

# Similarity Search

## Set Similarity Join

Nikolaus Augsten

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



WS 2025/26

Version November 6, 2025

## 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

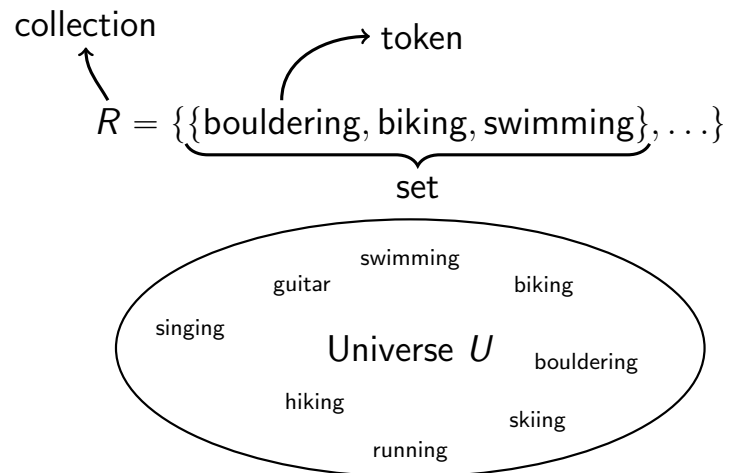
## Application Scenario

- **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}            |
| ... | ...         | ...                                     |

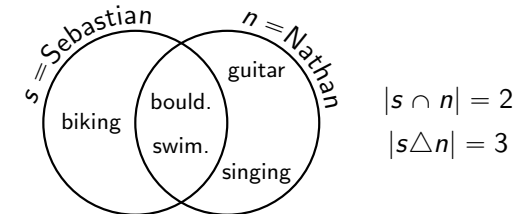
- **Task:** Recommend new friends based on similar interests!

## Notation



## 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  $|r \triangle s| = |(r \setminus s) \cup (s \setminus r)| = |r \cup s| - |r \cap s|$
- **Example:**



## 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

$$R \bowtie S = \{(r, s) \in R \times S \mid \text{Sim}(r, s) \geq t\}$$

for a similarity function  $\text{Sim}$  or

$$R \bowtie S = \{(r, s) \in R \times S \mid \text{Dis}(r, s) \leq t\}$$

for a distance function  $\text{Dis}$  and threshold  $t$ .

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

## Naive Join Example

- **Example:** self-join  $R \bowtie R$ , 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}            |

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

- **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  $t = 4$

| $ R $  | #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 ( $\approx 150$  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  $\text{Sign}$  is a function that maps a set of tokens to a set of signatures such that for any two sets of tokens,  $r$  and  $s$ :

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

for a similarity function  $\text{Sim}$  and

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

for a distance function  $\text{Dis}$ .

- **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  $\text{Sim}(r, s) \geq t$  or  $\text{Dis}(r, s) \leq t$  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  $\text{Sign}(s)$  for set  $s$
  2. Store a pointer to  $s$  in the list  $I_{sig}$  of each signature  $sig \in \text{Sign}(s)$

- **Example:**

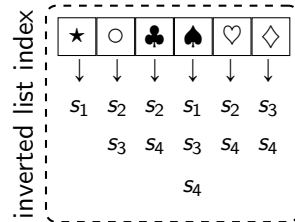
$$R = \{s_1, s_2, s_3, s_4\}$$

$$\text{Sign}(s_1) = \{\star, \spadesuit\}$$

$$\text{Sign}(s_2) = \{\circ, \clubsuit, \heartsuit\}$$

$$\text{Sign}(s_3) = \{\circ, \spadesuit, \diamondsuit\}$$

$$\text{Sign}(s_4) = \{\clubsuit, \spadesuit, \heartsuit, \diamondsuit\}$$



- 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  $M \subseteq R \times S$

$I \leftarrow \emptyset$  // inverted list index

**forall**  $s \in S$  **do** // indexing

**forall** signatures  $sig \in \text{Sign}(s)$  **do**

$I_{sig} \leftarrow I_{sig} \cup \{s\}$

$M \leftarrow \emptyset, C \leftarrow \emptyset$

**forall**  $r \in R$  **do** // probing

**forall** signatures  $sig \in \text{Sign}(r)$  **do**

$C \leftarrow C \cup \{(r, s) \mid s \in I_{sig}\}$

**forall** candidate pairs  $(r, s) \in C$  **do**

$M \leftarrow M \cup (r, s)$  if  $\text{Sim}(r, s) \geq t$  (or  $\text{Dis}(r, s) \leq t$ )

