LATEST: Learning-Assisted Selectivity Estimation Over Spatio-Textual Streams

Mayur Patil[a,b]
[a]Department of Computer Science and Engineering
[b]Center for Geospatial Sciences
University of California, Riverside
Email: mpati005@ucr.edu

Amr Magdy[a,b]
[a]Department of Computer Science and Engineering
[b]Center for Geospatial Sciences
University of California, Riverside
Email: amr@cs.ucr.edu

***Abstract*—Selectivity and cardinality estimation are main driving factors for developing cheap query plans and ultimately faster query processing. Traditionally, database systems use estimation data structures, e.g., histograms, to maintain data summaries. Machine learning models have recently been employed, acting as black boxes, in several database tasks, including cardinality estimation. In the dynamic streaming environments, both estimation data structures and machine learning models struggle with adaptation for dynamic changes in data and query workloads. This paper proposes *LATEST*; a system module that uses machine learning to enable dynamic adaptation of estimation data structures. For spatial-keyword queries in a streaming environment, it shows on par or better performance than the state-of-the-art estimators. *LATEST* builds an incremental supervised learning model over a moving time window that helps the underlying system to switch among several estimation structures to keep estimation accuracy high at all times. As an incremental learner, *LATEST* effectively adapts to dynamic changes of both data and queries in streaming environments. Our extensive experiments on three real datasets and various query workloads verify the effectiveness of *LATEST* with higher accuracy and lower response times over the state-of-the-art estimators.**

I. Introduction

Spatial-keyword queries have received significant attention over the past decade [2], [8], [11], [12] with many applications with location-based services such as social media platforms [69], online retail systems [38], supply chain systems [34], points of interests (POIs) search [15], news search [61], route planning [7], targeted advertising [20], visualization [31], sensor-based networks [5], and safety-critical applications, e.g., COVID-19 outbreak [47], fire rescue [70], and evacuations [64]. The importance of these spatial-keyword queries has increased notably in contemporary times, and as a result, recent systems provide system-level infrastructure to handle spatial-keyword queries [43], [49]. As a system utility, a system-level module should provide flexibility to work for various query workloads. For example, a query on geo-textual data could involve a spatial predicate, a keyword predicate, or both of them. The system should provide appropriate infrastructure to support flexible query workloads, including indexing, query optimization, and query processing.

Recently, streaming versions of spatial-keyword queries have received greater attention for both snapshot queries [8] and continuous queries [11] on streaming data. The main focus in existing streaming literature is on indexing and query processing for these queries. However, there are no efforts that explore query optimization modules for systems in this context. Selectivity and cardinality estimation are among major query optimization techniques that are understudied in this literature, despite having various crucial applications that benefit from these techniques in real time. For example, first responders of a rescue team can issue an estimation query that estimates the number of tweets with the keyword '*fire*' that lie within '*Downtown Thousand Oaks, California*' to estimate the number of affected people seeking help, and in turn estimate resource allocations for the fire incident[1]. Such real-time queries become more frequent for disasters that span extended spatial and temporal ranges. Streaming user-generated data has already shown great help in real-time disaster management situations, such as Hurricane Harvey [64] and Hurricane Irma [70]. Another example is targeted advertisement applications that gauge the popularity of product-related keywords in real time in different areas to place advertisements effectively. The lack of system-supported modules to serve such queries efficiently hinders the full potential of streaming user-generated data.

This paper proposes *LATEST*; a system-level module that provides efficient learning-assisted selectivity estimation for *streaming spatial-keyword queries*. Selectivity estimation has been the topic of study in the database community for a long time [19], [51], [58], [63], [67] to produce fast and accurate estimates as a crucial part of achieving a cheap query plan. A few of these efforts consider spatial-keyword queries, and only some focus on streaming versions of these queries [67]. Although high dynamism in query workloads is a quantified phenomenon in streaming user-generated data [40] on which spatial-keyword queries are prominent, existing techniques do not provide selectivity estimation models for streaming spatial-keyword objects to adapt with changing query workloads in real-time.

To fill the existing gap, *LATEST* employs a supervised machine learning model to provide efficient selectivity estimation for spatial-keyword queries on streaming data in terms of estimation accuracy and latency measures. Traditionally, machine learning models replace the underlying data structures,

[1]https://www.fire.ca.gov/incidents/2021/1/14/erbes-fire/

such as estimators [30], or indexes [37]. Various techniques utilize query parameters and results to train models, eliminating the need for data structures [36], [37], [55]. In turn, models act as indexes or estimators and are used for query answering as black boxes. However, in a spatial streaming environment, the models cannot cope with the rapidly changing stream distribution and query workload flow. Our proposed solution intends to incrementally learn a model that adapts to the changes in the stream. Besides, we use the model not to replace the underlying data structures but to recommend the most efficient data structure that boosts the estimation performance during this part of the stream lifetime. Therefore, the learned model does not answer the estimation query directly but decides which data structure to use for query answering based on the most recent window of streaming activities.

LATEST uses six estimators as famous widely-used examples from the existing literature that are derived from existing samplers, histograms, and spatial quadtrees. At any time point along the stream lifetime, only one of these estimators is used to answer estimation queries. The stream lifetime is divided into two main phases, a pre-training phase and an incremental learning phase. In the beginning, a pre-training phase is triggered to obtain training data for the learning model from all six estimators. At this phase, each query is fed to all estimators, then query latency and accuracy are evaluated based on the system logs of actual query selectivity. The machine learning model is then trained based on this data to be used in the following incremental learning phase.

After the pre-training phase is concluded, the incremental learning phase starts and consists of back-to-back disjoint periods that span the rest of the stream lifetime. The first period begins with employing the default estimator for answering selectivity estimation queries. The incoming query results are then used to get additional training data fed to the learning model to improve its recommendations. Meanwhile, the estimation accuracy is monitored based on system logs of actual query selectivity. Once estimation accuracy falls below a certain threshold based on recent queries, the machine learning model is consulted to switch to another estimator. The new estimator is pre-filled and used for a new period that repeats the same mechanism as the previous period.

A distinguishing feature of *LATEST* is the dynamic adaptation with the changing query workloads, e.g., spatial-dominated, keyword-dominated, or hybrid workloads of various ratios, through incremental learning over the stream lifetime. *LATEST* is experimentally evaluated based on three real datasets of spatio-textual objects and spatial-keyword queries that form different workloads. Our experiments have shown the effectiveness of *LATEST* to switch the system selection to the most accurate and the fastest estimators based on changes in query workloads in real-time. Our contributions in this paper can be summarized as follows:

- We identify and demonstrate the need for spatial-keyword streaming estimators to accommodate fast-paced changes in data and workloads.
- We identify the important features to train machine learning models for adaptivity in selectivity estimation.
- We develop a learning-assisted module *LATEST* that uses both workload information and actual system logs to adapt the best performing estimator along the stream lifetime dynamically.
- We provide an extensive experimental evaluation that shows the effectiveness of our techniques on three real datasets.

The rest of this paper is organized as follows. Section II covers the related work. Section III provides a formal problem definition, and Section IV discusses the preliminary data structures in use for selectivity estimation. Our proposed system-level module, *LATEST*, is described in Section V. The experimental evaluation and results are discussed in Section VI followed by conclusions in Section VII.

II. Related Work

Our related work lies in three main areas: *selectivity estimation on spatial streams*, and *learning for selectivity estimation* and *learning in database management systems*, each outlined below.

Selectivity Estimation on Spatial Streams. Selectivity estimation in databases has been studied for decades [16]. For spatial data, existing literature mainly use multi-dimensional histograms [6], [24], [51], [63], sampling [25], [58], or hybrid approaches [9], [50] that combine the strengths of both histograms and sampling. However, few other applications that use different techniques, such as kernel density models [28], and techniques based on using spatial indexes [1]. Some of these techniques are adapted for the streaming environment utilizing a four-ary tree with min heap for estimation [29] or hierarchical reservoir sampling [14] for range counting estimation. With several techniques proposed, there is no clear winner that can be used throughout the stream lifetime for all use cases. In contrast to these works, *LATEST* does not invent a new estimation technique. Instead, *LATEST* innovates a learning model that adaptively selects the appropriate technique among the existing techniques to provide the best performance for the current portion of the stream.

In another work, streaming spatio-keyword estimation is studied using the augmented adaptive space partitioning tree with local correlations [67]. However, this work tightly couples spatial and keyword predicates, and so it limits its flexibility to be used at a system level and provide very low accuracy for queries with only spatial or keyword predicates. In contrast, *LATEST* is a system-level module that targets flexibility in supporting query workloads that involve both spatial and keyword predicates with different ratios, which is not currently supported in any approach.

