Aster : Space Efficient Pre-Trained Cardinality Estimation Model using Positional Embeddings

Machine Learning in Databases CSCI-699, Spring 2023

Submitted by: Ashwini Ainchwar

Code Repo: https://anonymous.4open.science/r/iris-F3DB

Abstract

Estimating cardinalities is a widely studied problem in databases literature. Recently, machine learning based cardinality estimation techniques have emerged as a promising alternative to traditional methods due to their superior performance under memory and latency constraints. However, these learning based techniques require training separate models for every dataset and retraining for data updates or workload drifts, which makes these techniques prohibitively expensive in practice. To address these issues we propose Aster, which utilizes pre-trained models to generate column sketches of structured datasets which can be used for cardinality estimation. This paper provides a design overview of Aster and investigates the feasibility of using positional embedding for correlating independent column sketches for cardinality estimation. This approach provides an 8x reduction in memory usage over the state-of-the-art without any degradation in estimation accuracy and supports new column additions to the dataset with minimal overhead.

1 Introduction

A recent technique Iris [1], proposed by Lu et al.(2022) employs a single pre-trained model to generate mergeable row summaries for any structured dataset while avoiding per-dataset training. However, this technique requires modeling every column pair in the dataset at the least, leading to at least quadratic space complexity in the number of columns, and combinatorial space at worst for generating and storing summaries. The quadratic complexity forces them to use fewer learned summaries because of storage and latency constraints. Also, the choice of choosing which columns sets to summarize jointly and selecting summaries for cardinality estimation at inference time is driven by heuristics.

Aster leverages column sketches with positional embeddings to provide a linear space solution in the number of columns. It primarily consists of two components, a Sketcher, and an Aggregator model. The sketcher model encodes a dataset using column sketches. The aggregator model is responsible for taking an input query and combining these column sketches to provide a cardinality estimate.

1.1 Design Goals

We lay out the following design objectives to maximize the utility of the system.

1. **Same per column memory usage:** Traditional approaches like histograms allocate a certain memory budget per column in production systems. Aster uses a lower memory budget to store our column sketches and uses the max budget as a hyper-parameter.
2. **Low Latency:** Aster uses small sketches and will use a small aggregator model to provide CE within a millisecond.
3. **Off-the-shelf model for CE:** Pre-train a model for CE which does not need to be created or trained for a new dataset.

2 Background and Related Work

Current cardinality estimation methods primarily belong to one of the two categories, (1) Data driven, (2) Query/ Workload driven. Data driven models suffer from independence assumptions, higher memory footprints, and failure to handle data updates. Query driven models need a custom training set of queries generated for the dataset and cannot deal with data updates.

	Pre-trained	Handles Workload Drifts	Handles Data Updates	Low Memory Footprint
DeepDB [4]	✗	✓	✓	✗
Naru [3]	✗	✓	✗	✗
Iris [1]	✓	✓	✓	✓
MSCN [2]	✗	✓	✗	✓
Aster	✓	✓	✓	✓

Figure 1: Comparison of current CE approaches

2.1 Pre-trained models for cardinality estimation

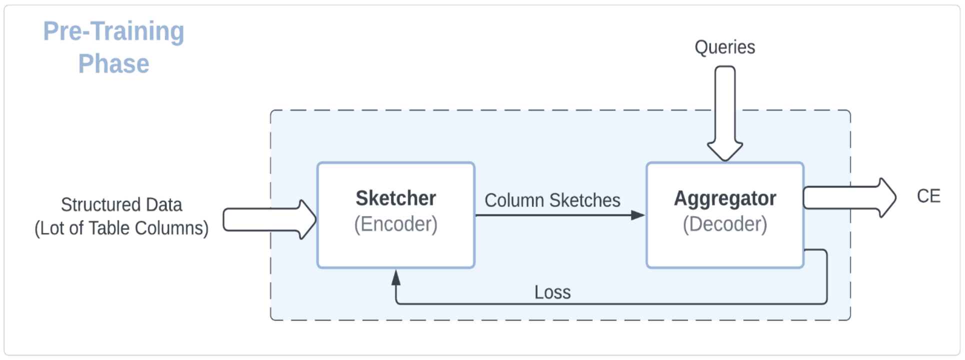
The Iris system utilizes an approach that involves exploring all feasible column subsets in a given table and producing row summaries using sparse indexes, histograms, or Iris summaries. The columnsets may overlap. To mitigate the computational cost associated with maintaining summaries for all possible column sets, the system restricts the size of column sets to a maximum of two, i.e. $O(n^2)$. For each row in the dataset, multiple summaries are generated by using a pre-trained encoder model. Each summary corresponds to a specific columnset. During inference, the system selects the summaries that have the most number of intersections with query columns. The row summaries are mergeable in case of data insertions or deletions. Heuristics are used for the appropriate method for summarizing a columnset and which columnsets to use for cardinality estimation.

