Robust and Efficient Fuzzy Match for Online Data Cleaning

S. Chaudhuri, K. Ganjan, V. Ganti, R. Motwani

Presented by Aaditeshwar Seth

Motivation

- Data warehouse: Many input tuples
- Tuples can be erroneous
  - Spelling mistakes
  - Different syntactic representation
- How to clean them automatically?
- Assumptions
  - Tuples are structured: Eg. schema matching has already been done
  - Some tuple fields are supposed to contain the same values

Methodology

- Clean tuples are stored in reference table
- Fuzzy matching done to find best matching clean tuples
Challenges

- Scalability
  - Reference table can be very large
  - Volume of input tuples can be very large
- Domain specific enhancements should be possible to add
- Should be able to build upon existing relational DBMS
  - No complex data structures for persistence

Solution outline

- Transform input tuple into reference tuple
- Similarity metric = 1 – (transformation cost)
- Flat edit distance not good
  - Within a field, cannot distinguish between more and less informative tokens
    - Intuitively know Boing is more informative than Corporation
    - Hence, match on Boing should mean high similarity
    - Implies, should be expensive to transform input token into Boing than into Corporation
  - Attach weights to transformation costs for each token

Solution outline: Cont...

- Use inverse token frequency for weight assignment
  - Tokens like Boing occur less than tokens like Corporation
  - Just a heuristic – pathological cases can exist
- Optimizations
  - Do not compute exact transformation cost
    - Match on sets of substrings instead
  - Design efficient index on reference table
Fuzzy matching similarity function

- Input tuple $u$, reference tuple $v$
- Split into tokens
  - Each token has weight $w_{\text{TOKEN}} = \log(\text{Ref Set}) / \text{Freq}$
- Find $tc$ = edit distance for each token
- $tc(u, v) = \text{Summation of } tc$ for each token
- $\text{fms}(u, v) = 1 - \min(tc(u, v) / \text{Sum}(w_{\text{TOKEN}}), 1.0)$
- Not a problem with $\text{fms}_\text{approx}$

How is the minimum edit distance calculated?

Many transformation paths are possible.

Approximate FMS

- Match on $q$-grams

<table>
<thead>
<tr>
<th>Boeing</th>
<th>Boeing</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H_1$</td>
<td>$H_1$</td>
</tr>
<tr>
<td>3 2 1 0</td>
<td>8 2 1 0</td>
</tr>
<tr>
<td>$H_2$</td>
<td>$H_2$</td>
</tr>
<tr>
<td>4 5 6 7</td>
<td>9 5 6 7</td>
</tr>
</tbody>
</table>

Min hash = {ing, boe}  Min hash = {ing, uei}

$\text{ms}(t_i, r_j) = \sum_{q} \{ \text{ms}(\text{QG}(t_i), \text{QG}(r_j)) \}$

- Hash functions avoid string comparison

Approximate FMS: Cont...

1. Test for similarity of token $t_i$ with all tokens $r_j$ in same column, and select Max
2. Multiply by weight of chosen Max tuple $r_j$
3. Repeat over all tokens in same column
4. Repeat over all columns, and divide by total weight

$\text{sim}(QG(t), QG(r)) = \frac{1}{d^q}$

$\text{sim}(QG(t), QG(r)) = 0.57$
Error Tolerant Index

- Index reference set on \( tid \)
- Index ETI on \( \{q\text{-gram, coordinate, column}\} \)

<table>
<thead>
<tr>
<th>Input</th>
<th>Coordinate</th>
<th>Column</th>
<th>Frequency</th>
<th>ETI</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>19</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- Each q-gram belongs to a token
- Each token has a weight
- Token belongs to a column
- Coordinate indicates

What if same q-gram belongs to multiple tokens? Overwriting!

Query processing

- Find all tokens of input tuple \( u \)
- Find min-hash signature of all tokens
- For all q-grams in min-hash signature
  - Find ETI(\( q\text{-gram, coordinate, column} \))
  - Find token \( t \) to which q-gram belongs, and weight of this token
  - Increment similarity metric of matching \( tid \) by \( w(t)/|mh(t)| \)

Fetch best \( K \) matching \( tid \)'s with similarity > \( c\text{.threshold} \)

Query processing: Cont...

- Optimizations
  - When incrementing similarity metric with matching \( tid \)'s, only need to do it with new \( tid \)'s if the maximum score possible with all remaining q-grams is > \( c\text{-threshold} \)
- Optimistic short circuiting
  - Order q-grams according to their weights
  - Process only the first \( i \) q-grams
  - Fetch matching \( tid \)'s
  - But only fetch new \( tid \)'s if
    \[ \min_{j=1}^{i} (S(j) + |mh(t)|) > c\text{-threshold} \]
  - Stop when FMS of all \( K \) \( tid \)'s > \( c\text{-threshold} \)
  - If don’t stop then increment \( i \) and repeat
Extensions

- Consider token as another q-gram
  - Split importance equally among itself and its min-hash signature
- Assign weights to columns
  - Domain dependant
- Token transposition

Experiments

- Clean reference set
- Error injection methods for unclean input tuples
- Accuracy
  - FMS better than edit distance
  - Min-hash signatures are better than token-only
  - Accuracy improves with more hash functions
  - Having tokens does not negatively impact

Experiments: Cont...

- Efficiency: Processing time
  - Much faster than naïve
  - Query processing time decreases with signature size
  - Use of tokens improves processing time
- Efficiency: ETI construction
  - About 7 times the amount of time taken to process 1 tuple using the naïve algorithm
  - But cost is amortized over repeat queries
Experiments: Cont...

- Average number of tid’s fetched per input tuple
  - More q-grams decrease set sizes by better distinguishing similarity scores
- Average number of tid’s processed per input tuple
  - More q-grams increase the number of tid’s processed
  - Compensated by decrease in number of tid’s fetched

Discussion topics

- Relevance: In what scenarios does IDF work and distinguishes between more and less informative tokens
  - Cluster tokens together?
- Does weight calculation become an issue with optimizations and optimal short circuiting?
- How to update ETI with new tuples or outdated tuples?
- What is the role of the factor 2 in fmsapx?
- What if same q-gram belongs to multiple tokens in the same column?