Learning for Selectivity Estimation There are several works associated with the use of learning for estimating selectivity. However, none of them target streaming environments for spatial keyword queries. Particularly, existing learning-based techniques for selectivity estimation include using deep learning for point and range queries [26], neural networks for

approximate bounded-range selectivity functions [41], curve-fitting [10], wavelet-based histograms [46], multi-set convolutional network upon sampling [35], probabilistic graphical models [22], and a study revealing the space and time trade-offs across several architectures for deep learned estimation [54]. In contrast to all these techniques, *LATEST* targets adaptive streaming environments for spatial-keyword workloads that change over time.

This literature can also be categorized into *workload-driven models* and *data-driven models*. Workload-driven models use query information and results to train a model that directly answers future estimation queries, while data-driven models use the raw data for training. Existing workload-driven models use neural networks [39], historical statistics [33], deep learning [27], or a hybrid neural network with tree ensemble model [19]. Existing data-driven models use the joint data distribution [71], statistical relational learning [21], a mixture model [19], deep learning [30] and local bayesian networks [67]. *LATEST* can be categorized as a workload-driven approach. However, unlike existing approaches, it builds a workload-driven model that is not used as a replacement for an estimation data structure, but it incrementally learns the best data structure to use through the time-changing workload information.

Learning in Database Management Systems. Machine learning models are used in different parts of database systems as surveyed in [32], highlighting new trends in learned data systems and their applications. Apart from selectivity estimation, learning is being explored to replace index structures [37], [52], to automatically select indexes [42], [65], in conjunctive querying processing [53], to generate query execution plans [45], to integrate ML agents in database systems [55], or even build a complete learned database system [36]. *LATEST* is inspired by the idea of automatic switching of the data structure in use, based on the changing query workload in our case. Compared to all these techniques, *LATEST* focuses on multi-dimensional streaming environments, which is more complex for learning models to perform accurately. Hence, we adopt a hybrid approach that uses an adaptive learning model with underlying data structures.

III. Problem Definition

We address the problem of building an efficient selectivity estimation module, *LATEST*, that is flexible to work at the system level for spatial-keyword queries on streaming data. The targeted spatial-keyword queries are snapshot queries that are posted on a streaming dataset S that consists of geo-textual objects. Each object $o \in S$ is represented with the four main attributes (oid, loc, kw, $timestamp$), where oid is the object identifier, loc is the object location in a two-dimensional space represented with latitude/longitude coordinates, kw is a set of associated keywords, and $timestamp$ is the time when the object is posted. S_T is a time window of the dataset S that represents the past T time units, so every object $o \in S_T$ has $o.timestamp \geq (NOW - T)$, where NOW represents the current time. *LATEST* receives snapshot estimation queries at a time window T to adapt with stream dynamism and returns the count estimate based on S_T. In particular, *LATEST* answers *RC-DVQ* estimation query that is defined as follows:

Range-Counting Distinct-Value Query (RC-DVQ): given a query q = (spatial range R, set of keywords W), and S_T that is a time window of the dataset S of size T, RC-DVQ returns number of estimated objects $o_i \in S_T$ so that: (1) $o_i.loc$ lies inside $q.R$, and (2) $o_i.kw \cap q.W \neq \phi$.

RC-DVQ estimates the number of objects that lie within the given spatial range R and contains at least one of the query keywords W over the time window of the past T time units. This query combines two of the most prominent estimation queries in the existing literature: a range counting query [14] that uses only spatial predicates and a distinct value query [4] that uses only keyword predicates. *RC-DVQ* definition has *R* and *W* as optional parameters, which allows the absence of either of them. This enables the user to convert the query into either a range counting query [14] or a distinct value query [4], both works on a time window to handle the data's streaming nature. In specific, a query q = (*R*) represents a range counting query, and a query q = (*W*) represents a distinct value query. As shown in a recent survey [13], these queries are the major estimation queries that cover a significant portion and play an indispensable role in modern management systems and numerous applications ranging from social media platforms, emergency services, route management, targeted advertising. Such flexibility is needed at the system level as previously motivated. By employing a time window, *LATEST* ensures the summary generated for the stream of data is always up-to-date.

IV. Preliminaries

This section introduces the preliminary estimators and estimation data structures solicited from the existing literature and used as precursors in our proposed technique. Selecting these estimators are based on their wide popularity in the existing literature. However, system administrators can select a different set of estimators that fit their needs and applications. In fact, our work is orthogonal from which estimators to use. Instead, our technique enables the power of using multiple estimators simultaneously and switching among them automatically to optimize the system performance. This enables systems to support real-world applications even if they constantly encounter changes in query types.

We mainly use the two most popular estimation approaches: sampling and histogram. Additionally, we use two other approaches: augmented adaptive spatial trees and hybrid approaches that combine both histograms and sampling. These approaches support both spatial and keyword predicate estimation, as imposed by our query definition in Section III. As we will briefly discuss in this section, and evidently show in our experimental evaluation (Section VI), each estimator has different performance pros and cons on different types of query workloads. This has motivated our work's main idea to employ all of them collaboratively to adapt to the changing query workloads along the stream lifetime.

- **Two-dimensional histogram**. As shown in Figure 1(a), a two-dimensional (2D) histogram divides the whole 2D

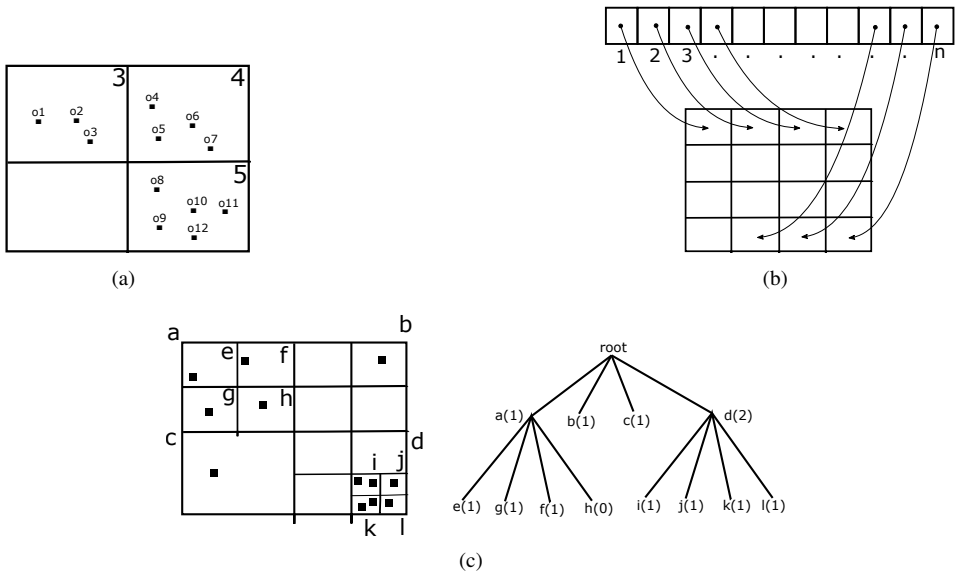


Fig. 1. Preliminary data structures: (a) A two-dimensional histogram. (b) A hybrid structure that is a reservoir sampling list indexed by a two-dimensional histogram. (c) An adaptive space partition tree with count summaries.

spatial space into a set of disjoint cells of equal size. Its structure is a regular spatial grid, where each cell stores only the count of points that lie within the cell's spatial boundaries rather than the actual points. This is a 2D generalization for the regular one-dimensional histogram with uniform binning [9] that is used for single-dimensional data. Two-dimensional histograms are appropriate for range counting queries that employ pure spatial predicates.

