A Framework for Learned Approximate Query Processing for Tabular Data with Trajectory

Kihyuk Nam, Sung-Soo Kim, Choon Seo Park, Taek Yong Nam, Taewhi Lee
ETRI
Daejeon, Korea
{nam, sungsoo, parkcs, tynam, taewhi}@etri.re.kr

Abstract— Approximate query processing has been well established for enhancing performance of aggregation queries on ever-increasing big data by statistically equivalent approximations. Recent popularity of mobile devices creates tremendous spatio-temporal data that require different treatment than relational ones. Among spatio-temporal data, we focus on trajectories in a tabular form and analyzes the problem, its requirements, and suggest a general-purpose framework for learned approximate query processing by providing a common encoding/embedding layer for embracing diverse state-of-the-art ML models, on top of which resides a probabilistic circuit for efficiency and efficacy with error bounds.

Keywords—Spatio-temporal Approximate Query Processing, AQP, Machine Learning

I. Introduction

Ever-growing amounts of spatio-temporal data are ceaselessly generated by a variety of devices from IoT sensors to hand-held mobile devices and autonomous driving cars. Those data have different characteristics from traditional tabular or relational data since the dependency along the time axis forms their intrinsic meaning.

The approximate query processing (AQP) has become a well-established component of big data analytics by fast producing statistically equivalent results of exact matching [1]. Furthermore, recent development of machine learning (ML) and deep learning (DL) technologies opened new opportunities in this direction. Although there has been proposed a variety of spatio-temporal data analytics methods that are based on traditional statistical techniques, ML/DL techniques are rarely applied to them, presumably due to the idiosyncratic nature of spatio-temporal data.

This paper analyzes the problem of approximating spatio-temporal queries using ML & DL, and requirements for applying state-of-the-art ML/DL models to AQPs, then suggests a general solution with a work-in-progress prototype. Among spatio-temporal data, we focus on the trajectories and their core operators as a starting point. Finally, we present an evaluation strategy and future plan.

II. Related Works

Li et. al., tackle almost the same problem except they use sampling on spatio-temporal index for approximation [2]. DeepSPACE is a deep learning-based approximate geospatial query processing engine that can answer flexible aggregation queries while keeping the required state to a small data size [3]. NeuroSketch handles range aggregate queries (RAQs) by leveraging neural networks [4]. It focuses on modeling queries rather than data in a geospatial database environment. Vu et. al, suggests a spatio-temporal pattern mining technique for LBS [5], and Kim et. al, presents 3D spatio-temporal models [6]. To the best of our knowledge, there are no learned spatio-temporal AQP for queries on relational data with trajectory, which guarantees efficiency, efficacy, and error bounds.

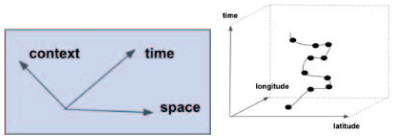


Fig. 1. Basic target space and its example

III. Problem Analysis

Among a variety of spatio-temporal problems, this paper deals with an AQP for aggregate trajectory range query.

A. Definitions

Problem: Given a region and/or a time condition in a target space, find an aggregation (e.g., count, sum, average) of rows satisfying both spatio-temporal and non-spatio-temporal conditions.

Spatio-temporal data are assumed to be a (m+n+1)-dimensional relational model, $((S_1, \dots, S_m, T), C_1, \dots, C_n)$ with (m+1) dimension spatio-temporal column (Fig 1, left).

A **trajectory** is a time-stamped sequence of spatial coordinates from a moving object. It can be seen as a curve or region in a hypercube. The spatial coordinates can be GPS data or check-in points from vehicles or smartphones.

The **dimension** of target spaces would be arbitrary. It can be a 3D space consisting of latitude, longitude, and time dimension (Fig 1, right). It could be a 4D space with latitude, longitude, altitude, and time dimension. Each dimension can be discrete, continuous, and categorical.

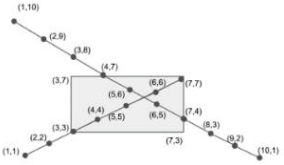
Querying on k-dimensional relations with spatio-temporal columns can be seen as searching in k-dimension space, or distinct value estimation problem [2].

B. Requirements and Limitations of DNNs

By using recent deep learning technologies, the following methods can be considered for our learned model.

