Ladislav Floriš

THRESHOLD TOP-K OPERÁTOR

Published May 1st 2021

Abstract

In this project I solve a problem in the field of similarity querying, namely the identification of the best for objects with respect to the N-tuple of attributes and the used aggregation function. For a more efficient search (especially with low selectivity due to the size of the entire dataset), I implement the Threshold top-k algorithm.

The input is the selected N-tuple of attributes, aggregation function, parameter size k and selected query method - naive sequential or Threshold top-k. The output is the top k objects with respect to the specified parameters, sorting is from the smallest to the largest. Querying is made available through a simple web interface that allows you to change parameters and displays the best objects in the table.

Solution

I perform all queries on the dataset stored in memory (no access to the database), the objects are sorted according to individual attributes at the start of the application.

I implement a naive sequential pass through a data source in which each object must be accessed, the value of the aggregation function calculated, and then the objects relative to the values of the aggregation function must be sorted and the best k objects are returned.

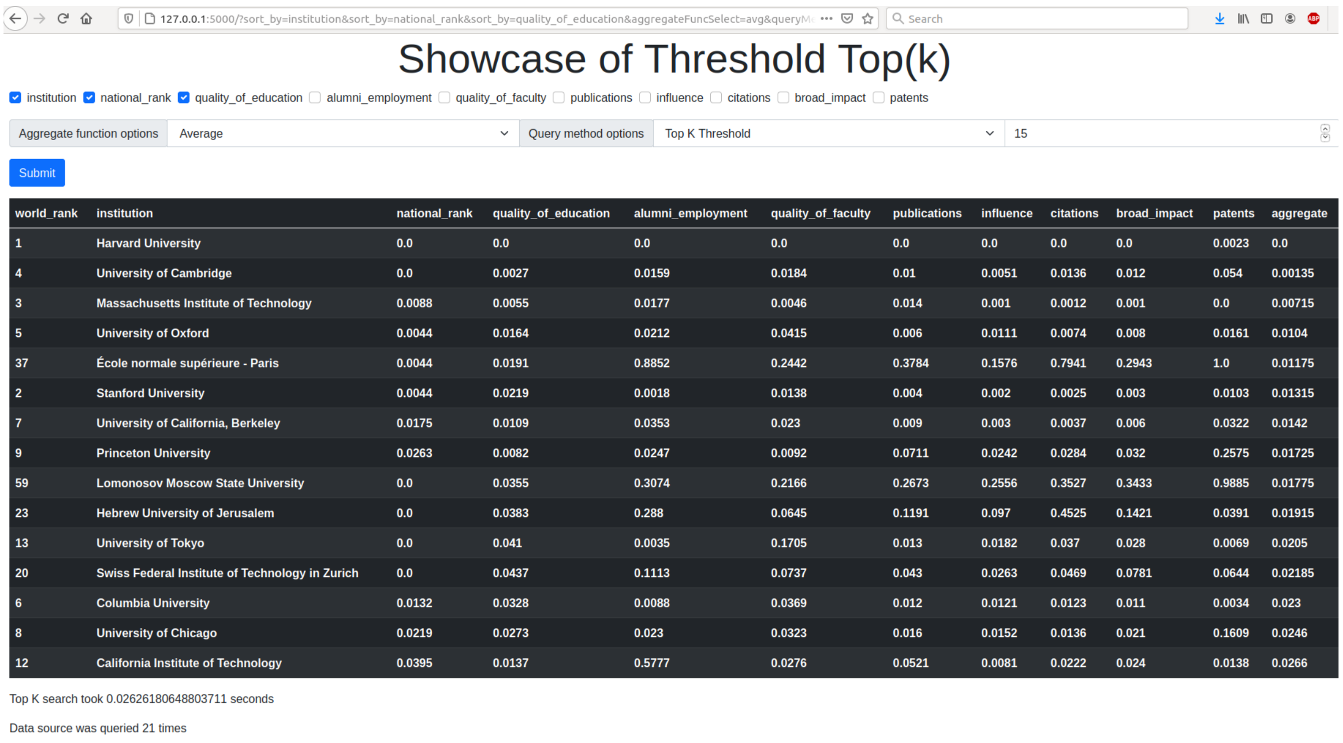
As a second way of querying, I implement the Threshold top-k algorithm which passes in parallel the individual fields sorted according to the specified attributes, and after passing one line, the threshold value is calculated, which is the theoretical optimum that can be achieved. The individual objects are maintained in the maximum heap, which reaches the maximum size (number of objects) k. As soon as there are all k objects in the heap and the worst of them is less than the threshold value, the pass is completed and the objects from the heap are returned in response.