3 Design Overview

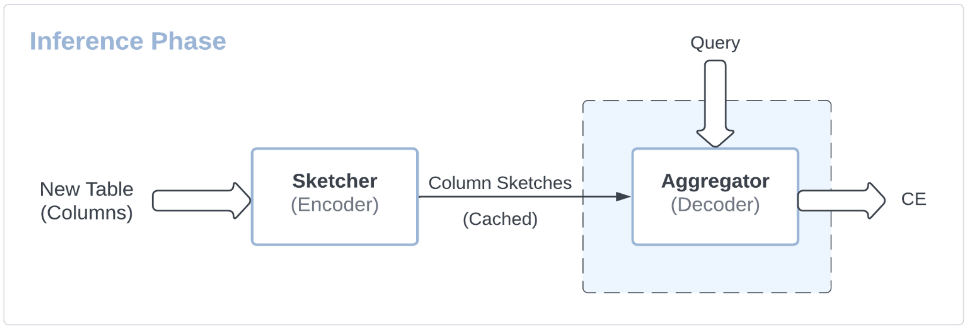
We use the following insights to propose a new approach that optimizes the accuracy of cardinality estimation as well as the memory required to store these sketches, to give a linear space solution:

1. We pre train a sketcher model on a corpus of datasets to generate column sketches for any new unseen dataset.
2. Use positional embeddings to correlate the column sketches
3. Pre-train an aggregator model jointly optimized to predict the final cardinality using the learned column sketches.

As shown in figure 2 (a), during the pre-training phase we do end-to-end training of the sketcher models with many datasets and the aggregator model with queries corresponding to the dataset. For a new dataset, we use the pre-trained sketcher model to generate column sketches once and store them. At inference time, a query is given to the aggregator which uses the stored summaries to generate a CE.



(a)



(b)

Figure 2: High Level Design of Aster. Sketcher model generates a unique sketch for each column, the aggregator model combines all the column sketches to produce the final cardinality estimates.

4 Implementation

We preprocess the dataset with mixed data types, such as real, categorical, and others, by employing quantization on each column. This pre-processing step is crucial for handling diverse data types effectively. Our method utilizes row numbers as positional embeddings, allowing us to maintain the context and order of the data. As a proof of concept, we use the Iris models row summaries for generating our column sketches.

We extend Iris to use only n columns sets of the form (positional index, column_i). Iris typically generates row summaries; however, by crafting column sets (b) in this manner, we effectively obtain n column sketches for each row. When a user provides a query, column sketches corresponding to all the columns involved in the query are used for estimating the cardinality.

The aggregator model is a simple multiplication of predicted cardinalities for each column sketch. e.g. If our table has 15 columns we store 15 column sketches generated by Iris. Given a query of the form

$$\begin{aligned} \text{SELECT * FROM T WHERE } & lb_1 <= C_1 <= ub_1 \\ \text{AND } & lb_7 <= C_7 <= ub_7 \\ \text{AND } & lb_{10} <= C_{10} <= ub_{10} \end{aligned}$$

The aggregator column will simply use column sketches of columns 1, 7, and 10.

5 Evaluation

5.1 Experimental Setup

Using Iris to generate column sketches for Aster without any fine tuning. Row summaries/ Column sketches are built by sampling 310000 rows, ~0.052% of the dataset. We evaluate the performance on TPCH_Lineitem dataset and use GMQ as our error metric.

5.2 Evaluation Metric

To evaluate the effectiveness of our proposed cardinality estimation approach, we adopt the GMQ error metric described below.
For a given predicate we measure the error as follows,

$$q_\theta(g, \hat{g}) = \max\left(\frac{max(g, \theta)}{max(\hat{g}, \theta)}, \frac{max(\hat{g}, \theta)}{max(g, \theta)}\right)$$
$$\text{where } g, \hat{g} \text{ are estimated and actual cardinalities}$$

$q\theta$, relative error after skipping values below θ. We use $\theta = 10$.

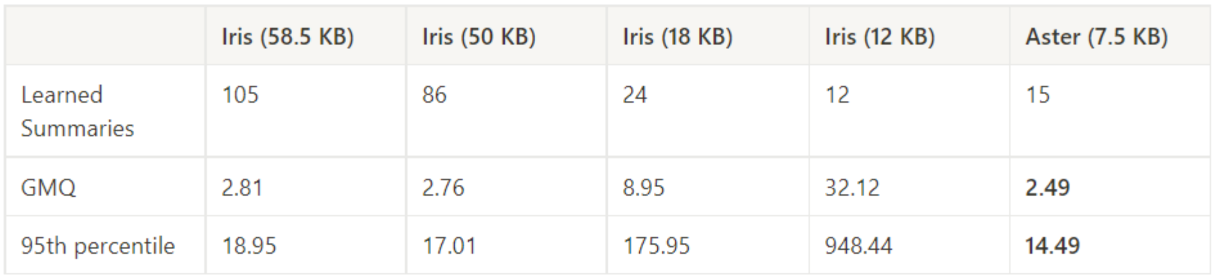
GMQ is the geometric mean of $q\theta$ over all predicates. Intuitively, the geometric mean is more robust to outliers; a $q\theta$ value of $1 + x$ indicates that the relative error in cardinality estimates is $100 * x$%; and, perfect estimates have a value of 1 i.e. x=0.

