Set Similarity Joins - Introduction

Nikolaus Augsten

University of Salzburg, Austria

October 29, 2018

Thanks to Willi Mann for providing many of the slides.

How to find data in a database?

1. Scan: Search whole database
   - very slow (1TB from hard disk $\rightarrow$ lasts multiple hours)

How to find data in a database?

1. Scan: Search whole database
   - very slow (1TB from hard disk $\rightarrow$ lasts multiple hours)
2. Sort: phone book
   - Sorting allows directed search

How to find data in a database?

1. Scan: Search whole database
   - very slow (1TB from hard disk → lasts multiple hours)
2. Sort: phone book
   - Sorting allows directed search
3. Hashing: digit sum / pigeon holes
   - digit sum determines pigeon hole



Standard Approaches do not work

- Sort: Errors destroy ordering

Customers

Name ↓	Location
Frieda	Bozen
Frodo	Auenland
Maria	Meran

Customers

Name ↓	Location
Frieda	Bozen
Maria	Meran
Vrodo	Auenland

Standard Approaches do not work

- Sort: Errors destroy ordering

Customers

Name (A–Z ↓)	Location
Frieda	Bozen
Frodo	Auenland
Maria	Meran

Customers

Name (A–Z ↓)	Location
Frieda	Bozen
Maria	Meran
Vrodo	Auenland

- Hashing: Errors change digit sum

0	1	2	3	4
	Frodo Frieda		Maria	

0	1	2	3	4
	Frieda	Vrodo	Maria	

Similarity of Sets

- Many similarity criteria can be expressed as set overlaps
- Examples:
  - Words in text documents or websites
  - Friends on Facebook
  - Products in a shopping basket
  - Tags on Flickr
  - Click stream: Links clicked by a user
  - ...

Set Similarity Join
Definition

- $R, S \ldots$ collections of sets

Set Similarity Join
Definition

- $R, S \ldots$ collections of sets
- Similarity function sim, e.g.
  - $O(x, y) = |x \cap y|$
  - $J(x, y) = \frac{|x \cap y|}{|x \cup y|}$

Set Similarity Join

Definition

- $R, S \ldots$ collections of sets
- Similarity function sim, e.g.
  - $O(x, y) = |x \cap y|$
  - $J(x, y) = \frac{|x \cap y|}{|x \cup y|}$
- Threshold t

Set Similarity Join

Definition

- $R, S \ldots$ collections of sets
- Similarity function sim, e.g.
  - $O(x, y) = |x \cap y|$
  - $J(x, y) = \frac{|x \cap y|}{|x \cup y|}$
- Threshold t

$$R \overset{\text{sim}}{\bowtie} S = \{(r, s) \in R \times S | \operatorname{sim}(r, s) \geq t\}$$

Similarity Join: Find all Similar Pairs

- Data set: Titles and authors of $n = 873.524$ publications (DBLP)
  - Sets with an average word count of 15 (min. 2, max: 289)
  - Zipf-distributed word frequencies, 408.824 different words
  - Goal: find pairs of sets sharing many elements (e.g., 90%)

$$\text{sim}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Similarity Join: Find all Similar Pairs

- Data set: Titles and authors of $n = 873.524$ publications (DBLP)
  - Sets with an average word count of 15 (min. 2, max: 289)
  - Zipf-distributed word frequencies, 408.824 different words
  - Goal: find pairs of sets sharing many elements (e.g., 90%)

$$\text{sim}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

- Naive solution: Compute all overlaps
  1. Compute overlap for all pairs of sets
  2. Return pairs for which the overlap is large enough
     $\rightarrow$ very inefficient: $O(n^2)$ computed overlaps (˜381 billion)

Clever Solution

- Compare lengths and start computation for sets of similar size
- Terminate as soon as it is guaranteed the overlap is not reachable anymore
  $\rightarrow$ single comparisons are very efficient (6-20ns)

Clever Solution

- Compare lengths and start computation for sets of similar size
- Terminate as soon as it is guaranteed the overlap is not reachable anymore
  $\rightarrow$ single comparisons are very efficient (6-20ns)
- However, still too many comparisons ...
  $\rightarrow$ still $O(n^2)$ comparisons (˜381 billion)