Implementation

I implemented the app in Python due to simplicity and rich libraries set.
Used frameworks and libraries:

- **pandas** - for data manipulation and processing
- **Flask** - simple web framework
- **matplotlib** - for drawing charts of the experiments

Showcase



127.0.0.1:5000/?sort_by=institution&sort_by=national_rank&sort_by=quality_of_education&aggregateFuncSelect=avg&queryM

Showcase of Threshold Top(k)

[✓ institution] [✓ national_rank] [✓ quality_of_education] alumni_employment quality_of_faculty publications influence citations broad_impact patents

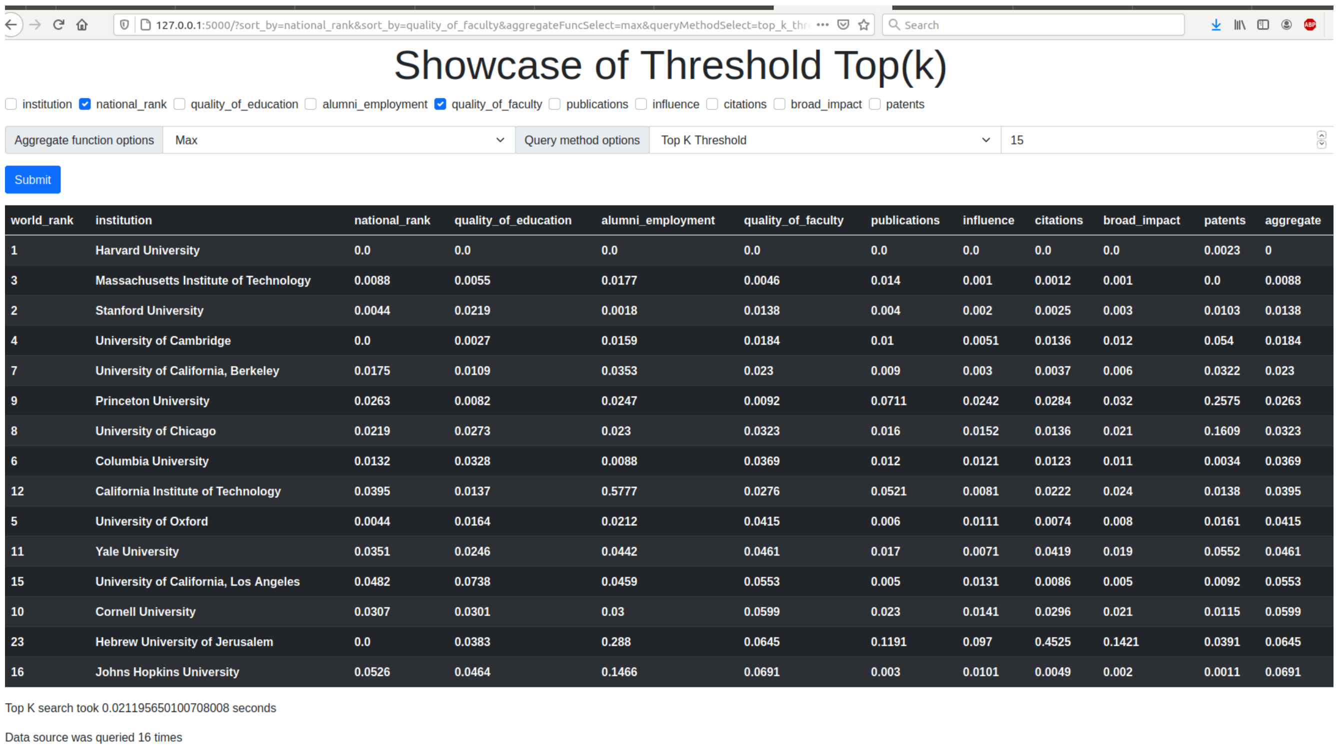
Aggregate function options: Average | Query method options: Top K Threshold | 15

