Similarity Search

Set Similarity Join

Nikolaus Augsten
nikolaus.augsten@plus.ac.at
Department of Computer Science
University of Salzburg

database research group
https://dbresearch.uni-salzburg.at

WS 2022/23

Version January 24, 2023

Outline

1. Filters for the Set Similarity Join
   - Motivation
   - Signature-based Filtering
   - Signatures for Overlap Similarity
   - Signatures for Hamming Distance
2. Implementations of Set Similarity Joins
   - Other Similarity Functions
   - Table of Set Similarity Join Algorithms and their Signatures
3. Conclusion

Outline

1. Filters for the Set Similarity Join
   - Motivation
   - Signature-based Filtering
   - Signatures for Overlap Similarity
   - Signatures for Hamming Distance
2. Implementations of Set Similarity Joins
   - Other Similarity Functions
   - Table of Set Similarity Join Algorithms and their Signatures
3. Conclusion

Application Scenario

- Scenario: A social network company stores user interests.
- Example: user table with interests:

R

id	name	interests
s	Sebastian	{bouldering, biking, swimming}
n	Nathan	{bouldering, swimming, guitar, singing}
p	Philippides	{hiking, running}
m	Maria	{bouldering, hiking, running}
r	Rosa	{bouldering, skiing, hiking}
...	...	...

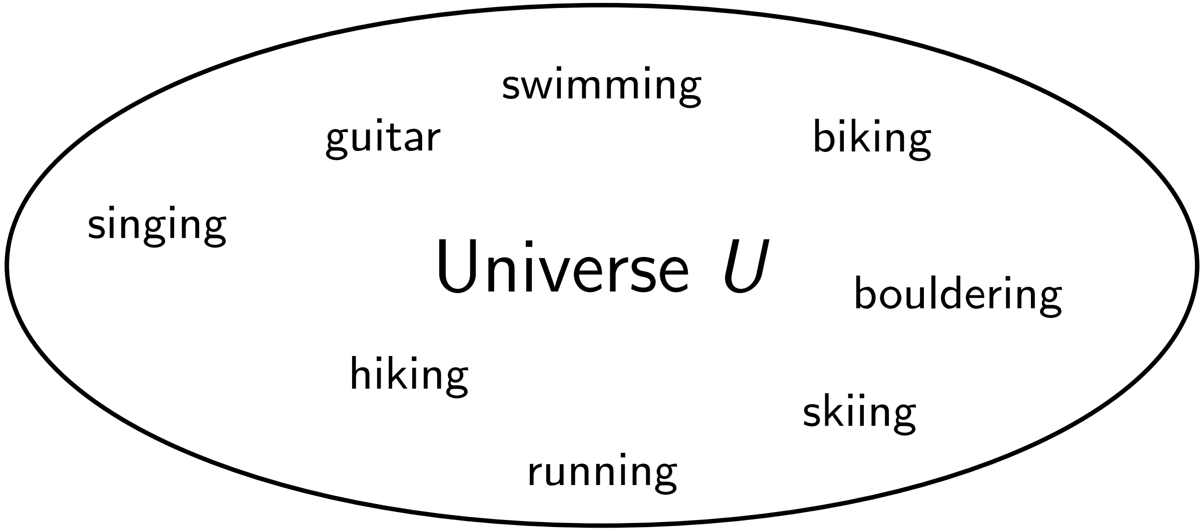
- Task: Recommend new friends based on similar interests!

Notation

collection

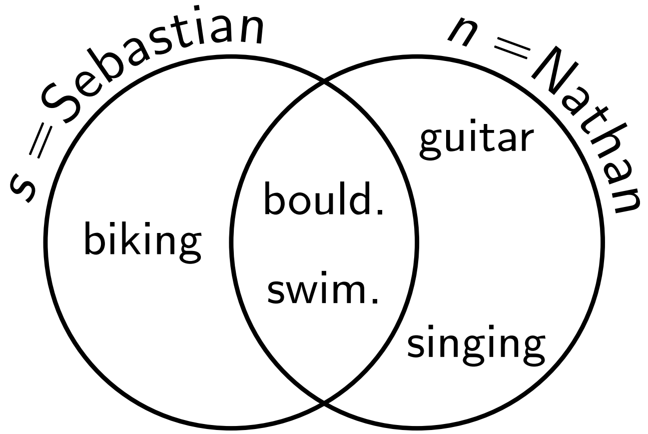
token

$$R = \{\underbrace{\{\text{bouldering}, \text{biking}, \text{swimming}\}}_{\text{set}}, \ldots\}$$



Measuring Similarity of Sets

- Goal: measure the similarity of two sets r, s
- Similarity Function:
  - $\text{Sim}(r, s)$ is *high* for similar sets, *low* for dissimilar sets
  - *Example:* Overlap $|r \cap s|$
- Distance Function:
  - $\text{Dis}(r, s)$ is *low* for similar sets, *high* for dissimilar sets
  - *Example:* Hamming distance $|r \triangle s| = |(r \backslash s) \cup (s \backslash r)| = |r \cup s| - |r \cap s|$
- Example:



$$|s \cap n| = 2$$
$$|s \triangle n| = 3$$

The Join Approach

- Solution: Compute the set similarity join

Definition (Set Similarity Join)

Given two collections of sets R and S, the set similarity join computes

$$R \mathbin{\tilde{\bowtie}} S = \{(r, s) \in R \times S \mid \mathrm{Sim}(r, s) \geqslant t\}$$

for a similarity function Sim or

$$R \mathbin{\tilde{\bowtie}} S = \{(r, s) \in R \times S \mid \mathrm{Dis}(r, s) \leqslant t\}$$

for a distance function Dis and threshold t.

- Naive Approach:
  1. Compute all pairs $R \times S$
  2. Test if $\mathrm{Sim}(r, s) \geqslant t$ or $\mathrm{Dis}(r, s) \leqslant t$ on each tuple

Naive Join Example

- Example: self-join $R \tilde{\bowtie} R$, overlap similarity, threshold $t = 2$

R

id	name	interests
s	Sebastian	{bouldering, biking, swimming}
n	Nathan	{bouldering, swimming, guitar, singing}
p	Philippides	{hiking, running}
m	Maria	{bouldering, hiking, running}
r	Rosa	{bouldering, skiing, hiking}

$$R \tilde{\bowtie} R = \{(s,n),(p,m),(m,r)\}$$

- 10 (non-reflexive, non-symmetric) comparisons!

