Similarity Search

The q-Gram Distance

Nikolaus Augsten

nikolaus.augsten@sbg.ac.at

Dept. of Computer Sciences
University of Salzburg
http://dbresearch.uni-salzburg.at

Version November 9, 2016

Wintersemester 2016/2017

Outline

Outline

Application Scenario

- 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:

A

ID	name	...
1023	Frodo Baggins	...
21	J. R. R. Tolkien	...
239	C.S. Lewis	...
863	Bilbo Baggins	...
...	...	...

B

ID	name	...
948483	John R. R. Tolkien	...
153494	C. S. Lewis	...
494392	Fordo Baggins	...
799294	Biblo Baggins	...
...	...	...

- Task: Created unified customer view!

The Join Approach

- Solution: Join customer tables on name attribute (Q1):

```
SELECT * FROM A,B
WHERE A.name = B.name
```

- Exact Join: Does not work!
- Similarity Join: Allow k errors...
  (1) Register UDF (User Defined Function) for the edit distance:

$$\text{ed}(x, y)$$

  retuns the union cost edit distance between the strings x and y.
  (2) Rewrite query Q1 as similarity join (Q2):

```
SELECT * FROM A,B
WHERE ed(A.name, B.name) <= k
```


Effectiveness and Efficiency of the Approximate Join

- Effectiveness: Join result for $k = 3$:

ID	name	ID	name
1023	Frodo Baggins	494392	Fordo Baggins
21	J. R. R. Tolkien	948483	John R. R. Tolkien
239	C.S. Lewis	153494	C. S. Lewis
863	Bilbo Baggins	799294	Biblo Baggins

$\Rightarrow$ 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 can not match, without actually computing the distance.

Outline

Filter Properties

- Error Types:

		Correct Result	
		positive	negative
Filter	positive	true positive	false positive
Test	negative	false negative	true negative

- 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 $\text{ed}(x, y) \not\leq k$.
- Good filters have
  - no false negatives (i.e., miss no correct results)
  - few false positive (i.e., avoid unnecessary distance computations)

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 lb(A.name, B.name) <= k AND
      ed(A.name, B.name) <= k
```

  - `lb(A.name, B.name)` is a cheap function
  - database will optimize query: compute `ed(A.name,B.name)` only if `lb(A.name,B.name) > k`
- No false negatives!

Outline

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:

```
SELECT * FROM A,B
WHERE ABS(LENGTH(A.name)-LENGTH(B.name)) <= k AND
      ed(A.name, B.name) <= k
```

Example: Length Filtering

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

A

ID	name
1023	Frodo Baggins$_{13}$
21	J. R. R. Tolkien$_{16}$
239	C.S. Lewis$_{10}$
863	Bilbo Baggins$_{13}$

B

ID	name
948483	John R. R. Tolkien$_{18}$
153494	C. S. Lewis$_{11}$
494392	Fordo Baggins$_{13}$
799294	Biblo Baggins$_{13}$

- 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`) except (`J. R. R. Tolkien`, `John R. R. Tolkien`) are pruned

Outline

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 $\# \notin \Sigma$
- Example: $x =$ `Frodo`, $q = 3$

```
extended: ##Frodo##
q-grams:  ##F
           #Fr
            Fro
             rod
              odo
               do#
                o##
```

- q-Gram Profile G_x: bag of all q-grams of x
- Profile size: $|G_x| = |x| + q - 1$

Single Edit Operations and Changing q-Grams

- Intuition: Strings within small edit distance share many q-grams.
- How many q-grams ($q = 3$) change/remain?

x	$\lvert G_x\rvert$	y	$\lvert G_y\rvert$	$\lvert G_x \Cap G_y\rvert$
peter	7	meter	7	4
peter	7	peters	8	5
peter	7	peer	6	4

- $ed(x, y) = 1 \Rightarrow |G_x \Cap G_y| = \max(|G_x|, |G_y|) - q$

Multiple Edit Operations and Changing q-Grams

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

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 → G_x = {##p, #pe, pet, ete, ter, er#, r##}
petal → 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 edit distance 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: tables A and B with schema $(id, name)$
  - *id* is the key attribute
  - *name* is string-valued
- Compute auxiliary tables QA and QB with schema $(id, qgram)$:
  - each tuple stores one q-gram
  - string x of attribute *name* is represented by its $|x| + q - 1$ q-grams
  - $QA.id$ is the key value ($A.id$) of a tuple with $A.name = x$
  - $QA.qgram$ is one of the q-grams of x
- Example:

A

id	name
1023	Frodo Baggins
21	J. R. R. Tolkien
239	C.S. Lewis
863	Bilbo Baggins

QA

id	qgram
1023	##F
1023	#Fr
...	...
21	##J
21	#J.
...	...

Count Filtering Query

- Query Q5 with Count Filtering:

