Approximation: Theory and Algorithms
The q-Gram Distance

Nikolaus Augsten

Free University of Bozen-Bolzano
Faculty of Computer Science
DIS

Unit 3 – March 13, 2009
Outline

1 Motivation

2 Filters for the Edit Distance
   - Lower Bound Filters
   - Length Filter
   - \( q \)-Grams: Count Filter
   - \( q \)-Grams: Position Filtering
   - Experiments

3 The \( q \)-Gram Distance
   - Definition and Properties

4 Conclusion
**Scenario:**

- A company offers a number of services on the Web.
- You can subscribe for each service independently.
- Each service has its own database (no unique key across databases).

**Example:** customer tables of two different services:

<table>
<thead>
<tr>
<th>ID</th>
<th>name</th>
<th>...</th>
<th>ID</th>
<th>name</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1023</td>
<td>Frodo Baggins</td>
<td>...</td>
<td>948483</td>
<td>John R. R. Tolkien</td>
<td>...</td>
</tr>
<tr>
<td>21</td>
<td>J. R. R. Tolkien</td>
<td>...</td>
<td>153494</td>
<td>C. S. Lewis</td>
<td>...</td>
</tr>
<tr>
<td>239</td>
<td>C.S. Lewis</td>
<td>...</td>
<td>494392</td>
<td>Fordo Baggins</td>
<td>...</td>
</tr>
<tr>
<td>863</td>
<td>Bilbo Baggins</td>
<td>...</td>
<td>799294</td>
<td>Biblo Baggins</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**Task:** Created unified customer view!
**Motivation**

The **Join Approach**

- **Solution:** Join customer tables on name attribute (Q1):
  
  ```sql
  SELECT * FROM A,B
  WHERE A.name = B.name
  ```

- **Exact Join:** Does not work!

- **Approximate Join:** Allow $k$ errors...
  
  1. Register UDF (User Defined Function) for the edit distance:
     
     ```sql
     levenshtein(x, y)
     ```
   
     returns the union cost edit distance between the strings $x$ and $y$.

  2. Rewrite query Q1 as approximate join (Q2):
     
     ```sql
     SELECT * FROM A,B
     WHERE levenshtein(A.name, B.name) <= k
     ```
Effectiveness and Efficiency of the Approximate Join

**Effectiveness:** Join result for \( k = 3 \):

<table>
<thead>
<tr>
<th>ID</th>
<th>name</th>
<th>ID</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1023</td>
<td>Frodo Baggins</td>
<td>494392</td>
<td>Fordo Baggins</td>
</tr>
<tr>
<td>21</td>
<td>J. R. R. Tolkien</td>
<td>948483</td>
<td>John R. R. Tolkien</td>
</tr>
<tr>
<td>239</td>
<td>C.S. Lewis</td>
<td>153494</td>
<td>C. S. Lewis</td>
</tr>
<tr>
<td>863</td>
<td>Bilbo Baggins</td>
<td>799294</td>
<td>Biblo Baggins</td>
</tr>
</tbody>
</table>

⇒ very good (100% correct)

**Efficiency:** How does the DB evaluate the query?

1. compute \( A \times B \)
2. evaluate UDF on each tuple \( t \in A \times B \)

**Experiment** [GIJ+01]: Self-join on string table (average string length = 14):

- 1K tuples: ca. 30min
- 14K tuples: > 3 days!

**Prohibitive runtime!**
Using a Filter for Search Space Reduction

- **Search space**: \( A \times B \) \( \Rightarrow \) \(|A| \cdot |B|\) edit distance computations
- **Filtering (Pruning)**: Remove tuples that cannot match, without actually computing the distance.
Outline

1 Motivation

2 Filters for the Edit Distance
   - Lower Bound Filters
     - Length Filter
     - $q$-Grams: Count Filter
     - $q$-Grams: Position Filtering
     - Experiments

3 The $q$-Gram Distance
   - Definition and Properties

4 Conclusion
### Error Types:

<table>
<thead>
<tr>
<th>Filter</th>
<th>Test</th>
<th>Correct Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>positive</td>
<td>positive</td>
<td>true positive</td>
</tr>
<tr>
<td>negative</td>
<td>negative</td>
<td>false negative</td>
</tr>
</tbody>
</table>

#### Example: “Are $x$ and $y$ within edit distance $k$?”
- **Correct result**: compute edit distance and test $\text{ed}(x, y) \leq k$
- **Filter test**: give answer without computing edit distance
- **False negatives**: $x$ and $y$ are pruned although $\text{ed}(x, y) \leq k$.
- **False positives**: $x$ and $y$ are not pruned although $\text{ed}(x, y) \ngeq k$.

