PS Ähnlichkeitssuche in großen Datenbanken

Task 3 - Inverted List Index

Thomas Hütter, WS 2020

Exercise

Set Similarity Self Joins

- Given: a collections of sets *R*, a similarity function *Jaccard*, and a similarity threshold *0.75*.

R r_1 = | 3 | 8 | 9 | r_2 = | 2 | 4 | 7 | 9 | r_3 = | 2 | 4 | 6 | 7 | 9 |

- Task: calculate the number of “touched” elements for:

Nested Loop:

Length Filter:

Length+Prefix Filter:

Example

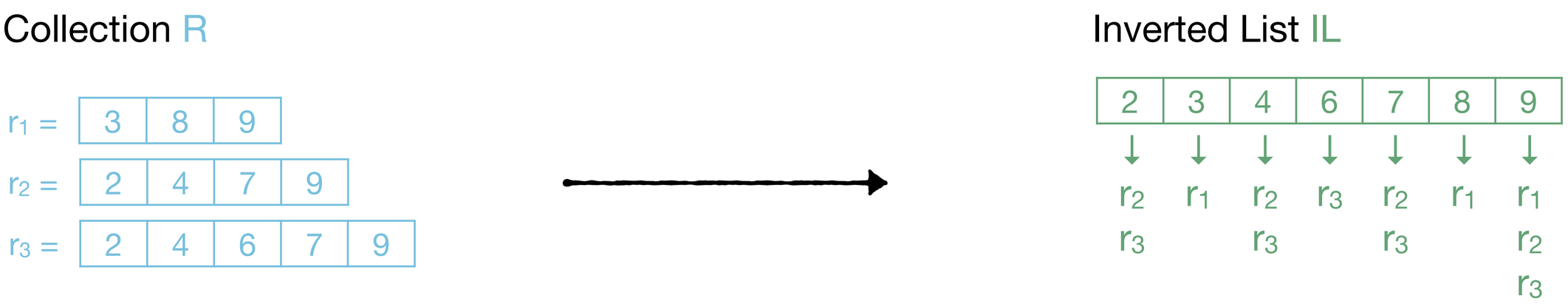
Set Similarity Self Joins

- Given: a collection of sets R with $|R| = 1000000$. Computing the Jaccard distance between any pair of sets $(r_1, r_2) \in R \times R$ takes 1µs.
- Computing the self-join takes approximately 6 days.

Inverted List Index

Optimization

- Motivation: considering all set pairs is expensive for large collections.
- Idea: lookup set pairs that are similar.
- Inverted List Index: given a set element, the index returns all sets of the collection that contain this particular element.



Algorithm: Index Nested Loop Join

Inverted List Index

- Collection R ordered by size and global order among all set elements.

```
Algorithm 3: setsimjoin(R, t_J):
input: R collection of sets, t_J similarity threshold
output: res set of result pairs (similarity at least t_J)
res = []; // Stores the result pairs.
I = {}; // Inverted list index.
for r in R:
    C = {}; // Candidates for set r.
    for pos = 0 to π_r - 1: // Apply prefix filter.
        for s in I[r[pos]]: // Sets with element r[pos].
            if |s| < |r| * t_J: // Apply length filter.
                remove s from I[r[pos]];
            else
                C = C ∪ s; // Increase intersection.
    for pos = 0 to π_r^I - 1:
        I[r[pos]] = I[r[pos]] ○ r; // Append r to index.
    for s in C:
        if verify(r, s, t_J):
            res = res ∪ (r, s);

return res;
```