- **Reservoir sampling** is a sampling approach that has shown good performance in representing a stream of data [56]. We use the *algorithm R* [66] version of reservoir sampling. It employs a list of a fixed size N. The first N elements in the stream are inserted into the list. Then, a random replacement policy is used to process new elements. In specific, the algorithm generates a random number every time a new element arrives in the stream. If the random number generated is smaller than N, the corresponding item in the list is replaced. Otherwise, the new element is discarded. The reservoir list structure allows estimation for both spatial and spatial-keyword queries as it contains actual data points with all attributes.
- **Augmented adaptive space partition tree**. The Augmented adaptive space partition (AASP) tree [67] consists of a KMV synopses of distinct elements of the stream and a set of adaptive space partition (ASP) trees. KMV synopses [3] estimates the number of distinct keywords in a data stream. It maps the keywords onto the range [0, 1] using distinct hash functions. The synopses maintain the k smallest hash values of the elements in the set. An ASP tree [29] is a four-ary compressed quadtree [60] as depicted in Figure 1(c). Each tree node has a corresponding spatial cell and a counter, where each data point is counted by exactly one node. The tree uses a split threshold to adapt the spatial distribution changes in streaming data by dynamically splitting cells based on density.
- **Hybrid structures**. Hybrid structures [50], [72] combine multiple estimators, e.g., both 2D histograms and samplers, to make use of their joint pros. There are several variations of hybrid structures, such as 2D histograms with each cell holding a list of objects, modified quadtrees, and others. There are also different strategies to build two-dimensional counting cells, such as modified quadtrees, modified R-tree and its variations, and non-uniform binning. Sampling lists have variations as well that include changes to replacement policies, windowed lists, etc. In our work, we use *a reservoir sampling list indexed by a 2D histogram*. It is a hybrid structure that reduces iteration overhead over reservoir sampling lists to answer queries efficiently. As shown in Figure 1(b), every element in the list is mapped to a grid cell in the histogram to use the histogram in retrieving efficiently. It is beyond the scope of this paper to include all hybrid structures. It is an open area of investigation to evaluate different structures on various query workloads.

V. Learning-assisted estimation

This section presents the selectivity estimation module, *LATEST*, that selects estimators adaptively to cope with the evolving nature of streaming data and query workloads. *LATEST* utilizes the collection of data structures presented in Section IV with the help of a learning model to switch between these structures adaptively over time to continuously improve the performance in terms of query accuracy and latency. The learning model is particularly built based on understanding the issues pertaining to other learning models' efficiency. The rest of this section gives the overview of *LATEST* in Section V-A,

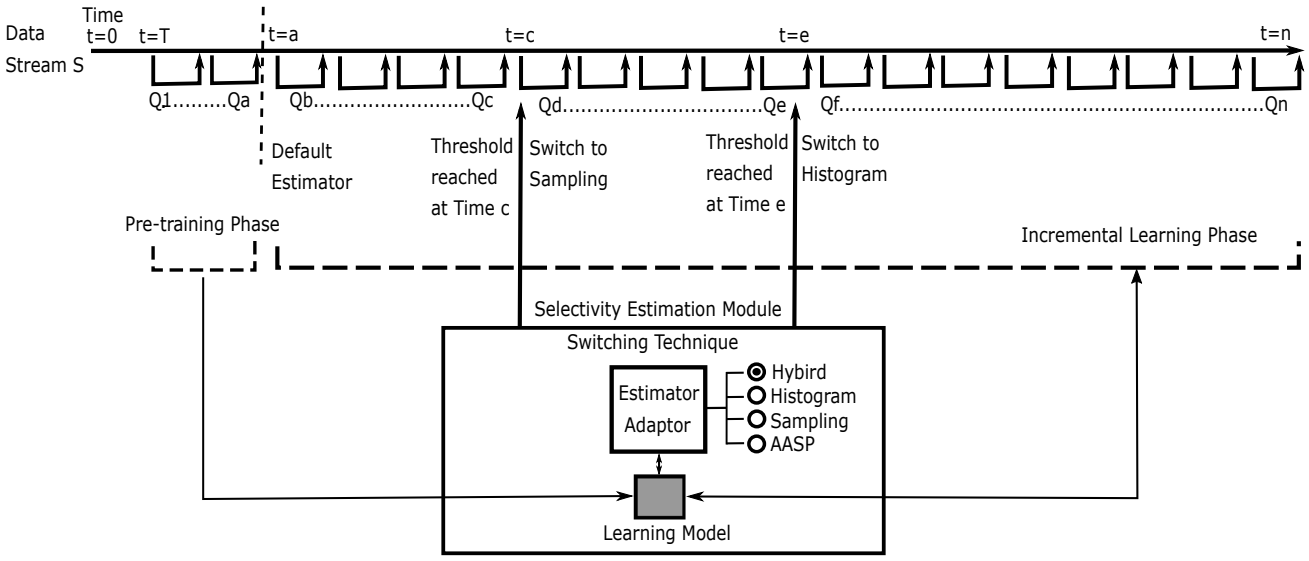


Fig. 2. Overview of *LATEST*

then Sections V-B to V-D detail the learning model and different phases of *LATEST* real-time operations.

A. LATEST Overview

Figure 2 depicts an overview for *LATEST* along the stream lifetime from time $t = 0$ to $t = n$. Along the stream lifetime, both data objects and queries are being received. As depicted in the figure, the stream lifetime is divided into three phases: (1) The *warm-up* phase of size T time units, from $t = 0$ to $t = T$. This phase is an idle phase that waits for enough data to arrive so that the time window can be applied to incoming queries. (2) The *pre-training* phase, from $t = T$ to $t = a$, when all incoming queries, Q_1 to Q_a, are executed on all underlying data structures that are introduced in Section IV to collect training data for the learning model. (3) The *incremental learning* phase that continuously answers incoming estimation queries using the currently employed estimator. This period starts with employing a default estimator; then *LATEST* keeps switching the estimation techniques through an *Estimator Adaptor*. The estimator adaptor is triggered based on changes in the query accuracy thresholds over time and chooses among different estimators based on the trained learning model. In the figure, two switching events happen at $t = c$ and $t = e$. The following sections detail the learning model choice and design (Section *V-B*), the *pre-training* phase (Section *V-C*), and the *incremental learning* phase (Section *V-D*).

B. The Learning Model

Design considerations. A recent study [23] has explored data stream learning models that update their models based on a continuous data flow. Existing learning models that are used in selectivity estimations are feed-forward neural networks [62], regular decision trees [59], and sum-product networks (SPNs) [57]. The first two models are used as workload-driven models that use training data from query parameters and results rather than raw data. In contrast, the third model is a data-driven model that uses raw data for training. However, applying these three techniques to our spatial streaming environment has shown severe limitations that make them impractical to be employed in real systems. In particular, we have applied a feed-forward neural network with multiple variations of hidden layers to a stream of 75 million geotagged tweets and 900 thousand queries that compose different types of query workloads, pure spatial, pure keywords, and hybrid queries. For different workloads, different architectures of the network have high variations of error rates, which clearly shows the model cannot adapt to changing query workloads over the stream lifetime to maintain a satisfying performance. Similar results are produced for the regular decision tree prediction. Without the constant model updates, which is the core requirement for streaming data, the decision tree fails to produce reasonably accurate results. An extended random forest also has similar issues, which is the reason we do not endorse the use of regular decision trees or random forest classifiers for the streaming environment. The third model is the sum-product network (SPN) that uses raw data with a sliding window for model training and updates. However, this model has a very high computational intensity to update the model with high-velocity data constantly.

In a streaming environment, the data arrives at high speeds, and its dynamic nature makes it difficult to depend on a pre-trained learning model as it will not provide high estimation accuracy for the whole stream lifetime. In fact, due to constant updates to the window of objects, the estimation accuracy of a model can even continuously decrease. A possible way to handle such dynamic nature is retraining the model often, which is not practical due to the high cost of both computing resources and system operation interruptions in real-time. Even with the use of query workloads and results for training, the

wide variety of query workloads makes it nearly impossible to train for all query type combinations.

***LATEST* model**. To overcome problems in existing solutions, we use a recent variation of *Hoeffding tree* [18] that is introduced in [44], a very fast decision tree (VFDT) algorithm that works efficiently as an incremental learning model for streaming data. In a streaming environment, all the historical data cannot be stored or used for learning. Hoeffding tree eliminates the need to reuse all instances from the beginning of the stream to determine the best split of features. Instead, it uses a sample of training records that are read incrementally to build a tree that converges similar to a batch learner tree with sufficiently large data. This sample's size is determined using the Hoeffding bound that quantifies the number of observations needed to estimate statistics with high precision.

With the use of Hoeffding bound, VFDT tree is created in constant time per training record, and therefore, it is much suited for learning in data streams. The strength of the Hoeffding tree is that it requires each example to be read at most once as it is incrementally built, and it is independent of the probability distribution generating the observations. Also, its learning accuracy significantly improves over time as training data is continuously updated based on incoming queries and their results, as shown in our experiments.