Demonstration

- Experiment: Naive approach
  - self-join with varying $|R|$
  - average set size 10
  - universe size 1000, uniformly distributed
  - overlap similarity with threshold $t = 4$

$\lvert R \rvert$	#comparisons	runtime [s]
1000	$5 \cdot 10^5$	0.022
10000	$5 \cdot 10^7$	2.288
100000	$5 \cdot 10^9$	218.773

- A *single* similarity computation is fast (≈150 CPU cycles)
- But the search space grows fast: $\Theta(|R|^2)$

Outline

1. Filters for the Set Similarity Join
   - Motivation
   - Signature-based Filtering
   - Signatures for Overlap Similarity
   - Signatures for Hamming Distance
2. Implementations of Set Similarity Joins
   - Other Similarity Functions
   - Table of Set Similarity Join Algorithms and their Signatures
3. Conclusion

Reducing the Search Space using Filters

- Filtering: Reduce the search space by removing dissimilar pairs of sets
- Set similarity Join: Most filters are *signature*-based

Definition (Signature Scheme)

A signature scheme Sign is a function that maps a set of tokens to a set of signatures such that for any two sets of tokens, r and s:

$$\mathrm{Sim}(r, s) \geqslant t \Rightarrow \mathrm{Sign}(r) \cap \mathrm{Sign}(s) \neq \emptyset$$

for a similarity function Sim and

$$\mathrm{Dis}(r, s) \leqslant t \Rightarrow \mathrm{Sign}(r) \cap \mathrm{Sign}(s) \neq \emptyset$$

for a distance function Dis.

- Intuition: Similar sets share at least one signature.

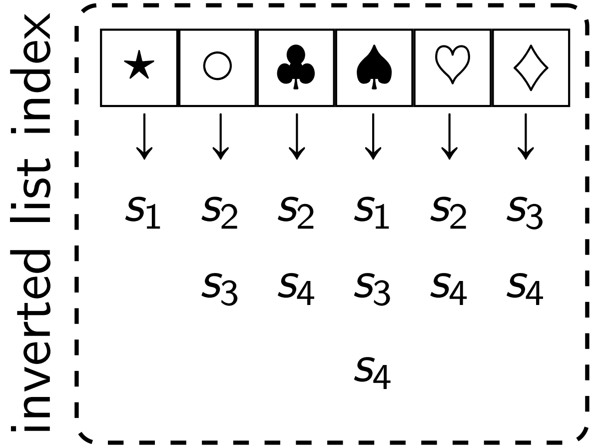
Signature-based Set Similarity Join

- Idea: Similar sets share signatures.
  1. Find all pairs sharing signatures (*candidates*)
  2. Test if $\text{Sim}(r, s) \geqslant t$ or $\text{Dis}(r, s) \leqslant t$ on each tuple
- How do we find pairs sharing signatures?
  1. Compute all pairs $R \times S$
  2. Test if $\text{Sign}(r) \cap \text{Sign}(s) \neq \emptyset$ on each tuple
- Likely slower than naive approach!
- Index: Build a simple index to find sets for each signature

Inverted-list Index

- Inverted-list Index: Stores mappings from content (e.g., signatures) to locations (e.g., sets)
  1. Compute signatures Sign(s) for set s
  2. Store a pointer to s in the list I_{sig} of each signature $sig \in \text{Sign}(s)$
- Example:

$$
\begin{aligned}
R &= \{s_1, s_2, s_3, s_4\} \\
\text{Sign}(s_1) &= \{\star, \spadesuit\} \\
\text{Sign}(s_2) &= \{\circ, \clubsuit, \heartsuit\} \\
\text{Sign}(s_3) &= \{\circ, \spadesuit, \diamondsuit\} \\
\text{Sign}(s_4) &= \{\clubsuit, \spadesuit, \heartsuit, \diamondsuit\}
\end{aligned}
$$



- A good signature scheme is both easy to compute and results in few false positives (= number of unnecessary verifications).

Signature-based Framework

Algorithm 1: Signature-based Framework

Data: Collection R, threshold t

Result: All similar pairs $M \subseteq R \times S$

```
I ← ∅  // inverted list index
forall s ∈ S do                                        // indexing
    forall signatures sig ∈ Sign(s) do
        I_sig ← I_sig ∪ {s}
M ← ∅, C ← ∅
forall r ∈ R do                                        // probing
    forall signatures sig ∈ Sign(r) do
        C ← C ∪ {(r, s) | s ∈ I_sig}
forall candidate pairs (r, s) ∈ C do
    M ← M ∪ (r, s) if Sim(r, s) ⩾ t        (or Dis(r, s) ⩽ t)
return M
```

Outline

1. Filters for the Set Similarity Join
   - Motivation
   - Signature-based Filtering
   - Signatures for Overlap Similarity
   - Signatures for Hamming Distance
2. Implementations of Set Similarity Joins
   - Other Similarity Functions
   - Table of Set Similarity Join Algorithms and their Signatures
3. Conclusion

Identity Signature

- Simplest signature scheme (for overlap) is identity (Sign = Id):

$$|r \cap s| \geqslant t \Rightarrow \mathrm{Id}(r) \cap \mathrm{Id}(s) \neq \varnothing$$
$$\Leftrightarrow |r \cap s| \geqslant t \Rightarrow |r \cap s| \geqslant 1 \qquad \text{assuming } t \geqslant 1$$

- Every token is a signature
- Example:

id	interests
s	{**B**ould., b**I**king, s**W**im.}
n	{**B**ould., s**W**im., **G**uitar, **S**ing.}
p	{**H**iking, **R**unning}
m	{**B**ould., **H**iking, **R**unning}
r	{**B**ould., s**K**iing, **H**iking}

inverted list index

B	I	W	G	S	H	R	K
↓	↓	↓	↓	↓	↓	↓	↓
s	s	s	n	n	p	p	r
n		n			m	m	
m					r		
r							

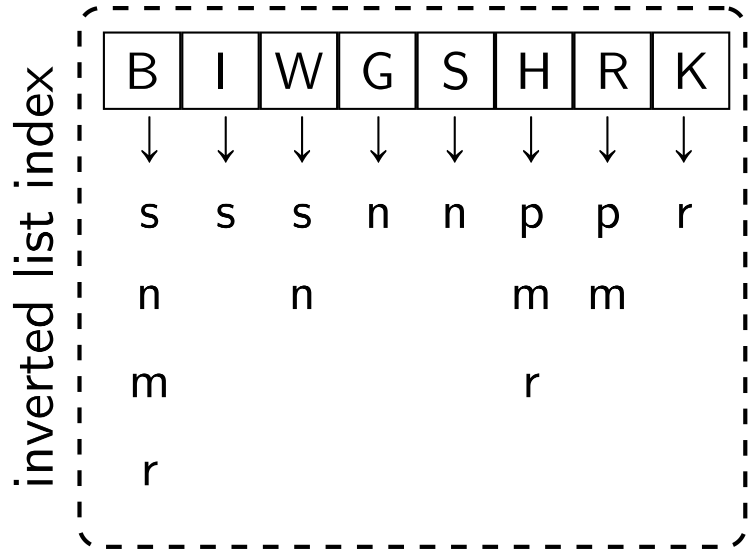
Identity Signature

- Simplest signature scheme (for overlap) is identity (Sign = Id):

$$|r \cap s| \geqslant t \Rightarrow \mathrm{Id}(r) \cap \mathrm{Id}(s) \neq \varnothing$$
$$\Leftrightarrow |r \cap s| \geqslant t \Rightarrow |r \cap s| \geqslant 1 \qquad \text{assuming } t \geqslant 1$$