Furthermore, to evaluate the worst case accuracy we also report the error values at 95th percentile. We also consider the size of learned summaries to keep it comparable to memory usage in current production systems.

5.3 Aster vs Iris : Forcing Iris to use only learned summaries

To minimize the memory footprint, Iris constructs a bag of summaries where different columnsets are encoded using a mix of learned and histogram based methods. To perform a fair evaluation of the efficacy of using column sketches, we force Iris to only use learned summaries. Iris maintains a single row summary for a pair of columns, and multiple row summaries a single row. It can use at max (n * (n-1))/2 summaries for n columns, i.e. 105 for TPCH-Lineitem as it has 15 columns.

As we can see in Figure3, if Iris uses only learned summaries the performance is significantly worse than Aster even though it uses ~8x more memory. Using multiple row summaries needs more storage, we can see that if the storage budget for Iris is decreased the accuracy degrades severely, and is not comparable to Aster at similar memory budget.



	Iris (58.5 KB)	Iris (50 KB)	Iris (18 KB)	Iris (12 KB)	Aster (7.5 KB)
Learned Summaries	105	86	24	12	15
GMQ	2.81	2.76	8.95	32.12	**2.49**
95th percentile	18.95	17.01	175.95	948.44	**14.49**

Figure 3 : Accuracy of CE by Iris (using only learned summaries) vs Aster over TPCH-Lineitem at different storage budgets of 4 KB per column.

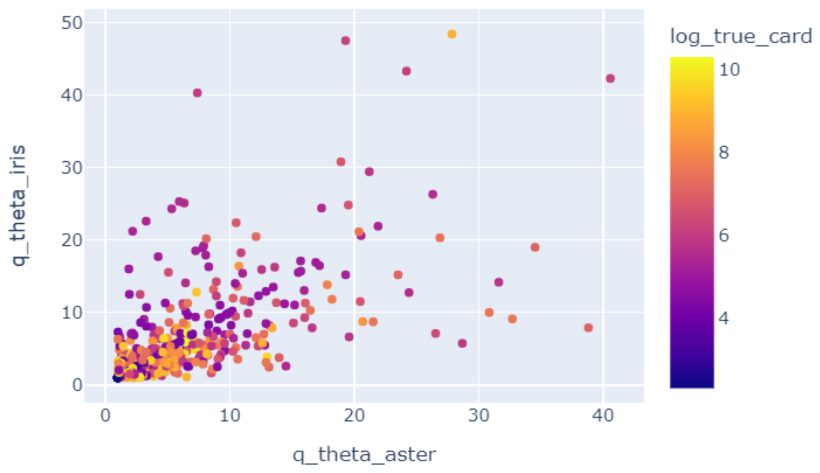


Figure 4: Error distribution for Aster and Iris. Points b

Figure 4 and Figure 5, give insights on how the error in CE varies with increasing true cardinalities. To improve the performance of our model, we can focus on queries with medium selectivity. This could also be a result of non-uniform sampling of the database.

As seen in Figure 5, Iris performs slightly better than Aster for $q\theta < 50$. However, if we see the $q\theta$ error for all ranges as shown in Figure 6, the error for Iris varies a lot and is really high.