- Probabilistic: probabilistic circuits [8] (e.g., SPNs [9])
- Convolutional (e.g., CNNs): Trajectories can be seen as 3D or 4D figures. Currently the spatial dimension should be less than or equal to 3 since most convolutional models are built for 2D or 3D inputs.
- Sequential (e.g., RNNs, Transformers): Trajectories naturally fit into sequential models due to the time dependency.

While recently developed deep neural networks (DNNs) show great representational power, it's difficult to provide uncertainty scores for reliable decision making with efficient learning and inference. Furthermore, they cannot provide various combinations of conditionals, marginals for processing typical select-project-join queries.



1	**Trajectories on 6/1, after 12:00, intersecting with the area (3,3)-(7,7)**
	kisql> select id, name, st_lifetime(**st_intersection**(position, st_importfromwkt('**MPOLYGON((2023/06/01 12:00:01,((3,3,7,3,7,7,3,7,3,3))))**'))) from nam_phone_user1; --- id name st_lifetime(…) ---- ------ --- 2 [Kim] Period[2023-06-01 12:00:07.0000~2023-06-01 12:00:07.0000] 1 [Nam] Period[2023-06-01 12:00:03.0000~2023-06-01 12:00:05.0000] --- select row count [2] RESULT : 0.000419
2	**Number of trajectories from 5/31, 12:00 to 6/1, 12:11, passing the area (3,3)-(7,7)**
	kisql> select count(*) from nam_phone_user1 where **st_passes**(position, st_importfromwkt('MPOLYGON((2023/05/31 12:00:03,((3,3,7,3,7,7,3,7,3,3))), (2023/06/01 12:10:59,((3,3,7,3,7,7,3,7,3,3))))'))=1; --- count(*) --- 2 --- select row count [1] RESULT : 0.000356

Fig. 2. A sample queries on 2 trajectories

TABLE I. QUERIES IN PROBABILITY EXPRESSION

1	Which street is the most likely to have traffic jam at 7pm?
	$\text{argmax}_{\text{Street}}$ P(Jam(Street) | 7pm)
2	When is the most likely to have traffic jam in Street_k?
	$\text{argmax}_{\text{Time}}$ P(Time | Jam(Street_k) = 1)
3	What's the probability of having traffic jam at Street_k on Monday?
	P(Day=Monday, Jam(Street_k) = 1)
4	When is the most likely to have traffic jam along the trajectory of my commute?
	argmax P(Time | $V_{t \in \text{Trajectory}} Jam(i)$)

* Time_k and Day are random variables, Jam(s) returns whether s has traffic jam.

TABLE II. OPERATORS

Type	Operators
Set	buffer, difference, intersection, union
Temporal	precedes, equals, meets, overlaps, contains
Spatio-temporal	equals, disjoints, touches, contains, within, crosses, overlaps, intersect, relate
Trajectory	enters, leaves, passes, meets, insides

TABLE III. SEMANTICS OF TEMPORAL OPERATORS

precedes(x, y)	x=[a, b) precedes y=[c, d) → b <= c
equals(x, y)	[a, b) equals [c, d) → (a=c) and (b=d)
meets(x, y)	[a, b) meets [c, d) → b=c
overlaps(x, y)	(∃t∈[a, b), t∈[c, d)) or (∃t∈[c, d), t∈[a, b))
contains(x, y)	(a<c) and (b>d)

C. Queries in SQL

Trajectory queries can be given in SQL statements for OGC-compliant, moving object database systems. Figure 2 shows some SQL statements for Kairos MO SQL [7]. This paper analyzes only aggregates with AVG, SUM, COUNT.

D. Queries in Probability Expression

Most trajectory queries can be represented as probability expressions. Table 1 shows some traffic examples, which require calculating conditionals and marginals (and joint probabilities from them) to answer those queries.

E. Predicates and Operators

Apart from comparison and logical operators such as <, =, >, !, 'and', and 'or', spatio-temporal operators or predicates should be supported. Major operators supported by OGC-compliant systems can be summarized as in table 2. Their semantics can be formally defined for being applied to ML models. For example, the semantics of temporal operators can be defined as table 3.