Overlap	Runtime
0.95%	2201 s
0.90%	3136 s
0.80 %	4280 s
0.70 %	7770 s

Complexity

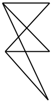
R	S
$\{a, b\}_{r_1}$	$\{a, b\}_{s_1}$
$\{d, g\}_{r_2}$	$\{c, e\}_{s_2}$
$\{c, f\}_{r_3}$	$\{c, g\}_{s_3}$

Complexity

R	S
$\{a, b\}_{r_1}$	$\{a, b\}_{s_1}$
$\{d, g\}_{r_2}$	$\{c, e\}_{s_2}$
$\{c, f\}_{r_3}$	$\{c, g\}_{s_3}$

$|r_1 \cap s_1| = 2 \quad |r_1 \cap s_2| = 0 \quad |r_1 \cap s_3| = 0$

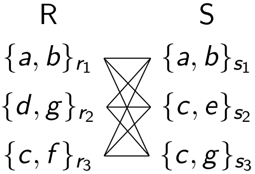
Complexity

R		S
$\{a, b\}_{r_1}$		$\{a, b\}_{s_1}$
$\{d, g\}_{r_2}$		$\{c, e\}_{s_2}$
$\{c, f\}_{r_3}$		$\{c, g\}_{s_3}$

$|r_1 \cap s_1| = 2 \quad |r_1 \cap s_2| = 0 \quad |r_1 \cap s_3| = 0$

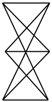
$|r_2 \cap s_1| = 0 \quad |r_2 \cap s_2| = 0 \quad |r_2 \cap s_3| = 1$

Complexity



$$
\begin{array}{lll}
|r_1 \cap s_1| = 2 & |r_1 \cap s_2| = 0 & |r_1 \cap s_3| = 0 \\
|r_2 \cap s_1| = 0 & |r_2 \cap s_2| = 0 & |r_2 \cap s_3| = 1 \\
|r_3 \cap s_1| = 0 & |r_3 \cap s_2| = 1 & |r_3 \cap s_3| = 1
\end{array}
$$

Complexity / Inverted List Index

R		S
$\{a, b\}_{r_1}$		$\{a, b\}_{s_1}$
$\{d, g\}_{r_2}$		$\{c, e\}_{s_2}$
$\{c, f\}_{r_3}$		$\{c, g\}_{s_3}$

$|r_1 \cap s_1| = 2 \quad |r_1 \cap s_2| = 0 \quad |r_1 \cap s_3| = 0$

$|r_2 \cap s_1| = 0 \quad |r_2 \cap s_2| = 0 \quad |r_2 \cap s_3| = 1$

$|r_3 \cap s_1| = 0 \quad |r_3 \cap s_2| = 1 \quad |r_3 \cap s_3| = 1$

S

a: s_1

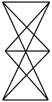
b: s_1

c: s_2, s_3

e: s_2

g: s_3

Complexity / Inverted List Index

R		S
$\{a, b\}_{r_1}$		$\{a, b\}_{s_1}$
$\{d, g\}_{r_2}$		$\{c, e\}_{s_2}$
$\{c, f\}_{r_3}$		$\{c, g\}_{s_3}$

$|r_1 \cap s_1| = 2 \quad |r_1 \cap s_2| = 0 \quad |r_1 \cap s_3| = 0$

$|r_2 \cap s_1| = 0 \quad |r_2 \cap s_2| = 0 \quad |r_2 \cap s_3| = 1$

$|r_3 \cap s_1| = 0 \quad |r_3 \cap s_2| = 1 \quad |r_3 \cap s_3| = 1$

R	S
	a: s_1
$\{a, b\}_{r_1}$	b: s_1
$\{d, g\}_{r_2}$	c: s_2, s_3
$\{c, f\}_{r_3}$	e: s_2
	g: s_3

Complexity / Inverted List Index

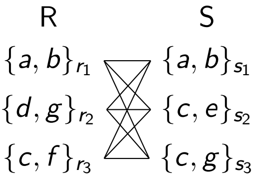
R		S
$\{a, b\}_{r_1}$		$\{a, b\}_{s_1}$
$\{d, g\}_{r_2}$		$\{c, e\}_{s_2}$
$\{c, f\}_{r_3}$		$\{c, g\}_{s_3}$