#### Good filters have
- no false negatives (i.e., miss no correct results)
- few false positive (i.e., avoid unnecessary distance computations)
Filters for the Edit Distance

Lower Bound Filters

- Lower bound (lb) for distance \( \text{dist}(x, y) \):
  \[
  \text{dist}(x, y) \geq \text{lb}_\text{dist}(x, y)
  \]

- Query Q3 with **Lower Bound Filter**:
  
  ```
  SELECT * FROM A,B
  WHERE \text{lb}(A.name, B.name) \leq k \text{ AND }
  \text{levenshtein}(A.name, B.name) \leq k
  
  \text{lb}(A.name, B.name) \text{ is a cheap function}
  \text{database will optimize query: compute}
  \text{levenshtein}(A.name, B.name) \text{ only if } \text{lb}(A.name, B.name) > k
  
  \text{No false negatives!}
  ```
Outline

1 Motivation

2 Filters for the Edit Distance
   - Lower Bound Filters
   - Length Filter
   - $q$-Grams: Count Filter
   - $q$-Grams: Position Filtering
   - Experiments

3 The $q$-Gram Distance
   - Definition and Properties

4 Conclusion
Length Filtering

Theorem (Length Filtering [GIJ+01])

If two strings $x$ and $y$ are within edit distance $k$, their lengths cannot differ by more than $k$:

$$\text{ed}(x, y) \geq \text{abs}(|x| - |y|)$$

- **Proof:** At least $\text{abs}(|x| - |y|)$ inserts are needed to bring $x$ and $y$ to the same length.

- Query Q4 with **Length Filtering**:

  ```sql
  SELECT * FROM A,B
  WHERE ABS(LENGTH(A.name)-LENGTH(B.name)) <= k AND
  levenshtein(A.name, B.name) <= k
  ```
Example: Length Filtering

Execute query without/with length filter ($k = 3$):

<table>
<thead>
<tr>
<th>ID</th>
<th>A name</th>
<th>ID</th>
<th>B name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1023</td>
<td>Frodo Baggins$_{13}$</td>
<td>948483</td>
<td>John R. R. Tolkien$_{18}$</td>
</tr>
<tr>
<td>21</td>
<td>J. R. R. Tolkien$_{16}$</td>
<td>153494</td>
<td>C. S. Lewis$_{11}$</td>
</tr>
<tr>
<td>239</td>
<td>C.S. Lewis$_{10}$</td>
<td>494392</td>
<td>Fordo Baggins$_{13}$</td>
</tr>
<tr>
<td>863</td>
<td>Bilbo Baggins$_{13}$</td>
<td>799294</td>
<td>Biblo Baggins$_{13}$</td>
</tr>
</tbody>
</table>

Without length filter: 16 edit distance computations

With length filter ($k = 3$): 12 edit distance computations

- J. R. R. Tolkien $\leftrightarrow$ C. S. Lewis is pruned
- all pairs (..., John R. R. Tolkien) but
  (J. R. R. Tolkien, John R. R. Tolkien) are pruned
Outline

1 Motivation

2 Filters for the Edit Distance
   - Lower Bound Filters
   - Length Filter
   - q-Grams: Count Filter
   - q-Grams: Position Filtering
   - Experiments

3 The q-Gram Distance
   - Definition and Properties

4 Conclusion
What is a $q$-Gram?

- **Intuition:**
  - slide window of length $q$ over string $x \in \Sigma^*$
  - characters covered by window form a $q$-gram
  - where window extends string: fill with dummy character $\#$ \(!\in\Sigma\)

- **Example:** $x = \text{Frodo}$, $q = 3$

  extended: $\#\#\text{Frodo}\#\#$
  
  $q$-grams: $\#\#F$
  
  $\#F$
  
  $F$
  
  $r$
  
  $o$
  
  $d$
  
  $o$
  
  $d$
  
  $o$
  
  $d$
  
  $o$
  
  $\#$

- **$q$-Gram Profile** $G_x$: bag of all $q$-grams of $x$

  **Profile size:** $|G_x| = |x| + q - 1$
**Intuition**: Strings within small edit distance share many $q$-grams.

How many $q$-grams ($q = 3$) change/remain?