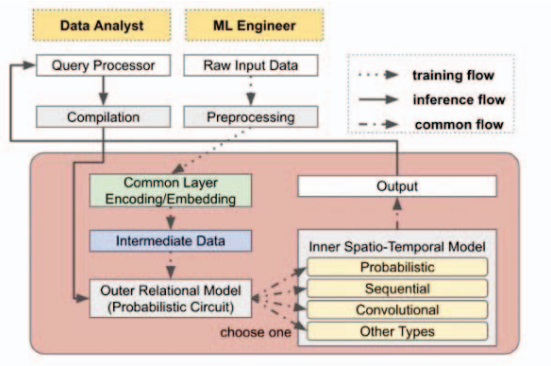


Fig. 3. PC-based Architecture

IV. SOLUTION

To provide a general methodology and a framework where new models can be easily adopted, we need to introduce a common intermediate layer on top of baseline architecture [15][16], that generalizes input data, learning/inference procedure and underlying subsystems.

A. Two-Staged Processing

As mentioned in Section 3-A, data are prepared in relational models, $((S_1, \dots, S_n, T), C_1, \dots, C_n)$, so we need to process relational data as well as spatio-temporal (trajectory) data. The exact order depends on the result of query planning.

B. PC-based Architecture

We chose probabilistic circuits as a front-end model for our learned AQP systems because of efficiency, efficacy, error bounds and incremental updating [8]. It is proven that any decomposable and smooth probabilistic circuit is valid probability distribution, and its inference is tractable (linear in the size of a PC). By using PCs as front-end, outer model, uncertainty scores can be generated, which helps reliable decision making. Furthermore, PC-based models can be updated partially by modifying only the relevant leaves and weights of sum nodes. In general, there can be three types of PC-based solution (Fig. 3).

- PC + PC: Both Spatio-temporal and relational (non-spatio-temporal) columns are modeled in PCs. Outer PC models relations, and inner PC represents trajectories. It's efficient but may lack representation power than DNN-based inner models.
- PC + Transformer: PC front-end with transformer-based leaves and attention-based weights of sums. The intermediate trajectories can also be added to positional encodings.
- PC + Convolution: Trajectories can be seen as 3D or 4D figures, which naturally fits in convolution-based models. Currently the spatial dimension should be less than or equal to 3 since most convolutional models are built for 2D or 3D inputs. The non-spatio-temporal columns are represented in the outer PC model.

Other types of DNNs not specified above can be connected into the outer PC as well. It is also possible to adopt all-in-one PC models that synthesized with recent DNN concepts (e.g., transformer, attention, convolution, etc.).

```
[[[ 0  1  2] [ 3  4  5] [ 6  7  8]]
 [[ 9 10 11] [12 13 14] [15 16 17]]
 [[18 19 20] [21 22 23] [24 25 26]]]
```

Fig. 4. An example of linear indexing of (3D) hypercube

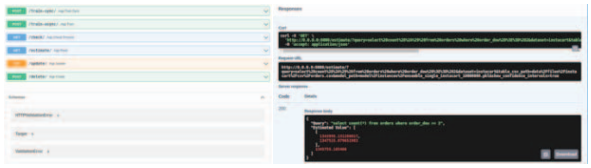


Fig. 5. REST API + Pure PC model

C. Common Layer – Encoding and Embedding

To make our solution as general as possible, we define a common layer that takes charge of encoding and embedding so that the downstream subsystems can be independent of our framework. We assume that all of the input data are pre-processed into 'one row, one trajectory' format. Many of open data such as Uber movements [10] are segment-based, i.e., one row is one segment, so that one trajectory can be spread over several rows. Those data are assumed to be aligned to 1:1 format before entering our learned AQP model.

Users can define the level of detail for each axis. For example, GPS data consists of latitude and longitude with ranges [-90, 90] and [-180, 180] respectively. They can be discretized by truncating below the decimal points or binning. Time values can also be discretized or categorized as well (e.g., [0, …, 23] hours, morning/afternoon, or Sunday-Saturday). Such a user-defined LoD value must be given to the system configuration.

Right after the common layer comes a PC-based front-end, outer model that summarizes entire calculations including query answers and their uncertainty scores.

For the PC front-end and DNN-based inner models to be more efficient and effective for learning and inference tasks, the spatio-temporal part of $((S^i, T), C^j)$ is structured as grids using the method introduced in [9].

The dimension of the grid-structure can be arbitrary and therefore called hypercubes. The grids of the hypercubes are represented by two extreme coordinates. For example, ((1, 5, 2), (8, 9, 11)) represents 3D grids ranging from 1 to 8 for the first axis, from 5 to 9 for the second axis, and from 2 to 11 for the third axis.