- Every token is a signature
- Example:

id	interests
s	{B, I, W}
n	{B, W, G, S}
p	{H, R}
m	{B, H, R}
r	{B, K, H}



Identity Signature Example

id	interests
s	{B, I, W}
n	{B, W, G, S}
p	{H, R}
m	{B, H, R}
r	{B, K, H}

inverted list index

B	I	W	G	S	H	R	K
↓	↓	↓	↓	↓	↓	↓	↓
s	s	s	n	n	p	p	r
n		n			m	m	
m					r		
r							

- Probing:
  1. *Set s:* $C \leftarrow \emptyset \cup \{(s, n), (s, m), (s, r)\}$
  2. *Set n:* $C \leftarrow C \cup \{(n, m), (n, r)\}$
  3. *Set p:* $C \leftarrow C \cup \{(p, m), (p, r)\}$
  4. *Set m:* $C \leftarrow C \cup \{(m, r)\}$
- 8 (non-reflexive, non-symmetric) comparisons!
- Most candidates are the result of the long list B.

Demonstration

- Experiment: Identity signature[1]
  - self-join with varying $|R|$
  - average set size 10
  - universe sizes $|U| = 1000$ and $|U| = 10000$, uniformly distributed
  - overlap similarity with threshold $t = 4$

$\|R\|$	$\|U\|$	#comparisons	runtime [s]
1000	1000	$4.7 \cdot 10^4$	0.0
	10000	$4.8 \cdot 10^3$	0.0
10000	1000	$4.7 \cdot 10^6$	0.01
	10000	$5 \cdot 10^5$	0.005
100000	1000	$4.7 \cdot 10^8$	1.6
	10000	$4.9 \cdot 10^7$	0.19
1000000	1000	$4.7 \cdot 10^{10}$	241
	10000	$4.9 \cdot 10^9$	23

- Large improvement over naive approach
- Universe size heavily influences performance

[1] Implementation includes some optimizations compared to framework

Prefix Signature

- Idea: Exploit token order to construct a signature that is based on a *subset* of tokens.

Definition (Prefix Signature Scheme)

The prefix signature $\text{Pre}(r)$ of a set r for overlap threshold t is constructed as follows:

1. Order the tokens of r by any fixed global[a] order.
2. Each of the first $|r| - t + 1$ tokens in the ordered set is a prefix signature.

[a] global: the same order must be used for all sets

- Example:

Set n, $t = 2$	B	W	G	S
1. order (e.g., alphabetically)	B	G	S	W
2. take first $4 - 2 + 1 = 3$ tokens	B	G	S	W

- $\text{Pre}(n) = \{B, G, S\}$

Prefix Signature Correctness I

Lemma

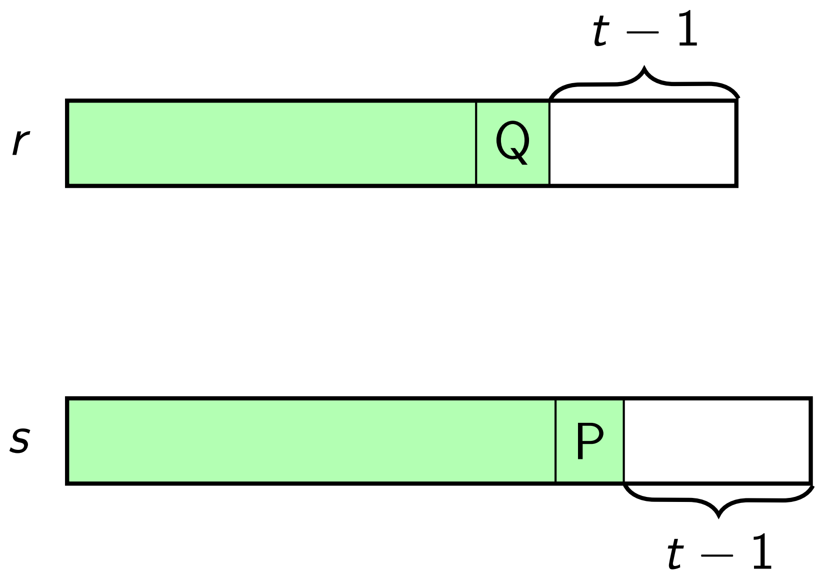
Pre *is a signature scheme for overlap similarity, i.e.,*

$$|r \cap s| \geqslant t \Rightarrow |\text{Pre}(r) \cap \text{Pre}(s)| \geqslant 1$$

We show the contraposition, i.e.,

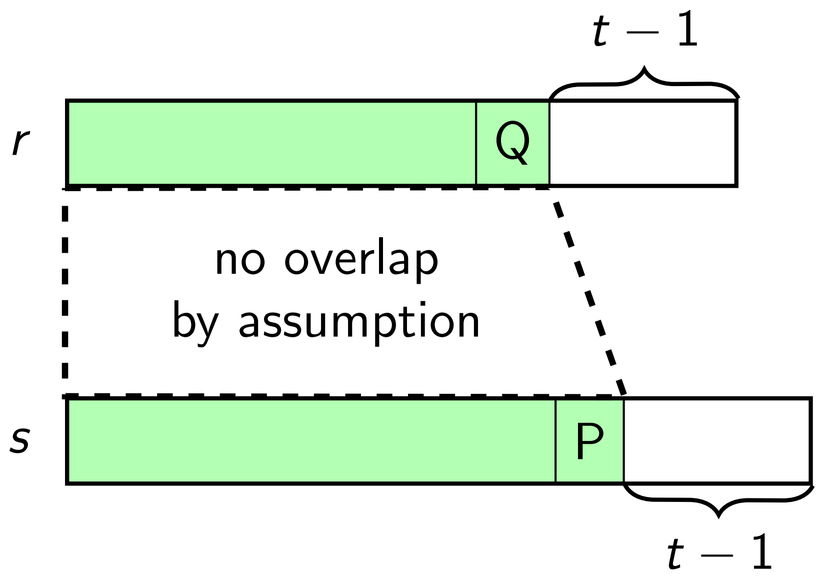
$$|\text{Pre}(r) \cap \text{Pre}(s)| = 0 \Rightarrow |r \cap s| < t$$

