Paper Title

Jaouhara Chanchaf
UM6P

Karima Echihabi
UM6P

ABSTRACT

300 word description of the project

PVLDB Reference Format:
Jaouhara Chanchaf and Karima Echihabi. Paper Title, 15(10): XXX-XXX, 2022.
doi:XX.XX/XXX.XX

1 INTRODUCTION

- **Use case 1: Keyword Query**
  A data scientist wants to retrieve datasets with information related to Biomass Power Companies. Initially, The user decides to start the search with a keyword-query $Q_{0,0} = (\{$"biomass", "power", "companies"$\}$, $k = 10)$. The search engine returns thirty datasets but none seem relevant to the user. To retrieve more results the user decides to run the same query and increase k, $Q_{0,1} = (\{$"biomass", "power", "companies"$\}$, $k = 20)$. After the second attempt the search engine returns Table 1 (at position #31) which contains data about biomass power plants per company. The user decides to keep Table 1 and continue to search for other relevant tables.

- **Use case 2: Join Query** Table 1 is relevant to $Q_{0,1}$ as it contains a list of biomass power plants, their location and capacity in Mega-Watt. However the user wants to include other information related to the prime mover of each plant, its status (operational or not), its start date etc. For that the user selects a set of plants based in California, and perform a join query on the "Plant" column in Table 1 to explore other tables that may complement Table 1 with more information.
  To avoid running the join query multiple times, the user chooses a high k value at the expense of query time. $Q_{1,0} = (\sigma_{Location="\%CA\%"}$(Table 1), Join column : "Plant", $k = 100)$. The search engine returned 381 results. After skimming through the list of result, the user finds Table 2 at position #315. Table 2 can be joined with Table 1 on column "Plant name".

Proceedings of the VLDB Endowment, Vol. 15, No. 10 ISSN 2150-8097.
doi:XX.XX/XXX.XX

Because the user has no prior knowledge of the total dataset size nor the optimal k value to retrieve relevant results in the least time possible, the user chooses k values randomly until he/she finds a relevant table.
In use case 2 the user is unaware that the same result could be retrieved at position #5 with $k = 10$.
Due to a large number of results, It is also possible that the user does not notice the desired result and decides to further increase k. For example, suppose that in use case 2 the user did not notice the result at position #315 and decided to submit $Q_{1,1} = (\sigma_{Location="\%CA\%"}$(Table 1), Join column : "Plant", $k = 200)$. The search engine will return 755 results, and Table 2 would be at position #235.

2 LITERATURE REVIEW

Dataset Discovery.

Keyword and Join Queries.

Incremental Query Answering.

3 PROPOSED APPROACH

Algorithm 1: BUILDINDEX

Algorithm 2: HEURISTICKNNSEARCH

Input: A query vector q, and k.
Output: k Nearest vectors to q.

```
KnnResults[k] ← {∞_1, ..., ∞_k};
initialize stack ← {};
N_curr = N_root;
while !N_curr.IsLeaf() do
    SP = N_curr.SplitPolicy();
    N_curr = N_curr.routeToChild(q, SP);
KnnResults ← GetNeirestVectors(N_curr, q, k);
return KnnResutls;
```

Algorithm 4: ExactKnnSearch

Input: A sequence of query vectors $Q = \{q_1, ..., q_n\}, k$.
Output: $k * n$ Nearest vectors to the the query vectors.

Array $KnnResults[n][k] \leftarrow \{\{+\infty_1, ..., +\infty_k\}_{q_1}, ..., \{+\infty_1, ..., +\infty_k\}_{q_n}\}$;
Queue $pq_1, ..., pq_n$;

foreach $q_i \in Q$ **do**
  $KnnResults[i] \leftarrow$HeuristicKnnSearch(q_i);
$WorkerThread$ reaches $SendUpdatesBarrier$;
`// update knn results in global array`
foreach q_i *in* Q **do**
  $ArrayCopy(AllKnnResults[i], KnnResutls[i])$;
`// initialize priority queues`
foreach q_i *in* Q **do**
  $pq_i \leftarrow \{\}$;
  $pq_i.Add(N_{root}, D_{lb}(N_{root}, q[i]))$;
while $!Finished$ *and* $\exists q_j \in Q$, $!pq_j.Empty()$ **do**
  foreach q_i *in* Q **do**
    $N_{curr} = pq_i.Pop()$;
    if $N_{curr}.IsLeaf()$ **then**
      $d_{curr} = calcMinDist(N_{curr}, q_i)$;
      if $d_{curr} < KnnResults[i][k-1]$ **then**
        $UpdateKnnResults(N_curr, KnnResults[i])$;
    else
      foreach N_{child} *in* $N_{curr}.ChildNodes()$ **do**
        if $D_{lb}(N_{child}, q_i) < KnnResults[i][k-1]$ **then**
          $pq_i.Add(N_{child}, D_{lb}(N_{child}, q[i]))$;
  $WorkerThread$ reaches SendUpdatesBarrier;
  `// update knn results in global array`
  foreach q_i *in* Q **do**
    $ArrayCopy(AllKnnResults[i], KnnResutls[i])$;
$Finished \leftarrow True$;

