i.d. drawn from \( R(A) \)
- So by CLT, sample mean \( \hat{\mu} \sim N(\mu, \sigma^2/K) \)
- Replacing \( \sigma^2 \) with a sample variance estimator \( \hat{\sigma}^2 \) and accounting for finite population, we can derive a confidence interval: \( \hat{\mu} \pm \lambda \sqrt{\frac{\hat{\sigma}}{K}} \)
Alternative: Hoeffding’s

Let $X_1, X_2, \ldots, X_K$ be a sequence of independent random variables in $[0,1]$ with $E[\bar{X}] = \mu$ where $\bar{X} = (X_1 + \cdots + X_K)/K$; then $P[|\bar{X} - \mu| \geq t] \leq 2e^{-2Kt^2}$

- You can “scale” $[0,1]$ to another range as needed
- More widely applicable—no need for the large-sample assumption
- But resulting confidence intervals are typically more conservative than those based on CLT
- See application to online aggregation: Hellerstein, Haas, Wang, SIGMOD 1997
Generalization

• Given a query with selection (filter) predicate $\theta$ and aggregating attribute $A$, define $\phi(t)$ for each tuple
  - **COUNT:** $\phi(t) = \theta(t) \cdot N$
  - **SUM:** $\phi(t) = \theta(t) \cdot N \cdot t[A]$
  - **AVG:** $\phi(t) = \theta(t) \cdot \left( \frac{K}{K_\theta} \right) \cdot t[A]$

  where $N$ is total # of tuples in the table and $K_\theta \leq K$ is # of tuples in the sample that satisfy $\theta$

• Then estimate result as \( \frac{1}{K} \Sigma_{t \in S} \phi(t) \)

• **GROUP BY** can be regarded one query per group
Example

Query: sum of citations of papers published after 2007
\[ \phi(t) = \theta(t) \cdot N \cdot t[A] \]

<table>
<thead>
<tr>
<th>id</th>
<th>title</th>
<th>pub_year</th>
<th>citation</th>
<th>predicate</th>
<th>( \phi )</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>CrowDB</td>
<td>2011</td>
<td>144</td>
<td>True</td>
<td>144*6</td>
</tr>
<tr>
<td>t2</td>
<td>TinyDB</td>
<td>2005</td>
<td>1569</td>
<td>False</td>
<td>0</td>
</tr>
<tr>
<td>t3</td>
<td>YFilter</td>
<td>2002</td>
<td>298</td>
<td>False</td>
<td>0</td>
</tr>
<tr>
<td>t4</td>
<td>Aqua</td>
<td>1999</td>
<td>106</td>
<td>False</td>
<td>0</td>
</tr>
<tr>
<td>t5</td>
<td>DataSpace</td>
<td>2008</td>
<td>107</td>
<td>True</td>
<td>107*6</td>
</tr>
<tr>
<td>t6</td>
<td>CrowER</td>
<td>2012</td>
<td>34</td>
<td>True</td>
<td>34*6</td>
</tr>
</tbody>
</table>

Full data

Real result

\[
\text{mean}(144*6+0+0+0+107*6+34*6)
\]

Estimation

\[
\text{mean}(0+107*6+34*6)
\]

Uncertainty

\[
1.96 \sqrt{\frac{\text{var}(0,107*6,34*6)}{3}}
\]
Challenge: what if data is dirty?

- Three types of errors are modeled
- Example: average # citations for papers published after 2000

**Dirty Data**

<table>
<thead>
<tr>
<th>P</th>
<th>id</th>
<th>title</th>
<th>pub_year</th>
<th>citation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/6</td>
<td>t1</td>
<td>CrowDB</td>
<td>11</td>
<td>144</td>
</tr>
<tr>
<td>1/6</td>
<td>t2</td>
<td>TinyDB</td>
<td>2005</td>
<td>1</td>
</tr>
<tr>
<td>1/6</td>
<td>t3</td>
<td>YFilter</td>
<td>2002</td>
<td>298</td>
</tr>
<tr>
<td>1/6</td>
<td>t4</td>
<td>Aqua</td>
<td>1999</td>
<td>106</td>
</tr>
<tr>
<td>1/6</td>
<td>t5</td>
<td>YFilter-ICDE</td>
<td>2002</td>
<td>298</td>
</tr>
<tr>
<td>1/6</td>
<td>t6</td>
<td>CrowER</td>
<td>2012</td>
<td>34</td>
</tr>
</tbody>
</table>

Duplication increases the probability of “YFilter” being sampled
Error correction

• Assume $c(t)$ completely removes condition/value errors of $t$

• Assume $d(t)$ returns # of times that $t$ appears in the table (as duplicates)
  • Gotcha: scope of cleaning is not limited to the sample!