Prefix Signature Correctness II



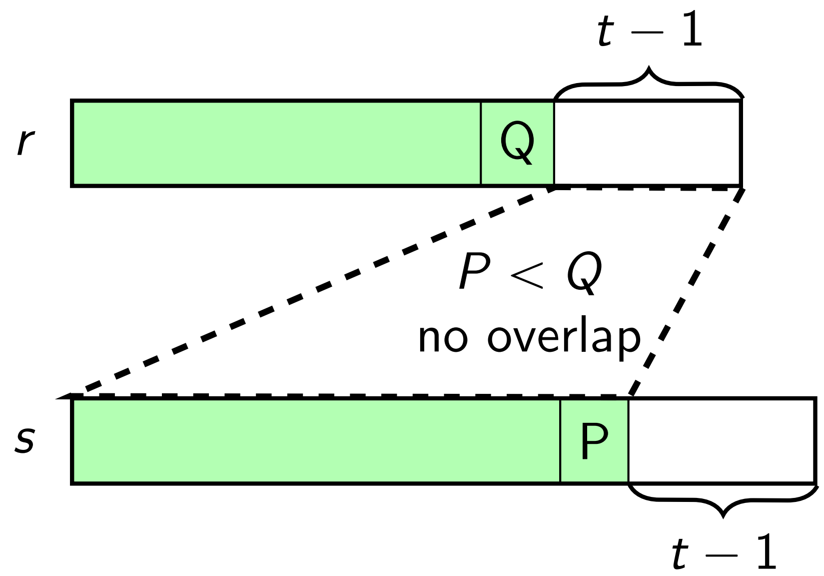
- Consider sets r, s; Q and P are the largest tokens in the respective prefixes, wlog. assume $P < Q$.
- Assume $\text{Pre}(r) \cap \text{Pre}(s) = \emptyset$. We bound $|r \cap s|$:
  - $|\text{Pre}(r) \cap \text{Pre}(s)| = 0$ by assumption
  - $|(r \setminus \text{Pre}(r)) \cap \text{Pre}(s)| = 0$ as $P < Q$
  - $|r \cap (s \setminus \text{Pre}(s))| \leqslant t - 1$ as $|s \setminus \text{Pre}(s)| = t - 1$
  - Hence, $|r \cap s| < t$.

Prefix Signature Correctness II



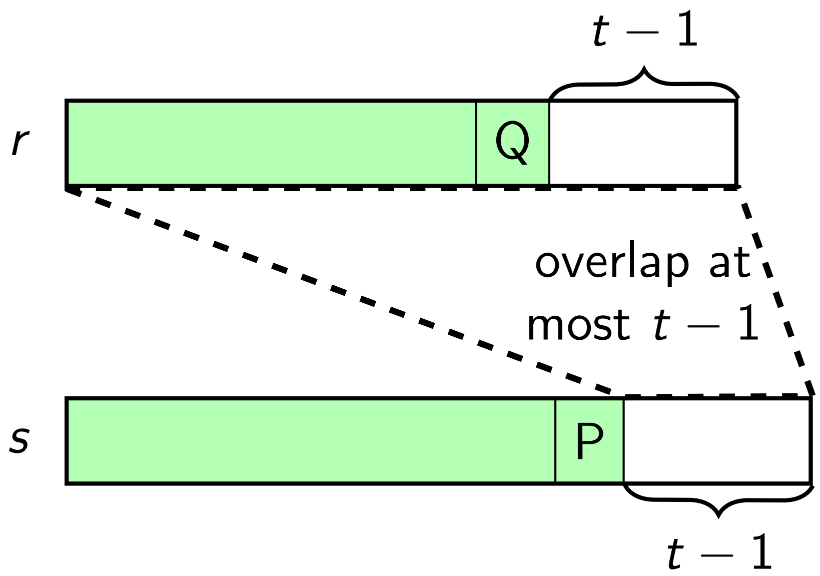
- Consider sets r, s; Q and P are the largest tokens in the respective prefixes, wlog. assume $P < Q$.
- Assume $\text{Pre}(r) \cap \text{Pre}(s) = \emptyset$. We bound $|r \cap s|$:
  - $|\text{Pre}(r) \cap \text{Pre}(s)| = 0$ by assumption
  - $|(r \setminus \text{Pre}(r)) \cap \text{Pre}(s)| = 0$ as $P < Q$
  - $|r \cap (s \setminus \text{Pre}(s))| \leqslant t - 1$ as $|s \setminus \text{Pre}(s)| = t - 1$
  - Hence, $|r \cap s| < t$.

Prefix Signature Correctness II



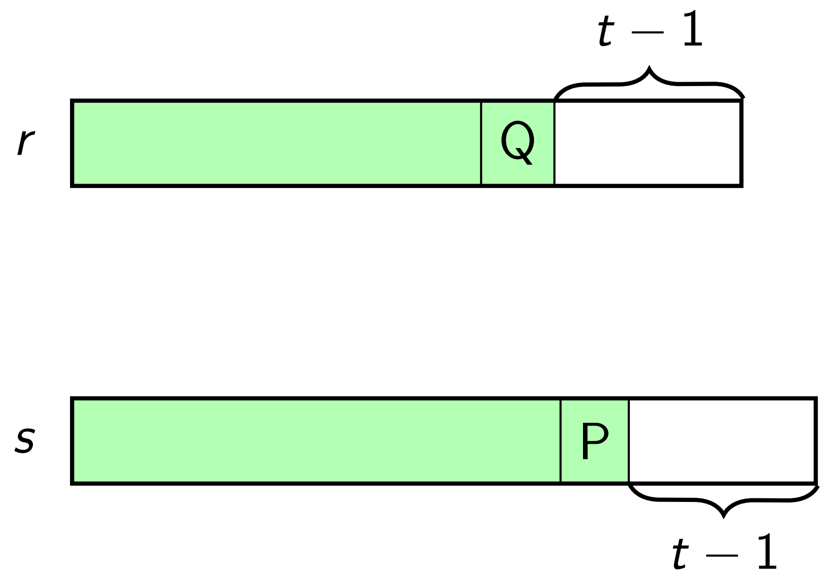
- Consider sets r, s; Q and P are the largest tokens in the respective prefixes, wlog. assume $P < Q$.
- Assume $\text{Pre}(r) \cap \text{Pre}(s) = \emptyset$. We bound $|r \cap s|$:
  - $|\text{Pre}(r) \cap \text{Pre}(s)| = 0$ by assumption
  - $|(r \setminus \text{Pre}(r)) \cap \text{Pre}(s)| = 0$ as $P < Q$
  - $|r \cap (s \setminus \text{Pre}(s))| \leqslant t - 1$ as $|s \setminus \text{Pre}(s)| = t - 1$
  - Hence, $|r \cap s| < t$.

Prefix Signature Correctness II



- Consider sets r, s; Q and P are the largest tokens in the respective prefixes, wlog. assume $P < Q$.
- Assume $\text{Pre}(r) \cap \text{Pre}(s) = \varnothing$. We bound $|r \cap s|$:
  - $|\text{Pre}(r) \cap \text{Pre}(s)| = 0$ by assumption
  - $|(r \setminus \text{Pre}(r)) \cap \text{Pre}(s)| = 0$ as $P < Q$
  - $|r \cap (s \setminus \text{Pre}(s))| \leqslant t - 1$ as $|s \setminus \text{Pre}(s)| = t - 1$
  - Hence, $|r \cap s| < t$.