```
SELECT    A.id, B.id, A.name, B.name
FROM      A, QA, B, QB
WHERE     A.id = QA.id AND
          B.id = QB.id AND
          QA.qgram = QB.qgram AND
          ABS(LENGTH(A.name)-LENGTH(B.name)) <= k
GROUP BY  A.id, B.id, A.name, B.name
HAVING    COUNT(*) >= LENGTH(A.name)-1-(k-1)*q AND
          COUNT(*) >= LENGTH(B.name)-1-(k-1)*q AND
          ed(A.name,B.name) <= 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 $\mathrm{ed}(x, y) <= k$.
- Example ($q = 3, k = 2$):
  WHERE-clause prunes x and y, although $\mathrm{ed}(x, y) <= k$

  $x = \texttt{IBM}$ $G_x = \{\texttt{\#\#I}, \texttt{\#IB}, \texttt{IBM}, \texttt{BM\#}, \texttt{M\#\#}\}$ $|G_x| = 5$
  $y = \texttt{BMW}$ $G_y = \{\texttt{\#\#B}, \texttt{\#BM}, \texttt{BMW}, \texttt{MW\#}, \texttt{W\#\#}\}$ $|G_y| = 5$
- 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., abc and 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
SELECT A.id, B.id, A.name, B.name
FROM   A, B
WHERE  LENGTH(A.name)+q-1 <= k*q AND
       LENGTH(B.name)+q-1 <= k*q AND
       ABS(LENGTH(A.name) - LENGTH(B.name)) <= k AND
       ed(A.name,B.name) <= k
```

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

Outline

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: $x = \texttt{Frodo}$

extended string: ##Frodo##
positional q-grams:
(1,##F)
(2,#Fr)
(3,Fro)
(4,rod)
(5,odo)
(6,do#)
(7,o##)

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 for table A in SQL (Q7):