Each coordinate in a hypercube corresponds to a random variable. The 3D grids above, for example, consists of 8 * 5 * 10 = 400 random variables. The random variables are linearly indexed. For example, the index (0, 1, …, 26) of (3, 3, 3) hypercube can be indexed as Figure 4. Sum nodes and leaves in PCs correspond to sub-hypercubes, and their linear indices are scope for the nodes.

For pure PC models, learning algorithms that are based on K-means or EM algorithms, can directly be applied to the grid-structured intermediate data by splitting and clustering the grids.

V. Evaluation Strategy

We built a front-end system with REST API to facilitate easy integration and evaluation (Fig. 5) [16]. Currently only the pure PC model based on RSPN [11] is prototyped but the architecture facilitates easy extension of other models. The datasets for evaluation can be NYC taxi, Uber movements, and Foursquare [9][10][11].

VI. Conclusion and Future Works

This paper suggests a methodology and framework for learned spatio-temporal approximate query processing. By introducing a common encoding/embedding layer with a PC-based front-end model, it can embrace state-of-the-art models with efficient and effective AQP with error bounds. Future works include prototyping hybrid models such as 'PC+Transformer' and performance comparison with different combinations of models and traditional spatio-temporal data store systems.

Acknowledgment

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2021-0-00231, Development of Approximate DBMS Query Technology to Facilitate Fast Query Processing for Exploratory Data Analysis)

References

[1] Li, K., Li, G. Approximate Query Processing: What is New and Where to Go?. Data Sci. Eng. 3, 379–397 (2018).

[2] Y. Li, CY. Chow, K. Deng, M. Yuan, J. Zeng, JD. Zhang, Q. Yang, and JL. Zhang. 2015. Sampling Big Trajectory Data. In Proceedings of the 24th ACM International on Conference on Information and Knowledge Management. Association for Computing Machinery, New York, NY, USA, 941–950. https://doi.org/10.1145/2806416.2806422

[3] D. Vorona, A. Kipf, T. Neumann, and A. Kemper, "DeepSPACE: Approximate Geospatial Query Processing with Deep Learning," in Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2019, Chicago, IL, USA, November 5-8, 2019. ACM, 2019, pp. 500–503.

[4] S. Zeighami, C. Shahabi, and V. Sharan, "NeuroSketch: Fast and Approximate Evaluation of Range Aggregate Queries with Neural Networks," Proc. ACM Manag. Data, vol. 1, no. 1, pp. 100:1–100:26, 2023. [Online]. Available: https://doi.org/10.1145/3588954

[5] Vu, T.H.N., Lee, J.W. and Ryu, K.H. (2008), Spatiotemporal Pattern Mining Technique for Location-Based Service System. ETRI Journal, 30: 421-431. https://doi.org/10.4218/etrij.08.0107.0238.

[6] Kim, H.-G., Shin, S.-S., Kim, S.-W. and Lee, G.Y. (2021), No-reference quality assessment of dynamic sports videos based on a spatiotemporal motion model. ETRI Journal, 43: 538-548. https://doi.org/10.4218/etrij.2020-0160.

[7] http://www.realtimetech.co.kr/bbs/page.php?hid=p202_3

[8] A Vergari, YJ Choi, R Peharz, G Van den Broeck. "Probabilistic circuits: Representations, inference, learning and applications", Tutorial at the The 34th AAAI, 2020

[9] H. Poon and P. Domingos, "Sum-product networks: a new deep architecture," in Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence (UAI), pp. 337–346, 2011.

[10] https://github.com/cambridge-mlg/EinsumNetworks.

[11] B. Hilprecht, A. Schmidt, M. Kulessa, A. Molina, K. Kersting, and C. Binnig. 2020. DeepDB: learn from data, not from queries! Proc. VLDB Endow. 13, 7 (March 2020), 992–1005

[12] https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page

[13] https://movement.uber.com/

[14] https://location.foursquare.com/developer/

[15] T. Lee, K. Nam, C. S. Park, and S. Kim, "Exploiting Machine Learning Models for Approximate Query Processing," in *IEEE International Conference on Big Data, Big Data 2022, Osaka, Japan, December 17-20, 2022*. IEEE, 2022, pp. 6752–6754.

[16] K. Nam, S. -S. Kim, C. S. Park, T. Y. Nam and T. Lee, "Designing ML-based Approximate Query Processing Services on Time-Varying Large Dataset for Distributed Systems," *2022 13th International Conference on Information and Communication Technology Convergence (ICTC)*, Jeju Island, Korea, Republic of, 2022, pp. 1979-1982, doi: 10.1109/ICTC55196.2022.9952398.