Submit

world_rank	institution	national_rank	quality_of_education	alumni_employment	quality_of_faculty	publications	influence	citations	broad_impact	patents	aggregate
1	Harvard University	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0023	0.0
4	University of Cambridge	0.0	0.0027	0.0159	0.0184	0.01	0.0051	0.0136	0.012	0.054	0.00135
3	Massachusetts Institute of Technology	0.0088	0.0055	0.0177	0.0046	0.014	0.001	0.0012	0.001	0.0	0.00715
5	University of Oxford	0.0044	0.0164	0.0212	0.0415	0.006	0.0111	0.0074	0.008	0.0161	0.0104
37	École normale supérieure - Paris	0.0044	0.0191	0.8852	0.2442	0.3784	0.1576	0.7941	0.2943	1.0	0.01175
2	Stanford University	0.0044	0.0219	0.0018	0.0138	0.004	0.002	0.0025	0.003	0.0103	0.01315
7	University of California, Berkeley	0.0175	0.0109	0.0353	0.023	0.009	0.003	0.0037	0.006	0.0322	0.0142
9	Princeton University	0.0263	0.0082	0.0247	0.0092	0.0711	0.0242	0.0284	0.032	0.2575	0.01725
59	Lomonosov Moscow State University	0.0	0.0355	0.3074	0.2166	0.2673	0.2556	0.3527	0.3433	0.9885	0.01775
23	Hebrew University of Jerusalem	0.0	0.0383	0.288	0.0645	0.1191	0.097	0.4525	0.1421	0.0391	0.01915
13	University of Tokyo	0.0	0.041	0.0035	0.1705	0.013	0.0182	0.037	0.028	0.0069	0.0205
20	Swiss Federal Institute of Technology in Zurich	0.0	0.0437	0.1113	0.0737	0.043	0.0263	0.0469	0.0781	0.0644	0.02185
6	Columbia University	0.0132	0.0328	0.0088	0.0369	0.012	0.0121	0.0123	0.011	0.0034	0.023
8	University of Chicago	0.0219	0.0273	0.023	0.0323	0.016	0.0152	0.0136	0.021	0.1609	0.0246
12	California Institute of Technology	0.0395	0.0137	0.5777	0.0276	0.0521	0.0081	0.0222	0.024	0.0138	0.0266

Top K search took 0.02626180648803711 seconds

Data source was queried 21 times



127.0.0.1:5000/?sort_by=national_rank&sort_by=quality_of_faculty&aggregateFuncSelect=max&queryMethodSelect=top_k_thr

Showcase of Threshold Top(k)

institution [✓ national_rank] quality_of_education alumni_employment [✓ quality_of_faculty] publications influence citations broad_impact patents