```
CREATE TABLE QA AS
       SELECT A.id, N.i AS pos,
           SUBSTRING(CONCAT(
               SUBSTRING('#..#', 1, q - 1),
               LOWER(A.name),
               SUBSTRING('#..#', 1, q - 1)),
           N.i, q) AS qgram
       FROM A, N
       WHERE N.i <= LENGTH(A.name) + q - 1
```


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' = ##abaabaabaabaaba##
  - edit distance is 1 (delete Z from x)
  - $(7, \text{aba})$ in x corresponds to $(6, \text{aba})$ in y
  - ... but not to $(9, \text{aba})$

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 then 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 Count and Position Filtering:

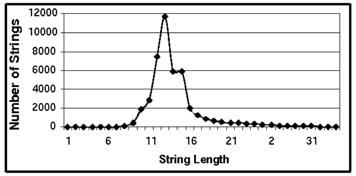
```
SELECT   A.id, B.id, A.name, B.name
FROM     A, QA, B, QB
WHERE    A.id = QA.id AND
         B.id = QB.id AND
         QA.qgram = QB.qgram AND
         ABS(LENGTH(A.name)-LENGTH(B.name)) <= k AND
         ABS(QA.pos-QB.pos)<=k
GROUP BY A.id, B.id, A.name, B.name
HAVING   COUNT(*) >= LENGTH(A.name)-1-(k-1)*q AND
         COUNT(*) >= LENGTH(B.name)-1-(k-1)*q AND
         ed(A.name,B.name) <= k
```

Outline

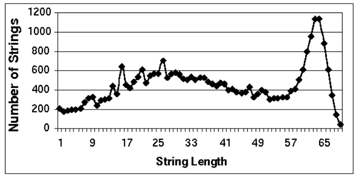
Experimental Data

- 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

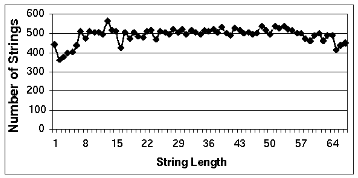
String Length Distributions



Set 1



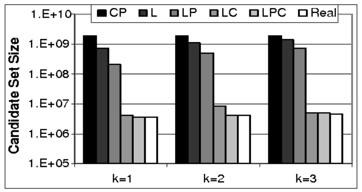
Set 2



Set 3

Candidate Set Size

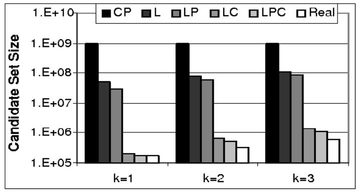
- Question: How many edit distances do we have to compute?
- Show candidate set size for different filters (small is good).
- $q = 2$
- Caption:
  - CP: cross product
  - L: length filtering, P: position filtering, C: count filtering
  - Real: number of real matches



Set 1

Candidate Set Size

- Question: How many edit distances do we have to compute?
- Show candidate set size for different filters (small is good).
- $q = 2$
- Caption:
  - CP: cross product
  - L: length filtering, P: position filtering, C: count filtering
  - Real: number of real matches

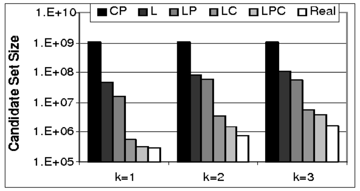


Set 2

Filters for the Edit Distance Experiments

Candidate Set Size

- Question: How many edit distances do we have to compute?
- Show candidate set size for different filters (small is good).
- $q = 2$
- Caption:
  - CP: cross product
  - L: length filtering, P: position filtering, C: count filtering
  - Real: number of real matches

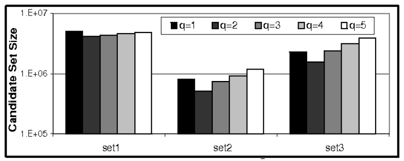


Set 3

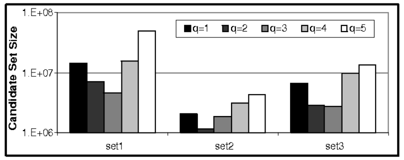
Filters for the Edit Distance Experiments

Various q-Gram Lengths

- Question: How does the choice of q influence the filter effectiveness?
- Show candidate set size for different q values (small is good).



Edit Distance Threshold $k = 2$

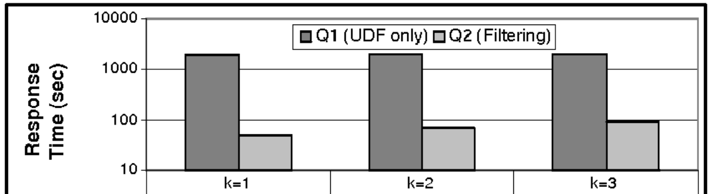


Edit Distance Threshold $k = 3$

Filters for the Edit Distance Experiments

Response Time

- Approximate self-join on small sample of 1000 tuples (set 1) (full dataset $>$ 3 days without filters!)
- Measure response time (small is good).
- Caption:
  - k: edit distance threshold
  - $Q1$: edit distance without filters
  - $Q2$: edit distance with filters



	k=1	k=2	k=3
Q1 (UDF only)	1954	2028	2044
Q2 (Filtering)	48	68	91

Outline

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,

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

- Example: $q = 2, x = \texttt{abab}, y = \texttt{abcab}$

