Work Progress

kNN Search with Parallel Incremental Query Answering

Jaouhara Chanchaf

Wednesday Feb 15th, 2023

1. Summary

Done:

AI 1	Use Min Max Heap structure to store kNN and compare query time with Kashif using sorted array and Kashif using OSTree.
AI 2	Change Kashif code to reduce kNN heap size each time we return incremental results.

In progress:

AI 3	Read literature on word / column embeddings.

2. Kashif: Average query time, storing kNNs in a sorted array vs OS-Tree vs Min Max Heap

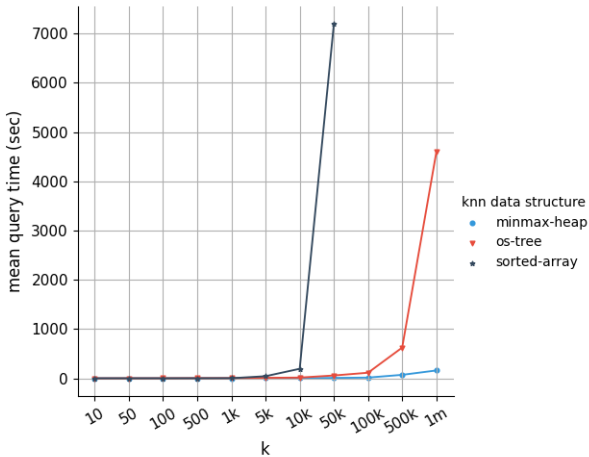

Figure 1: Kashif mean query time
(10 queries, query size = 100, dataset = 100k tables, 490k cols, 5M vectors)

2. Kashif: Average query time, storing kNNs in a sorted array vs OS-Tree vs Min Max Heap

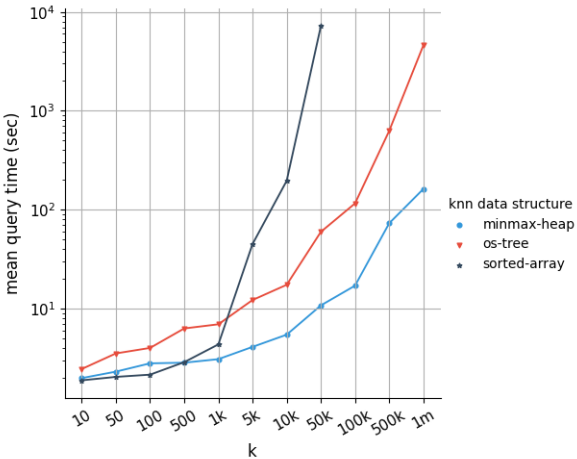

Figure 2: Kashif mean query time (log scale)
(10 queries, query size = 100, dataset = 100k tables, 490k cols, 5M vectors)

3. Average query time at k_{max} Kashif (min max heap) Vs PEXESO

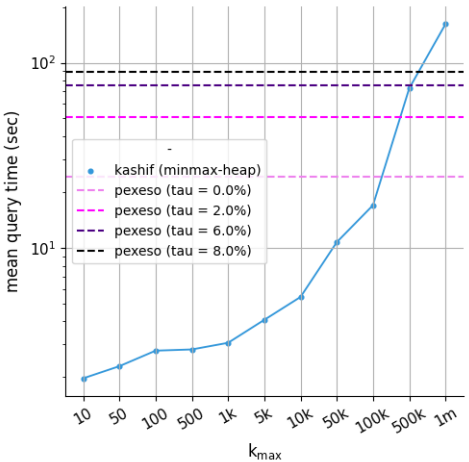

Figure 3: Kashif mean query time
(10 queries, query size = 100, dataset = 100k tables, 490k cols, 5M vectors)

4. Average query time at k_{max} Kashif (min max heap) Vs PEXESO

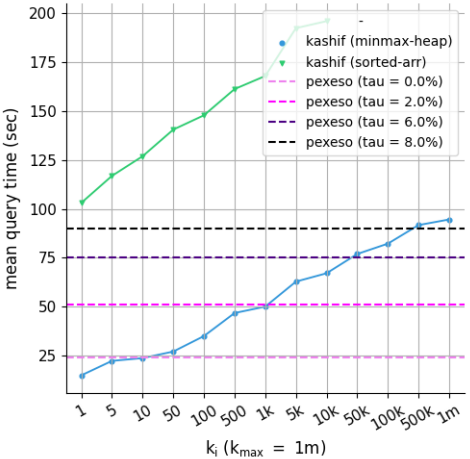

Figure 4: Kashif mean query time
(10 queries, query size = 100, dataset = 100k tables, 490k cols, 5M vectors)

3. Discussion

- When using Min-Max Heap to store kNNs we cannot keep the heap size constant. Every time a new incremental result is returned we must remove it from the heap to get the next incremental result. (Note: Min Max Heap is not an ordered data structure)
- Min Max Heap out performed the sorted array and OSTree in terms of query time (Currently measuring the memory usage for each data structure).
- We need to prove that Kashif achieves a good recall for $k_i \leq 50k$.

Worst Case Time Complexity:

	Sorted Array	Order Statistics Tree	Min Max Heap
Insert/Delete(d)	$O(n^2)$	$O(log(n))$	$O(log(n))$
Select(i) (ith smallest element)	$O(1)$	$O(log(n))$	-
GetMax/GetMin()	$O(1)$	$O(log(n))$	$O(1)$