PS Ähnlichkeitssuche in großen Datenbanken

Task 4 - Efficient Verification

Thomas Hütter, WS 2020

Task 3

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; // Add s as candidate.
    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;
```

Optimize Verification

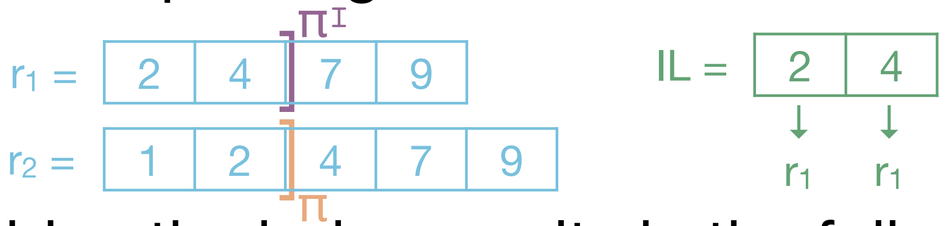
Matching Prefix Elements

- Problem: while looking up the index, the number of matching prefix elements for each pair is identified but disregarded afterwards.
- Idea: keep track of matching prefix elements and leverage the information to verify more efficiently. Based on the overlap in the prefix, a certain number of elements of the two sets can be skipped during verification.

Exercise

Set Similarity Self Joins

- Given: a probing set r_2 and r_1 inserted into an inverted list IL.



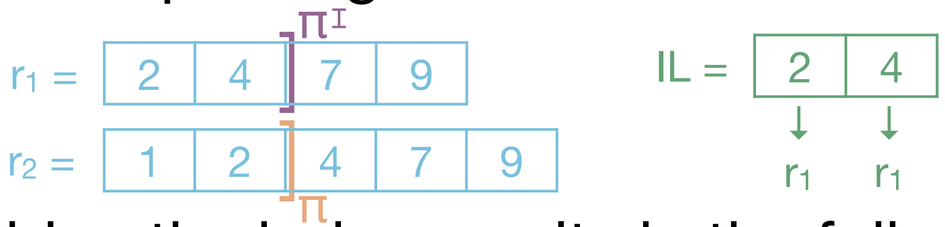
  Probing the index results in the following candidates: (r_2, r_1) with overlap 1.

- Task: what is the rightmost positions in r_1 (resp. r_2) to start the verification?

Exercise - Solution

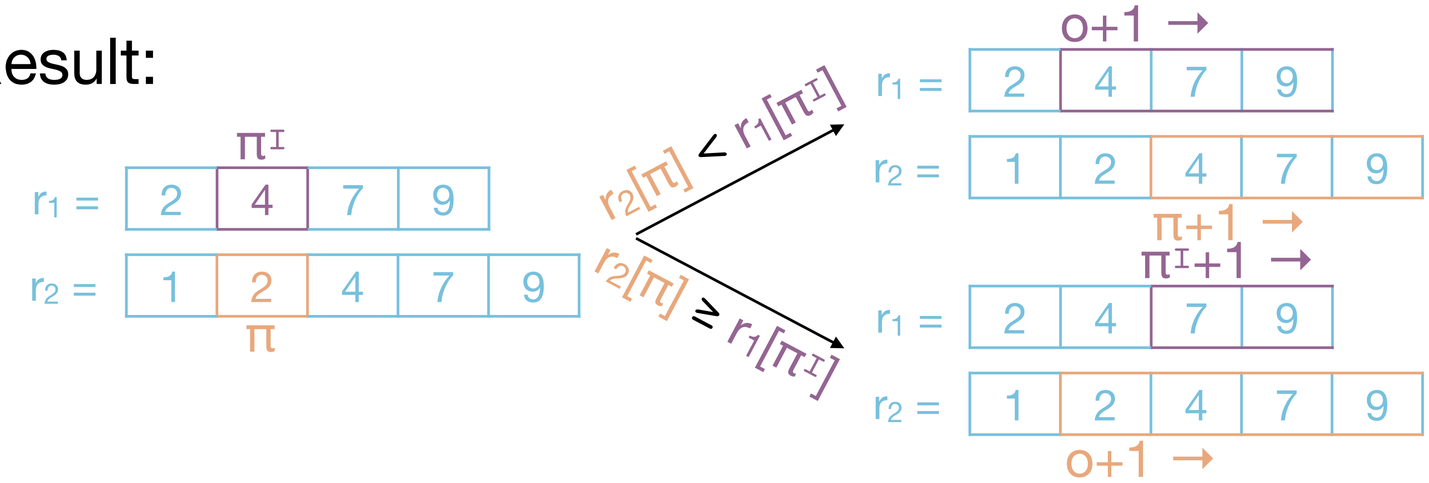
Set Similarity Self Joins

- Given: a probing set r_2 and r_1 inserted into an inverted list IL.



  Probing the index results in the following candidates: (r_2, r_1) with overlap = 1.

- Task: what is the rightmost positions in r_1 (resp. r_2) to start the verification?
- Result:



Task 4

Efficient Verification

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

```
Algorithm 4: 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[s] += 1; // Increase intersection.
    for pos = 0 to π_r^I - 1:
        I[r[pos]] = I[r[pos]] ○ r; // Append r to index.
    res = res U verify_candidate(r, C, t_J);

return res;
```

```
Algorithm 4: verify_candidate(r, C, t_J):
input: r set, C list of pairs, t_J threshold
output: res list of result pairs
res = []; // Stores the result pairs.
for (s, o) in C:
    eqo = ⌈(t_J / (t_J + 1))(|r| + |s|)⌉;
    if r[π_r] < s[π_s^I]:
        ret = verify(r, s, eqo, o, π_r+1, o+1);
    else:
        ret = verify(r, s, eqo, o, o+1, π_s^I+1);
    if ret is true:
        res = res U (r, s);

return res;
```

verify() by Mann et al.[1]

Efficient Verification

- The pseudocode on the previous slide is calling the verification function by Mann et al.[1], Page 5, Algorithm 1.
- Task:
  - Implement the function.
  - Explain variables maxr, maxs, and each stopping condition of the while loop in a comment block above the verify function.
- The paper mentions 7 set similarity join algorithms. Combining Tasks 1-4 results in the AllPairs algorithm (ALL). Bonus: check out the other algorithms.

[1] http://www.uni-salzburg.at/fileadmin/multimedia/Computerwissenschaften/3_Forschung/2016-03_Mann_Augsten_Bouros.pdf

Homework

Task 4

- Implement the efficient verification algorithm.
  - Implement the pseudocode of Task 4. (0.5 point)
  - Implement the verify() function by Mann et al. (0.5 point)
  - Explain how the verify function works (in the comments). (0.5 point)
- 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 19.01.2021, 23:55.