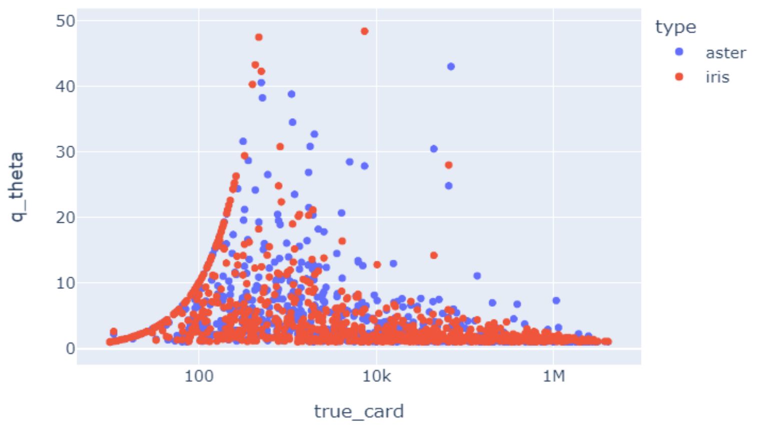
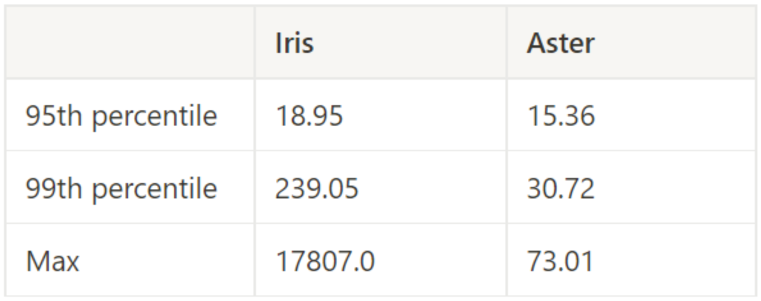
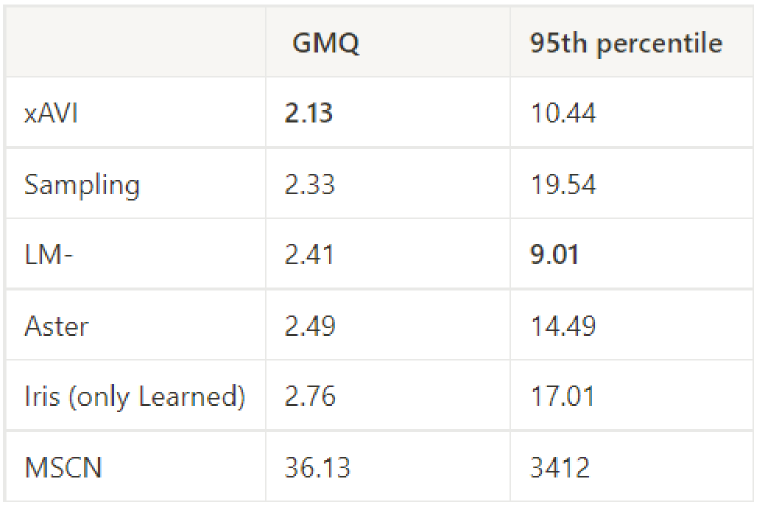


Figure 5: Error analysis for Aster and Iris, we illustrate the error as a function of true cardinality for each query in the dataset while restricting per query error to < 50.



	Iris	Aster
95th percentile	18.95	15.36
99th percentile	239.05	30.72
Max	17807.0	73.01

Figure 6: $q\theta$ error ranges



	GMQ	95th percentile
xAVI	**2.13**	10.44
Sampling	2.33	19.54
LM-	2.41	**9.01**
Aster	2.49	14.49
Iris (only Learned)	2.76	17.01
MSCN	36.13	3412

Figure 7: Accuracy of CE for different baselines

6 Discussion & Future Work

In the future, we want to experiment with training an aggregator model to obtain the cardinality estimates from column sketches, which would enable us to eliminate the need for heuristic approach used at present. To evaluate the generalizability and performance of our proposed method, we intend to test the system on additional datasets. We should investigate the impact of sampling and selectivity on the accuracy and efficiency of the system.

We also intend to explore non-uniform quantization schemes for generating column sketches to potentially achieve a more accurate CE. Non-uniform quantization can capture a finer granularity of data distribution, leading to improved results.

Furthermore, we plan to jointly train the sketcher model responsible for generating column sketches alongside an aggregator model fine-tuned to merge these sketches effectively. This end-to-end training approach will enable the two models to work in synergy. By refining the aggregation process and considering the inherent characteristics of the column sketches, we aim to enhance the overall accuracy.

7 Conclusions

In this paper, we introduced Aster, an approach that utilizes a pre-trained model for cardinality estimation. This approach mitigates the burden of training for a new dataset and provides an out-of-the-box solution. Through preliminary experiments, we show that using column summaries with positional embedding yields promising results. Aster outperforms Iris when using only learned summaries while taking ~8x less space without any fine tuning. In the future, we intend to implement the proposed aggregator model, which would further help improve the accuracy of the system.

References

[1] Lu, Yao, et al. "Pre-training summarization models of structured datasets for cardinality estimation." *Proceedings of the VLDB Endowment* 15.3 (2021): 414-426.

[2] Kipf, Andreas, et al. "Learned cardinalities: Estimating correlated joins with deep learning." *arXiv preprint arXiv:1809.00677* (2018).

[3] Yang, Zongheng, et al. "Deep unsupervised cardinality estimation." *arXiv preprint arXiv:1905.04278* (2019).

[4] Hilprecht, Benjamin, et al. "Deepdb: Learn from data, not from queries!." *arXiv preprint arXiv:1909.00607* (2019).