**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 ( $\text{Sign} = \text{Id}$ ):

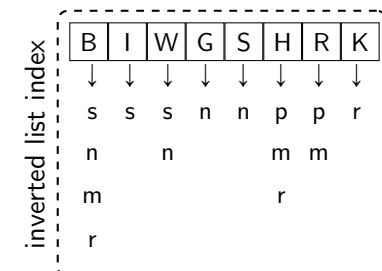
$$|r \cap s| \geq t \Rightarrow \text{Id}(r) \cap \text{Id}(s) \neq \emptyset$$

$$\Leftrightarrow |r \cap s| \geq t \Rightarrow |r \cap s| \geq 1 \quad \text{assuming } t \geq 1$$

- Every token is a signature

- **Example:**

| id | interests                      |
|----|--------------------------------|
| s  | {Bould., bKing, sWim.}         |
| n  | {Bould., sWim., Guitar, Sing.} |
| p  | {Hiking, Running}              |
| m  | {Bould., Hiking, Running}      |
| r  | {Bould., sKiing, Hiking}       |



## Identity Signature

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

$$|r \cap s| \geq t \Rightarrow \text{Id}(r) \cap \text{Id}(s) \neq \emptyset$$

$$\Leftrightarrow |r \cap s| \geq t \Rightarrow |r \cap s| \geq 1 \quad \text{assuming } t \geq 1$$

- 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}    |

| inverted list index | B | I | W | G | S | H | R | K |
|---------------------|---|---|---|---|---|---|---|---|
| s                   | s | s | s | n | n | p | p | r |
| n                   | n |   | n |   |   | m | m |   |
| m                   | m |   |   |   |   | r |   |   |
| r                   | r |   |   |   |   |   |   |   |

## 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 | s | n | n | p | p | r |
| n                   | n |   | n |   |   | m | m |   |
| m                   | m |   |   |   |   | r |   |   |
| r                   | r |   |   |   |   |   |   |   |

- Probing:

- Set  $s$ :  $C \leftarrow \emptyset \cup \{(s, n), (s, m), (s, r)\}$
- Set  $n$ :  $C \leftarrow C \cup \{(n, m), (n, r)\}$
- Set  $p$ :  $C \leftarrow C \cup \{(p, m), (p, r)\}$
- 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<sup>1</sup>
  - 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

<sup>1</sup>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:

- Order the tokens of  $r$  by any fixed global<sup>a</sup> order.
- Each of the first  $|r| - t + 1$  tokens in the ordered set is a prefix signature.

<sup>a</sup>global: the same order must be used for all sets

- Example:

Set  $n$ ,  $t = 2$

|   |   |   |   |
|---|---|---|---|
| B | W | G | S |
|---|---|---|---|

- order (e.g., alphabetically)

|   |   |   |   |
|---|---|---|---|
| B | G | S | W |
|---|---|---|---|

- take first  $4 - 2 + 1 = 3$  tokens

|   |   |   |   |
|---|---|---|---|
| B | G | S | W |
|---|---|---|---|

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

## Prefix Signature Correctness I

## Lemma

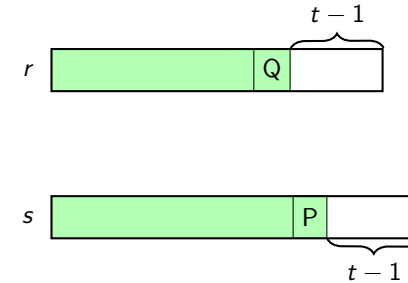
*Pre* is a signature scheme for overlap similarity, i.e.,

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

We show the contraposition, i.e.,

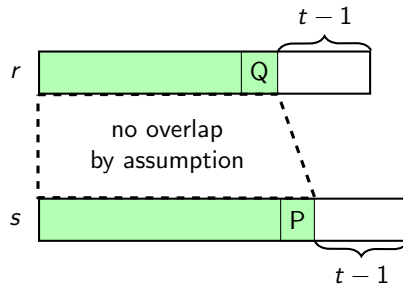
$$|\text{Pre}(r) \cap \text{Pre}(s)| = 0 \Rightarrow |r \cap s| < t$$