|   | $|G_x|$ |   | $|G_y|$ | $|G_x \cap G_y|$ |
|---|--------|---|--------|-----------------|
| peter | 7     | meter | 7 | 4               |
| peter | 7     | peters | 8 | 5               |
| peter | 7     | peer  | 6 | 4               |

$ed(x, y) = 1 \Rightarrow |G_x \cap G_y| = \max(|G_x|, |G_y|) - q$
Filters for the Edit Distance

$q$-Grams: Count Filter

**Multiple Edit Operations and Changing $q$-Grams**

- $ed(x, y) = 1 \Rightarrow |G_x \cap G_y| = \max(|G_x|, |G_y|) - q$
- What if $ed(x, y) = k > 1$?

| $x$   | $|G_x|$ | $y$   | $|G_y|$ | $|G_x \cap G_y|$ |
|-------|--------|-------|--------|------------------|
| peter | 7      | meters| 8      | 2                |
| peter | 7      | petal | 7      | 3                |

- Multiple edit operations may affect the same $q$-gram:
  - $peter \rightarrow G_x = \{\#\#p, \#pe, pet, ete, ter, er\#, r\#\}$
  - $petal \rightarrow G_x = \{\#\#p, \#pe, pet, eta, tal, al\#, l\#\}$

- Each edit operation affects at most $q$ $q$-grams.
Count Filtering

Theorem (Count Filtering [GIJ+01])

Consider two strings $x$ and $y$ with the $q$-gram profiles $G_x$ and $G_y$, respectively. If $x$ and $y$ are within an edit distance of $k$, then the cardinality of the $q$-gram profile intersection is at least

$$|G_x \cap G_y| \geq \max(|G_x|, |G_y|) - kq$$

Proof (by induction):

- true for $k = 1$: $|G_x \cap G_y| \geq \max(|G_x|, |G_y|) - q$
- $k \to k + 1$: each additional edit operation changes at most $q$ $q$-grams.
Implementation of $q$-Grams

- **Given:** table $R$ with schema $(A_0, A_1, \ldots, A_m)$
  - $A_0$ is the key attribute
  - $A_i, i > 0$ are string-valued

- **Compute** auxiliary tables $RA_iQ$ with schema $(A_0, Qgram)$:
  - each tuple stores one $q$-gram
  - string $x$ of attribute $A_i$ is represented by its $|x| + q - 1$ $q$-grams
  - $RA_iQ.A_0$ is the key value ($R.A_0$) of a tuple with $R.A_i = x$
  - $RA_iQ.Qgram$ is one of the $q$-grams of $x$

- **Example:**

<table>
<thead>
<tr>
<th>$R$</th>
<th>$RA_1Q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_0$</td>
<td>$A_1$</td>
</tr>
<tr>
<td>1023</td>
<td>Frodo Baggins</td>
</tr>
<tr>
<td>21</td>
<td>J. R. R. Tolkien</td>
</tr>
<tr>
<td>239</td>
<td>C.S. Lewis</td>
</tr>
<tr>
<td>863</td>
<td>Bilbo Baggins</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Count Filtering Query

Query Q5 with **Count Filtering**:

```sql
SELECT R1.A0, R2.A0, R1.Ai, R2.Aj
FROM R1, R1AiQ, R2, R2AjQ
WHERE R1.A0 = R1AiQ.A0 AND
     R2.A0 = R2AjQ.A0 AND
     R1AiQ.Qgram = R2AjQ.Qgram AND
     ABS(LENGTH(A.name)-LENGTH(B.name)) <= k
GROUP BY R1.A0, R2.A0, R1.Ai, R2.Aj
HAVING COUNT(*) >= LENGTH(R1.Ai)-1-(k-1)*q AND
       COUNT(*) >= LENGTH(R2.Aj)-1-(k-1)*q AND
       levenshtein(R1.Ai,R2.Aj) <= k
```
Problem with Count Filtering Query