C. Pre-training Phase

In the pre-training phase, a set of queries are evaluated on all the underlying estimation data structures. In addition, after queries are executed on actual data, the system logs are used to obtain accuracy information for different estimation structures. So, query workload information is collected and used as features to train the Hoeffding tree. The collected features are: (1) data structure that we use for estimation, (2) query type, (3) query accuracy based on evaluation against actual data, (4) query latency, and (5) error rate. The estimator performance is mainly captured by two features: query accuracy and query latency. Both features are scaled through min-max normalization within the range $[0, 1]$. Then, both features are additionally scaled to represent their relative importance through a parameter α, where $0 \leq \alpha \leq 1$.

To allow different estimation data structures to evaluate queries in the pre-training phase, all these data structures are *pre-filled* during the warm-up phase that is introduced in Section V-A. In the warm-up phase, *LATEST* receives data, but not queries, for T time units. During this period, the incoming data is used to fill the estimation data structures. For example, the data objects are inserted in the sampling lists, the histogram counters are updated accordingly, and AASP trees are structured based on this data. Therefore, once the pre-training phase starts to receive estimation queries, these structures can be used to obtain the training data. This pre-filling does not guarantee that the estimator is filled up to its capacity, which could get unreal accuracy for some of the pre-training phase results. For example, if a reservoir sampler has a sampling list of one million objects capacity, there is no guarantee that T time units of the warm-up phase has one million objects to fill the list completely. To overcome this problem, the system administrator either stretches the warm-up phase or adjusts the length of the pre-training phase so estimators are completely filled, and hence the training data records reflect the real performance of different estimators.

After the pre-training phase is concluded, all estimation structures are wiped out to reduce the system overhead, except the one that will be used at the beginning of the next phase. Having this means only a single estimation structure is actively maintained at a time. The following phase shows the way *LATEST* switches from one active estimator to another while still maintaining low system overhead.

D. Incremental Learning Phase

In modern database systems, log files are used for maintaining essential debugging information [48] as well as statistics about index selectivity, estimation accuracy, etc. We exploit system logs to continuously improve the prediction abilities of our learning model throughout the stream lifetime. After the pre-training phase is concluded, the incremental learning phase starts with employing a default estimator to answer incoming queries. After answering each query, its results are kept for later accuracy evaluation. When the query plan goes from the query optimizer to the query processor, and the query is executed on actual data, system logs will include actual query selectivity information. This information will be used along with estimation results to calculate the estimation accuracy of this query. This accuracy is used in two ways. First, it is fed to the Hoeffding tree as an additional training record to improve its prediction capabilities. Second, combined with the accuracy of queries that arrived in the past time window, we compute an average accuracy score that represents the current *LATEST* performance. Once this average accuracy falls under a user-defined threshold τ, *LATEST* decides to switch to a different estimation technique, which is recommended by consulting the Hoeffding tree.

A problem that arises in the switching process is that the system maintains a single active estimator at any given time. Once *LATEST* decides to switch to a different estimator, this estimator is an empty data structure, e.g., sampling list or histogram counters. To overcome this, the actual decision is anticipated earlier when the average estimation accuracy falls below a *pre-filling threshold* $\beta * \tau$, where $0 \leq \beta \leq 1$. When this threshold is reached, the pre-filling phase starts. The following query in the queue is fed into the learning model to have a recommended estimator. Then, the recommended estimator begins to be filled with incoming data objects so that when the accuracy falls under τ, the new estimator is ready to be used. If the average accuracy starts to increase again, the pre-filled estimator is discarded, and the process repeats to keep monitoring the accuracy over time. The value of β represents a trade-off between system overhead and estimation accuracy. However, it does not significantly affect the system overhead due to the lightweight estimation data structures.

Over time, the Hoeffding tree model accuracy increases with incoming training records. Our empirical evaluation shows

that the model achieves a stable level of accuracy after 100K queries are used in both pre-training and incremental learning phases. Exploring systematic ways to tune the learning model parameters, e.g., splitting criteria or leaf prediction strategy, may expedite achieving stability. However, this remains an open area of research and exploring it in detail requires a significant effort that we consider beyond the scope of this paper. One of the major learning model issues in a streaming environment is how long the model remains viable before it has to be retrained to account for changes in the query types on the stream, or in essence, how often do we have to retrain the model. In *LATEST*, retraining of the model occurs for two reasons. First, the Hoeffding tree retrains itself for achieving the best splits when there are a minimum number of objects it has not seen previously. Second, the more important aspect of triggering the retraining process manually is the increase in latency times or overall error rate, which is calculated as the total relative error for all queries since the last training.

VI. Experimental Evaluation

This section presents the experimental evaluation of *LATEST*. Section VI-A presents the experimental setup. Section VI-B shows *LATEST* performance along stream lifetime with various query workloads, while Section VI-C shows the impact of trading estimation accuracy and latency. Sections VI-D and VI-E evaluate the impact of both spatial and keyword predicates, respectively. Finally, Section VI-F studies the impact of the estimation memory budget.

A. Experimental Setup

We evaluate the effectiveness and performance of *LATEST* using a real implementation for the technique based on WEKA library[2] implementation of Hoeffding tree along with six estimators listed below. For WEKA Hoeffding tree options, our leaf prediction strategy is *Majority Class*, splitting criteria is *Info Gain*, and other options are set to the library default values. Our performance measures include estimation accuracy and estimation query latency. All experiments are based on Java 8 implementation and using an AMD Ryzen 7 2700X server with 8-core CPUs and 4.125GHz clock speed, 64GB RAM, NVIDIA GeForce RTX 2070 GPU with 8GB frame buffer and 1620 MHz boost clock and running Windows 10.

Selectivity estimators. Our evaluation uses six major estimators from the literature (introduced in Section IV): two-dimensional histogram with 4096 cells (denoted as H4096), reservoir sampling list (denoted as RSL) with 1 million objects, a modified version of augmented adaptive space partition tree (denoted as AASP) with split value of 0.5, and the hybrid reservoir sampling hashmap (denoted as RSH) with 1 million objects and grid size of 4096 cells. In addition, we use workload driven feed-forward neural network (denoted as FFN) [62] based on WEKA library with hyperparameters of a learning rate of 0.3, momentum of 0.2, unipolar sigmoid activation function, and is trained until the generalization gap stops shrinking at 483 epochs, and data-driven sum-product network (denoted as SPN) [57] based on LibSPN library[3] through single op nodes. The reservoir sampling hashmap (RSH) [72] estimator works similar to the reservoir sampling list, except the data objects are stored in a two-dimensional grid instead of a list. The default employed estimator is RSH. The currently employed estimator is marked with a *dotted line* along the experiments.

Evaluation datasets. We use three evaluation datasets.

1) *Twitter* dataset: We simulate a data stream that runs for 10 hours with a total of 75 million geotagged tweets, collected from Twitter streaming APIs. All tweets are geotagged with either exact latitude/longitude coordinates or common places, e.g., cities, approximated with centroid points. Hashtags from tweet text act as keywords, otherwise, random words are used.
2) *eBird* dataset: We use 41+ million records to simulate a data stream that runs for 6 hours from the eBird dataset [17] from UCR Star[4] collected from Cornell Lab of Ornithology [5]. The records are used to simulate a data stream. Each record contains exact latitude/longitude coordinates and protocol type, breeding category, trip comments, and species comments that act as keywords.
3) *CheckIn* dataset: We simulate a data stream of 973,358 Foursquare check-ins [73] with exact latitude/longitude coordinates and tags that act as keywords. The data is obtained from UCR Star[4] originated from ITEE at the University of Queensland[6].

Query workloads. We use three sets of query workloads.