Algorithm 3: Kashif: ParallelIncremental-QueryAnswering

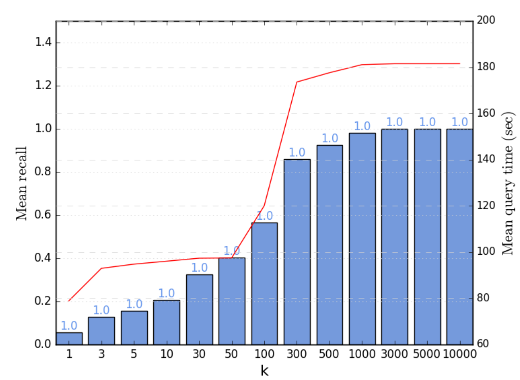
Input: A sequence of query vectors $Q = \{q_1, ..., q_n\}, k$ and the recall threashold r_{th}.
Output: $k * n$ Nearest vectors to the the query vectors.

Shared Array $AllKnnResults[n][k] \leftarrow \{\{+\infty_1, ..., +\infty_k\}_{q_1}, ..., \{+\infty_1, ..., +\infty_k\}_{q_n}\}$;
Shared Boolead $Finished \leftarrow False$;
Float $CurrentRecall \leftarrow 0$;
Barrier $SendUpdatesBarrier$ *for* $workerThread$;
Barrier $GetUpdatesBarrier$ *for* $CoordinatorThread$;

initialize $WorkerThread$;
$WorkerThread$ runs an instance of ExactKnnSearch(Q, k);
do
  $CoordinatorThread$ blocks on $GetUpdatesBarrier$;
  $CurrentRecall \leftarrow ComputeRecall(AllKnnResults)$;
while $!Finished$ *and* $CurrentRecall < r_{th}$;
$Finished \leftarrow True$;
return $AllKnnResutls$;

4 EXPERIMENTAL EVALUATION

Figure 1: Query-time recall and precision, Experiment on 100k tables ≈ 500k columns and 25M vectors



ACKNOWLEDGMENTS

We sincerely thank X, Y and Z.

REFERENCES

[1] K. Echihabi, K. Zoumpatianos, T. Palpanas, and H. Benbrahim. The Lernaean Hydra of Data Series Similarity Search: An Experimental Evaluation of the State of the Art. *PVLDB*, 12(2), 2018.

[2] PhDComics. Graduate Student Work Output. https://phdcomics.com/comics/archive.php?comicid=124, 2022.

Company	**Plant**	**Location**	**Feedstock**	**Capacity (MW)**
Wheelabrator Technologies Inc.	Wheelabrator Shasta Energy Co. Inc.	Anderson - CA	Logging and Mill Residue/Ag Residue	50
Greenleaf Power LLC	Desert View	Mecca - CA	Ag Residue/Urban Wood Waste	47
Greenleaf Power LLC	Honey Lake	Wendel - CA	Mill and Logging Residue/Forest Thinning/Urban Woodwaste	30
Covanta	Covanta Delano	Delano - CA	Orchard and Vineyard Prunings/Nut Shells/Stone Fruit Pits	58
...	...	...	...	...

Table 1: U.S. Biomass Power Plants

Category	Plant ID	Plant Name	Unit	Status	Start Date	Retire Date	Prime mover ID	Prime Mover Description	Capacity	net MWh
E	E0027	Desert View Power (Mecca Plant)	GEN1	OP	1991/11/1	-	ST	Steam Turbine	54.15	351291
E	E0041	HL Power Company (Honey Lake)	GEN 1	OP	1989/7/26	-	ST	Steam Turbine	35.5	200712
E	E0029	Covanta Delano, Inc	Delano 1-2	OP	1990/6/12	-	ST	Steam Turbine	58	322731
E	E0086	Wheelabrator Shasta	Units 1-3	OP	1987/1/1	-	ST	Steam Turbine	54.9	405628
...	...	...	...	...	...	...	...	...	...	...

Table 2: Annual Generation - Plant Unit