  $G_x = \{\texttt{\#a}, \texttt{ab}, \texttt{ba}, \texttt{ab}, \texttt{b\#}\}$
  $G_y = \{\texttt{\#a}, \texttt{ab}, \texttt{bc}, \texttt{ca}, \texttt{ab}, \texttt{b\#}\}$
  $G_x \uplus G_y = \{\texttt{\#a}, \texttt{ab}, \texttt{ba}, \texttt{ab}, \texttt{b\#}, \texttt{\#a}, \texttt{ab}, \texttt{bc}, \texttt{ca}, \texttt{ab}, \texttt{b\#}\}$
  $G_x \Cap G_y = \{\texttt{\#a}, \texttt{ab}, \texttt{ab}, \texttt{b\#}\}$
  $\text{dist}_q(x, y) = |G_x \uplus G_y| - 2|G_x \Cap G_y| = 11 - 2 \cdot 4 = 3$

Pseudo Metric q-Gram Distance

- 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)
  - $\text{dist}_q(x, y) = 0 \Leftarrow x = y$
- 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:

  $\text{dist}_q(\texttt{axybxycxyd}, \texttt{axycxybxyd}) = 0$
  $G_x = G_y = \{\texttt{\#\#a}, \texttt{\#ax}, \texttt{axy}, \texttt{xyb}, \texttt{ybx}, \texttt{bxy}, \texttt{xyc}, \texttt{ycx}, \texttt{cxy}, \texttt{xyd}, \texttt{yd\#}, \texttt{d\#\#}\}$

Distance Normalization (1/3)

- What is a good threshold?

  ed(International Business Machines Corporation,
  International Bussiness Machine Corporation) = 2
  ed(IBM, BMW) = 2
  ed(Int. Business Machines Corp.,
  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: $0 \leq dist_q(x, y) \leq |G_x \uplus G_y| - |G_x \Cap G_y|$
- Normalized q-Gram Distance: $0 \leq \text{norm-dist}_q(x, y) \leq 1$

$$\text{norm-dist}_q(x, y) = \frac{dist_q(x, y)}{|G_x \uplus G_y| - |G_x \Cap G_y|} = 1 - \frac{|G_x \Cap G_y|}{|G_x \uplus G_y| - |G_x \Cap G_y|}$$

- Dividing by $|G_x \uplus G_y|$ also normalizes to [0..1], but the metric properties (triangle inequality) get lost [ABG10].

Distance Normalization (3/3)

- Normalized edit distance:

norm-ed(`International Business Machines Corporation`,
`International Bussiness Machine_ Corporation`) = 0.047
norm-ed(`IBM`, `BMW`) = 0.66
norm-ed(`Int. Business Machines Corp.`,
`International Business Machines Corporation`) = 0.4

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

norm-dist_q(`International Business Machines Corporation`,
`International Bussiness Machine_ Corporation`) = 0.089
norm-dist_q(`IBM`, `BMW`) = 1.0
norm-dist_q(`Int. Business Machines Corp.`,
`International Business Machines Corporation`) = 0.36

Edit Distance vs. q-Gram Distance

- Edit distance can not handle block-moves well:

$x =$ `Nikolaus Augsten` $y =$ `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` $y =$ `(39 06 467422)`
$\text{norm-ed}(x, y) = 0.4$
$\text{norm-dist}_q(x, y) = 1.0 \quad (q = 3)$

- Nikolaus Augsten, Michael Böhlen, and Johann Gamper.
The *pq*-gram distance between ordered labeled trees.
ACM Transactions on Database Systems (TODS), 35(1):1–36, 2010.

- Luis Gravano, Panagiotis G. Ipeirotis, H. V. Jagadish, Nick Koudas, S. Muthukrishnan, and Divesh Srivastava.
Approximate string joins in a database (almost) for free.
In *Proceedings of the International Conference on Very Large Databases (VLDB)*, pages 491–500, Roma, Italy, September 2001. Morgan Kaufmann Publishers Inc.

- Luis Gravano, Panagiotis G. Ipeirotis, H. V. Jagadish, Nick Koudas, S. Muthukrishnan, and Divesh Srivastava.
Approximate string joins in a database (almost) for free — Erratum.
Technical Report CUCS-011-03, Department of Computer Science, Columbia University, 2003.

- Esko Ukkonen.
Approximate string-matching with *q*-grams and maximal matches.

Theoretical Computer Science, 92(1):191–211, January 1992.