1) Workloads for Twitter dataset: All queries we use in these workloads contain a point location taken from *real queries of 'Bing' mobile search engine*, and optional keywords that are randomly selected from evaluation data. We compose nine query workloads similar to previous literature [11], [13], [68] each of 100K queries, with various percentages of pure spatial, pure keyword, or mixed queries. For example,
   - TwQW1: One-third of pure spatial, pure keyword, and spatial-keyword queries each.
   - TwQW2: 100% of pure spatial queries.
   - TwQW3: 50% pure spatial and 50% spatial-keyword queries.
   - TwQW4: 100% single keyword queries.
   - TwQW5: 100% multi-keyword queries.
   - TwQW6: One-third of pure spatial, pure keyword, and spatial-keyword queries, each with query types frequently changing in a different order compared to TwQW1.
2) Workloads for the eBird dataset: We use a real workload from the 'Da Vinci' server of UCR Star. This workload contains 40K ***real requests*** for dataset search with spatial range. We use it combined with keywords selected

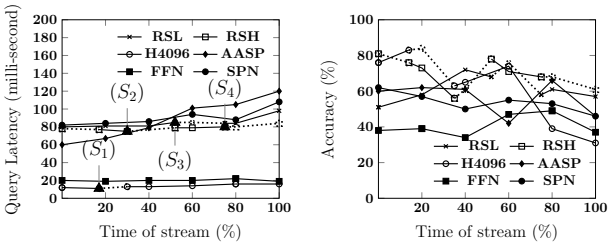
[2]https://www.cs.waikato.ac.nz/ml/weka/

[3]https://www.libspn.org/

[4]https://star.cs.ucr.edu/

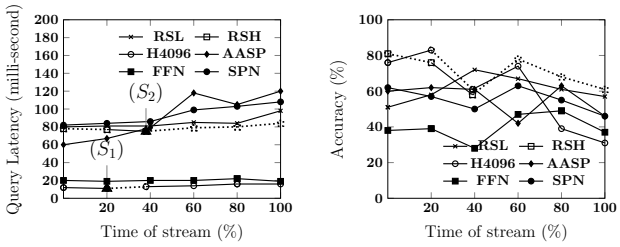
[5]https://www.birds.cornell.edu/

[6]https://sites.google.com/site/dbhongzhi/



(a) Query latency (b) Estimation accuracy

Fig. 3. Estimator switches for query workload with changing order in TwQW1



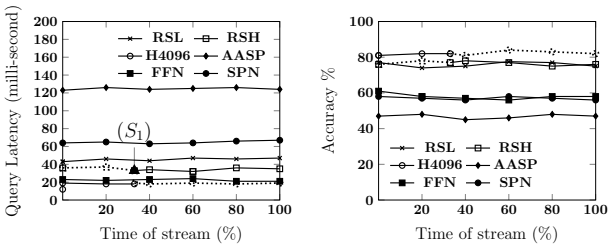
(a) Query latency (b) Estimation accuracy

Fig. 4. Estimator switches for query workload TwQW6

randomly from the dataset to generate a total of six workloads of different query type distributions. We use workload EbRQW1 that contains 100% of spatial queries.

3) Workloads for the CheckIn dataset: Similar to the Twitter dataset workloads, we use Bing search locations to generate three workloads of different distributions of query types. We use CiQW1 workload that has 100K of single keyword queries.

For brevity, we have only described the workloads above that are used in further evaluation sections.



(a) Query latency (b) Estimation accuracy

Fig. 5. Estimator switches for query workload EbRQW1

B. Real-time Estimator Switching

This section presents an experiment to observe the effectiveness of our switching technique on a stream starting from time t_0 to time t_{100} that represent the start and end times of the incremental learning phase, respectively. So, t_0 is the time point directly after concluding the pre-training phase, and t_{100} is the last time point of running the stream during the experiment. Intermediate times are symbolized with interpolated subscript between 0 and 100, so, for example, t_{50} is the middle timestamp for the incremental learning phase, and so on. Figures 3, 4, and 5 show *LATEST* switching for different query workloads encountering one or more switches over its lifetime. The horizontal axis represents a timeline for the stream, and the vertical axes represent estimation query latency and accuracy. In the beginning, the default estimator is RSH. Along the timeline, the currently employed estimator is marked with a dotted line.

Query workload TwQW1 clearly shows the effectiveness of *LATEST* switching among different estimators more than all other workloads. This workload has one-third of each of pure spatial, pure keyword, and spatial-keyword queries, so the types of queries are heavily changing over time. On this workload, as depicted in Figure 3, the stream starts with RSH estimator, then *LATEST* makes four switches at t_{18}, t_{31}, t_{53}, and t_{75}, to H4096, RSH, RSL, and RSH estimators, respectively, switches are marked as S_1, S_2, S_3, and S_4 in Figure 3(a). In each of these switches, *LATEST* compromises estimation accuracy with latency. At all times, the H4096 estimator provides the best latency, but it does not consistently provide the best accuracy as it works better for pure spatial queries while poorly estimates queries with keyword predicates. From t_{18} to t_{31}, when pure spatial queries are dominating the workload, *LATEST* switches to H4096 estimator to provide the best of both latency and accuracy. At other times, *LATEST* switches back and forth between RSH and RSL that provide the best accuracy in their corresponding times (as shown in Figure 3(b)) even with higher latency. This experiment shows the importance of parameter α that weighs the importance of accuracy and latency during the learning process, so the model suggests an estimator with a favorable performance measure in case of contradiction between different measures. We evaluate the different values of α later in Section VI-C. For all switches, *LATEST* chooses the model with the best estimation accuracy (Figure 3(b)), which shows its effectiveness in handling changing query workloads in real time on spatial streaming data.

For query workload TwQW6, *LATEST* performs two switches at t_{18} and t_{39} as shown in Figure 4. The first switch (S_1) discards RSH estimator and employs H4096 estimator, while the second switch S_2 discards H4096 estimator and switches back to RSH till t_{100}. As shown in Figures 4(a) and 4(b), H4096 has the lowest latency, but it has lower accuracy on this workload that includes a significant ratio, two thirds, of keyword queries. Between t_{18} and t_{39}, when the one-third of pure spatial queries are dominating, H4096 estimator provides better accuracy in addition to its better latency, so it is chosen by *LATEST*. After that, when keyword predicates resume flowing in the stream, H4096 estimation accuracy falls again, and hence *LATEST* switches back to RSH estimator, which provides the best accuracy among all estimators. This experiment clearly shows again that *LATEST* is able to switch back and forth among estimators based on their performance on changing query workloads in real time. This confirms our experiment on query workload 9.

For the real query workload EbRQW1, *LATEST* makes one switch at t_{34} as shown in Figure 5. This switch discards the default RSH estimator and employs H4096 estimator. As shown

TABLE I
INDEX OVERHEAD COMPARISON

Dataset & Index Latency	Estimator	
	Average Latency	Average Accuracy
eBird, Grid (322ms)	H4096 (20ms)	76%
eBird, Grid (322ms)	RSL (53ms)	68%
eBird, Grid (322ms)	RSH (34ms)	71%
eBird, QuadTree (291ms)	AASP (111ms)	48%
CheckIn, Grid (460ms)	RSL (48ms)	70%
CheckIn, Grid (460ms)	RSH (41ms)	66%
CheckIn, QuadTree (402ms)	AASP (118ms)	64%
Twitter, Grid (340ms)	H4096 (19ms)	75%
Twitter, Grid (340ms)	RSL (71ms)	71%
Twitter, Grid (340ms)	RSH (74ms)	72%
Twitter, QuadTree (265ms)	AASP (96ms)	44%

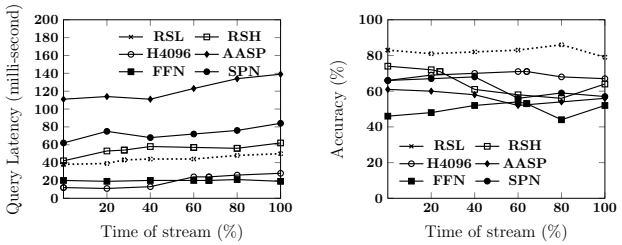
TABLE II
IMPACT OF α ON QUERY WORKLOAD TWQW3

α	$LATEST$ choice		
	t=20	t=60	t=100
0	RSL	RSL	RSL
0.3	RSL	RSL	RSL
0.5	RSL	RSL	RSL
0.7	H4096	H4096	H4096
1	H4096	H4096	FFN

in Figures 5(a) and 5(b), H4096 has the lowest latency and highest accuracy among the estimators. RSH also has similar accuracy but due to sustained higher latency than H4096, the switch is performed. This confirms *LATEST* ability to adjust estimators for real query workloads.

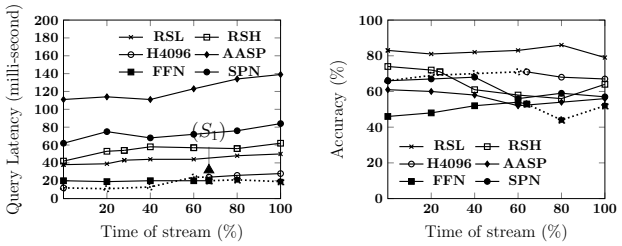
It is worth emphasizing that other query workloads that are excessively dominated by either pure spatial predicates or pure keyword predicates do not encounter switches in *LATEST* decisions over time, yet, it still provides the best accuracy at all time points. This highlights two important points. First, *LATEST* decision is truly dependant on the underlying workload, and it clearly distinguishes workloads that need several switches from those that need no switches at all. Second, *LATEST* is more effective with dynamically changing workloads where types of queries are heavily changing over the stream lifetime. So, it effectively adapts to those workloads, which is the main goal behind designing *LATEST*. This emphasizes its appeal for system designers as a flexible system-level module that supports various types of query workloads.