$\lvert r_1 \cap s_1 \rvert = 2$	$\lvert r_1 \cap s_2 \rvert = 0$	$\lvert r_1 \cap s_3 \rvert = 0$
$\lvert r_2 \cap s_1 \rvert = 0$	$\lvert r_2 \cap s_2 \rvert = 0$	$\lvert r_2 \cap s_3 \rvert = 1$
$\lvert r_3 \cap s_1 \rvert = 0$	$\lvert r_3 \cap s_2 \rvert = 1$	$\lvert r_3 \cap s_3 \rvert = 1$

R	S
	a: s_1
$\{a, b\}_{r_1}$	b: s_1
$\{d, g\}_{r_2}$	c: s_2, s_3
$\{c, f\}_{r_3}$	e: s_2
	g: s_3

$$|r_1 \cap s_1| = 2$$

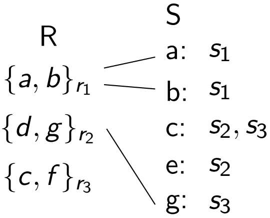
Complexity / Inverted List Index



$|r_1 \cap s_1| = 2$ $\quad |r_1 \cap s_2| = 0$ $\quad |r_1 \cap s_3| = 0$

$|r_2 \cap s_1| = 0$ $\quad |r_2 \cap s_2| = 0$ $\quad |r_2 \cap s_3| = 1$

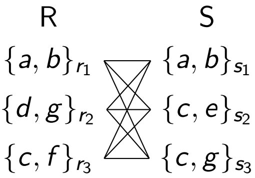
$|r_3 \cap s_1| = 0$ $\quad |r_3 \cap s_2| = 1$ $\quad |r_3 \cap s_3| = 1$



$|r_1 \cap s_1| = 2$

$|r_2 \cap s_3| = 1$

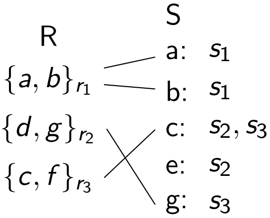
Complexity / Inverted List Index



$|r_1 \cap s_1| = 2$ $\quad |r_1 \cap s_2| = 0$ $\quad |r_1 \cap s_3| = 0$

$|r_2 \cap s_1| = 0$ $\quad |r_2 \cap s_2| = 0$ $\quad |r_2 \cap s_3| = 1$

$|r_3 \cap s_1| = 0$ $\quad |r_3 \cap s_2| = 1$ $\quad |r_3 \cap s_3| = 1$



$|r_1 \cap s_1| = 2$

$|r_2 \cap s_3| = 1$

$|r_3 \cap s_2| = 1$ $\quad |r_3 \cap s_3| = 1$

Background - Prefix Filter[1]

$|r \cap s| \geq 4$

- Example
  - 2 sets
  - Sorted alphabetically
  - Scan from left to right to compute overlap:

r:	?	?	?	?	?
s:	?	?	?	?	?

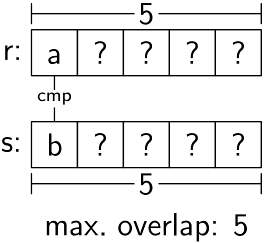
max. overlap: 5

[1] S. Chaudhuri, V. Ganti, and R. Kaushik. A primitive operator for similarity joins in data cleaning. In Proc. ICDE, pages 5–16, 2006.

Background - Prefix Filter[1]

$|r \cap s| \geq 4$

- Example
  - 2 sets
  - Sorted alphabetically
  - Scan from left to right to compute overlap:

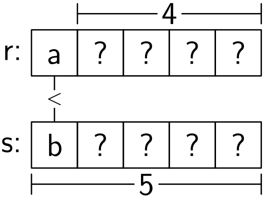


[1]S. Chaudhuri, V. Ganti, and R. Kaushik. A primitive operator for similarity joins in data cleaning. In Proc. ICDE, pages 5–16, 2006.

Background - Prefix Filter[1]

$|r \cap s| \geq 4$

- Example
  - 2 sets
  - Sorted alphabetically
  - Scan from left to right to compute overlap:



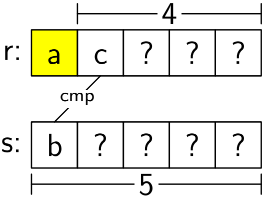
max. overlap: 4

[1] S. Chaudhuri, V. Ganti, and R. Kaushik. A primitive operator for similarity joins in data cleaning. In Proc. ICDE, pages 5–16, 2006.