## Prefix Signature Correctness II



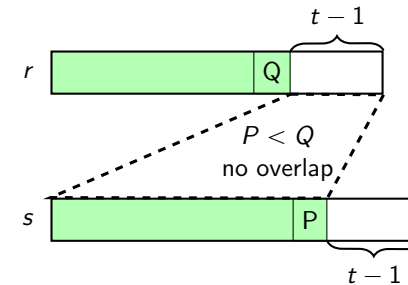
- Consider sets  $r, s$ ;  $Q$  and  $P$  are the largest tokens in the respective **prefixes**, wlog. assume  $P < Q$ .
- Assume  $\text{Pre}(r) \cap \text{Pre}(s) = \emptyset$ . We bound  $|r \cap s|$ :
  - $|\text{Pre}(r) \cap \text{Pre}(s)| = 0$  by assumption
  - $|(r \setminus \text{Pre}(r)) \cap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

## Prefix Signature Correctness II



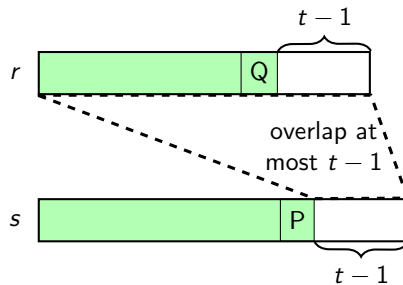
- Consider sets  $r, s$ ;  $Q$  and  $P$  are the largest tokens in the respective **prefixes**, wlog. assume  $P < Q$ .
- Assume  $\text{Pre}(r) \cap \text{Pre}(s) = \emptyset$ . We bound  $|r \cap s|$ :
  - $|\text{Pre}(r) \cap \text{Pre}(s)| = 0$  by assumption
  - $|(r \setminus \text{Pre}(r)) \cap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

## Prefix Signature Correctness II



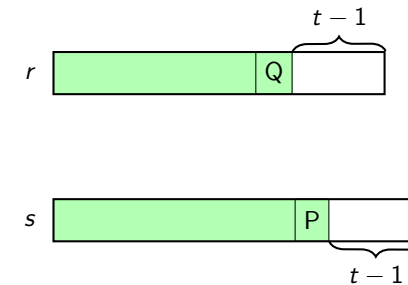
- Consider sets  $r, s$ ;  $Q$  and  $P$  are the largest tokens in the respective **prefixes**, wlog. assume  $P < Q$ .
- Assume  $\text{Pre}(r) \cap \text{Pre}(s) = \emptyset$ . We bound  $|r \cap s|$ :
  - $|\text{Pre}(r) \cap \text{Pre}(s)| = 0$  by assumption
  - $|(r \setminus \text{Pre}(r)) \cap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

## Prefix Signature Example

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

|   |   |   |   |   |
|---|---|---|---|---|
| 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**:
  - Set  $s$ :  $C \leftarrow \emptyset \cup \{(s, n), (s, m), (s, r)\}$
  - Set  $n$ :  $C \leftarrow C \cup \{(n, m), (n, r)\}$
  - Set  $p$ :  $C \leftarrow C \cup \{(p, m), (p, r)\}$
  - 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

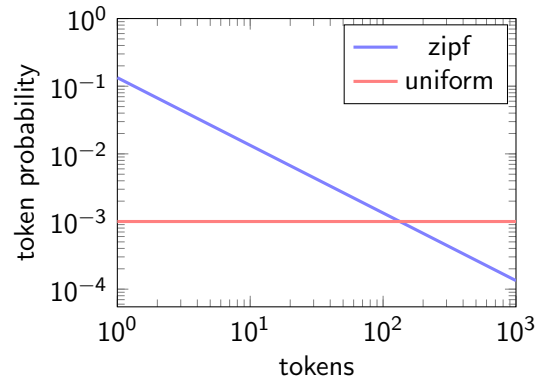
| id | interests (ordered by frequency) |                     |
|----|----------------------------------|---------------------|
| s  | {I, W, B}                        | inverted list index |
| n  | {G, S, W, B}                     |                     |
| p  | {R, H}                           |   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

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

- Overlap threshold  $t = 2$
- Indexing**: all tokens except the last, ordered by ascending frequency
- Probing**:
  - Set  $s$ :  $C \leftarrow \emptyset \cup \{(s, n)\}$
  - Set  $n$ :  $C \leftarrow C \cup \emptyset$
  - Set  $p$ :  $C \leftarrow C \cup \{(p, m)\}$
  - 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