- Previous query Q5 works fine for $kq < \max(|G_x|, |G_y|)$.
- However: If $kq \geq \max(|G_x|, |G_y|)$, no $q$-grams may match even if $\text{ed}(x, y) \leq k$.
- Example ($q = 3$, $k = 2$):
  - WHERE-clause prunes $x$ and $y$, although $\text{ed}(x, y) \leq k$
  - $x = \text{IBM}$ \quad $G_x = \{##I, #IB, IBM, BM#, M##\}$ \quad $|G_x| = 5$
  - $y = \text{BMW}$ \quad $G_y = \{##B, #BM, BMW, MW#, W##\}$ \quad $|G_y| = 5$

- False negatives:
  - short strings with respect to edit distance (e.g., $|x| = 3$, $k = 3$)
  - even if within given edit distance, matches tend to be meaningless (e.g., $\text{abc}$ and $\text{xyz}$ are within edit distance $k = 3$)
Fixing Count Filtering Query

- Fix query to avoid false negatives [GIJ+03]:
  - Join pairs \((x, y)\) with \(kq \geq \max(|G_x|, |G_y|)\) using only length filter.
  - Union results with results of previous query Q5.

- Query Q6 without false negatives (extends previous query Q5):

  ```
  UNION ALL
  SELECT R1.A0, R2.A0, R1.Ai, R2.Aj
  FROM R1, R2
  WHERE
    LENGTH(R1.Ai)+q-1 <= k*q AND
    LENGTH(R2.Aj)+q-1 <= k*q AND
    ABS(LENGTH(R1.Ai) - LENGTH(R2.Aj)) <= k
    levenshtein(R1.Ai,R2.Aj) <= k
  ```

- Note: We omit this part in subsequent versions of the query, as it remains unchanged.
Outline

1 Motivation

2 Filters for the Edit Distance
   - Lower Bound Filters
   - Length Filter
   - $q$-Grams: Count Filter
   - $q$-Grams: Position Filtering
   - Experiments

3 The $q$-Gram Distance
   - Definition and Properties

4 Conclusion
Positional $q$-grams

- Enrich $q$-grams with position information:
  - extended string: prefix and suffix string $x$ with $q - 1$ characters #
  - slide window of length $q$ over extended string $x'$
  - characters covered by window after shifting it $i$ times form the $q$-gram at position $i + 1$

- Example: $x = \text{Frodo}$

  extended string: 
  positional $q$-grams: 
  
  
  
  
  
  
  
  
  

Filters for the Edit Distance  \(q\)-Grams: Position Filtering

Computing Positional \(q\)-Grams in SQL

- Given: table \(N\)
  - \(N\) has a single attribute \(i\)
  - \(N\) is filled with numbers from 1 to \(max\) (\(max\) is the maximum string length plus \(q - 1\))

- Positional \(q\)-grams in SQL (Q7):

\[
\text{INSERT INTO RAiQ}
\begin{align*}
\text{SELECT } & \text{R.A0, N.i,} \\
& \text{SUBSTRING(CONCAT(} \\
& \text{SUBSTRING ('#..#', 1, q - 1),} \\
& \text{LOWER(R.Ai),} \\
& \text{SUBSTRING ('#..#', 1, q - 1)),} \\
& \text{N.i, q)} \\
\text{FROM } & \text{R, N} \\
\text{WHERE } & \text{N.i} \leq \text{LENGTH(R.Ai) + q - 1}
\end{align*}
\]
Corresponding $q$-Grams

- **Corresponding $q$-gram:**
  - Given: positional $q$-grams $(i, g)$ of $x$
  - transform $x$ to $y$ applying edit operations
  - $(i, g)$ “becomes” $(j, g)$ in $y$
  - We define: $(i, g)$ corresponds to $(j, g)$

- **Example:**
  - $x' = ##abaZabaabaaba##$, $y' = ##abaabaabaabaabaaba##$
  - edit distance is 1 (delete $Z$ from $x$)
  - $(7, aba)$ in $x$ corresponds to $(6, aba)$ in $y$
  - ... but not to $(9, aba)$
Position Filtering

Theorem (Position Filtering [GIJ+01])

If two strings $x$ and $y$ are within edit distance $k$, then a positional $q$-gram in one cannot correspond to a positional $q$-gram in the other that differs from it by more than $k$ positions.