Background - Prefix Filter[1]

$|r \cap s| \geq 4$

- Example
  - 2 sets
  - Sorted alphabetically
  - Scan from left to right to compute overlap:



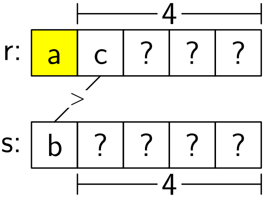
max. overlap: 4

[1] S. Chaudhuri, V. Ganti, and R. Kaushik. A primitive operator for similarity joins in data cleaning. In Proc. ICDE, pages 5–16, 2006.

Background - Prefix Filter[1]

$|r \cap s| \geq 4$

- Example
  - 2 sets
  - Sorted alphabetically
  - Scan from left to right to compute overlap:



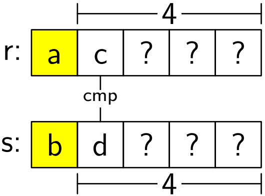
max. overlap: 4

[1]S. Chaudhuri, V. Ganti, and R. Kaushik. A primitive operator for similarity joins in data cleaning. In Proc. ICDE, pages 5–16, 2006.

Background - Prefix Filter[1]

$|r \cap s| \geq 4$

- Example
  - 2 sets
  - Sorted alphabetically
  - Scan from left to right to compute overlap:



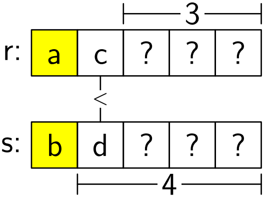
max. overlap: 4

[1] S. Chaudhuri, V. Ganti, and R. Kaushik. A primitive operator for similarity joins in data cleaning. In Proc. ICDE, pages 5–16, 2006.

Background - Prefix Filter[1]

$|r \cap s| \geq 4$

- Example
  - 2 sets
  - Sorted alphabetically
  - Scan from left to right to compute overlap:



max. overlap: 3

[1] S. Chaudhuri, V. Ganti, and R. Kaushik. A primitive operator for similarity joins in data cleaning. In Proc. ICDE, pages 5–16, 2006.

Background - Prefix Filter[1]

$|r \cap s| \geq 4$

- Example
  - 2 sets
  - Sorted alphabetically
  - Scan from left to right to compute overlap:

r:	a	c	e	f	g
s:	b	d	e	f	g

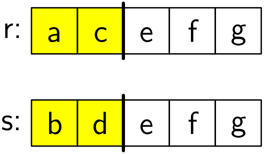
max. overlap: 3

[1]S. Chaudhuri, V. Ganti, and R. Kaushik. A primitive operator for similarity joins in data cleaning. In Proc. ICDE, pages 5–16, 2006.

Background - Prefix Filter[1]

$|r \cap s| \geq 4$

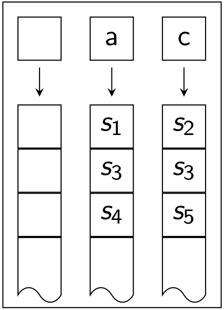
- Example
  - 2 sets
  - Sorted alphabetically
  - Scan from left to right to compute overlap:



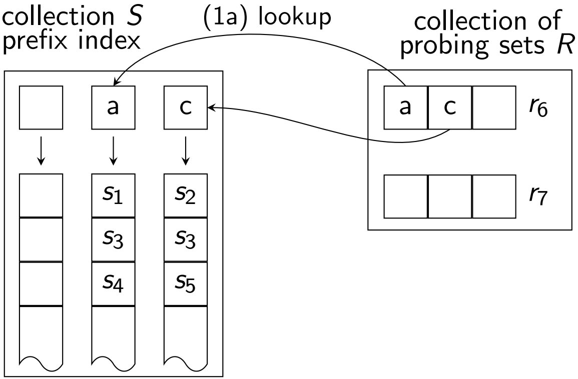
[1] S. Chaudhuri, V. Ganti, and R. Kaushik. A primitive operator for similarity joins in data cleaning. In Proc. ICDE, pages 5–16, 2006.