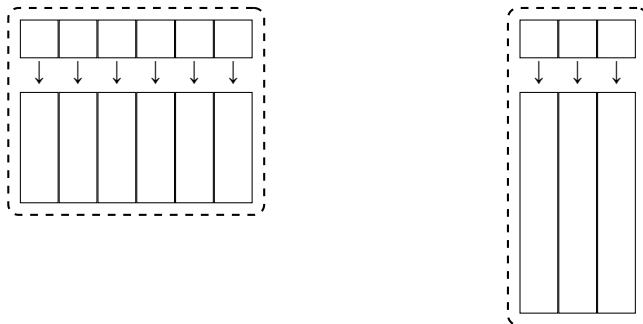
- **Experiment:** Identity signature vs. Prefix signature<sup>2</sup>
  - self-join with  $|R| = 100000$
  - average set size 10
  - universe sizes  $|U| = 10000$ , uniform and zipfian distribution
  - overlap similarity with threshold  $t \in \{4, 6, 8\}$
  - global order: ascending token frequency

| Identity |     |                  |             | Prefix |     |                  |             |
|----------|-----|------------------|-------------|--------|-----|------------------|-------------|
| dist.    | $t$ | #comp.           | runtime [s] | dist.  | $t$ | #comp.           | runtime [s] |
| uni.     | 4   | $5.0 \cdot 10^7$ | 0.187       | uni.   | 4   | $2.9 \cdot 10^7$ | 0.449       |
|          | 6   | $5.0 \cdot 10^7$ | 0.186       | uni.   | 6   | $1.8 \cdot 10^7$ | 0.349       |
|          | 8   | $5.0 \cdot 10^7$ | 0.186       | uni.   | 8   | $8.9 \cdot 10^6$ | 0.145       |
| zipf     | 4   | $3.4 \cdot 10^9$ | 16.358      | zipf   | 4   | $3.1 \cdot 10^8$ | 3.935       |
|          | 6   | $3.4 \cdot 10^9$ | 16.862      | zipf   | 6   | $6.2 \cdot 10^7$ | 0.873       |
|          | 8   | $3.4 \cdot 10^9$ | 16.842      | zipf   | 8   | $1.2 \cdot 10^7$ | 0.197       |

<sup>2</sup>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

$$|r \cap s| \geq t$$

$$\Leftrightarrow$$

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

- 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

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

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:**  $n = \{B, W, G, S\}$ ,  $t = 2$

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

## 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

## Demonstration

- **Experiment:** Prefix signature vs. Subsets signature<sup>3</sup>
  - 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 |     |             | Subsets |     |             |
|--------|-----|-------------|---------|-----|-------------|
| $ U $  | $t$ | runtime [s] | $ U $   | $t$ | runtime [s] |
| 1000   | 3   | 336         | 1000    | 3   | 25.6        |
|        | 4   | 233         |         | 4   | 41.0        |
|        | 5   | 138         |         | 5   | 42.8        |
| 10000  | 3   | 40.7        | 10000   | 3   | 32.0        |
|        | 4   | 24.7        |         | 4   | 53.9        |
|        | 5   | 14.9        |         | 5   | 66.2        |

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

<sup>3</sup>Implementation includes some optimizations compared to framework

## Partitioning I

### Definition (Partition)

A partition  $P$  of universe  $U$  is a family of sets  $P = \{p_1, \dots, p_n\}$  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  $p_i \cap p_j = \emptyset$

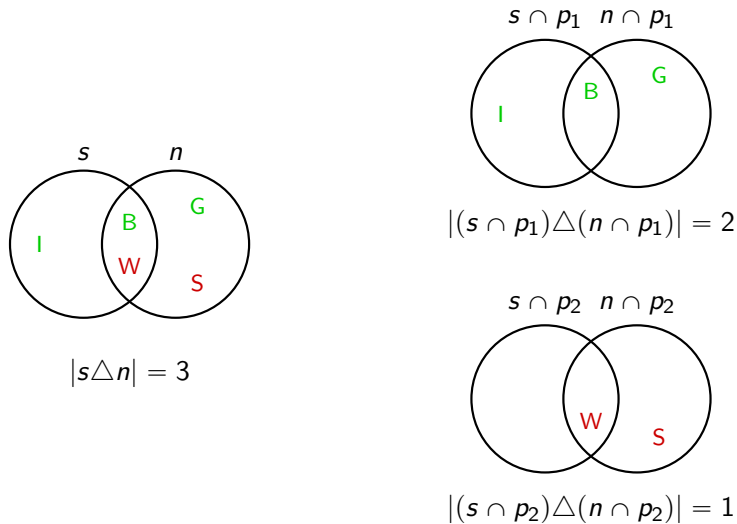
### Lemma

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

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

## Partitioning II

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



## Partition Signature

## Definition (Partition Signature)

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

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

- 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)

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

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

## 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  $\text{Ham}(r \cap p_i, s \cap p_i) \geq 1$ .
- Hence,  $\text{Ham}(r, s) = \sum_{p \in P} \text{Ham}(r \cap p, s \cap p) \geq |P| = t + 1 > t$ .