Prefix Signature Correctness II



- Consider sets r, s; Q and P are the largest tokens in the respective prefixes, wlog. assume $P < Q$.
- Assume $\text{Pre}(r) \cap \text{Pre}(s) = \emptyset$. We bound $|r \cap s|$:
  - $|\text{Pre}(r) \cap \text{Pre}(s)| = 0$ by assumption
  - $|(r \setminus \text{Pre}(r)) \cap \text{Pre}(s)| = 0$ as $P < Q$
  - $|r \cap (s \setminus \text{Pre}(s))| \leqslant t - 1$ as $|s \setminus \text{Pre}(s)| = t - 1$
  - Hence, $|r \cap s| < t$.

Prefix Signature Example

id	interests (ordered alphabetically)
s	{B, I, W}
n	{B, G, S, W}
p	{H, R}
m	{B, H, R}
r	{B, H, K}

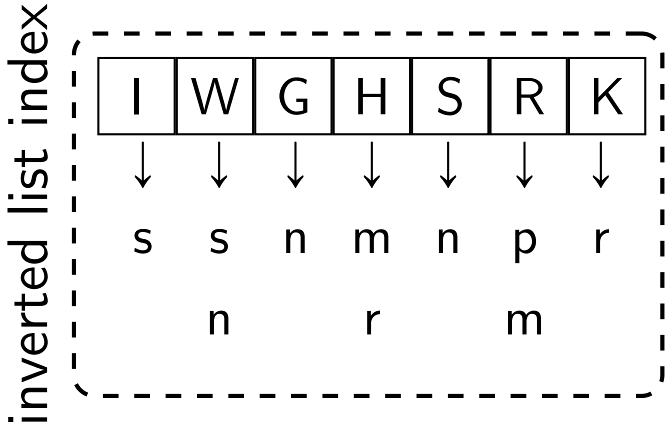
inverted list index

B	I	G	H	S
↓	↓	↓	↓	↓
s	s	n	p	n
n			m	
m			r	
r				

- Overlap threshold $t = 2$
- Indexing: all tokens except the last, alphabetical order
- Probing:
  1. *Set s:* $C \leftarrow \emptyset \cup \{(s, n), (s, m), (s, r)\}$
  2. *Set n:* $C \leftarrow C \cup \{(n, m), (n, r)\}$
  3. *Set p:* $C \leftarrow C \cup \{(p, m), (p, r)\}$
  4. *Set m:* $C \leftarrow C \cup \{(m, r)\}$
- still 8 comparisons!
- removing B from the prefix could help

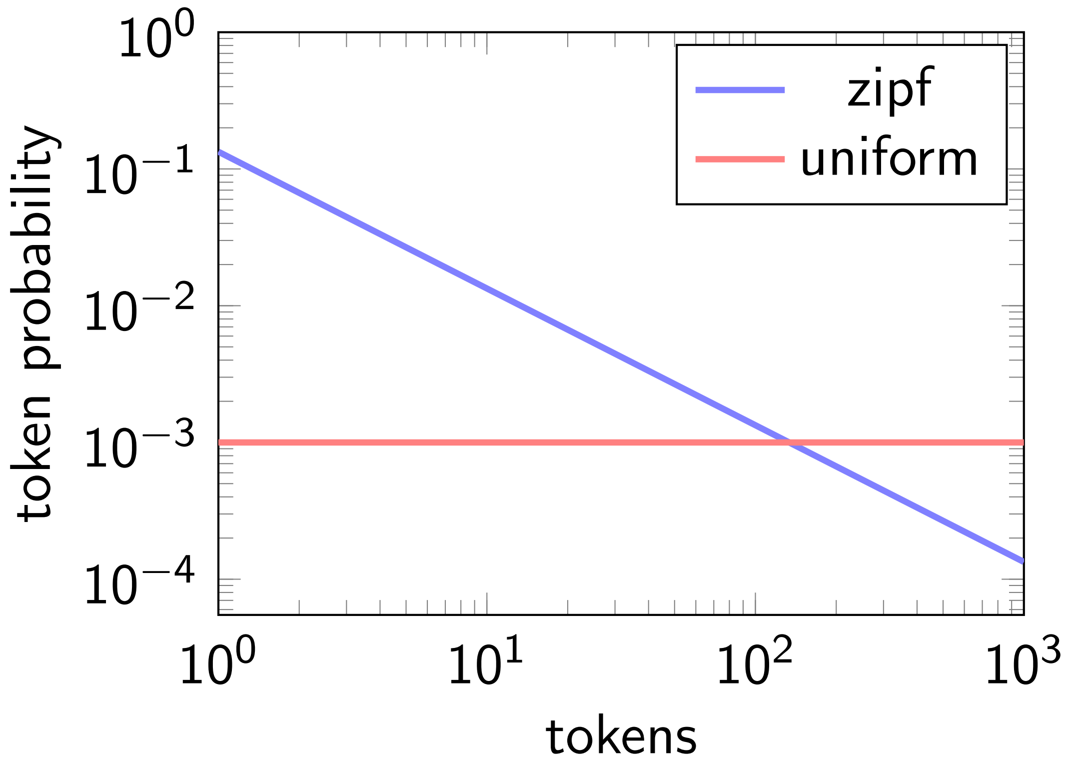
Prefix Signature Example: Order Makes a Difference

id	interests (ordered by frequency)
s	{I, W, B}
n	{G, S, W, B}
p	{R, H}
m	{R, H, B}
r	{K, H, B}



- Overlap threshold $t = 2$
- Indexing: all tokens except the last, ordered by ascending frequency
- Probing:
  1. *Set s:* $C \leftarrow \emptyset \cup \{(s, n)\}$
  2. *Set n:* $C \leftarrow C \cup \emptyset$
  3. *Set p:* $C \leftarrow C \cup \{(p, m)\}$
  4. *Set m:* $C \leftarrow C \cup \{(m, r)\}$
- only 3 comparisons!
- Heuristic: Ordering by ascending token frequency reduces candidates

Two Distributions



- Distribution: Real-world set-data often follow a zipfian distribution
- Skew: Some tokens appear frequently, a large number of tokens is uncommon. This favors the prefix signature.

Demonstration

- Experiment: Identity signature vs. Prefix signature[2]
  - self-join with $|R| = 100000$
  - average set size 10
  - universe sizes $|U| = 10000$, uniform and zipfian distribution
  - overlap similarity with threshold $t \in \{4, 6, 8\}$
  - global order: ascending token frequency

Identity

dist.	t	#comp.	runtime [s]
uni.	4	$5.0 \cdot 10^7$	0.187
	6	$5.0 \cdot 10^7$	0.186
	8	$5.0 \cdot 10^7$	0.186
zipf	4	$3.4 \cdot 10^9$	16.358
	6	$3.4 \cdot 10^9$	16.862
	8	$3.4 \cdot 10^9$	16.842