Proof:
- each increment (decrement) of a position requires an insert (delete);
- a shift by $k$ positions requires $k$ inserts/deletes.
Position Filtering

Query Q8 with \textbf{Count and Position Filtering}:

\begin{verbatim}
SELECT   R1.A0, R2.A0, R1.Ai, R2.Aj
FROM      R1, R1AiQ, R2, R2AjQ
WHERE      R1.A0 = R1AiQ.A0 AND
            R2.A0 = R2AjQ.A0 AND
            R1AiQ.Qgram = R2AjQ.Qgram AND
            \text{ABS}(\text{LENGTH}(A.name)-\text{LENGTH}(B.name)) \leq k AND
            \text{ABS}(R1AiQ.Pos-R2AjQ.Pos) \leq k
GROUP BY  R1.A0, R2.A0, R1.Ai, R2.Aj
HAVING    \text{COUNT}(*) \geq \text{LENGTH}(R1.Ai)-1-(k-1)*q AND
            \text{COUNT}(*) \geq \text{LENGTH}(R2.Aj)-1-(k-1)*q AND
            \text{levenshtein}(R1.Ai,R2.Aj) \leq k
\end{verbatim}
Outline

1 Motivation

2 Filters for the Edit Distance
   - Lower Bound Filters
   - Length Filter
   - $q$-Grams: Count Filter
   - $q$-Grams: Position Filtering
   - Experiments

3 The $q$-Gram Distance
   - Definition and Properties

4 Conclusion
Experimental Data

- All experimental results taken from [GIJ+01]
- Three string data sets:
  - set1: 40K tuples, average length: 14 chars
  - set2: 30K tuples, average length: 38 chars
  - set3: 30K tuples, average length: 33 chars
String Length Distributions

Set 1

Set 2

Set 3
Candidate Set Size

- Question: How many edit distances do we have to compute?
- $q = 2$
- Caption:
  - CP: cross product
  - L: length filtering, P: position filtering, C: count filtering
  - Real: number of real matches

Set 1
Candidate Set Size

Question: How many edit distances do we have to compute?

\[ q = 2 \]

Caption:
- CP: cross product
- L: length filtering, P: position filtering, C: count filtering
- Real: number of real matches

Set 2
Candidate Set Size

- Question: How many edit distances do we have to compute?
- \( q = 2 \)
- Caption:
  - CP: cross product
  - L: length filtering, P: position filtering, C: count filtering
  - Real: number of real matches

Set 3
Question: How does the choice of $q$ influence the filter effectiveness?

- Edit Distance Threshold $k = 2$
- Edit Distance Threshold $k = 3$
Filters for the Edit Distance Experiments

Response Time

- Approximate self-join on sample of 1000 tuples (set 1)
  (full dataset > 3 days without filters!)
- $k$: edit distance threshold
- $Q_1$: edit distance without filters
- $Q_2$: edit distance with filters

<table>
<thead>
<tr>
<th>Response Time (sec)</th>
<th>$k=1$</th>
<th>$k=2$</th>
<th>$k=3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q_1$ (UDF only)</td>
<td>1354</td>
<td>2026</td>
<td>2044</td>
</tr>
<tr>
<td>$Q_2$ (Filtering)</td>
<td>48</td>
<td>68</td>
<td>91</td>
</tr>
</tbody>
</table>
Outline

1 Motivation

2 Filters for the Edit Distance
   - Lower Bound Filters
   - Length Filter
   - $q$-Grams: Count Filter
   - $q$-Grams: Position Filtering
   - Experiments

3 The $q$-Gram Distance
   - Definition and Properties

4 Conclusion
The q-Gram Distance

Definition (q-Gram Distance [Ukk92])

Let $G_x$ and $G_y$ be the q-gram profiles of the strings $x$ and $y$, respectively. The q-gram distance between two strings is the number of q-grams in $G_x$ and $G_y$ that have no match in the other profile,

$$\text{dist}_q(x, y) = |G_x \cup G_y| - 2|G_x \cap G_y|.$$ 

Example: $q = 2, x = abab, y = abcab$

- $G_x = \{#a, ab, ba, ab, b#\}$
- $G_y = \{#a, ab, bc, ca, ab, b#\}$
- $G_x \cup G_y = \{#a, ab, ba, ab, b#, #a, ab, bc, ca, ab, b#\}$
- $G_x \cap G_y = \{#a, ab, ab, b#\}$
- $\text{dist}_q(x, y) = |G_x \cup G_y| - 2|G_x \cap G_y| = 11 - 2 \cdot 4 = 3$
The $q$-gram distance is a **pseudo metric**:
For all $x, y, z \in \Sigma^*$
- $\text{dist}_q(x, y) + \text{dist}_q(y, z) \geq \text{dist}_q(x, z)$ (triangle inequality)
- $\text{dist}_q(x, y) = \text{dist}_q(y, x)$ (symmetric)
- $\text{dist}_q(x, y) = 0 \iff x = y$