• Tweak $\phi(t)$ to $\phi_c(t)$ as follows
  • COUNT: $\phi_c(t) = \theta(c(t)) \cdot N \cdot \frac{1}{d(t)}$
  • SUM: $\phi_c(t) = \theta(c(t)) \cdot N \cdot c(t)[A] \cdot \frac{1}{d(t)}$
  • AVG: $\phi_c(t) = \theta(c(t)) \cdot \left(\frac{K}{K_\theta}\right) \cdot c(t)[A] \cdot \frac{\bar{d}}{d(t)}$, where $\bar{d} = \frac{K}{\sum_{t \in S} \frac{1}{d(t)}}$ is the duplication rate of the sample
Two estimations methods

SampleClean will choose the better result from RawSC and NormalizedSC as final estimation.
RawSC estimation

- Just query the cleaned sample for estimation $\mu_c$

- CLT confidence interval: $\pm \lambda \sqrt{\frac{\sigma_c^2}{K}}$, where $\sigma_c^2$ is computed over the cleaned samples

- Estimation quality unaffected by how bad the errors are—nice!

- It completely ignores the dirty data though
NormalizedSC estimation

• Idea: don’t ignore dirty data we are given—think about sampling “errors” (differences between dirty and clean) instead of sampling clean data

• Observation: errors are separable and additive
  • Define $\epsilon(t) = \phi(t) - \phi_c(t)$

• Compute query result on full dirty data: $f(D)$

• Given sample $S \subseteq D$, clean each $t \in S$

• Compute $Q = \{\epsilon(t) | t \in S\}$, and $\mu_q$ and $\sigma_q^2$ over $Q$

• Return $(f(D) - \mu_q) \pm \lambda \sqrt{\frac{\sigma_q^2}{K}}$
RawSC vs. NormalizedSC

<table>
<thead>
<tr>
<th></th>
<th>RawSC</th>
<th>NormalizedSC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idea</td>
<td>Clean estimation</td>
<td>Error correction</td>
</tr>
<tr>
<td>Error</td>
<td>$\sigma_c^2 / K$</td>
<td>$\sigma_q^2 / K$</td>
</tr>
<tr>
<td>Data queried</td>
<td>Sample</td>
<td>Full data</td>
</tr>
</tbody>
</table>

- Note that error is independent of the size of the full dataset!
- System can try both and return the tighter confidence interval!
Microsoft academic search

• Three authors, 1374 records, with real condition and duplication errors
• Goal: rank the three authors in # citations
• Strategy: keep cleaning until confidence intervals no longer overlap
  • How can we prioritize automatically?
  • Tech report version only talks about sample allocation across groups to minimize max length of confidence intervals
Intel Lab sensor data

• 44,460 records with different types of measurements and value errors

• Sample size = 500 (1.12%)

- Different queries have different error characteristics, making the choice of RawSC vs. NormalizedSC important
TPC-H with synthetic errors

- 6 millions records with synthetic errors
  - Value errors: perturb digits, based on OCR error model
  - Condition errors: remove at random
  - Duplication errors: 80%+1, 15%+2, 5%+3

RawSC is flat because it’s insensitive to value error distribution

Funny drop @ high%:
high dup% means most tuples will have the same dup#, making AVG easier to estimate
RawSC vs. NormalizedSC on TPC-H

- Less dirty: %3 value, %1 condition, %2 dup errors
- Very dirty: %30 value, %10 condition, %20 dup errors

Note the reversal in performance between RawSC and NormalizedSC
Imperfect cleaning

- Assume cleaning only identifies & cleans a fixed percentage of errors
  - Even AllClean below here is imperfect

- You cannot hope to get to 0%, but you can still approach AllClean
Summary

• Even simple sampling works well with cleaning
• Sampling clean data vs. sampling errors

• Selective predicates over data with lots of condition errors? Joins? Ranking?
  • Better sampling schemes than uniform?
  • Besides connection to SAQP, also data monitoring
• Cleaning up duplicates requires matching the sample with the entire database!
• How about missing data?
• Better modeling of imperfect cleaning?
  • What if the cleaned value is wrong?
• Cost of cleaning in general ≠ # tuples cleaned
  • Writing a script to clean up one particular error may allow you to clean an entire dataset w.r.t. this error; how can the sampling framework handle this?