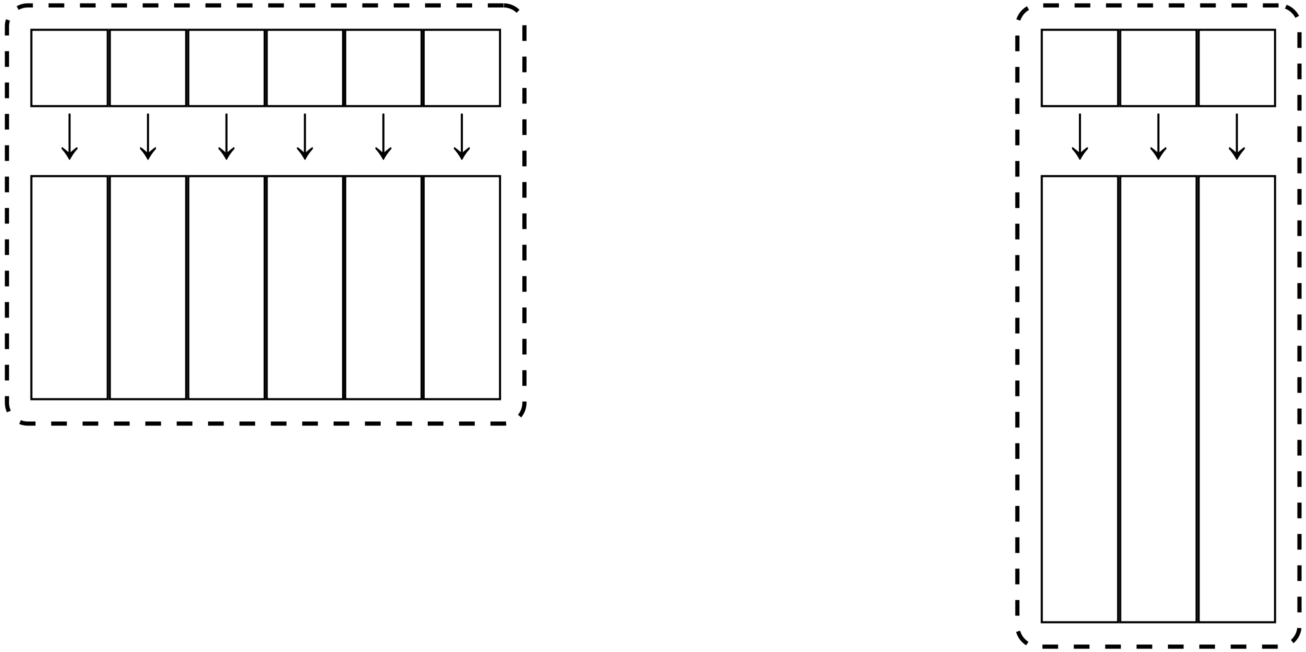
Prefix

dist.	t	#comp.	runtime [s]
uni.	4	$2.9 \cdot 10^7$	0.449
	6	$1.8 \cdot 10^7$	0.349
	8	$8.9 \cdot 10^6$	0.145
zipf	4	$3.1 \cdot 10^8$	3.935
	6	$6.2 \cdot 10^7$	0.873
	8	$1.2 \cdot 10^7$	0.197

[2] Implementation includes some optimizations compared to framework

Impact of Universe Size

- Identity and Prefix: individual tokens used as signatures
- Runtime: proportional to sum of all pairs in each list
- Problem: small universe size reduces filtering effectiveness



- Uniform distribution: halving the universe doubles the list lengths and runtime
- Idea: use a more selective signature than individual tokens

Subset Signature I

Lemma

$$|r \cap s| \geqslant t$$

$$\Leftrightarrow$$

$$\exists p \subseteq U : |p| = t \wedge p \subseteq r \wedge p \subseteq s$$

- Similar sets have at least one common subset of size t. This proves correctness of the following signature:

Definition (Subsets Signature)

The subsets signature $\text{Sub}(r)$ of a set r is defined as:

$$\text{Sub}(r) = \{p \subseteq r \mid |p| = t\}$$

for overlap threshold t.

Subset Signature II

- Sub is stronger than required for signature schemes. It also holds that

$$\text{Sign}(r) \cap \text{Sign}(s) \neq \emptyset \Rightarrow \text{Sim}(r, s) \geqslant t.$$

  Therefore, verification is not necessary.
- For set r and threshold t, we have $|\,\text{Sub}(r)| = \binom{|r|}{t}$, growing very quickly depending on both $|r|$ and t.
- Example: $n = \{B, W, G, S\}$, $t = 2$

$$\begin{aligned}\text{Sub}(n) = \{&\{B, W\}, \{B, G\}, \{B, S\}\\ &\{W, G\}, \{W, S\}, \{G, S\}\}\end{aligned}$$

Demonstration

- Experiment: Prefix signature vs. Subsets signature[3]
  - self-join with $|R| = 1000000$
  - average set size 6
  - universe sizes $|U| \in \{1000, 10000\}$, uniform distribution
  - overlap similarity with threshold $t \in \{3, 4, 5\}$

Prefix

$\|U\|$	t	runtime [s]
1000	3	336
	4	233
	5	138
10000	3	40.7
	4	24.7
	5	14.9

Subsets

$\|U\|$	t	runtime [s]
1000	3	25.6
	4	41.0
	5	42.8
10000	3	32.0
	4	53.9
	5	66.2

- Sub outperforms Pre for small set sizes and small universes
- Pre scales better wrt. set size and threshold for large universes

[3] Implementation includes some optimizations compared to framework

Outline

1. Filters for the Set Similarity Join
    - Motivation
    - Signature-based Filtering
    - Signatures for Overlap Similarity
    - Signatures for Hamming Distance
2. Implementations of Set Similarity Joins
    - Other Similarity Functions
    - Table of Set Similarity Join Algorithms and their Signatures
3. Conclusion

Partitioning I

Definition (Partition)

A partition P of universe U is a family of sets $P = \{p_1, \ldots, p_n\}$ with the following properties:

1. $\emptyset \notin P$
2. $\bigcup_{p \in P} p = U$
3. For any $p_i, p_j, i \neq j$, we have $p_i \cap p_j = \emptyset$

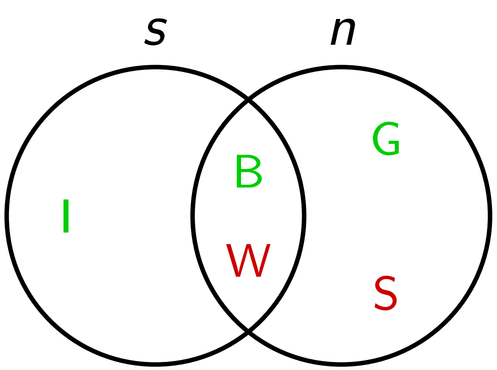
Lemma

For any partition P of universe U and any two sets $r, s \subseteq U$, we have

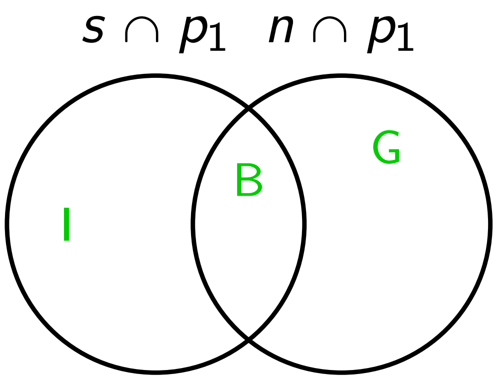
$$Ham(r, s) = \sum_{p \in P} Ham(r \cap p, s \cap p)$$