Background - Abstract Model of Algorithms

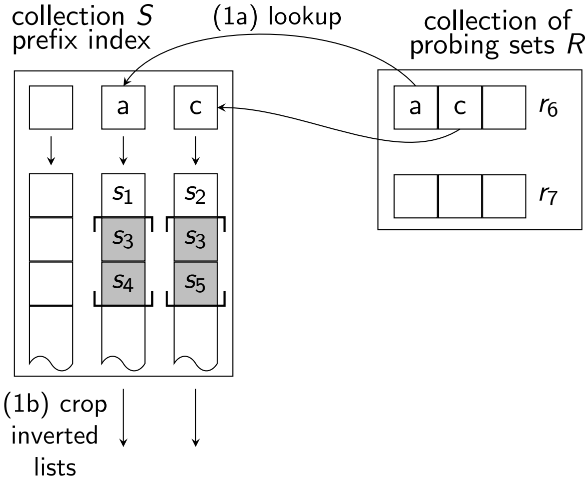
collection S
prefix index



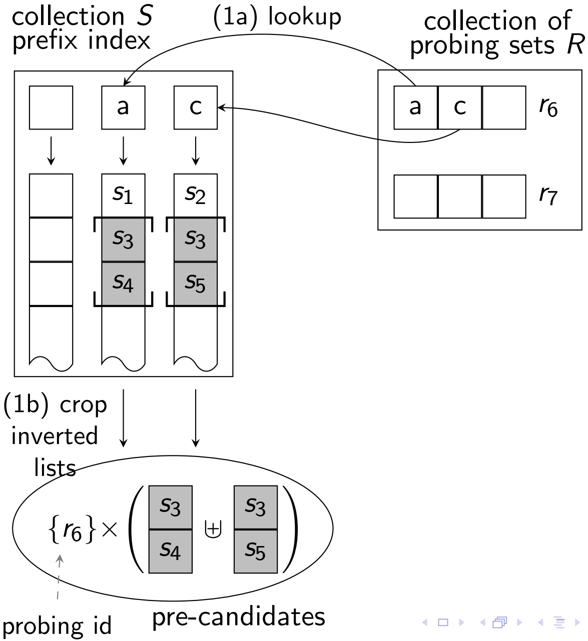
Background - Abstract Model of Algorithms



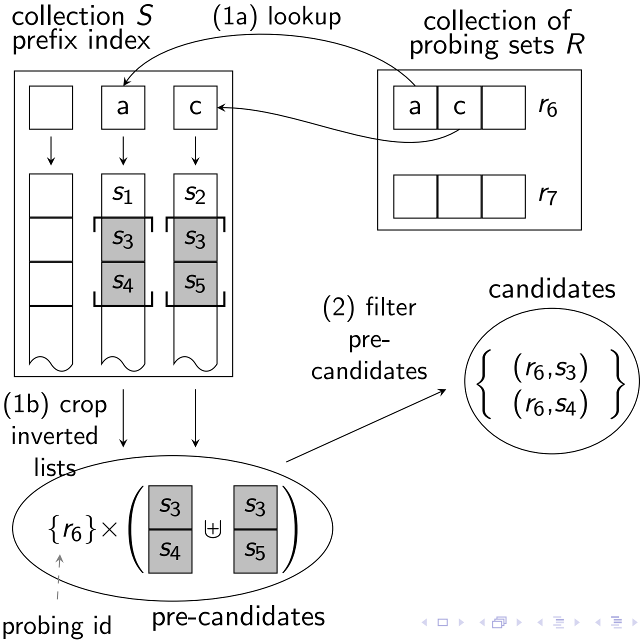
Background - Abstract Model of Algorithms