To further emphasize the strength of *LATEST*, Table I shows the querying overhead of spatial grid and quadtree index structures in comparison to different estimators. For the eBird dataset on query workload EbRQW1, estimator latency values change between 20ms to 111ms while indexes take 1450%-1600% more time than the estimator chosen by *LATEST* (H4096). This index performance is similar to our previous extensive study on spatial keyword indexes [2]. Similar results are also seen with CheckIn's CiQW1 workload and Twitter's TwQW4 workload. This clearly shows the merit of *LATEST* for our motivating applications.



(a) Query latency (b) Estimation accuracy

Fig. 6. Estimator switches for query workload TwQW3 for $\alpha = 0$



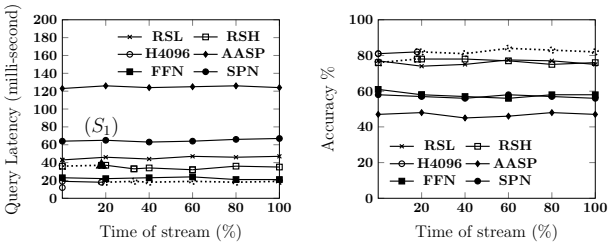
(a) Query latency (b) Estimation accuracy

Fig. 7. Estimator switches for query workload TwQW3 for $\alpha = 1$

C. Relative Importance of Learning Model Features

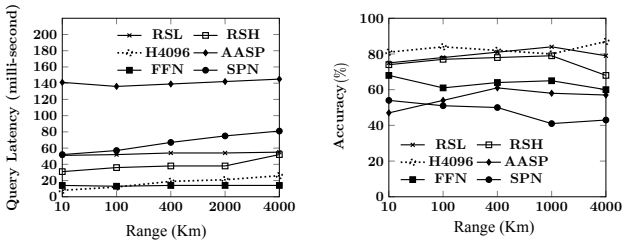
This section evaluates the performance of *LATEST* for various settings of parameter α that weighs the relative importance of estimation query accuracy and query latency as *training features* for the model. If $\alpha = 0$, then estimation accuracy is the only weighted factor, and latency is totally ignored, while $\alpha = 1$ gives the whole weight to query latency and ignores accuracy. Other values of α between 0 and 1 give partial importance for each factor as detailed in Section V-C. All previous experiments are presented with default $\alpha = 0.5$, while this section closely investigates the effect of changing α value on the model decisions along the data stream lifetime.

Figures 6 and 7 show estimator switches over stream lifetime for query workload TwQW3, for both extreme values of α, $\alpha = 0$ in Figure 6 and $\alpha = 1$ in Figure 7. As shown in Figure 6(b), *LATEST* always chooses the best accuracy while the estimator latency is sub-optimal (Figure 6(a)) as it is not considered in the training features. The opposite behavior can be seen in Figure 7, where *LATEST* always chooses the best latency (Figure 7(a)) regardless of the sub-optimal accuracy (Figure 7(b)). This confirms the impact of α on *LATEST* decisions during the system operations. So, in its default state, *LATEST* model tends to favor estimation accuracy over estimator latency. This is a favorite performance for several systems as estimation accuracy drives inducing cheap query plans and saves significant time in actual query processing. Overall, the significant changes in model decisions clearly show the impact of α on driving *LATEST* decisions, which provides system administrators with a solid tuning knot for system performance based on the desired output characteristics. When accuracy is favored over other factors, the system administrator is able to boost the system performance to provide the best estimation accuracy and the same for latency.



(a) Query latency (b) Estimation accuracy

Fig. 8. Estimator switches for query workload EbRQW1 for $\alpha = 1$

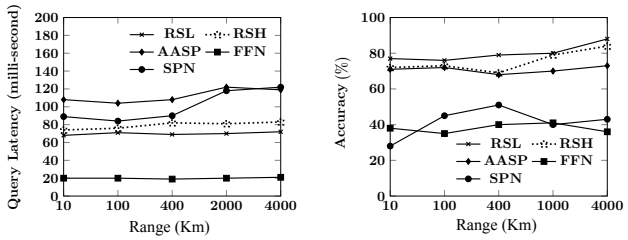


(a) Query latency (b) Estimation accuracy

Fig. 9. Varying spatial ranges on query workload TwQW1

Figures 8(a) and 8(b) show the adaptability and effectiveness of *LATEST* on the eBird dataset. This experiment is similar to the previous experiment shown in Figures 5(a) and 5(b) with the key difference of emphasis on latency as the performance measure ($\alpha = 1$), which highlights similar behavior for different datasets as *LATEST* successfully switches to the estimator with the lowest latency.

Table II shows the impact of various α values on query workload TwQW3. The table shows *LATEST* choices at three time points, $t = 20$, $t = 60$, and $t = 100$ for each value of α. The corresponding values of estimation query latency and accuracy are depicted in Figures 3, and 4 respectively. The values of accuracy and latency do not depend on the value of α, as they are provided by the estimator based only on the incoming data and queries, regardless of whether a certain estimator is selected by *LATEST* or not. However, the value of α solely affects the estimator selection based on the relative importance of accuracy and latency. In Table II, changing the value of α greatly affects the estimator selection decision, so at $0 \leq \alpha \leq 0.5$, RSL estimator is chosen as it provides the best accuracy. For $0.5 < \alpha \leq 1$, latency is given a priority, and hence H4096 and FFN estimators dominate the selection regardless of their low accuracy. This impact changes depending on the query workload. Thus, while the streaming workload characteristics change dynamically over time, *LATEST* dynamically adapts to select the best estimator based on the pre-set α value. So, the effect of the tuning parameter α on different query workloads adapt with changing both workload characteristics and estimator's performance. In all cases, it provides system administrators with a powerful and flexible tool to tune the system performance effectively.



(a) Query latency (b) Estimation accuracy

Fig. 10. Varying spatial ranges on query workload TwQW4

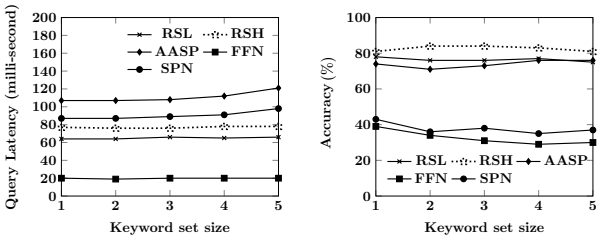
D. Spatial Impact

This section shows the impact of different spatial ranges on estimation latency and accuracy of different estimators, highlighting *LATEST* choices in each case. Figures 9 and 10 show the spatial impact on different query workloads. Each figure reports estimation latency and accuracy, on the vertical axes, for queries of different spatial ranges, on the horizontal axis, for the corresponding workload. The general observation in all these figures is that in most cases, increasing the spatial range has an insignificant effect for a certain estimator, with a few exceptions, yet, different estimators significantly vary from each other. Specifically, Figure 9 shows the superiority of the H4096 histogram estimator selected by *LATEST* for different spatial ranges, while the hierarchical structure, AASP encounters the highest latency with mean accuracy. All estimators outperform AASP, and H4096 estimator shows the highest difference that clearly shows the insufficiency of the tightly coupled approach in [67] to work with dynamic workloads with mixed spatial and keyword predicates. However, all estimators encounter almost stable or slight differences for different spatial ranges.

For both presented query workloads, *LATEST* always selects the estimator with the highest estimation accuracy for different spatial ranges. This confirms the effectiveness and adaptivity of *LATEST* model to deal with various query workloads in streaming environments.