Aggregate function options: Max | Query method options: Top K Threshold | 15

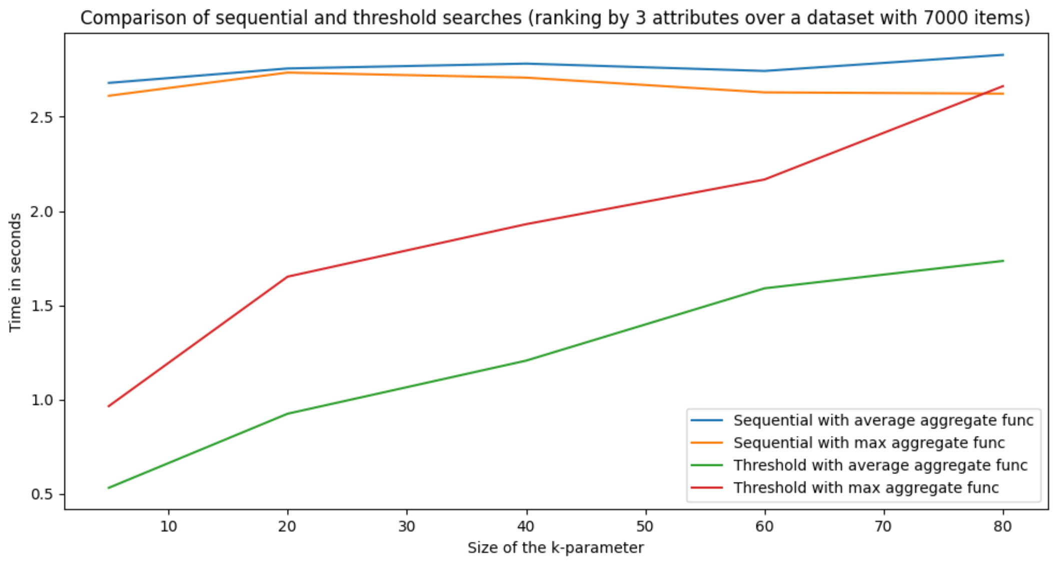
Submit

world_rank	institution	national_rank	quality_of_education	alumni_employment	quality_of_faculty	publications	influence	citations	broad_impact	patents	aggregate
1	Harvard University	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0023	0
3	Massachusetts Institute of Technology	0.0088	0.0055	0.0177	0.0046	0.014	0.001	0.0012	0.001	0.0	0.0088
2	Stanford University	0.0044	0.0219	0.0018	0.0138	0.004	0.002	0.0025	0.003	0.0103	0.0138
4	University of Cambridge	0.0	0.0027	0.0159	0.0184	0.01	0.0051	0.0136	0.012	0.054	0.0184
7	University of California, Berkeley	0.0175	0.0109	0.0353	0.023	0.009	0.003	0.0037	0.006	0.0322	0.023
9	Princeton University	0.0263	0.0082	0.0247	0.0092	0.0711	0.0242	0.0284	0.032	0.2575	0.0263
8	University of Chicago	0.0219	0.0273	0.023	0.0323	0.016	0.0152	0.0136	0.021	0.1609	0.0323
6	Columbia University	0.0132	0.0328	0.0088	0.0369	0.012	0.0121	0.0123	0.011	0.0034	0.0369
12	California Institute of Technology	0.0395	0.0137	0.5777	0.0276	0.0521	0.0081	0.0222	0.024	0.0138	0.0395
5	University of Oxford	0.0044	0.0164	0.0212	0.0415	0.006	0.0111	0.0074	0.008	0.0161	0.0415
11	Yale University	0.0351	0.0246	0.0442	0.0461	0.017	0.0071	0.0419	0.019	0.0552	0.0461
15	University of California, Los Angeles	0.0482	0.0738	0.0459	0.0553	0.005	0.0131	0.0086	0.005	0.0092	0.0553
10	Cornell University	0.0307	0.0301	0.03	0.0599	0.023	0.0141	0.0296	0.021	0.0115	0.0599
23	Hebrew University of Jerusalem	0.0	0.0383	0.288	0.0645	0.1191	0.097	0.4525	0.1421	0.0391	0.0645
16	Johns Hopkins University	0.0526	0.0464	0.1466	0.0691	0.003	0.0101	0.0049	0.002	0.0011	0.0691

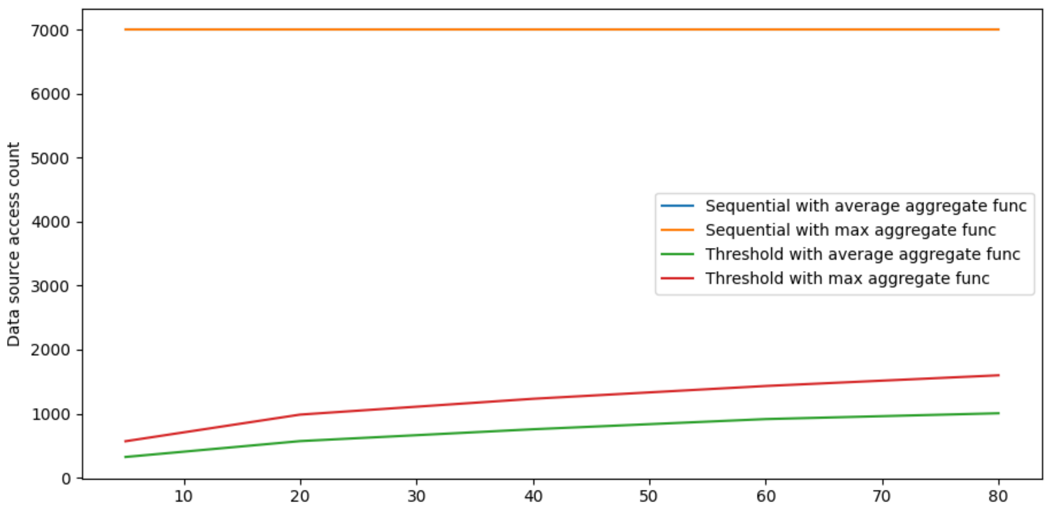
Top K search took 0.021195650100708008 seconds