**Note:** Identity condition relaxed: $\text{dist}_q(x, y) = 0 \not\Rightarrow x = y$

i.e., the $q$-gram distance between two different strings can be 0

**Example:**
\[
\text{dist}_q(axybxy cxyd, axy cxyb xyd) = 0
\]
\[
G_x = G_y = \{##a, #ax, axy, xyb, ybx, bxy, yxc, ycx, cxy, yxd, yd#, d##\}
What is a good threshold?

\[
ed(\text{International Business Machines Corporation,} \\
\text{International Business Machines Corporation}) = 2 \\
ed(\text{IBM, BMW}) = 2 \\
ed(\text{Int. Business Machines Corp.,} \\
\text{International Business Machines Corporation}) = 17
\]

Problem: Absolute numbers not always meaningful...

Solution: Compute error relative to string length!
Distance Normalization (2/3)

- Normalize distance such that $\delta(x, y) \in [0..1]$
- **Edit Distance**: $0 \leq ed(x, y) \leq \max(|x|, |y|)$
- Normalized Edit Distance: $0 \leq \text{norm-ed}(x, y) \leq 1$

$$\text{norm-ed}(x, y) = \frac{ed(x, y)}{\max(|x|, |y|)}$$

- **$q$-Gram Distance**: $0 \leq \text{dist}_q(x, y) \leq |G_x| + |G_y|$
- Normalized $q$-Gram Distance: $0 \leq \text{norm-dist}_q(x, y) \leq 1$

$$\text{norm-dist}_q(x, y) = \frac{\text{dist}_q(x, y)}{|G_x| + |G_y|} = 1 - \frac{|G_x \cap G_y|}{|G_x \cup G_y|}$$
Normalized edit distance:

\[
\text{norm-ed}(\text{International Business Machines Corporation}, \\
\text{International Business Machine Corporation}) = 0.047
\]
\[
\text{norm-ed}(\text{IBM}, \text{BMW}) = 0.66
\]
\[
\text{norm-ed}(\text{Int. Business Machines Corp.}, \\
\text{International Business Machines Corporation}) = 0.4
\]

Normalized \(q\)-gram distance \((q = 3)\):

\[
\text{norm-dist}_q(\text{International Business Machines Corporation}, \\
\text{International Business Machine Corporation}) = 0.089
\]
\[
\text{norm-dist}_q(\text{IBM}, \text{BMW}) = 1.0
\]
\[
\text{norm-dist}_q(\text{Int. Business Machines Corp.}, \\
\text{International Business Machines Corporation}) = 0.36
\]
Edit Distance vs. $q$-Gram Distance

- Edit distance can not handle block-moves well:
  \[ x = \text{Nikolaus Augsten} \quad y = \text{Augsten Nikolaus} \]
  \[ \text{norm-ed}(x, y) = 1.0 \]
  \[ \text{norm-dist}_q(x, y) = 0.39 \quad (q = 3) \]

- $q$-Gram distance may be too strict:
  \[ x = +39-06-46-74-22 \quad y = (39 \ 06 \ 467422) \]
  \[ \text{norm-ed}(x, y) = 0.4 \]
  \[ \text{norm-dist}_q(x, y) = 1.0 \quad (q = 3) \]
Approximate join with edit distance inefficient.

Edit distance filters speed up join:
- Length filter: based on the string length
- Count filter: based on $q$-Grams
- Position filter: based on positional $q$-Grams

$q$-Gram distance: a distance on its own
- pseudo metric
- can handle block moves
What’s Next?

- Hierarchical data and XML in a database system
  - (un)ordered, labeled trees
  - representing trees in a database
  - query and update efficiency
  - representing XML as a tree
Luis Gravano, Panagiotis G. Ipeirotis, H. V. Jagadish, Nick Koudas, S. Muthukrishnan, and Divesh Srivastava.
Approximate string joins in a database (almost) for free.

Luis Gravano, Panagiotis G. Ipeirotis, H. V. Jagadish, Nick Koudas, S. Muthukrishnan, and Divesh Srivastava.

Esko Ukkonen.