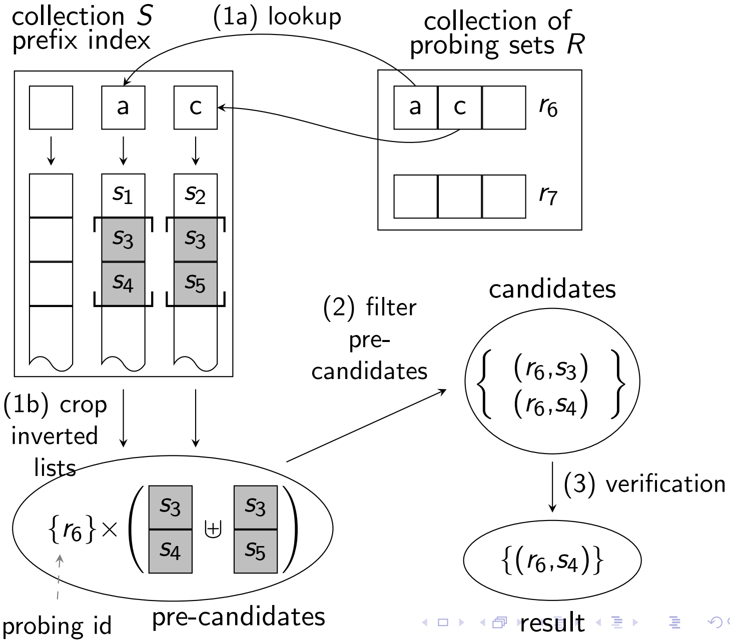
Background - Abstract Model of Algorithms



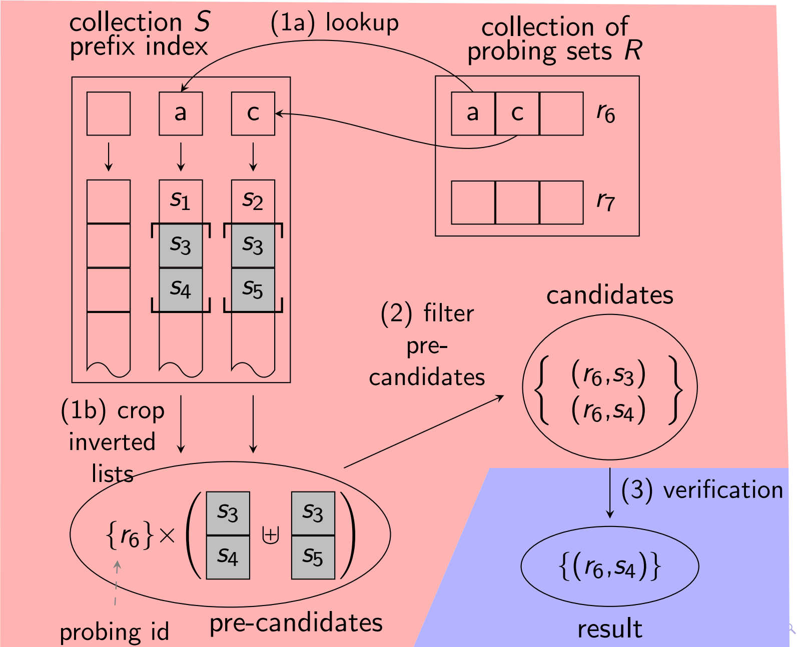
Background - Abstract Model of Algorithms



Background - Abstract Model of Algorithms



Background - Abstract Model of Algorithms



Preprocessing

Preprocessing:

Here you find the preprocessing instructions for the datasets in this project:
`http://ssjoin.dbresearch.uni-salzburg.at/datasets.html`

Preprocessing

Preprocessing:

1. Convert input objects into sets (arbitrary token data types).

Here you find the preprocessing instructions for the datasets in this project:

`http://ssjoin.dbresearch.uni-salzburg.at/datasets.html`

Preprocessing

Preprocessing:

1. Convert input objects into sets (arbitrary token data types).
2. Convert to integer sets. The larger the integer value, the more common the token is.

Here you find the preprocessing instructions for the datasets in this project:

`http://ssjoin.dbresearch.uni-salzburg.at/datasets.html`

Preprocessing

Preprocessing:

1. Convert input objects into sets (arbitrary token data types).
2. Convert to integer sets. The larger the integer value, the more common the token is.
3. Sort sets by tokens.

Here you find the preprocessing instructions for the datasets in this project:

`http://ssjoin.dbresearch.uni-salzburg.at/datasets.html`

Preprocessing

Preprocessing:

1. Convert input objects into sets (arbitrary token data types).
2. Convert to integer sets. The larger the integer value, the more common the token is.
3. Sort sets by tokens.
4. Sort set collection by set size.

Here you find the preprocessing instructions for the datasets in this project:

`http://ssjoin.dbresearch.uni-salzburg.at/datasets.html`