Data source was queried 16 times

Experiments - size of the k parameter

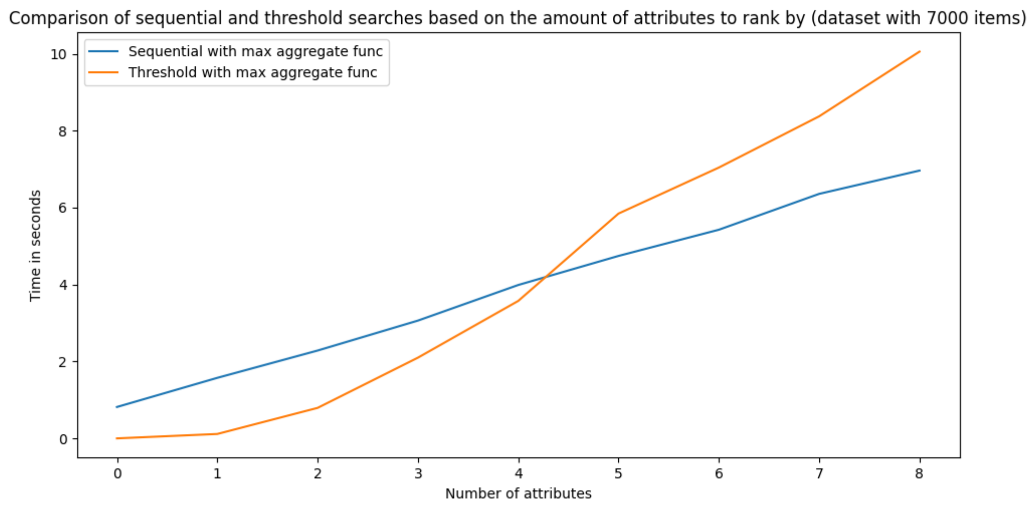


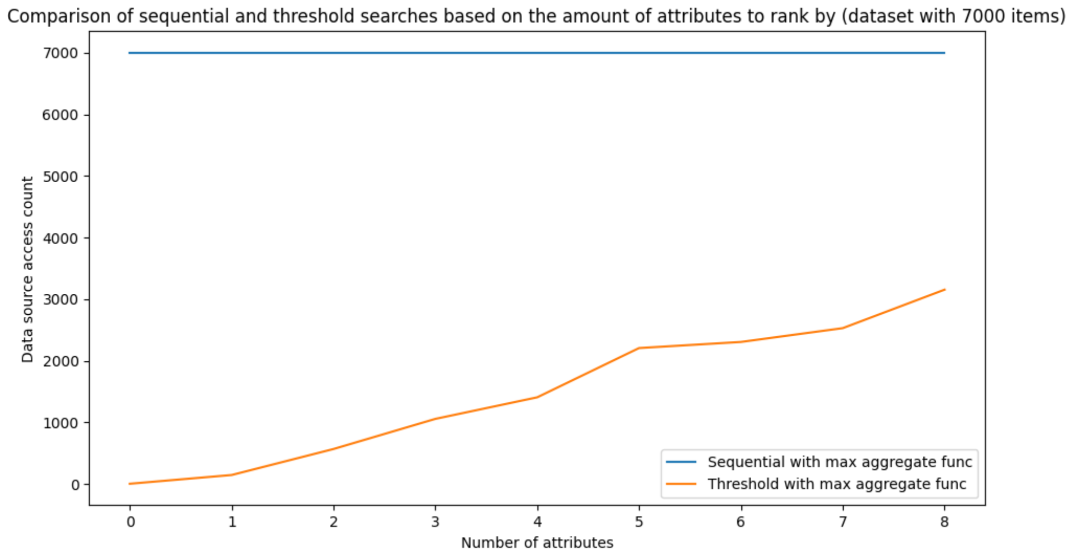
We see in the graph that the Threshold top-k algorithm achieves better times, but with increasing k parameter and thus greater selectivity, the algorithm must go through more records and the time overhead that results from maintaining the heap causes an increase in query duration.



If access to a data source is expensive, the Threshold top-k is much better due to the low demand on the number of accesses to the data source.

Experiments - number of attributes





We can see in the graphs that as the number of attributes increases, so does the number hits of Threshold top-k algorithm on the data source, which leads to higher latency.

Discussion

Although querying over data held in memory is as good as proof-of-concept, it would be good to extend this solution and test it on some "remote" data source, such as a database or data stream. As another shortcoming I see relatively high time requirements of the Threshold top-k algorithm with a higher number of selected attributes, here I would definitely see room for improvement and optimization of the existing solution.

Summary

In addition to the application of the Threshold top-k algorithm itself, I also solved other problems related to development, such as getting acquainted with and using the Flask framework for web application development, or manipulating and normalizing data used for testing through the pandas library.