Partitioning II

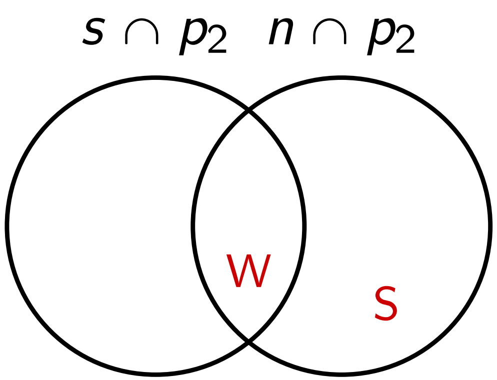
- Example: $P = \{\{B, G, H, I\}, \{K, R, S, W\}\} = \{p_1, p_2\}$



$$|s \triangle n| = 3$$



$$|(s \cap p_1) \triangle (n \cap p_1)| = 2$$



$$|(s \cap p_2) \triangle (n \cap p_2)| = 1$$

Partition Signature

Definition (Partition Signature)

The partition signature $\text{Par}(r)$ of a set r is constructed as follows:

1. Fix any partition P of U into $t+1$ parts (for Hamming distance t)
2. $\text{Par}(r) = \{(r \cap p_i, i) \mid p_i \in P\}$

- Example: $P = \{\{B, G, H, I\}, \{K, R, S, W\}\} = \{p_1, p_2\}$
  - $\text{Par}(\{I, B, W\}) = \{(\{I, B\}, 1), (\{W\}, 2)\}$
  - $\text{Par}(\{S, W\}) = \{(\emptyset, 1), (\{S, W\}, 2)\}$

Correctness of the Partition Signature

Lemma (Correctness of Par)

$$Ham(r, s) \leqslant t \Rightarrow \text{Par}(r) \cap \text{Par}(s) \neq \emptyset$$

We show the contraposition $\text{Par}(r) \cap \text{Par}(s) = \emptyset \Rightarrow Ham(r, s) > t$

Proof.

Assume $\text{Par}(r) \cap \text{Par}(s) = \emptyset$.

- For any $p_i \in P$, if $r \cap p_i \neq s \cap p_i$, then $Ham(r \cap p_i, s \cap p_i) \geqslant 1$.
- Hence, $Ham(r, s) = \sum_{p \in P} Ham(r \cap p, s \cap p) \geqslant |P| = t + 1 > t$.

□

Demonstration

- Experiment: Prefix signature[4] vs. Partition signature
  - self-join with $|R| = 1000000$
  - average set size 20
  - universe sizes $|U| \in \{1000, 10000\}$, uniform distribution
  - Hamming distance with threshold $t \in \{3, 4, 5\}$

Prefix

$\lvert U\rvert$	t	runtime [s]
1000	3	326
	4	451
	5	576
10000	3	34
	4	46
	5	70

Partition

$\lvert U\rvert$	t	runtime [s]
1000	3	4
	4	23
	5	144
10000	3	4
	4	23
	5	144

- Par outperforms Pre for large set sizes and small universes
- Par is less sensitive to universe size compared to Pre
- Pre works better for large universes and small set sizes

[4] Adapted to work with Hamming distance

The Empty Par Signature

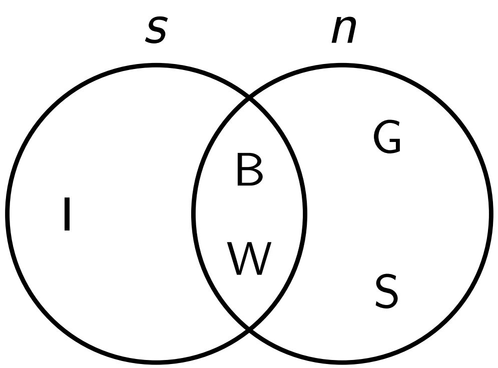
- Consider partitioning $P = \{\{A, B, C\}, \{D, E, F\}, \{G, H, I\}\}$, $t = 2$:

	A	B	C	D	E	F	G	H	I
r	1	1					1	1	
s	1		1				1		1
u		1	1					1	1

- Although all pairs of sets are at Hamming distance 4, they all share the red signature.
- Hence, all pairs of sets are candidates!
- This can happen for small sets or heavily skewed distributions.
- More sophisticated partitioning and more flexible searching in each partition can remedy this problem.

Enumeration: Idea

- How can we make s and n the same?



- By removing I from s and both S and G from n.
- Idea: By removing at most t tokens, all pairs of sets in Hamming distance t can be made equal.

Enumeration Signature

Definition (Enumeration Signature)

The enumeration signature $\mathrm{En}(r)$ of a set r for Hamming distance with threshold t is given by:

$$\mathrm{En}(r) = \{p \subseteq r \mid |p| \geqslant |r| - t\}$$

- Example: $t = 2$
  - $\mathrm{En}(\{I, B, G\}) = \{\{I, B, G\}, \{I, B\}, \{I, G\}, \{B, G\}, \{I\}, \{B\}, \{G\}\}$
  - $\mathrm{En}(\{H, R\}) = \{\{H, R\}, \{H\}, \{R\}, \varnothing\}$

Lemma (Correctness of En)

$$Ham(r, s) \leqslant t \Rightarrow \mathrm{En}(r) \cap \mathrm{En}(s) \neq \varnothing$$

Correctness of the Enumeration Signature

Proof.

- Assume $Ham(r, s) = |(r \backslash s) \cup (s \backslash r)| \leqslant t$.
- Hence, $|r \backslash (r \backslash s)| \geqslant |r| - t$ and $|s \backslash (s \backslash r)| \geqslant |s| - t$
- Consider the set $r \backslash (r \backslash s) = r \cap s = s \backslash (s \backslash r)$
- As $r \cap s \subseteq r$ and $|r \cap s| \geqslant |r| - t$, $r \cap s \in \text{En}(r)$.
- Similarly, $r \cap s \in \text{En}(s)$.
- So $r \cap s \in \text{En}(r) \cap \text{En}(s)$.

□

Demonstration

- Experiment: Partition signature vs. Enumeration signature[5]
  - self-join with $|R| = 1000000$
  - average set size $\overline{|r|} \in \{8, 16\}$
  - universe size $|U| = 10000$, uniform distribution
  - Hamming distance with threshold $t \in \{1, 2, 3\}$

Partition

$\overline{\lvert r \rvert}$	t	runtime [s]
8	1	4
	2	141
	3	1034
16	1	1
	2	3
	3	14

Enumeration

$\overline{\lvert r \rvert}$	t	runtime [s]
8	1	12
	2	42
	3	165
16	1	21
	2	170
	3	(out of memory)

- En can outperform Par for small thresholds and set sizes
- For large thresholds and sets, En generates too many signatures

[5] Implemented using an optimization called *asymmetric signature scheme* that avoids false positives.