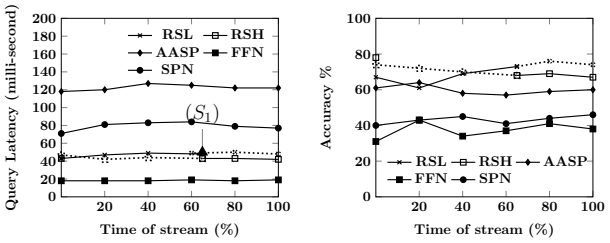
E. Keyword Impact

This section studies the effect of varying the number of keywords in a query on the performance of the estimators and *LATEST* selection decisions. We use query workload TwQW5 where we change the number of keywords in the input. As all queries are pure keyword queries, H4096 estimator is not included in the comparison as it uses purely spatial statistics. Figure 11 shows the performance of different estimators when the number of keywords changes from 1 to 5. The reported latency and accuracy are corresponding to the end of the incremental learning phase. The figure shows that *LATEST* consistently chooses RSH estimator (marked with dotted line), which always achieves the highest estimation accuracy for the different number of keywords. In addition, the figure shows stable latency for all estimators with a varying number of keywords. The estimation accuracy is high and stable for RSH, AASP, and RSL estimators, while it is lower for FFN and SPN and slightly decrease with increasing keywords. In addition to



(a) Query latency (b) Estimation accuracy

Fig. 11. Varying keyword set size for query workload TwQW5



(a) Query latency (b) Estimation accuracy

Fig. 12. Estimator switches for query workload CiQW1

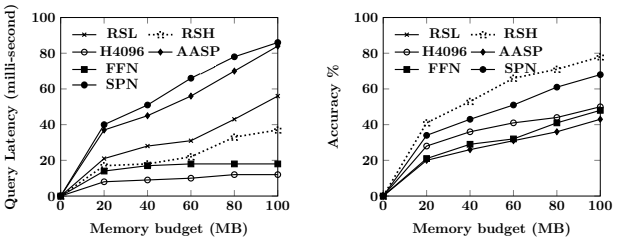
showing the effectiveness of *LATEST* choices, this experiment gives insights on the effectiveness of different sampling and adaptive tree streaming estimators on pure keyword queries that include multiple keywords. Figures 12(a) and 12(b) shows *LATEST*'s ability to sustain its effective performance for the CheckIn dataset on CiQW1 query workload. For this workload, we see that improving the accuracy of the RSL estimator drives the switch at t_{63} (marked as S_1).

F. Memory Budget Impact

This section studies the effects of varying the allocated memory budgets for different estimators. Although estimator memory consumption is modest compared to the available memory in existing systems, it affects the estimation accuracy and latency. So, we are more interested in studying its effect, even if it is relatively small. Figure 13 shows the performance of different estimators with varying memory budgets. *LATEST* choice still provides the best accuracy at all values. In addition, the figure shows high variability among estimators in the rate of increased latency as shown in Figure 13(a). As this figure shows, AASP and SPN estimators have a linear increase in latency with increasing memory budget, while the rest of the estimators have a sub-linear increase. However, all estimators have a similar uptrend in estimation accuracy (Figure 13(b)), where RSH estimator always performs the best, so it is chosen by *LATEST* for all values. These results give system administrators insights on the trade-off of improving accuracy while scarifying query latency and which estimators are able to maintain low latency while providing high accuracy.

VII. Conclusion

This paper studies the selectivity estimation problem for spatial-keyword queries in spatial streaming environments. We have shown the shortcomings of existing techniques to dynamically adapt to changing query workloads in spatial streaming environments. To overcome these shortcomings, we have built a system-level module *LATEST* that dynamically selects an appropriate estimator for currently flowing queries on the data stream. *LATEST* incorporates an incremental supervised learning model that switches among different estimators to appropriately employ the best estimator for each time period of the stream lifetime based on a time window streaming model. We used six example widely-used estimators for empirical evaluation. Our extensive experimental evaluation on real data shows the effectiveness of *LATEST* for various query workloads of mixed spatial and keyword predicates. *LATEST* is clearly able to distinguish workloads that need several switches in estimators over time from those that need no switches at all. Also, for dynamically changing workloads where types of queries heavily change over time, *LATEST* shows effective adaptation to provide the best combinations of estimation accuracy and latency along the stream lifetime.



(a) Query latency (b) Estimation accuracy

Fig. 13. Varying memory budget for queries on the Twitter dataset

Acknowledgement

This work is partially supported by the National Science Foundation, USA, under grants IIS-1849971, SES-1831615, and CNS-2031418 and the US DoE GAANN Fellowship.

References

[1] S. Acharya, V. Poosala, and S. Ramaswamy. Selectivity Estimation in Spatial Databases. In *SIGMOD*, 1999.

[2] A. Almaslukh and A. Magdy. Evaluating Spatial-Keyword Queries on Streaming Data. In *SIGSPATIAL*, 2018.

[3] Z. Bar-Yossef, T. S. Jayram, R. Kumar, D. Sivakumar, and L. Trevisan. Counting Distinct Elements in a Data Stream. In J. D. P. Rolim and S. Vadhan, editors, *Randomization and Approximation Techniques in Computer Science*, 2002.

[4] K. Beyer, P. J. Haas, B. Reinwald, Y. Sismanis, and R. Gemulla. On Synopses for Distinct-Value Estimation under Multiset Operations. In *SIGMOD*, 2007.

[5] M. Brown. The Utility Network: Introduction to the Next Generation of Network Management. https://proceedings.esri.com/library/userconf/seuc19/papers/SEUC_38.pdf, 2019.

[6] N. Bruno, S. Chaudhuri, and L. Gravano. STHoles: A Multidimensional Workload-Aware Histogram. In *SIGMOD*, 2001.

[7] X. Cao, L. Chen, G. Cong, and X. Xiao. Keyword-Aware Optimal Route Search. *VLDB*, 2012.

[8] X. Cao, G. Cong, C. S. Jensen, and B. C. Ooi. Collective Spatial Keyword Querying. In *SIGMOD*, 2011.

[9] H. Chasparis and A. Eldawy. Experimental Evaluation of Selectivity Estimation on Big Spatial Data. In *GeoRich*, 2017.

[10] C. M. Chen and N. Roussopoulos. Adaptive Selectivity Estimation Using Query Feedback. In *SIGMOD*, 1994.

[11] L. Chen, G. Cong, and X. Cao. An efficient query indexing mechanism for filtering geo-textual data. In *SIGMOD*, 2013.

[12] L. Chen, G. Cong, C. S. Jensen, and D. Wu. Spatial Keyword Query Processing: An Experimental Evaluation. *VLDB*, 2013.

[13] L. Chen, S. Shang, C. Yang, and J. Li. Spatial keyword search: a survey. *Geoinformatica*, 2020.

[14] E. Cohen, G. Cormode, and N. Duffield. Structure-Aware Sampling on Data Streams. In *SIGMETRICS*, 2011.

[15] G. Cong, C. S. Jensen, and D. Wu. Efficient Retrieval of the Top-k Most Relevant Spatial Web Objects. *VLDB*, 2009.

[16] G. Cormode, M. Garofalakis, P. J. Haas, and C. Jermaine. Synopses for Massive Data: Samples, Histograms, Wavelets, Sketches. *TRDB*, 2012.

[17] N. Y. Cornell Lab of Ornithology, Ithaca. ebird basic dataset. version: Ebd_reljun-2020. Retrieved from UCR-STAR https://star.cs.ucr.edu/?eBird.

[18] P. Domingos and G. Hulten. Mining High-Speed Data Streams. In *SIGKDD*, 2000.

[19] A. Dutt, C. Wang, A. Nazi, S. Kandula, V. Narasayya, and S. Chaudhuri. Selectivity Estimation for Range Predicates Using Lightweight Models. *VLDB*, 2019.

[20] M. Eftekhar and N. Koudas. Some Research Opportunities on Twitter Advertising. *IEEE Data Engineering Bulletin*, 2013.

[21] L. Getoor and L. Mihalkova. Learning Statistical Models from Relational Data. In *SIGMOD*, 2011.

[22] L. Getoor, B. Taskar, and D. Koller. Selectivity Estimation Using Probabilistic Models. In *SIGMOD*, 2001.

[23] H. M. Gomes, J. Read, A. Bifet, J. P. Barddal, and J. a. Gama. Machine Learning for Streaming Data: State of the Art, Challenges, and Opportunities. *SIGKDD Explorations*, 2019.

[24] D. Gunopulos, G. Kollios, J. Tsotras, and C. Domeniconi. Selectivity Estimators for Multidimensional Range Queries over Real Attributes. *VLDB Journal*, 2005.

[25] P. J. Haas, J. F. Naughton, and A. N. Swami. On the Relative Cost of Sampling for Join Selectivity Estimation. In *PODS*, 1994.

[26] S. Hasan, S. Thirumuruganathan, J. Augustine, N. Koudas, and G. Das. Multi-Attribute Selectivity Estimation Using Deep Learning. *CoRR. abs/1903.09999*, 2019.

[27] R. Hayek and O. Shmueli. Improved Cardinality Estimation by Learning Queries Containment Rates. eprint: 1908.07723, 2019.

