PS Ähnlichkeitssuche in großen Datenbanken

Task 1

Thomas Hütter, WS 2020

Motivation

Task 1

- All users are stored in a database with the following schema:

ID	Name	Hobbies
1	Peter	{guitar, biking, swimming}
2	Sarah	{skiing, hiking, singing}
3	John	{singing, hiking}
4	Kate	{guitar, skiing, swimming, running}
…	…	…

- For example, we could define “similar hobbies” as follows:
  - consider hobbies as sets of words,
  - define a threshold t = 0.75
  - using Jaccard similarity.

→ How can we compute the set similarity self join?

Exercise

Set Similarity Self Join Algorithm

- Given: a collections of sets, a similarity function, and a similarity threshold.
- Task: find a basic algorithm that computes the set similarity self join.

Algorithm: Set Similarity Self Join

Task 0

- Compute Jaccard for each pair of sets.

```
Algorithm 0: 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
for i = 0 to |R| - 1:
    for j = i + 1 to |R| - 1:
        if (R[i] ∩ R[j] / R[i] ∪ R[j]) ≥ t_J:
            res = res ∪ (i, j);
return res;
```

Exercise

Set Similarity Self Join Algorithm 2

- Given: a collections of sets R, a similarity function, and a similarity threshold.
- Task: using Algorithm 0, how many pairs are considered to compute $R \bowtie_{sim} R$?

Length Filter

Optimization

- Motivation: considering a quadratic number of pairs may not be feasible for large datasets.
- Idea: not all pairs are relevant because they are too different.
- Length filter: if the length difference of two sets is too high, the threshold cannot be reached even though their elements match.

Exercise

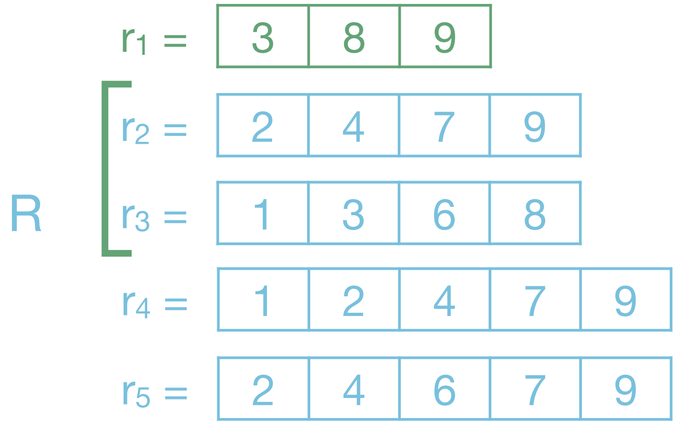
Length Filter

- Given: two sets r and s with |r| = 4, |s| = 6 and Jaccard similarity with threshold 0.7.
- Task: do we need to consider (r, s), i.e., is there a possibility that $J(r, s) \geq 0.7$?

Apply Length Filter

Optimization

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



- Idea: stop the inner loop of Algorithm 0 as soon as (r_i, r_j) cannot be higher than the threshold. For example, for r_1 is only compared to r_2 and r_3.

Algorithm: Set Similarity Self Join

Task 1

- Consider a collection of sets R ordered by size.

```
Algorithm 1: 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
for i = 0 to |R| - 1:
    for j = i + 1 to |R| - 1:
        if |R[j]| is to big for |R[i]| to reach t_J:
            break;
        if (R[i] ∩ R[j] / R[i] ∪ R[j]) ≥ t_J:
            res = res U (i, j);
return res;
```

Pre-processing

Set Similarity Self Join Algorithm

- Goal: introduce a common input format that can be processed efficiently and leveraged to optimize the algorithm.
- Idea: represent each set element by a single integer and sort them.

Original Dataset

Hobbies
{guitar, swimming}
{hiking, guitar, singing}
{singing, guitar, swimming}
{guitar, skiing, swimming, hiking}

Assign ascending integers.

Hobbies
{1, 2}
{3, 1, 4}
{4, 1, 5}
{1, 5, 2, 3}

Sort each set.

Pre-processed Dataset

Hobbies
{1, 2}
{1, 3, 4}
{1, 4, 5}
{1, 2, 3, 5}

Homework

Task 1

- Implement Algorithm 1: setsimjoin(R, t_J).
  - Include the length filter (input collection is ordered by size). (1 point)
  - Find a smart way to verify $J(r, s) \geq t_J$. (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.
- Deadline: 03.11.2020, 23:55