Index lookup / candidate generation

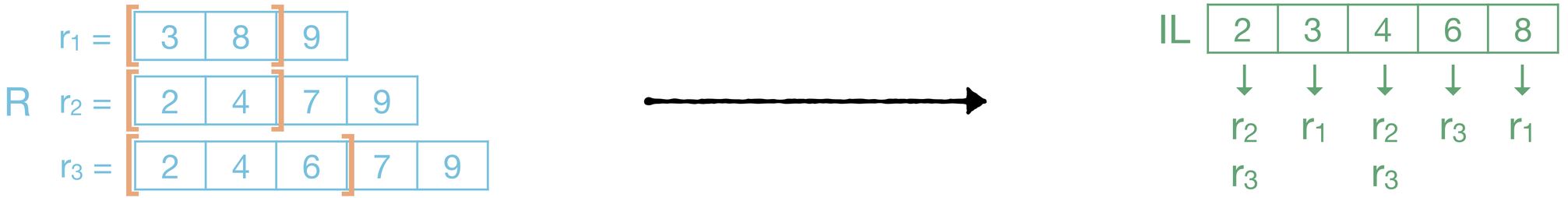
Build index incrementally

Verification

Inverted List Index

Build Index Incrementally

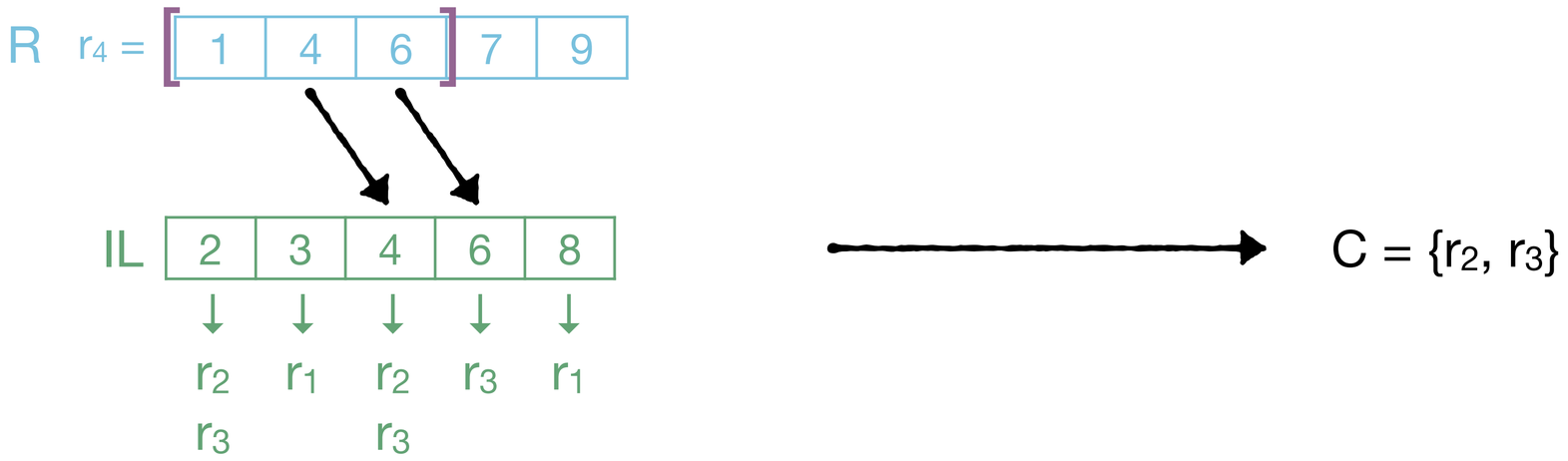
- Instead of building the full inverted list upfront, the index is incrementally build while processing the collection.
- Insert only the first π_r^I elements of each set r, called indexing prefix.



Inverted List Index

Index Lookup / Candidate Generation

- Index Lookup: only the first π_r elements of a set r, called probing prefix.



- Remove all pairs from the index that do not qualify due to the length filter.
  - Benefit: keeps the index small.

Hints

Task 3

- The indexing prefix may not be identical to the probing prefix, π_r^I != π_r.
- The prefix size is computed by $\pi = |r| - \lceil \frac{t_J}{t_J + 1} (|r| + |s|) \rceil + 1$.
  - Probing: |s| is not known. Approximate with the smallest set size in the index.
  - Indexing: Since the collection is processed in ascending size order, future probing sets are at least of size |r|.
- The needed intersection may be larger than the set size.

Homework

Task 3

- Use an inverted list compute the set similarity self join.
  - Correctness: perform all tests within 1 minute.
  - Compute the minimal probing and indexing prefix length. (0.5 points each)
  - Inverted list implementation (data structure, removal, etc.). (0.5 points)
- Verify your implementation with given datasets on the teaching website. Further datasets can be found at http://ssjoin.dbresearch.uni-salzburg.at/datasets.html.
- Follow the submission guidelines written on the teaching website.
- Submit via abgaben.cosy.sbg.ac.at until 15.12.2020, 23:55.