□

## Demonstration

- Experiment: Prefix signature<sup>4</sup> 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 |     |             | Partition |     |             |
|--------|-----|-------------|-----------|-----|-------------|
| $ U $  | $t$ | runtime [s] | $ U $     | $t$ | runtime [s] |
| 1000   | 3   | 326         | 1000      | 3   | 4           |
|        | 4   | 451         | 1000      | 4   | 23          |
|        | 5   | 576         | 1000      | 5   | 144         |
| 10000  | 3   | 34          | 10000     | 3   | 4           |
|        | 4   | 46          | 10000     | 4   | 23          |
|        | 5   | 70          | 10000     | 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

<sup>4</sup>Adapted to work with Hamming distance

## The Empty Par Signature

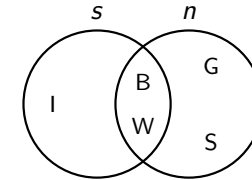
- Consider partitioning  $P = \{\{A, B, C\}, \{D, E, F\}, \{G, H, I\}\}$ ,  $t = 2$ :

|   | 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  $\text{En}(r)$  of a set  $r$  for Hamming distance with threshold  $t$  is given by:

$$\text{En}(r) = \{p \subseteq r \mid |p| \geq |r| - t\}$$

- Example:**  $t = 2$ 
  - $\text{En}(\{I, B, G\}) = \{\{I, B, G\}, \{I, B\}, \{I, G\}, \{B, G\}, \{I\}, \{B\}, \{G\}\}$
  - $\text{En}(\{H, R\}) = \{\{H, R\}, \{H\}, \{R\}, \emptyset\}$

### Lemma (Correctness of En)

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

## Correctness of the Enumeration Signature

### Proof.

- Assume  $\text{Ham}(r, s) = |(r \setminus s) \cup (s \setminus r)| \leq t$ .
- Hence,  $|r \setminus (r \setminus s)| \geq |r| - t$  and  $|s \setminus (s \setminus r)| \geq |s| - t$
- Consider the set  $r \setminus (r \setminus s) = r \cap s = s \setminus (s \setminus r)$
- As  $r \cap s \subseteq r$  and  $|r \cap s| \geq |r| - t$ ,  $r \cap s \in \text{En}(r)$ .
- 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<sup>5</sup>
  - self-join with  $|R| = 1000000$
  - average set size  $|r| \in \{8, 16\}$
  - universe size  $|U| = 10000$ , uniform distribution
  - Hamming distance with threshold  $t \in \{1, 2, 3\}$

| Partition |     |             | Enumeration |     |                 |
|-----------|-----|-------------|-------------|-----|-----------------|
| $ r $     | $t$ | runtime [s] | $ r $       | $t$ | runtime [s]     |
| 8         | 1   | 4           | 8           | 1   | 12              |
|           | 2   | 141         |             | 2   | 42              |
|           | 3   | 1034        |             | 3   | 165             |
| 16        | 1   | 1           | 16          | 1   | 21              |
|           | 2   | 3           |             | 2   | 170             |
|           | 3   | 14          |             | 3   | (out of memory) |

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

<sup>5</sup>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 *positions* 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  $\text{Jac}(r, s) \geq t$

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

- $\text{eqo}_J$  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.

## Outline

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

## Length Bounds for Jaccard

## Lemma

If  $\text{Jac}(r, s) \geq t$ , then  $t|r| \leq |s| \leq \frac{|r|}{t}$ .

## Lemma

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

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

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

| Algorithm                      | Signature <sup>6</sup>            | Remarks                                                                                                                                       |
|--------------------------------|-----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| AllPairs [BMS07]               | Length+ Pre                       |                                                                                                                                               |
| PPJoin [XWLY08]                | Length+ Pre                       | additional filter based on position of prefix matches                                                                                         |
| SkipJoin [WQL <sup>+</sup> 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 subsets, enumerates in each partition                                                                            |
| PartAlloc [DLWF15]             | Length+ Par+ En                   | more flexible than PartEnum, has tighter filtering condition                                                                                  |
| GPH [QXW <sup>+</sup> 21]      | Par+ En                           | more flexible than PartAlloc, optimizes how partitions are chosen                                                                             |

<sup>6</sup>Refers to the closest signature discussed during the lecture; the listed algorithms often use a more efficient variation of the respective 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

-  **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.

-  **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.

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.