[28] M. Heimel, M. Kiefer, and V. Markl. Self-Tuning, GPU-Accelerated Kernel Density Models for Multidimensional Selectivity Estimation. In *SIGMOD*, 2015.

[29] J. Hershberger, N. Shrivastava, S. Suri, and C. D. Toth. Adaptive Spatial Partitioning for Multidimensional Data Streams. *Algorithmica*, 2006.

[30] B. Hilprecht, A. Schmidt, M. Kulessa, A. Molina, K. Kersting, and C. Binnig. DeepDB: Learn from Data, Not from Queries! *VLDB*, 2020.

[31] B. Huang, B. Jiang, and H. Li. An integration of GIS, virtual reality and the Internet for visualization, analysis and exploration of spatial data. *International Journal of Geographical Information Science*, 2001.

[32] S. Idreos and T. Kraska. From Auto-Tuning One Size Fits All to Self-Designed and Learned Data-Intensive Systems. In *SIGMOD*, 2019.

[33] O. Ivanov and S. Bartunov. Adaptive Cardinality Estimation. *CoRR. abs/1711.08330*, 2017.

[34] E. P. Karan and J. Irizarry. Developing a spatial data framework for facility management supply chains. In *Construction Research Congress 2014: Construction in a Global Network*, 2014.

[35] A. Kipf, T. Kipf, B. Radke, V. Leis, P. A. Boncz, and A. Kemper. Learned Cardinalities: Estimating Correlated Joins with Deep Learning. *CoRR. abs/1809.00677*, 2018.

[36] T. Kraska, M. Alizadeh, A. Beutel, E. H. Chi, J. Ding, A. Kristo, G. Leclerc, S. Madden, H. Mao, and V. Nathan. SageDB: A Learned Database System. *CIDR*, 2019.

[37] T. Kraska, A. Beutel, E. H. Chi, J. Dean, and N. Polyzotis. The Case for Learned Index Structures. In *SIGMOD*, 2018.

[38] M. Krause-Traudes, S. Scheider, S. Rüping, and H. Meßner. Spatial data mining for retail sales forecasting. In *International Conference on Geographic Information Science*, 2008.

[39] M. S. Lakshmi and S. Zhou. Selectivity Estimation in Extensible Databases - A Neural Network Approach. In *VLDB*, 1998.

[40] J. J. Lin and G. Mishne. A Study of "Churn" in Tweets and Real-Time Search Queries. In *ICWSM*, 2012.

[41] H. Liu, M. Xu, Z. Yu, V. Corvinelli, and C. Zuzarte. Cardinality Estimation Using Neural Networks. In *CASCON*, 2015.

[42] L. Ma, D. Van Aken, A. Hefny, G. Mezerhane, A. Pavlo, and G. J. Gordon. Query-Based Workload Forecasting for Self-Driving Database Management Systems. In *SIGMOD*, 2018.

[43] A. R. Mahmood, A. M. Aly, T. Qadah, E. K. Rezig, A. Daghistani, A. Madkour, A. S. Abdelhamid, M. S. Hassan, W. G. Aref, and S. Basalamah. Tornado: A Distributed Spatio-Textual Stream Processing System. *VLDB*, 2015.

[44] C. Manapragada, G. I. Webb, and M. Salehi. Extremely Fast Decision Tree. In *SIGKDD*, 2018.

[45] R. Marcus, P. Negi, H. Mao, C. Zhang, M. Alizadeh, T. Kraska, O. Papaemmanouil, and N. Tatbul. Neo: A Learned Query Optimizer. *VLDB*, 2019.

[46] Y. Matias, J. S. Vitter, and M. Wang. Wavelet-Based Histograms for Selectivity Estimation. In *SIGMOD*, 1998.

[47] B. Michael and C. Corey. Announcing CHIME, A tool for COVID-19 capacity planning. http://predictivehealthcare.pennmedicine.org/2020/03/14/accouncing-chime.html, 2020.

[48] Microsoft. Logs in SQL Server. https://docs.microsoft.com/en-us/sql/relational-databases/logs/, visited: 2020-01-30.

[49] M. F. Mokbel and A. Magdy. Microblogs Data Management Systems: Querying, Analysis, and Visualization. In *SIGMOD*, 2016.

[50] M. Müller, G. Moerkotte, and O. Kolb. Improved Selectivity Estimation by Combining Knowledge from Sampling and Synopses. *VLDB*, 2018.

[51] M. Muralikrishna and D. J. DeWitt. Equi-Depth Multidimensional Histograms. *SIGMOD Record.*, 1988.

[52] V. Nathan, J. Ding, M. Alizadeh, and T. Kraska. Learning Multi-dimensional Indexes. In *SIGMOD*, 2020.

[53] H. Oosterhuis, J. S. Culpepper, and M. de Rijke. The Potential of Learned Index Structures for Index Compression. In *ADCS*, 2018.

[54] J. Ortiz, M. Balazinska, J. Gehrke, and S. S. Keerthi. An Empirical Analysis of Deep Learning for Cardinality Estimation. *CoRR. abs/1905.06425*, 2019.

[55] A. Pavlo, M. Butrovich, A. Joshi, L. Ma, P. Menon, D. V. Aken, L. J. Lee, and R. Salakhutdinov. External vs. Internal: An Essay on Machine Learning Agents for Autonomous Database Management Systems. *IEEE Data Engineering Bulletin*, 2019.

[56] A. Pečar, M. Zidar, and M. Kukar. Reservoir Sampling Techniques in Modern Data Analysis. In *BCI*, 2012.

[57] H. Poon and P. M. Domingos. Sum-Product Networks: A New Deep Architecture. *CoRR. abs/1202.3732*, 2012.

[58] M. Riondato, M. Akdere, U. undefinedetintemel, S. B. Zdonik, and E. Upfal. The VC-Dimension of SQL Queries and Selectivity Estimation through Sampling. In *ECML PKDD*, 2011.

[59] L. Rokach and O. Maimon. *Data Mining With Decision Trees: Theory and Applications*. World Scientific, 2014.

[60] H. Samet. The Quadtree and Related Hierarchical Data Structures. *ACS*, 1984.

[61] J. Sankaranarayanan, H. Samet, B. E. Teitler, M. D. Lieberman, and J. Sperling. TwitterStand: News in Tweets. In *SIGSPATIAL*, 2009.

[62] J. Schmidhuber. Deep Learning in Neural Networks: An Overview. *CoRR. abs/1404.7828*, 2014.

[63] M. Shekelyan, A. Dignös, and J. Gamper. DigitHist: A Histogram-Based Data Summary with Tight Error Bounds. *VLDB*, 2017.

[64] Time. Hurricane Harvey Victims Turn to Twitter and Facebook. https://time.com/4921961/hurricane-harvey-twitter-facebook-social-media/, 2017.

[65] G. Valentin, M. Zuliani, D. C. Zilio, G. Lohman, and A. Skelley. DB2 advisor: an optimizer smart enough to recommend its own indexes. In *ICDE*, 2000.

[66] J. S. Vitter. Random Sampling with a Reservoir. *TOMS*, 1985.

[67] X. Wang, Y. Zhang, W. Zhang, X. Lin, and W. Wang. Selectivity Estimation on Streaming Spatio-Textual Data Using Local Correlations. *VLDB*, 2014.

[68] X. Wang, Y. Zhang, W. Zhang, X. Lin, and W. Wang. Ap-tree: Efficiently support continuous spatial-keyword queries over stream. In *ICDE*, 2015.

[69] WSJ. Health Department Use of Social Media to Identify Foodborne Illness - Chicago, Illinois, 2013-2014. https://www.cdc.gov/mmwr/preview/mmwrhtml/mm6332a1, 2014.

[70] WSJ. In Irma, Emergency Responders' New Tools: Twitter and Facebook. https://www.wsj.com/articles/for-hurricane-irma/information-officialspost-on-social-media/1505149661, 2017.

[71] Z. Yang, E. Liang, A. Kamsetty, C. Wu, Y. Duan, X. Chen, P. Abbeel, J. M. Hellerstein, S. Krishnan, and I. Stoica. Deep Unsupervised Cardinality Estimation. *VLDB*, 2019.

[72] Yibei Ling, Wei Sun, N. D. Rishe, and Xianjing Xiang. A hybrid estimator for selectivity estimation. *TKDE*, 1999.

[73] H. Yin. Poi or location-based recommendation dataset twitter-foursquare. Retrieved from UCR-STAR https://star.cs.ucr.edu/?yin/foursquare.