Implementations of Set Similarity Joins

Real implementations of set similarity join algorithms typically

- also support similarity functions other than overlap and Hamming
- use a combination of multiple signature schemes
- extend the algorithmic framework to optimize for their signature schemes
- use additional filters (e.g., based on set *length* or the *position*s of matching signatures)

Outline

1. Filters for the Set Similarity Join
   - Motivation
   - Signature-based Filtering
   - Signatures for Overlap Similarity
   - Signatures for Hamming Distance
2. Implementations of Set Similarity Joins
   - Other Similarity Functions
   - Table of Set Similarity Join Algorithms and their Signatures
3. Conclusion

Other Similarity Functions

- Normalization: often, normalized similarity functions are preferred
  - $r = \{A, B, C\}, s = \{A, B\}, u = \{A, B, C, D, E, F, G, H, I\}$
  - The pair (r, u) has higher overlap than the pair (r, s)
  - Still, (r, s) might appear more similar due to fewer different tokens
  - Normalizations also consider set sizes and take values in $[0, 1]$
- Jaccard: $\text{Jac}(r, s) = \frac{|r \cap s|}{|r \cup s|} = \frac{|r \cap s|}{|r| + |s| - |r \cap s|}$
- Dice: $\text{Dice}(r, s) = \frac{2|r \cap s|}{|r| + |s|}$
- Cosine: $\text{Cos}(r, s) = \frac{|r \cap s|}{\sqrt{|r| \cdot |s|}}$
- Example:

$$\text{Jac}(r, s) = \frac{|\{A, B\}|}{|\{A, B, C\}|} = \frac{2}{3}$$

$$\text{Jac}(r, u) = \frac{|\{A, B, C\}|}{|\{A, B, C, D, E, F, G, H, I\}|} = \frac{3}{9} = \frac{1}{3}$$

Adapting the Prefix Signature for Jaccard

- The prefix signature operates with overlap similarity
- Idea: Bound minimum overlap s.t. two sets r, s can have $\text{Jac}(r, s) \geqslant t$

$$
\begin{aligned}
& \frac{|r \cap s|}{|r| + |s| - |r \cap s|} \geqslant t \\
\Leftrightarrow \quad & |r \cap s| \geqslant t(|r| + |s| - |r \cap s|) \\
\Leftrightarrow \quad & |r \cap s| \geqslant \frac{t}{1+t}(|r| + |s|) =: \text{eqo}_J(r, s)
\end{aligned}
$$

- eqo_J depends on the sizes of two sets. As we want to handle all possible size combinations, we have to get rid of one of them.
- Possible Solution: Assume minimal value $|s| = 1$, but better bounds are possible.

Length Bounds for Jaccard

Lemma

If $\text{Jac}(r,s) \geqslant t$*, then* $t|r| \leqslant |s| \leqslant \frac{|r|}{t}$.

Lemma

If $\text{Jac}(r,s) \geqslant t$*, then*

$$\begin{aligned}|r \cap s| &\geqslant \text{eqo}_J(r,s)\\ &\geqslant \text{eqo}_J(r,t|r|)\\ &= t|r|\end{aligned}$$

- Hence, for each set r use $\lceil t|r| \rceil$ as the overlap and proceed as in Pre.

Outline

1. Filters for the Set Similarity Join
   - Motivation
   - Signature-based Filtering
   - Signatures for Overlap Similarity
   - Signatures for Hamming Distance
2. Implementations of Set Similarity Joins
   - Other Similarity Functions
   - Table of Set Similarity Join Algorithms and their Signatures
3. Conclusion

Algorithm	Signature[6]	Remarks
AllPairs [BMS07]	Length + Pre	
PPJoin [XWLY08]	Length + Pre	additional filter based on position of prefix matches
SkipJoin [WQL$^+$19]	Length + Pre	tighter length filter using prefix positions, removes unmatchable entries from index, can leverage knowledge of set similarity "transitively"
SizeAware [DTL18]	Sub (small sets) + Id (large sets)	avoids enumerating all subsets in Sub by exploiting an order on subsets, handles small and large sets differently
PartEnum [AGK06]	Length + Par + En	partitions sets into smaller subets, enumerates in each partition
PartAlloc [DLWF15]	Length + Par + En	more flexible than PartEnum, has tighter filtering condition
GPH [QXW$^+$21]	Par + En	more flexible than PartAlloc, optimizes how partitions are chosen

[6] Refers to the closest signature discussed during the lecture; the listet algorithms often use a more efficient variation of the respectiv signatures.

Summary

- Naive set similarity join inefficient due to large search space
- Signature-based filters speed up join:
  - Id and Pre: single tokens as signatures
  - Sub: all subsets of overlap size as signatures
  - Par: non-overlapping subsets as signatures
  - En: all subsets in Hamming distance as signatures
- Performance depends on dataset's characteristics

Arvind Arasu, Venkatesh Ganti, and Raghav Kaushik.
Efficient exact set-similarity joins.
In Umeshwar Dayal, Kyu-Young Whang, David B. Lomet, Gustavo Alonso, Guy M. Lohman, Martin L. Kersten, Sang Kyun Cha, and Young-Kuk Kim, editors, *Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea, September 12-15, 2006*, pages 918–929. ACM, 2006.

Roberto J. Bayardo, Yiming Ma, and Ramakrishnan Srikant.
Scaling up all pairs similarity search.
In Carey L. Williamson, Mary Ellen Zurko, Peter F. Patel-Schneider, and Prashant J. Shenoy, editors, *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, pages 131–140. ACM, 2007.

Dong Deng, Guoliang Li, He Wen, and Jianhua Feng.
An efficient partition based method for exact set similarity joins.
Proc. VLDB Endow., 9(4):360–371, 2015.

Dong Deng, Yufei Tao, and Guoliang Li.
Overlap set similarity joins with theoretical guarantees.
In Gautam Das, Christopher M. Jermaine, and Philip A. Bernstein, editors, *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, pages 905–920. ACM, 2018.

Jianbin Qin, Chuan Xiao, Yaoshu Wang, Wei Wang, Xuemin Lin, Yoshiharu Ishikawa, and Guoren Wang.
Generalizing the pigeonhole principle for similarity search in hamming space.
IEEE Trans. Knowl. Data Eng., 33(2):489–505, 2021.

Xubo Wang, Lu Qin, Xuemin Lin, Ying Zhang, and Lijun Chang.
Leveraging set relations in exact and dynamic set similarity join.
VLDB J., 28(2):267–292, 2019.

Chuan Xiao, Wei Wang, Xuemin Lin, and Jeffrey Xu Yu.
Efficient similarity joins for near duplicate detection.

In Jinpeng Huai, Robin Chen, Hsiao-Wuen Hon, Yunhao Liu, Wei-Ying Ma, Andrew Tomkins, and Xiaodong Zhang, editors, *Proceedings of the 17th International Conference on World Wide Web, WWW 2008, Beijing, China, April 21-25, 2008*, pages 131–140. ACM, 2008.